

# ПРОГРАММИРОВАНИЕ ДЛЯ ДЕТЕЙ

SCRATCH!

Э Л С В Е Й Г А Р Т

ДЛЯ ДЕТЕЙ СТАРШЕ 8 ЛЕТ И ИХ РОДИТЕЛЕЙ



AL SWEIGART

# SCRATCH PROGRAMMING PLAYGROUND

LEARN TO PROGRAM BY MAKING COOL GAMES





# ПРОГРАМИРОВАНИЕ ДЛЯ ДЕТЕЙ

ЭЛ СВЕЙГАРТ

# **ПРОГРАММИРОВАНИЕ ДЛЯ ДЕТЕЙ**

ДЕЛАЙ ИГРЫ И УЧИ ЯЗЫК SCRATCH!



МОСКВА  
2017



УДК 087.5:004.43  
ББК 32.973.26-018.1  
С24

Copyright © 2016 by Al Sweigart. Title of English-language original: Scratch Programming Playground: Learn to Program by Making Cool Games, ISBN 978-1-59327-762-8, published by No Starch Press. All rights reserved.

**Свейгарт, Эл.**  
С24 Программирование для детей: делай игры и учи язык Scratch! / Эл Свейгарт ; [пер. с англ. М. Райтман]. — Москва : Эксмо, 2017. — 304 с. — (Программирование для детей).

ISBN 978-5-699-98943-0

Интересный, понятный и красочный самоучитель для детей по самому популярному в мире языку для начинающих программистов — Scratch. На примере создания веселых классических игр, таких как «Змейка» и «Фруктовый ниндзя», читатели не только осваивают Scratch, но и лучше понимают основные принципы программирования. Книга будет интересна и тем, кто никогда не программировал, и тем, кто хочет создавать собственные игры, но не знает как. Она подходит как для самостоятельного изучения Scratch, так и для совместных занятий с родителями или педагогом.

УДК 087.5:004.43  
ББК 32.973.26-018.1

ISBN 978-5-699-98943-0

© Райтман М., перевод на русский язык, 2017  
© Оформление. ООО «Издательство «Эксмо», 2017

# Оглавление

Об авторе . . . . .	11
---------------------	----

## **ВВЕДЕНИЕ** **13**

Для кого предназначена эта книга . . . . .	14
Об этой книге . . . . .	15
Как пользоваться этой книгой . . . . .	17
Веб-ресурсы . . . . .	18

## **ГЛАВА 1. НАЧАЛО РАБОТЫ СО SCRATCH** **19**

Запуск Scratch . . . . .	21
Автономный редактор . . . . .	22
Редактор Scratch и спрайты . . . . .	22
Графический редактор . . . . .	24
Работа с блоками кода . . . . .	27
Добавление блоков . . . . .	27
Удаление блоков . . . . .	28
Запуск программ . . . . .	29
Демонстрация ваших программ . . . . .	30
Получение помощи . . . . .	30
Окно подсказок . . . . .	31
Просмотр проектов других пользователей . . . . .	32
Заключение . . . . .	33

## **ГЛАВА 2. РАДУЖНЫЕ ЛИНИИ В КОСМОСЕ!** **35**

Эскиз проекта . . . . .	36
<b>А. Создание космического фона</b> . . . . .	38
1. Очистка и настройка сцены . . . . .	38
<b>Б. Создание трех движущихся точек</b> . . . . .	40
2. Рисование точки . . . . .	40
3. Добавление кода для спрайта <b>Точка 1</b> . . . . .	42
4. Дублирование спрайта <b>Точка 1</b> . . . . .	45
<b>В. Прорисовка линий радуги</b> . . . . .	45
5. Добавление кода спрайта <b>Рисующая точка</b> . . . . .	45
Завершенная программа . . . . .	48
Турбо-режим . . . . .	49
Версия 2.0: Радужные треугольники . . . . .	50
Версия 3.0: Две радужные линии . . . . .	51
Версия 4.0: Самостоятельная работа . . . . .	52
Заключение . . . . .	52
Обзорные вопросы . . . . .	54

## ГЛАВА 3. БЕГУЩИЙ В ЛАБИРИНТЕ

55

Эскиз проекта. . . . .	56
<b>А. Создание прогуливающегося кота</b> . . . . .	58
1. Добавление кода движения для спрайта игрока . . . . .	61
2. Дублирование кода движения для спрайта кота . . . . .	62
<b>Б. Создание уровней лабиринта</b> . . . . .	64
3. Загрузка изображений лабиринта . . . . .	64
4. Изменение фона . . . . .	64
5. Создание первого лабиринта . . . . .	65
<b>В. Ограничение движения кота при помощи стен</b> . . . . .	65
6. Проверим, касается ли кот стен . . . . .	66
<b>Г. Добавление награды в конце лабиринта</b> . . . . .	68
7. Создание спрайта яблока . . . . .	68
8. Выявление момента, когда игрок находит яблоко. . . . .	68
9. Добавление кода обработки сообщения в спрайт <b>Лабиринт</b> . . . . .	70
Законченная программа. . . . .	71
Версия 2.0: Режим для двух игроков . . . . .	72
Дублирование спрайта <b>Яблоко</b> . . . . .	72
Изменение кода спрайта <b>Яблоко2</b> . . . . .	73
Дублирование спрайта <b>Рыжий кот</b> . . . . .	74
Изменение кода спрайта <b>Синий кот</b> . . . . .	74
Возвращение в начальное положение. . . . .	75
Версия 3.0: Ловушки. . . . .	77
Создание нового спрайта для ловушек . . . . .	77
Создание второго костюма для ловушек . . . . .	78
Добавление скопированного кода для ловушек . . . . .	80
Изменение кода спрайта <b>Рыжий кот</b> . . . . .	81
Копирование кода из спрайта <b>Рыжий кот</b> в <b>Синий кот</b> . . . . .	83
Чит-режим: умение проходить сквозь стены. . . . .	85
Добавление кода для прохождения сквозь стены для рыжего кота . . . . .	85
Добавление кода для прохождения сквозь стены для синего кота . . . . .	86
Заключение. . . . .	87
Обзорные вопросы . . . . .	88

## ГЛАВА 4. БАСКЕТБОЛ С УЧЕТОМ СИЛЫ ТЯЖЕСТИ

89

Эскиз проекта. . . . .	90
<b>А. Обучение кота подпрыгиванию и приземлению.</b> . . . .	91
1. Добавление кода силы тяжести к спрайту кота . . . . .	91
2. Добавление кода уровня земли . . . . .	96
3. Добавление кода прыжков к спрайту <b>Кот</b> . . . . .	97
<b>Б. Обучаем кота перемещению влево и вправо</b> . . . . .	98
4. Добавление кода ходьбы к спрайту <b>Кот</b> . . . . .	99
<b>В. Создаем летающее баскетбольное кольцо</b> . . . . .	100
5. Создание спрайта кольца . . . . .	100
6. Создание хитбокса . . . . .	102
<b>Г. Обучаем кота бросать мяч в кольцо.</b> . . . .	104

7. Создание спрайта баскетбольного мяча . . . . .	105
8. Добавление кода для спрайта <b>Баскетбол</b> . . . . .	106
9. Учет успешных бросков . . . . .	107
10. Исправление ошибки в счете . . . . .	109
Законченная программа . . . . .	112
Версия 2.0: режим для двух игроков . . . . .	113
Дублируем спрайты <b>Кот</b> и <b>Баскетбол</b> . . . . .	113
Меняем код спрайта <b>Кот2</b> . . . . .	114
Меняем код спрайта <b>Баскетбол2</b> . . . . .	115
Чит-режим: остановка кольца . . . . .	116
Заключение . . . . .	117
Обзорные вопросы . . . . .	118

## ГЛАВА 5. АРКАНОИД

119

Эскиз проекта . . . . .	120
<b>А. Создание платформы-ракетки, перемещаемой влево/вправо</b> . . . . .	121
1. Создание спрайта платформы . . . . .	121
<b>Б. Настройка отскакивания мяча от стен</b> . . . . .	124
2. Создание спрайта мячика . . . . .	124
<b>В. Настройка отскакивания мячика от платформы</b> . . . . .	126
3. Добавление кода отскакивания к спрайту теннисного мяча . . . . .	126
<b>Г. Клонирование кирпичиков</b> . . . . .	129
4. Создание спрайта кирпичика . . . . .	129
5. Клонирование спрайта <b>Кирпичик</b> . . . . .	129
<b>Д. Настройка отскакивания мяча от кирпичиков</b> . . . . .	131
6. Добавление кода отскакивания к спрайту <b>Кирпичик</b> . . . . .	131
<b>Е. Создание сообщений о выигрыше и об окончании игры</b> . . . . .	132
7. Изменение кода спрайта <b>Мячик</b> . . . . .	133
8. Создание спрайта <b>Игра окончена</b> . . . . .	133
9. Создание спрайта <b>Вы выиграли</b> . . . . .	135
Законченная программа . . . . .	136
Версия 2.0: Придаем игре лоск . . . . .	137
Создание классного фона . . . . .	138
Добавление музыки . . . . .	139
Настройки градиента . . . . .	139
Изменение цвета платформы при попадании мяча . . . . .	140
Анимированное появление и исчезновение кирпичиков . . . . .	140
Звуковое сопровождение исчезновения кирпичиков . . . . .	143
Звуковое сопровождение мячика . . . . .	145
Добавление хвоста к мячику . . . . .	145
Анимация появления спрайта <b>Игра окончена</b> . . . . .	147
Анимация появления спрайта <b>Вы выиграли</b> . . . . .	148
Заключение . . . . .	150
Обзорные вопросы . . . . .	151

## ГЛАВА 6. ЗМЕ-Е-ЕЙКА!

153

Эскиз проекта. . . . .	155
<b>А. Создание головы змеи, поворачивающейся во все стороны. . . . .</b>	<b>155</b>
1. Создание спрайта головы . . . . .	156
<b>Б. Создание появляющихся яблок . . . . .</b>	<b>159</b>
2. Добавление спрайта <b>Яблоко</b> . . . . .	159
<b>В. Создание тела змеи . . . . .</b>	<b>160</b>
3. Создание спрайта <b>Тело</b> . . . . .	160
4. Создание второго костюма для спрайта <b>Тело</b> . . . . .	161
5. Добавление кода для спрайта <b>Тело</b> . . . . .	162
6. Определим, врежется ли змейка в саму себя или в стену . . . . .	163
Законченная программа. . . . .	166
Версия 2.0: добавление бонусных фруктов. . . . .	167
Чит-режим: Непобедимость . . . . .	168
Изменение головы. . . . .	168
Изменение кода спрайта <b>Тело</b> . . . . .	170
Чит-режим: отрезание хвоста . . . . .	171
Заключение. . . . .	171
Обзорные вопросы . . . . .	172

## ГЛАВА 7. ФРУКТОВЫЙ НИНДЗА

173

Эскиз проекта. . . . .	175
<b>А. Создание начальной экранной заставки . . . . .</b>	<b>175</b>
1. Рисование фона . . . . .	176
2. Добавление кода сцены . . . . .	178
<b>Б. Создание следа от разрезания фруктов . . . . .</b>	<b>179</b>
3. Создание спрайта <b>Ломтик</b> . . . . .	179
4. Создание списков и переменных для спрайта <b>Ломтик</b> . . . . .	183
5. Запись перемещений мыши . . . . .	184
6. Создание пользовательского блока для рисования разреза . . . . .	185
<b>В. Создание кнопки «Начать» . . . . .</b>	<b>189</b>
7. Создание спрайта для кнопки «Начать» . . . . .	189
<b>Г. Создание движущихся фруктов и бомб . . . . .</b>	<b>192</b>
8. Создание спрайта фруктов. . . . .	192
9. Создание костюмов разрезанных фруктов . . . . .	194
10. Добавление кода в спрайт <b>Фрукт</b> . . . . .	197
11. Добавление кода для клонов спрайта <b>Фрукт</b> . . . . .	200
<b>Д. Создание спрайта здоровья . . . . .</b>	<b>203</b>
12. Создание спрайта здоровья. . . . .	203
<b>Е. Подготовка концовки игры. . . . .</b>	<b>206</b>
13. Создание спрайта <b>Затухание</b> . . . . .	206
Версия 2.0: игровой счет. . . . .	209
Чит-режим: восстановление здоровья . . . . .	211
Обзорные вопросы . . . . .	213

## ГЛАВА 8. УНИЧТОЖИТЕЛЬ АСТЕРОИДОВ... В КОСМОСЕ! 215

Эскиз проекта. . . . .	216
<b>А. Создание движущегося космолета</b> . . . . .	218
1. Создание спрайта <b>Космолет</b> . . . . .	218
<b>Б. Выход космолета за края сцены.</b> . . . .	220
2. Добавление необходимого кода в спрайт <b>Космолет</b> . . . . .	220
3. Добавление кода случайных движений в спрайт <b>Космолет</b> . . . . .	222
<b>В. Прицеливание с помощью мыши и стрельба</b> клавишей «Пробел» . . . . .	223
4. Создание мощного бластера . . . . .	223
<b>Г. Создание астероидов.</b> . . . .	226
5. Создание спрайта <b>Астероид</b> . . . . .	226
<b>Д. Создание астероидов, раскалывающихся надвое</b> при попадании . . . . .	229
6. Добавление кода раскалывания астероида . . . . .	229
7. Добавление сообщения «попадание» в спрайт <b>Шар бластера</b> . . . . .	231
<b>Е. Ведение счета и создание таймера</b> . . . . .	232
8. Создание спрайта <b>Время вышло.</b> . . . .	232
<b>Ж. Взрыв космолета при столкновении с астероидом.</b> . . . .	234
9. Загрузка спрайта <b>Взрыв</b> . . . . .	235
10. Добавление кода спрайта <b>Взрыв</b> . . . . .	235
11. Добавление кода взрыва в спрайт <b>Космолет</b> . . . . .	236
Версия 2.0: Ограничение боезапаса . . . . .	237
Чит-режим: звездная бомба . . . . .	239
Заключение. . . . .	241
Обзорные вопросы . . . . .	242

## ГЛАВА 9. ПРОДВИНУТЫЙ ПЛАТФОРМЕР 243

Эскиз проекта. . . . .	244
<b>А. Имитация гравитации, падения и приземления</b> . . . . .	246
1. Создание спрайта <b>Земля</b> . . . . .	246
2. Добавление кода гравитации и приземления . . . . .	247
3. Обучение кота ходьбе и способности пересекать края сцены. . . . .	249
4. Удаление задержки подъема из земли . . . . .	250
<b>Б. Использование крутых склонов и стен</b> . . . . .	252
5. Добавление кода для крутого склона . . . . .	253
<b>В. Обучение кота высоким и низким прыжкам</b> . . . . .	256
6. Добавление кода прыжка . . . . .	257
<b>Г. Добавление обнаружения препятствий сверху</b> . . . . .	259
7. Добавление низкой платформы к спрайту <b>Земля</b> . . . . .	259
8. Добавление кода обнаружения препятствия сверху . . . . .	260
<b>Д. Использование хитбокса для спрайта Кот</b> . . . . .	263
9. Добавление костюма <b>Хитбокс</b> к спрайту <b>Кот</b> . . . . .	264
10. Добавление кода хитбокса . . . . .	265
<b>Е. Улучшение анимации ходьбы</b> . . . . .	266
11. Добавление новых костюмов к спрайту <b>Кот</b> . . . . .	267

12. Создание набора правильных блоков костюмов . . . . .	268
<b>Ж. Создание уровня . . . . .</b>	<b>274</b>
13. Загрузка и добавление фона для сцены. . . . .	274
14. Создание хитбокса для спрайта <b>Земля</b> . . . . .	274
15. Добавление кода спрайта <b>Земля</b> . . . . .	276
16. Добавление дополнительного кода в спрайт <b>Кот</b> . . . . .	277
<b>З. Добавление крабов и яблок. . . . .</b>	<b>278</b>
17. Добавление спрайта <b>Яблоко</b> и кода для него . . . . .	278
18. Создание спрайта <b>Краб</b> . . . . .	280
19. Разработка искусственного интеллекта врага . . . . .	281
20. Добавление спрайта <b>Время вышло</b> . . . . .	285
Заключение. . . . .	286
Обзорные вопросы . . . . .	288

## ДОПОЛНИТЕЛЬНЫЕ РЕСУРСЫ 289

## ОТВЕТЫ НА ВОПРОСЫ 291

Глава 2 . . . . .	291
Глава 3 . . . . .	292
Глава 4 . . . . .	292
Глава 5 . . . . .	293
Глава 6 . . . . .	293
Глава 7 . . . . .	294
Глава 8 . . . . .	294
Глава 9 . . . . .	295

## ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ 296

*Посвящается Сеймуру Паперту,  
До свидания и спасибо за черепашек!*

## ОБ АВТОРЕ

Эл Свейгарт — разработчик программного обеспечения, автор технических книг и человек, у которого всегда при себе полотенце<sup>1</sup>. Он написал несколько книг по программированию для начинающих, в том числе по языку Python. Его книги на языке оригинала можно бесплатно прочитать на сайте [www.inventwithpython.com](http://www.inventwithpython.com).

---

<sup>1</sup> В оригинале: «...who really knows where his towel is» — отсылка к роману «Автостопом по галактике» (1979 г.) Дугласа Адамса, британского писателя, оказавшего влияние на культовое для программистов на Python комедийное шоу «Летающий цирк Монти Пайтона» (Monty Python's Flying Circus) — *Примеч. ред.*





# ВВЕДЕНИЕ

**К**онечно, весело играть в видео-игры, но их разработка — это сложный творческий навык, овладев которым, можно научиться создавать собственные развлечения. Свободная среда программирования Scratch позволяет каждому желающему с легкостью развить в себе навыки программирования.

В первую очередь Scratch предназначен для детей 8–16 лет, но пользуются им люди всех возрастов, включая детей младшего возраста, которым помогают родители, и студентов, изучающих свой первый язык программирования.

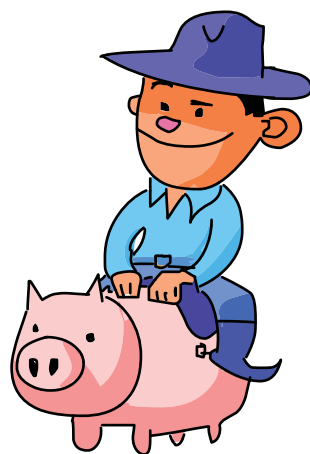
Со Scratch можно столько всего сделать, что бывает трудно понять, с чего начать. В этом вам поможет наша книга. Из нее вы узнаете, как с помощью Scratch создавать собственные видеоигры. Делая описанные здесь проекты, вы узнаете, какие блоки Scratch обычно используются для разработки игр. Работая с ними, вы создадите надежный фундамент из знаний, которые вам понадобятся в будущем для разработки собственных программ.

## ДЛЯ КОГО ПРЕДНАЗНАЧЕНА ЭТА КНИГА

Эта книга подойдет даже тем, у кого нет никакого опыта в программировании. Необходимы только базовые математические навыки — уметь совершать основные арифметические действия: сложение, вычитание, умножение и деление. Даже если вы не уверены в своем знании математики, не позволяйте сомнениям стать преградой на пути к программированию. И не забывайте, что компьютер будет выполнять все вычисления за вас!

Любую программу из этой книги легко написать, следуя пошаговой инструкции. Вы узнаете о блоках кода и концепциях программирования, а также собственноручно создадите при помощи всего этого несколько увлекательных игр. Вы можете начать знакомство с этой книгой прямо сейчас, и не важно, что вы умеете!

Дети могут заниматься с этой книгой без посторонней помощи. Проекты, описанные в ней, идеально подходят для занятий в выходных или в компьютерном клубе



после школы. Подходит она и для родителей или преподавателей, которые хотят познакомить своих детей или учеников с миром программирования. При этом, чтобы помочь ребенку с обучением, взрослому совершенно не обязательно самому быть программистом.

Если вам нужно полное руководство по всем функциям Scratch, вы можете посмотреть видеоуроки в Интернете на сайтах [scratch.mit.edu/help/videos/](https://scratch.mit.edu/help/videos/) и [inventwithscratch.com](https://inventwithscratch.com).

И помните: программирование – это практический навык, как карате или умение играть на гитаре, и вы конечно же не сможете полностью им овладеть, просто прочитав одну книгу. Однако создавая игры по мере ее прочтения, вы совершенно точно лучше поймете процесс программирования.

## ОБ ЭТОЙ КНИГЕ

Каждая глава посвящена разработке отдельной игры, и кроме того, по ходу книги разъясняются основы программирования. Подмечая, что происходит в конце каждой игры, читатели начнут понимать, из чего состоят программы. Затем вы узнаете, как написать каждую из частей, чтобы в результате собрать полноценную игру. После создания игры вы сможете добавить к ней специальные функции и чит-коды. Вопросы в конце каждой главы помогут проверить, насколько хорошо вы ее поняли.

- ▶ **Глава 1. Начало работы со Scratch.** В этой главе вы узнаете, как перейти на сайт Scratch и познакомитесь с редактором Scratch.
- ▶ **Глава 2. Радужные линии в космосе!** В этой главе вы создадите анимированный арт-проект с использованием базовых блоков кода и нескольких спрайтов, работающих вместе. Вы также узнаете о направлениях и порядках.

- ▶ **Глава 3. Бегущий в лабиринте.** В этой главе вы создадите игру, в которой игрок использует клавиатуру, чтобы провести кошку по лабиринту, состоящему из восьми различных уровней.
- ▶ **Глава 4. Баскетбол с учетом силы тяжести.** Здесь вы создадите баскетбольную игру с реалистичной гравитацией для прыгающих кошек и падающих баскетбольных мячей.
- ▶ **Глава 5. Арканойд.** Глава описывает методы создания игры, в которой простой кирпичик превращается в гладкий после попадания по нему. Игру можно сделать красивее при помощи анимации, звуковых эффектов и многого другого.
- ▶ **Глава 6: Зме-е-ейка!** Классическая компьютерная игра, в которой игрок управляет постоянно растущей змейкой, движущейся по экрану. В главе объясняется, как использовать функцию клонирования спрайтов в Scratch, чтобы тело змеи увеличивалось.
- ▶ **Глава 7. Фруктовый ниндзя.** В этой главе вы разработаете клон популярной игры для мобильных устройств Fruit Ninja, в которой игрок режет на кусочки пролетающие фрукты.
- ▶ **Глава 8. Уничтожитель астероидов... в космосе!** Это клон классического космического шутера Asteroids. Для управления космолетом вы будете использовать мышь и клавиатуру.
- ▶ **Глава 9: Продвинутый платформер.** В этой главе собрана информация, которая рассматривалась в предыдущих главах. Здесь объясняется, как создать игру-платформер с анимацией ходьбы и прыжков, платформами и врагами с искусственным интеллектом.



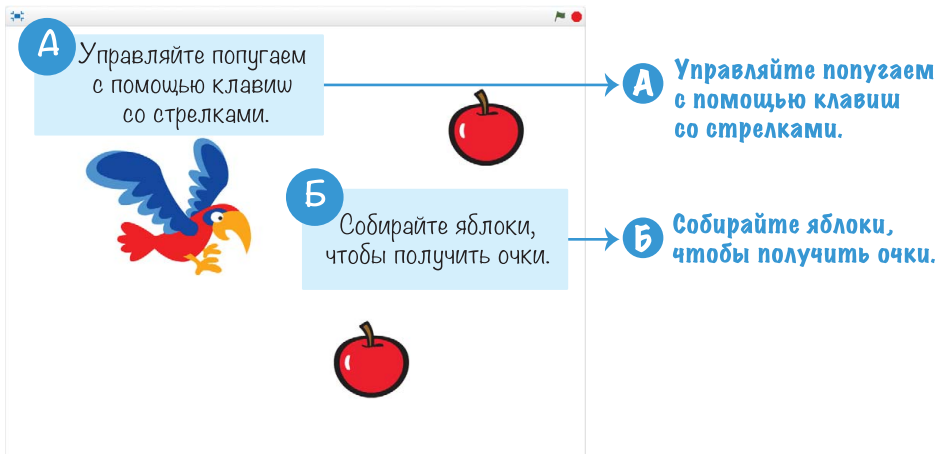
## КОНТРОЛЬНАЯ ТОЧКА

В этой книге вы увидите вот такие текстовые блоки с надписью «Контрольная точка». Поскольку вы будете создавать программы пошагово, часто будет возникать необходимость приостановить работу и запустить программу, даже если она еще не закончена. Так вы сможете увидеть, правильно ли она работает, и отловить возможные ошибки на этом этапе. Контрольные точки будут напоминать вам о том, что пора сохранить программу, выбрав команду меню **Файл ► Сохранить сейчас**.

## КАК ПОЛЬЗОВАТЬСЯ ЭТОЙ КНИГОЙ

Все проекты книги начинаются с эскиза игры, которую мы будем создавать. Подписи на эскизе описывают функции, которые мы добавим в игру с помощью кода.

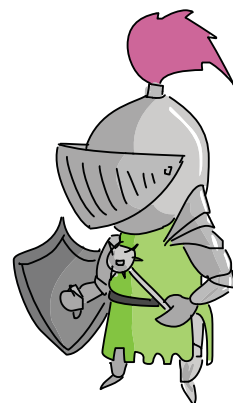
Чтобы управлять процессом создания игры, мы будем заниматься каждой ее частью по очереди. Заголовки с буквами АБВ в книге соответствуют этим функциям в эскизе.



Дробление большой задачи на несколько мелких поможет вам думать организованно и сделает большую задачу не такой пугающей, какой она может показаться на первый взгляд. После того как будет создана и запущена простая версия игры, мы добавим новые функции, чит-коды и т.д. Для создания собственной игры я рекомендую начать с простого эскиза.

## ВЕБ-РЕСУРСЫ

Хотя среда Scratch и включает множество собственных ресурсов, вам потребуется и несколько дополнительных файлов, чтобы реализовать проекты, описанные в этой книге. Эти файлы находятся в архиве, который можно загрузить по ссылке [eksmo.ru/files/Scratch\\_Sweigart.zip](http://eksmo.ru/files/Scratch_Sweigart.zip). Чтобы получить доступ к этим файлам, скачанный архив нужно распаковать на ваш жесткий диск.



Архив содержит файлы изображений, используемых в проектах книги, и файлы скелета проекта для каждой из программ. В этих файлах все начальные шаги уже сделаны, от вас требуется только добавить блоки кода. Поэтому если у вас возникли проблемы с тем, чтобы доделать программу, попробуйте начать с файла скелета проекта, а не создавать новый пустой проект.

Использование этих файлов будет также удобным для учителей, обучающих несколько учеников, когда время на работу ограничено. Ученикам нужно будет только добавить блоки кода, чтобы завершить программу.



1

## НАЧАЛО РАБОТЫ СО SCRATCH

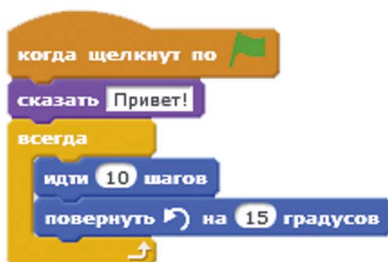
**Н**а сегодняшний день Scratch является одной из лучших образовательных программ. Ни один другой инструмент не делает программирование таким легким. Были созданы многие аналоги, но Scratch все равно остается самым популярным. С помощью Scratch вы можете создавать интерактивные игры, анимацию и научные проекты, и все это просто и весело!

Scratch представляет собой свободную среду программирования, которая открывается в вашем браузере. Он был разработан небольшой группой программистов Массачусетского технологического института. Пользователи Scratch, которых называют *скретчеры*, создают программы, совмещая в редакторе Scratch блоки кода. Хотя Scratch разрабатывался для детей от 8 до 16 лет, он давно перешагнул свои границы и им пользуются люди всех возрастов, в том числе и дети более младшего возраста при помощи родителей. Используя эту программу, любой может с легкостью развивать свои навыки программирования и решения всевозможных задач.



Поскольку Scratch работает в браузере, программное обеспечение не требует установки. Scratch никоим образом не может повредить файлы на вашем компьютере. Scratch – это бесплатная программа, в нем нет рекламы или предложений покупки, так что дети могут пользоваться сайтом Scratch совершенно свободно, а взрослым не нужно беспокоиться о непредвиденных расходах.

В Scratch можно использовать мышь, чтобы перетаскивать блоки кода, поэтому набирать текст почти не нужно. Ниже показан пример совмещенных блоков кода:



Визуальный редактор Scratch быстро откликается, так что не придется часами вводить загадочные команды, прежде чем вы сможете увидеть, что происходит с вашей программой. Программирование на Scratch – быстрый и интересный процесс. И, в отличие от других языков про-



граммирования, Scratch не выдает никаких сообщений об ошибках, которые могут запутать начинающего программиста. Если вам нужно изучить основы программирования (или помочь кому-то в изучении этого вопроса), ничего удобнее Scratch не найти.

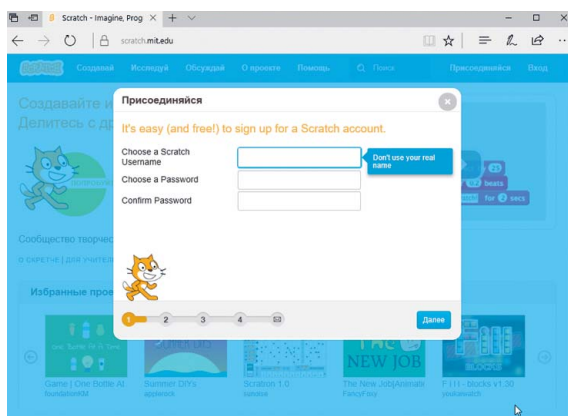
## ЗАПУСК SCRATCH

Чтобы начать знакомство со Scratch, откройте браузер и перейдите по адресу **scratch.mit.edu**. Не имеет значения, какой операционной системой вы пользуетесь – Windows, macOS или Linux, вы сможете запустить Scratch на своем ноутбуке или компьютере. Но обратите внимание, что Scratch не работает на планшетах или смартфонах.

**ПРИМЕЧАНИЕ.** Версия Scratch 2.0, описанная в этой книге, не работает на компьютере Raspberry Pi.

Регистрация для создания учетной записи бесплатна. Вы можете создавать программы в Scratch и без учетной записи, но наличие аккаунта позволяет сохранять ваши программы в Интернете. После этого вы можете продолжить работу над своим проектом с любого компьютера, подключенного к Сети.

Щелкните мышью по ссылке **Присоединяйся** в правом верхнем углу страницы, чтобы создать учетную запись. Откроется новое окно.



Выберите и укажите имя пользователя и пароль, а также введите информацию об учетной записи. Сайт Scratch никому не передаст ваш адрес электронной почты или личную информацию без вашего разрешения. Информация о его политике и полной конфиденциальности находится по ссылке **[scratch.mit.edu/privacy\\_policy/](https://scratch.mit.edu/privacy_policy/)**.

После того как вы вошли на сайт Scratch, щелкните мышью по ссылке **Создавай** в верхней части страницы, чтобы запустить редактор Scratch.

## АВТОНОМНЫЙ РЕДАКТОР

Автономный редактор позволяет программировать без подключения к Интернету. Если у вас нет доступа к Сети или если соединение нестабильно, вы можете установить автономный редактор на свой компьютер вместо того, чтобы использовать сайт Scratch. Единственное отличие состоит в том, что программы будут сохранены на вашем компьютере, а не на сайте Scratch. Позже вы сможете загрузить свои программы в Сеть или скопировать их на переносной диск, если вам нужно будет переместить их на другой компьютер.

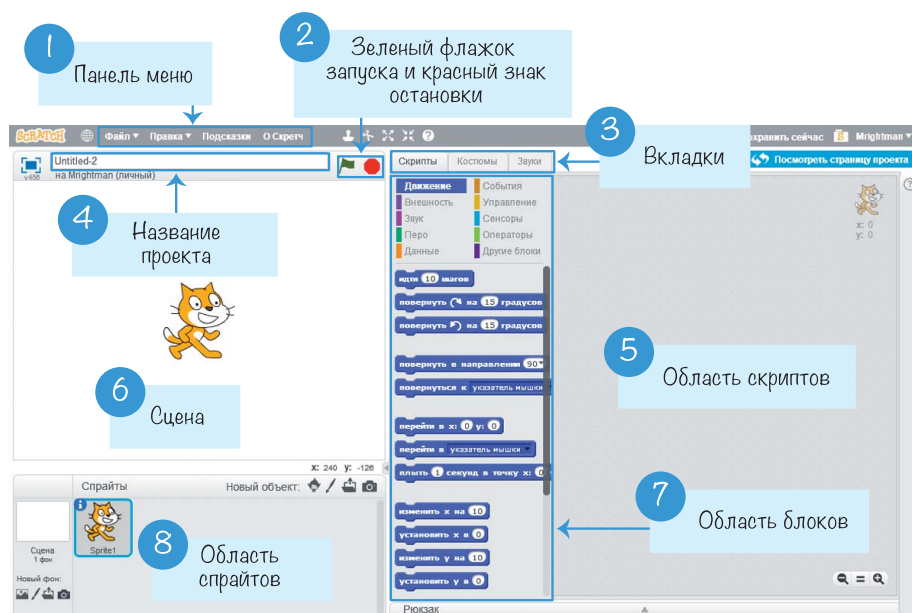
Автономный редактор Scratch доступен по адресу **[scratch.mit.edu/scratch2download/](https://scratch.mit.edu/scratch2download/)**.

**ПРИМЕЧАНИЕ.** *Вы можете найти более раннюю версию редактора, Scratch 1.4. Не используйте эту версию; она устарела и не имеет новых функций, которые есть в версии Scratch 2.0. Если вы используете Scratch в вашем браузере, вы используете Scratch 2.0. Если вы загрузили автономный редактор Scratch, убедитесь, что загрузили именно версию Scratch 2.0.*

## РЕДАКТОР SCRATCH И СПРАЙТЫ

В редакторе Scratch вы совмещаете блоки кода, чтобы создать игру, анимацию или иллюстрацию. Редактор от-

крывается при нажатии ссылки **Создавай** в верхней части страницы, как показано на рисунке ниже, и вы можете начать создавать программы в Scratch.



Основным объектом в Scratch является спрайт. Спрайты отображаются на сцене **6**, а их поведением управляют соответствующие блоки кода. Для всех новых проектов редактор автоматически запускается со спрайтом **Кот**, но вы можете добавить собственные спрайты. Программирование спрайта осуществляется путем добавления блоков кода в область скриптов **5** в правой части экрана. Несколько объединенных блоков кода в Scratch называются *скриптом*.

Текстовое поле в верхней части редактора содержит название проекта **4**. Проекту стоит дать имя, описывающее его суть. Помните, что время от времени нужно сохранять проект. Для этого выберите команду **Файл ► Сохранить Сейчас** в строке меню **1**. Так вы не потеряете вашу работу, если ваш браузер аварийно закроется.

В центре находится область блоков **7**, из которой вы будете брать блоки кода. В верхней части области блоков вы увидите категории блоков кода: **Движение**, **Внешность**,

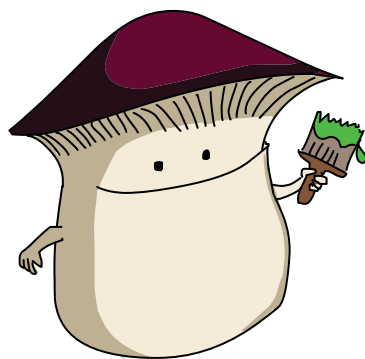
**Звук, Перо, Данные, Управление, События, Сенсоры, Операторы и Другие блоки.** Каждый блок кода относится к одной из групп и оформлен цветом этой группы. Например, блок **Сказать** относится к фиолетовой категории **Внешность**. Запас доступных блоков кода безграничен; просто выбирайте их в области блоков и перетаскивайте в область скриптов.

Каждый объект-спрайт имеет свои собственные скрипты. При нажатии кнопки спрайта в области спрайтов **8** в области скриптов будут отображаться скрипты выбранного спрайта. Выберите вкладку **Скрипты** **3** для отображения области скриптов. При выборе вкладки **Костюмы** вместо области скриптов появляется графический редактор, а при выборе вкладки **Звуки** – звуковой редактор.

Нажатие кнопки в виде зеленого флага запускает вашу программу, а если вы нажмете кнопку в виде красного значка остановки, программа завершит работу **2**.

## ГРАФИЧЕСКИЙ РЕДАКТОР

Есть несколько способов добавить спрайты в свою программу. Вы можете использовать спрайты, которые есть в Scratch, а также загрузить с вашего компьютера или нарисовать свои собственные. Если вы хотите сделать собственный спрайт, вы можете использовать графический редактор, который встроен в Scratch.



Он схож с другими графическими редакторами, такими как Microsoft Paint или Paintbrush. Чтобы нарисовать новый спрайт, нажмите кнопку **Нарисовать новый спрайт**, которая находится рядом с надписью **Новый объект**. Вы можете изменить внешний вид спрайтов, переключаясь между множеством разных костюмов. Чтобы создать новый

костюм для спрайта, выберите вкладку **Костюмы**, а затем нажмите кнопку **Нарисовать новый костюм**, которая находится под надписью **Новый костюм**.

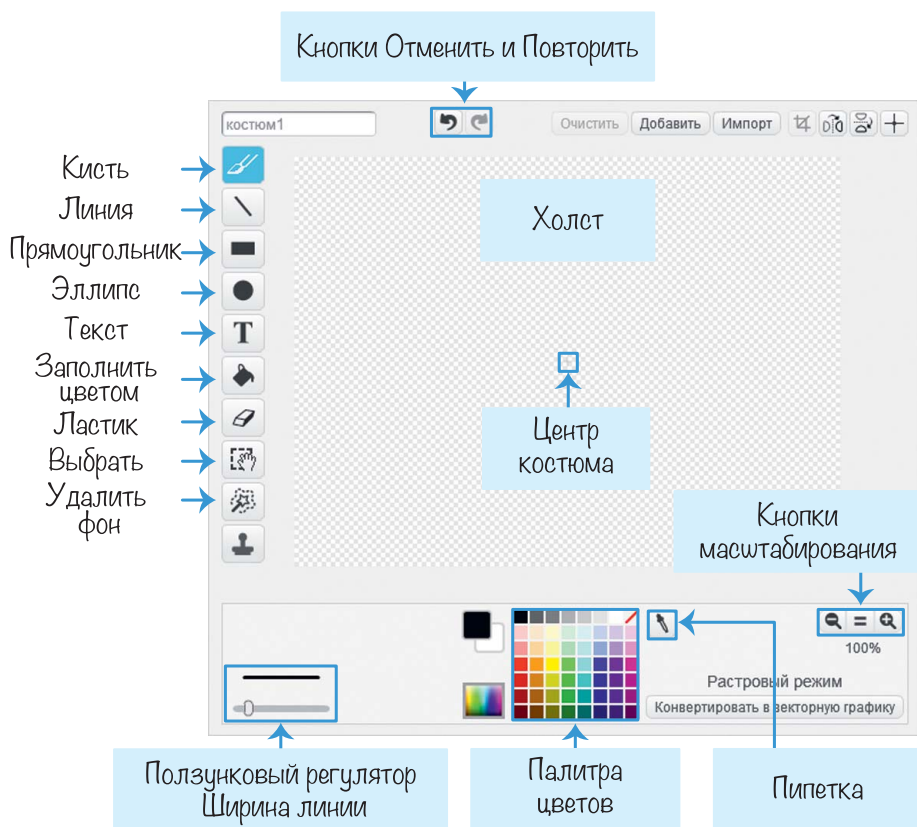
Ниже перечислены основные части графического редактора, встроенного в Scratch:

- ▶ инструменты для рисования, которые можно выбрать, используя кнопки, находящиеся в правой части окна;
- ▶ холст, на котором вы рисуете изображения;
- ▶ центр костюма, отображающийся с помощью кнопки в виде крестика;
- ▶ ползунковый регулятор ширины линии, который устанавливает толщину инструментов рисования;
- ▶ палитра цветов, при помощи которой можно изменить цвет рисования;
- ▶ кнопки масштабирования изображения для увеличения или уменьшения холста;
- ▶ кнопки **Отменить** и **Повторить**<sup>2</sup>, которые помогут исправить ошибки.

---

<sup>2</sup> В программе по ошибке обе кнопки переведены как Отменить. Кнопка Отменить находится слева, а кнопка Повторить – справа — *Примеч. пер.*

Графический редактор выглядит следующим образом:



Экспериментируйте с графическим редактором, нажимая кнопки инструментов рисования и перемещая мышью по холсту, чтобы увидеть, как они работают. Изменяйте цвет и ширину инструментов рисования с помощью палитры цветов и слайдера ширины линии. С помощью инструмента **Пипетка** можно выбрать цвет прямо на холсте, а не из палитры цветов. Если вы допустили ошибку, нажмите кнопку **Отменить**, которая находится в верхней части окна.

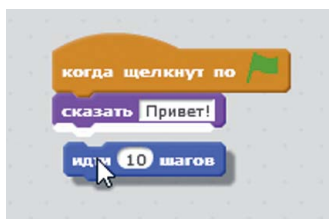
Перечень костюмов спрайтов расположен слева от инструментов рисования. Если вы хотите сохранить костюм в качестве графического файла, щелкните правой кнопкой мыши по костюму и выберите пункт **Сохранить локальный файл** в контекстном меню.

# РАБОТА С БЛОКАМИ КОДА

Перед тем как начать программировать, стоит получить представление о том, как блоки кода состыкуются между собой в редакторе. В этой книге вы узнаете, что делает каждый блок кода.

## Добавление блоков

Чтобы создать новый блок кода, перетащите его из области блоков, расположенной по центру, в область скриптов. Блоки кода, имеющие выемку на верхней стороне и выступ снизу, называются блоками стека. Чтобы соединить один блок стека с другим, перетащите первый блок ближе к нижней части второго. Когда появится белый контур, отпустите блок, чтобы объединить их в стек.



Блоки стека могут также помещаться между блоками. Посмотрите внимательно на место, где в скрипте появляется белый контур: здесь будет помещен блок стека. Рисунки ниже показывают перемещение блока **Ждать 1 секунду** в середину скрипта:



Вы можете изменить значение белого поля внутри блока, щелкнув мышью по этому полю и введя новое значение. В прямоугольных белых полях пишется текст; скругленные белые поля предназначены для числовых значений.

Скругленные блоки называются *блоками ссылки*. Они вставляются внутрь белых полей. Например, на следующих рисунках зеленый блок ссылки **Выдать случайное от 1 до 10** помещился внутри белого поля. Когда *левый край* блока ссылки оказывается на белом поле, вокруг этого поля появляется белый контур. Если левый край блока ссылки находится не над белым полем, белый контур не будет появляться, и блок ссылки не может быть помещен внутрь.



## Удаление блоков

Чтобы удалить блоки, перетащите их за пределы области скрипта. Если вы удаляете блок стека, вы одновременно удаляете стек блоков, расположенных под ним, как показано на следующем рисунке. Возможно, потребуется расположить эти блоки отдельно, если вы хотите вернуть некоторые из них в скрипт. Чтобы удалить блоки со сцены, перетащите их в центральную область блоков. Вы всегда можете добавить дополнительные блоки из области блоков, когда это потребуется.



Существует и другой способ удаления блоков: щелкните правой кнопкой мыши по блоку и выберите команду **Удалить** в контекстном меню. Однако при этом также будут удалены все блоки, находящиеся под ним. Если вы случайно удалили нужные блоки, их можно восстановить, выбрав команду **Правка ► Восстановить в строке меню**.



## Запуск программ

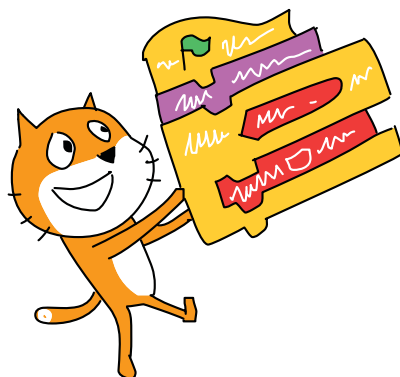
Создайте программу, показанную ниже, перетаскивая блоки из области блоков в область скриптов.



При нажатии кнопки в виде зеленого флажка в верхней части сцены программа запускается. Она начинает выполняться с верхнего блока (**после щелчка по зеленому флажку**), а затем запускается следующий блок кода в скрипте. В этом примере над спрайтом появляется всплывающее окно и отображается слово «Привет!». В цикле **Всегда** спрайт движется вперед на 10 шагов, а затем поворачивается против часовой стрелки на 15 градусов. Когда программа дойдет до выполнения последнего блока, *цикл* возвращает назад, к верхнему блоку в цикле. Все блоки, находящиеся в блоке **Всегда**, будут работать заиклленно. Выполнение программы прекращается только после нажатия кнопки, которая выглядит как красный знак остановки.

Кроме того, можно запустить скрипт или блок, дважды щелкнув по нему мышью. Но лучше запускать программы все же кнопкой в виде зеленого флажка.

В своей программе вы можете использовать столько спрайтов и блоков кода, сколько хотите. По мере создания программных проектов, описанных в этой книге, вы познакомитесь с различными типами блоков кода Scratch.



## ДЕМОНСТРАЦИЯ ВАШИХ ПРОГРАММ

Когда вы вошли в свою учетную запись Scratch, нажмите кнопку **Поделиться** в правом верхнем углу редактора, чтобы позволить другим пользователям Scratch видеть вашу программу. Они будут иметь возможность запускать вашу игру и оставлять комментарии. Если пользователям понравится игра, они могут поставить ей «лайк» или добавить в Избранное.



После того как вы закончили проект, вы можете добавить его на сайт *Scratch Programming Playground studio*. Здесь хранятся проекты и переделки, которые сделали вы и другие пользователи. После того как вы поделились своим проектом, скопируйте его URL-адрес и перейдите на страницу [inventwithscratch.com/studio/](https://inventwithscratch.com/studio/). Нажмите кнопку **Add projects**, вставьте URL-адрес в текстовое поле и нажмите кнопку **Add by URL**. Теперь другие пользователи смогут запускать вашу игру.

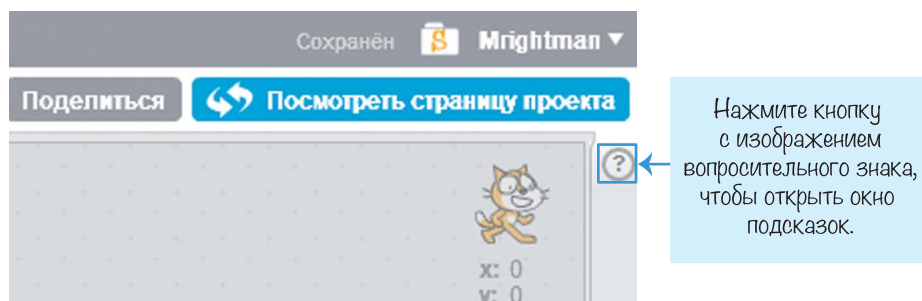
Не переживайте, если считаете, что ваша игра недостаточно хороша – каждый начинает свой путь программирования с простых вещей. Большинство людей на сайте Scratch такие же новички, как вы. Более 11 миллионов человек поделились своими программами на сайте Scratch, так что не расстраивайтесь, если у вашей программы мало просмотров – ее может быть сложно найти с таким количеством доступных программ.

## ПОЛУЧЕНИЕ ПОМОЩИ

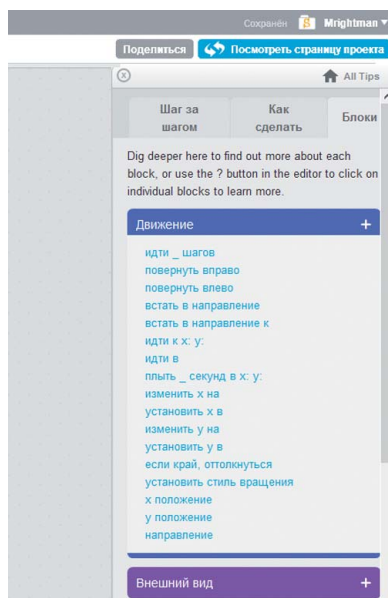
Можно стать отличным программистом, не имея ответов на все вопросы, но зная, как их найти. Следуя пошаговым инструкциям, разработанным для создания проектов в этой книге, вы можете задавать собственные вопросы.

## Окно подсказок

Первым делом при возникновении вопросов перейдите в справочный раздел редактора Scratch. Нажмите кнопку со знаком вопроса на правой стороне, чтобы открыть окно Все подсказки.



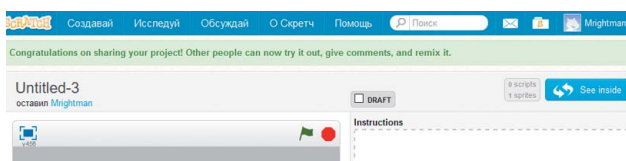
В окне подсказок вы можете узнать, за что отвечает конкретный блок кода, выбрав его на вкладке **Блоки**, как показано на рисунке ниже.



Там же вы можете изучить различные руководства. Помимо этого вы можете обратиться за помощью на специализированный форум, что часто быстрее, чем поиск ответа в окне советов.

## Просмотр проектов других пользователей

Вы можете узнать много новых методов, просматривая код программ других пользователей Scratch. Найдите на сайте Scratch-проект, который вам нравится, а затем нажмите кнопку **See inside**, как показано здесь:



Нажмите кнопку **See inside**, чтобы увидеть код, написанный другим пользователем Scratch

Вы можете копировать, изменять или переделывать код других пользователей Scratch. Все программы на сайте публикуются в соответствии с его правилами в свободном доступе, поэтому вам не нужно спрашивать разрешения у автора. Скретчеры часто переделывают программы друг друга, создавая свои собственные версии.

Вам все еще нужна помощь и вы хотите поговорить с другим пользователем? Щелкните мышью по ссылке **Обсуждай** в верхней части сайта, чтобы посетить форумы для обсуждения.

Scratch

СоздавайИсследуйОбсуждайО СкретчПомощь

Поиск

Мightman

Discuss Scratch

Discussion HomeПоиск

Показать новые сообщения, с момента Вашего последнего визита

Discussion Forums » Русский

« previous1234...13141516next »

Русский

Создать новую тему

Тема	Ответов	View	Last Post
Важно: ВСЕ ВОПРОСЫ СЮДА!! оставил Dmith (New Posts)	2010	80265	Май 17, 2017 15:47:33 оставил Nybik666
Важно: Скретч Вики на русском! [Регистрируйтесь, господа] оставил Dmith (New Posts)	160	9831	Март 12, 2017 14:20:43 оставил gauss06
Важно: Правила поведения на форуме оставил Dmith (New Posts)	4	753	Март 29, 2016 10:02:23 оставил Mr-Angry-Fruit13
Важно: Добавление русского в векторные картинки (Скретч 2.0) оставил Dmith (New Posts)	0	4219	Май 22, 2014 07:21:13 оставил Dmith
Важно: Russian only forum оставил Paddle2See (New Posts)	0	1794	Янв. 27, 2013 15:19:38 оставил Paddle2See
Это топик для тестирования кое-чего. оставил VitSem_Company (New Posts)	47	427	Вчера 03:19:20 оставил VitSem_Company
Игры можно опубликовать здесь! оставил АНОНА (New Posts)	41	1307	Май 17, 2017 14:49:58 оставил PrinsForUnicorns

## ЗАКЛЮЧЕНИЕ

Редактор Scratch – это инструмент для творчества с большими возможностями. На сайте Scratch вы увидите самые разные проекты: от игр, мультфильмов, моделирования до презентаций.

Теперь, когда вы знаете, как получить доступ к сайту Scratch, создать учетную запись, использовать редакторы скриптов и графики, а также совмещать блоки кода в скрипте, вы готовы следовать пошаговым инструкциям, которые приведены в следующих главах этой книги. Если у вас возникают вопросы, помните об использовании окна подсказок в редакторе и форумов на сайте Scratch.

Давайте создадим свою первую программу!



2

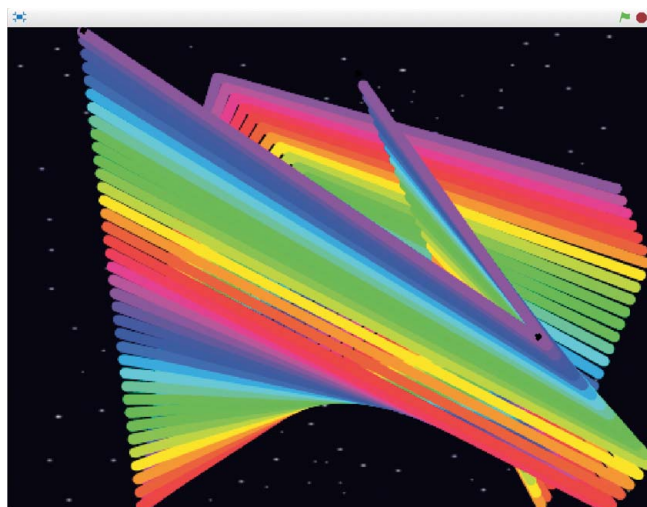
## РАДУЖНЫЕ ЛИНИИ В КОСМОСЕ!

**В** этой главе вы создадите классную анимацию — летящую V-образную радугу, оставляющую за собой яркий след.

Эта программа была вдохновлена *демо-сценой*, течением, появившимся в среде программистов 1980-х годов, начавших создавать при помощи языков программирования удивительные графические программы.

Участники демосцены создавали красивые сложные программы, называемые *демо*, которые демонстрировали их художественные и музыкальные таланты и навыки программирования. Но самое удивительное, что эти программы были крошечные – всего несколько килобайт! Программа, которую мы будем писать, не настолько мала, но она тоже эффектная и красочная. И в ней используется лишь несколько строк кода Scratch.

Перед тем как приступить к программированию, взгляните на рисунок ниже, чтобы увидеть, как будет выглядеть законченная программа. Затем перейдите по ссылке [scratch.mit.edu/studios/4188596/](https://scratch.mit.edu/studios/4188596/), чтобы посмотреть готовую анимацию.



Так же, как участники демосцены, вы можете создавать красивые программы. Давайте создадим наше собственное графическое демо в Scratch!

## ЭСКИЗ ПРОЕКТА

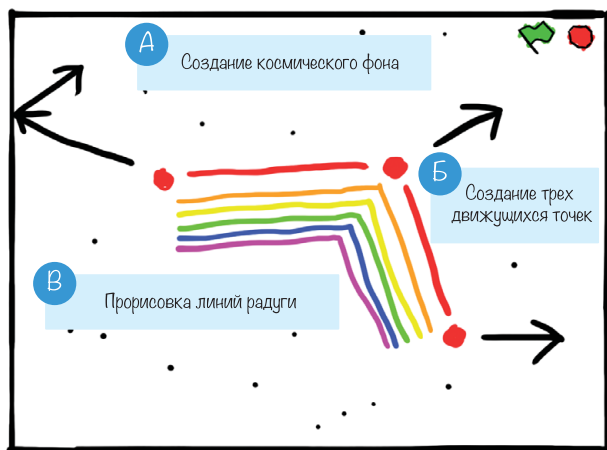
Первым шагом превращения идеи в программу является представление о том, как она должна выглядеть. Проектирование программы поможет выяснить, какие спрайты вам нужны и как они будут себя вести. Я рекомендую ри-

совать свои идеи на бумаге, чтобы вы смогли вычеркнуть их, если они вам не понравятся, а также записывать заметки и напоминания.

Лучше всего, чтобы проект оставался простым. Если вы начинаете с создания сложной игры, такой как *Minecraft* или *Zelda*, вы быстро поймете, что это очень трудоемкая работа. Законченный небольшой проект принесет больше пользы, чем создание незаконченной программы, в которую невозможно играть и которая будет вас только расстраивать.

После того как вы закончите свою простую игру, вы можете сделать ее сложнее – в этом и состоит идея итеративной разработки. Сначала вы создаете простую рабочую программу. Затем вы ее улучшаете. Вы всегда можете добавить элементы к основной программе после того, как закончите ее. Или, если программа становится слишком сложной, вы можете вернуться к своим первоначальным эскизам и решить, что нужно удалить.

Не старайтесь чтобы эскиз выглядел красиво. Намного важнее иметь четкий план для всех основных частей программы. В моем эскизе есть три части: А, Б и В. Мы будем над ними работать, пока полная программа еще не собрана.



После того как вы закончили эскиз своей программы, можно начинать программировать! Перейдите на сайт



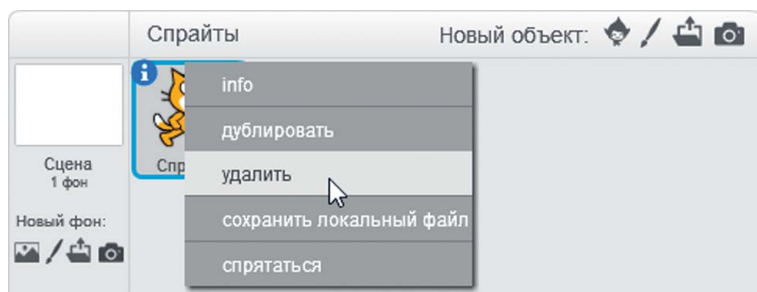
Scratch по ссылке **scratch.mit.edu**, зарегистрируйтесь на нем для получения учетной записи и авторизуйтесь в системе (наличие аккаунта позволяет сохранять созданные вами программы). После того как вы вошли в систему, нажмите кнопку **Создавай** в верхней части экрана, чтобы приступить к разработке своего первого Scratch-проекта. Затем щелкните по текстовому полю в левом верхнем углу, чтобы изменить название проекта с *Untitled* на **Радужные линии**. Давайте начнем с создания части А этой программы.

## **А** СОЗДАНИЕ КОСМИЧЕСКОГО ФОНА

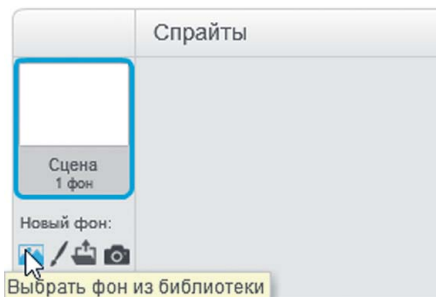
Во-первых, давайте уберем спрайты, которые мы не будем использовать, и создадим фон.

### 1. Очистка и настройка сцены

Каждый раз, когда вы создаете новый Scratch-проект, вы видите спрайт рыжего кота на пустой белой сцене. Для задуманной нами программы спрайт кота не понадобится, поэтому щелкните правой кнопкой мыши по элементу **Спрайт1** в области спрайтов и выберите команду **Удалить** в контекстном меню, чтобы убрать кота со сцены и из области спрайтов.

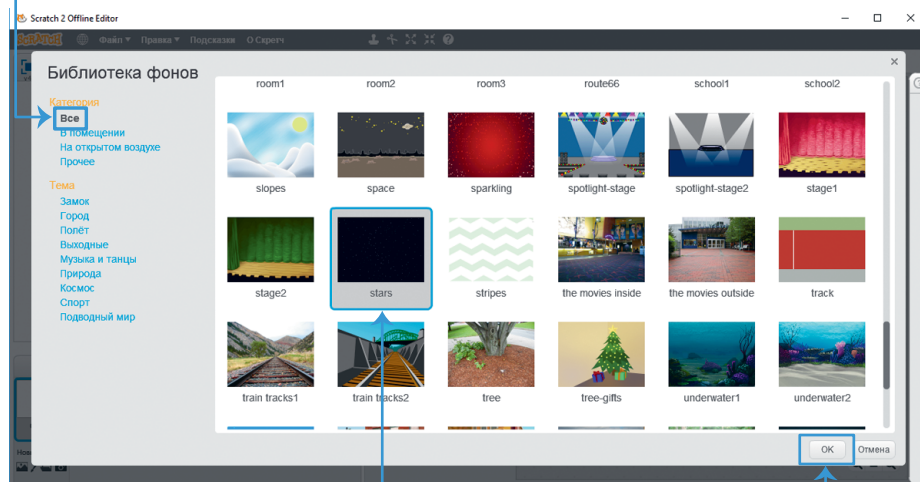


Нажмите кнопку **Выбрать фон из библиотеки** (которая выглядит как изображение ландшафта) под надписью **Новый фон**.



Откроется окно **Библиотека фонов**, в котором будут отображены все фоны, отсортированные в алфавитном порядке. Выберите фон **Stars** и нажмите кнопку **ОК**.

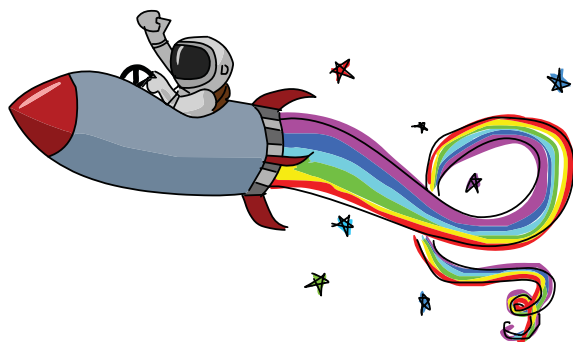
1 Убедитесь, что выбрана категория **Все**, иначе фон **Stars** не будет отображаться.



2 Выберите фон **Stars**. Для удобства вы можете переименовать фон, присвоив ему имя **Звезды**.

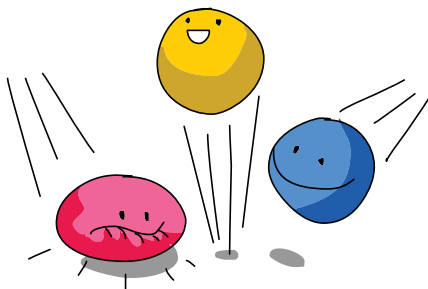
3 Нажмите кнопку **ОК**, чтобы подтвердить выбор.

Теперь сцена выглядит как космическое пространство!



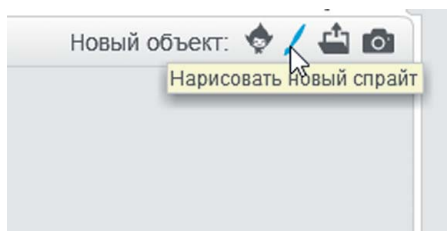
## Б СОЗДАНИЕ ТРЕХ ДВИЖУЩИХСЯ ТОЧЕК

Следующим шагом будет добавление трех новых спрайтов, которые представляют собой три точки летающей радуги в виде буквы V.

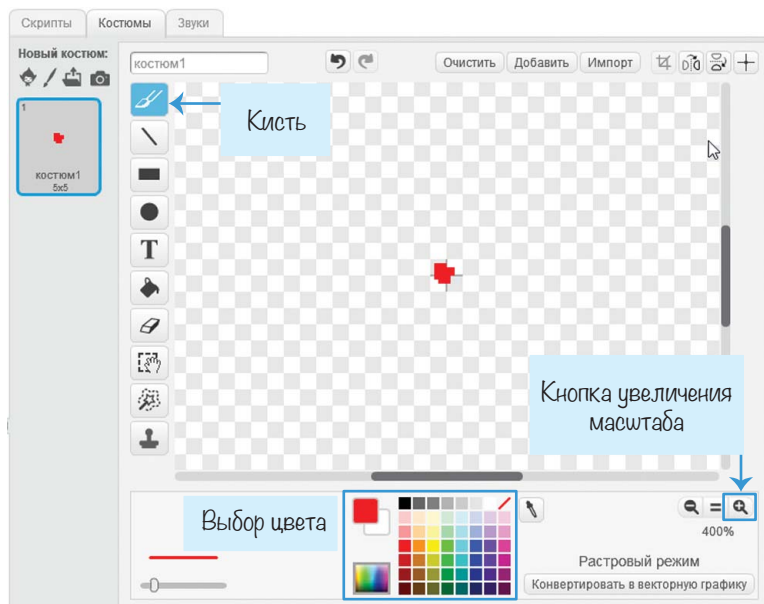


### 2. Рисование точки

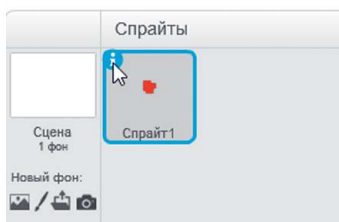
Нажмите кнопку **Нарисовать новый спрайт**, которая выглядит как кисточка и находится рядом с надписью **Новый объект**.




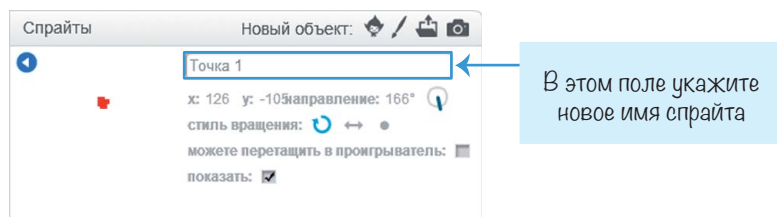
Новый спрайт с именем **Спрайт1** появится в области спрайтов. При нажатии этой кнопки программа переключается на вкладку **Костюмы**, в которой вы увидите графический редактор. Используйте инструмент **Кисть**, чтобы нарисовать маленькую красную точку возле крестика в графическом редакторе. Для большего удобства в графическом редакторе можно увеличить изображение, нажав кнопку масштабирования (которая выглядит как увеличительное стекло).



Нажмите кнопку **i** на значке **Спрайт1**, чтобы открыть панель информации спрайта первой точки. (Вы также можете открыть панель информации, щелкнув правой кнопкой мыши по спрайту и выбрав команду **Info** в контекстном меню.)

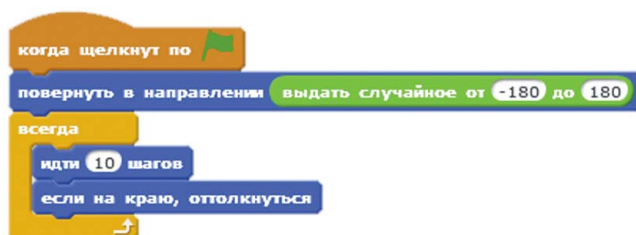


Измените имя спрайта **Спрайт1** на **Точка 1**. Затем нажмите кнопку , чтобы закрыть панель информации и вновь отобразить область спрайтов.



### 3. Добавление кода для спрайта **Точка 1**

Теперь мы можем начать программировать. Перейдите на вкладку **Скрипты**, чтобы отобразить область скриптов. Добавьте в нее код, показанный на следующем рисунке. Эти блоки вы можете найти в категориях **События** (коричневый), **Движение** (темно-синий), **Операторы** (зеленый) и **Управление** (желтый). Если вам не совсем понятно, как перетащить эти блоки, посмотрите анимацию по ссылке [www.nostarch.com/scratchplayground/](http://www.nostarch.com/scratchplayground/).

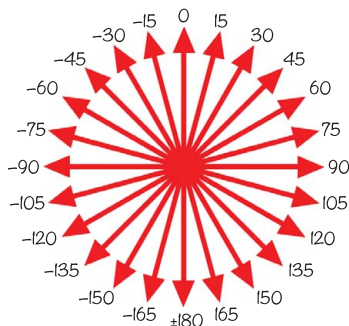


При нажатии кнопки в виде зеленого флага спрайт **Точка 1** появляется в случайной позиции между  $-180$  и  $180$  градусов. Затем цикл **всегда** перемещает спрайт вперед на 10 шагов, и при достижении точкой края сцены она от него отскакивает. Таким образом, спрайт будет все время перемещаться.

Обратите внимание на то, что спрайт **Точка 1** еще не рисует радужные линии. Мы сделаем это позже, когда мы создадим больше спрайтов.

## ЭТО ИНТЕРЕСНО: НАПРАВЛЕНИЕ И ГРАДУСЫ

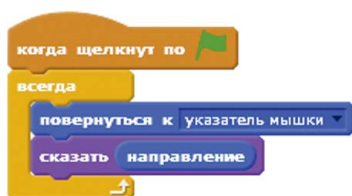
Такие слова, как *вверх* или *вправо*, хорошо понятны людям. Но компьютеру нужно числовое значение, чтобы указать точное направление. Все спрайты в Scratch имеют свое собственное значение направления. Значения направления находятся в диапазоне от  $-180$  до  $180$  градусов. Направление точно вверх задается значением  $0$  градусов. Указав значение в  $90$  градусов, вы задаете направление вправо. На следующем рисунке показано несколько направлений и их градусы. Обратите внимание на то, что по часовой стрелке происходит увеличение градуса, а против часовой стрелки – уменьшение. Кроме того, обратите внимание, что значения  $-180$  и  $180$  градусов указывают в том же направлении: вниз.



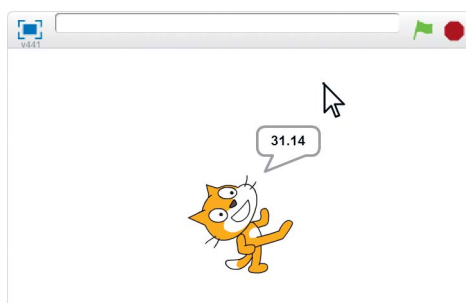
Блок **Выдать случайное от  $-180$  до  $180$**  выбирает случайное число для использования в качестве направления. Затем блок **Повернуть в направлении** указывает спрайту это направление. Это означает, что спрайт может быть направлен куда угодно.

Давайте напишем новый скрипт, который демонстрирует работу градусов. Откройте новую вкладку, нажав сочетание клавиш **Ctrl+T** в вашем браузере, и перейдите по ссылке [scratch.mit.edu](https://scratch.mit.edu), чтобы открыть новое окно редактора Scratch. Вы можете одновременно редактировать несколько программ Scratch.

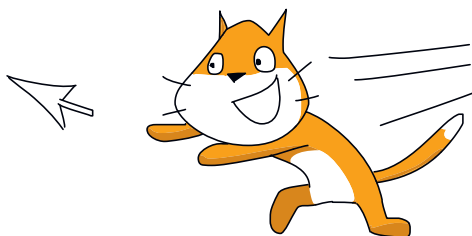
В области скриптов для спрайта кота с именем **Спрайт1**, добавьте код, показанный на следующем рисунке, используя блоки категорий **События** (коричневый), **Управление** (желтый), **Движение** (темно-синий) и **Внешность** (фиолетовый). Имейте в виду, что мы пишем совершенно отдельную от игры Радужные линии программу!



При запуске этой программы спрайт кота поворачивается в направлении мыши. Также он будет сообщать направление, в котором он повернулся.

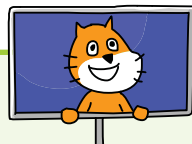


Обратите внимание на то, что числовое значение направления изменяется при повороте кота.



## 4. Дублирование спрайта Точка 1

Щелкните правой кнопкой мыши по спрайту **Точка 1** в области спрайтов и выберите команду **Дублировать в контекстном меню**. Сделайте это два раза, поскольку вам понадобятся два дубликата: **Точка 2** и **Точка 3**. (Scratch будет автоматически присваивать спрайтам имена с увеличением числа.)



### КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить уже готовый код. Убедитесь, что все три точки движутся и отскакивают от края сцены. При дублировании спрайта **Точка 1** также продублировался и код спрайта. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## В ПРОРИСОВКА ЛИНИЙ РАДУГИ

Теперь, когда мы создали все точки, отталкивающиеся от краев сцены, мы можем создать четвертый спрайт для рисования радужных линий. Мы напишем программу, с помощью которой точка этого спрайта будет быстро перемещаться между тремя спрайтами отталкивающихся точек, рисуя при этом линии. Этот процесс будет повторяться три раза, а затем через 10 секунд экран очистится.

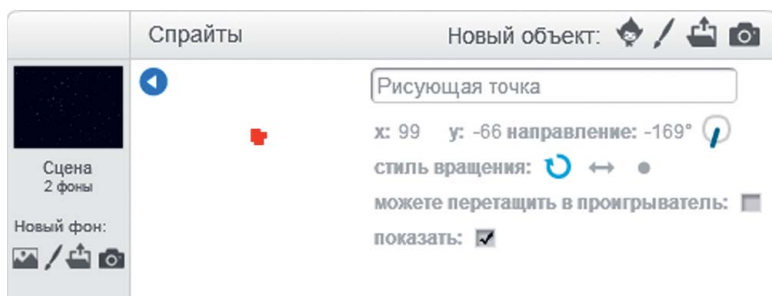
## 5. Добавление кода спрайта Рисующая точка

Щелкните правой кнопкой мыши по спрайту одной из отталкивающихся точек и выберите команду **Дублировать** в контекстном меню. Поскольку это дубликат, у него уже есть определенный код, который нам нужно будет удалить. В области скриптов нового спрайта удалите все блоки кода, щелкнув правой кнопкой мыши по блоку **Когда щелкнут**



по зеленому флагу, а затем выберите команду **Удалить** в контекстном меню. Также вы можете удалять блоки, перетаскивая их в область блоков в центре редактора.

Нажмите кнопку **i** и переименуйте новый спрайт, присвоив ему имя **Рисующая точка**.



Добавьте два скрипта, показанные на следующем рисунке, в спрайт **Рисующая точка**. Вы можете найти эти блоки в категориях **События** (коричневый), **Перо** (бирюзовый), **Управление** (желтый) и **Движение** (темно-синий).

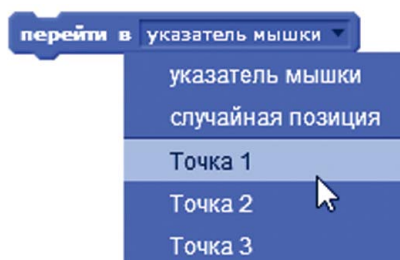
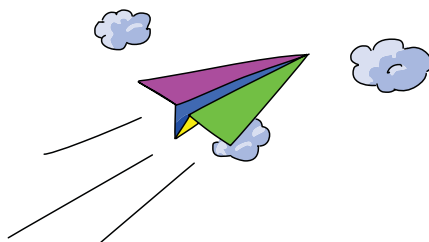


Убедитесь, что в скрипте **1** вы используете блок **Перейти в**, а не блок **Перейти в x: y:**. Кроме того, убедитесь, что вы поменяли в блоке **Перейти в** установленное по умолчанию значение **Указатель мыши**. Чтобы это сделать, щелкните мышью по черному треугольнику на этом блоке

и выберите в раскрывающемся списке имя соответствующего спрайта.

Перед тем как запустить код, давайте рассмотрим, как это работает. При нажатии кнопки в виде зеленого флага в скрипте ❶ спрайт **Рисующая точка**

запускает блок **Очистить**, чтобы убрать любое изображение, которое может быть на сцене. После этого запускается блок **Установить размер пера 8**, который присваивает перу размер в 8 пикселей. Затем скрипт запускает блок **Поднять перо** и переходит к спрайту **Точка 1**. Здесь перо опускается с помощью блока **Опустить перо**: когда спрайт движется, перо рисует линию на сцене.



Чтобы лучше понять, что делает блок **Опустить перо**, представьте себе, что вы ходите по большому листу бумаги, держа на нем открытый маркер: там, где вы пройдете, будет оставаться линия! Спрайт **Рисующая точка** перемещается к спрайту **Точка 1**, опускает вниз перо, движется к спрайту **Точка 2**, а затем к спрайту **Точка 3**. После этого блок **Изменить цвет пера на 10** плавно меняет цвет пера. (Вы можете увеличить это число, чтобы цвет изменялся быстрее.) Одновременно с этим, спрайты **Точка 1**, **Точка 2** и **Точка 3** продолжают передвигаться самостоятельно. Таким образом, V-образная линия, которую создает спрайт **Рисующая точка**, также движется.

Скрипт ❷ намного проще для понимания. Этот код ждет 10 секунд, а затем очищает экран от любых меток,

сделанных пером. Благодаря этому скрипту сцена не будет переполнена радужными линиями.



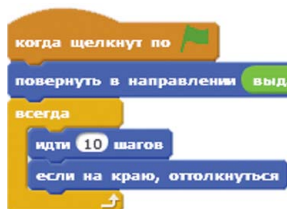
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Вы должны увидеть летающую радугу в форме буквы V, которая движется по сцене. Затем через каждые 10 секунд радуга исчезает. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу!

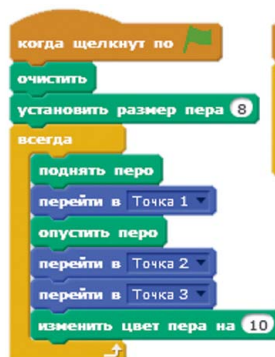
## ЗАВЕРШЕННАЯ ПРОГРАММА

Здесь вы видите окончательный код всей программы. Обратите внимание на то, что код для спрайтов **Точка 1**, **Точка 2** и **Точка 3** одинаков. Если ваша программа не работает должным образом, сравните свой код с этим:

точка 1   точка 2   точка 3

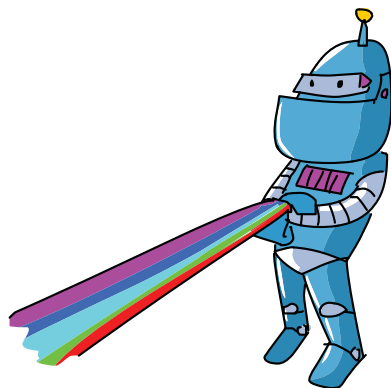


Рисующая точка

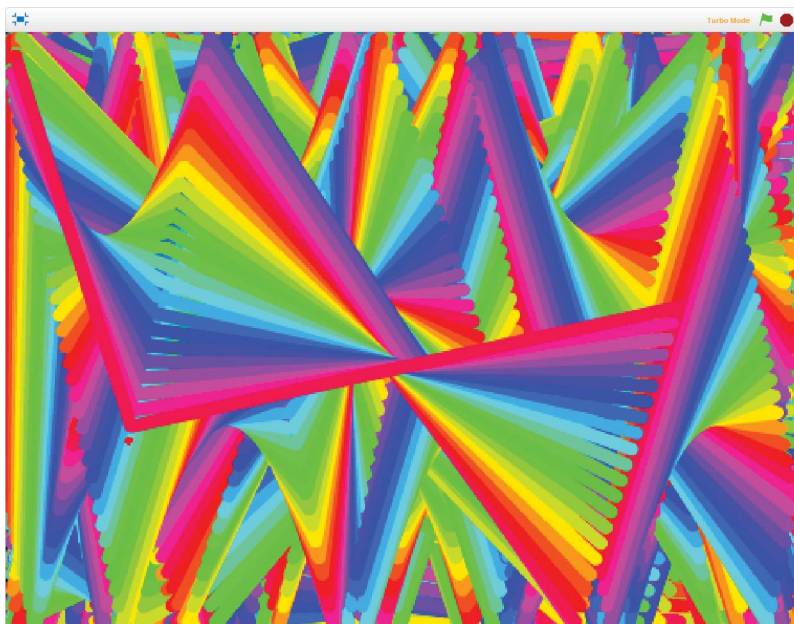


## ТУРБОРЕЖИМ

Если удерживать клавишу **Shift** при нажатии кнопки в виде зеленого флага, можно запустить программу в турбо-режиме. Компьютер, как правило, в состоянии запустить блоки кода быстро, но программа, которая рисует спрайты на экране, замедляет его работу. В турборежиме Scratch рисует на экране не после запуска каждого блока кода, а при прохождении сразу нескольких блоков. Человеческий глаз не замечает такие скачки, и поэтому кажется, что программа работает быстрее.



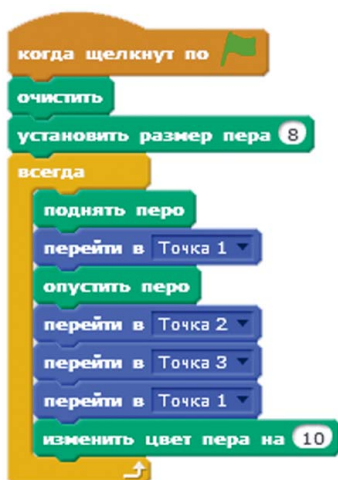
Нажмите кнопку в виде зеленого флага, удерживая клавишу **Shift**, чтобы запустить программу **Радужные линии** в турборежиме. Почти мгновенно экран будет заполнен! Для завершения турборежима снова нажмите кнопку в виде зеленого флага, удерживая клавишу **Shift**.



## ВЕРСИЯ 2.0: РАДУЖНЫЕ ТРЕУГОЛЬНИКИ

Ваша программа **Радужные линии** завершена, но вы можете перевести эту программу на следующий уровень. Если в вашу голову придут какие-либо идеи, вы всегда можете дополнить ими свою программу.

Так, для версии 2.0 только что созданной нами программы давайте соединим спрайты **Точка 3** и **Точка 1** так, чтобы вместо летающей радуги в виде буквы V, получился летающий радужный треугольник. Измените код спрайта **Рисующая точка**, чтобы он соответствовал коду, показанному на рисунке ниже, а остальную часть кода оставьте без изменений.



Новый блок **перейти в Точка 1** рисует линию от спрайта **Точка 3** до **Точка 1**, образуя треугольник.

### КОНТРОЛЬНАЯ ТОЧКА

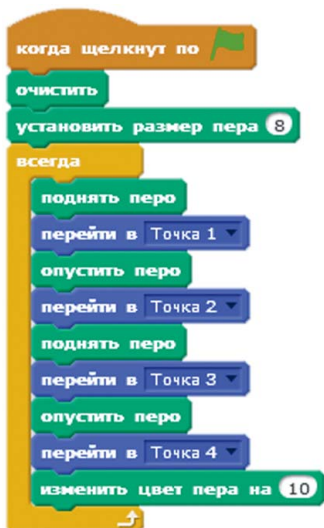


Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Убедитесь, что вы видите летающий радужный треугольник, который движется по сцене. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## ВЕРСИЯ 3.0: ДВЕ РАДУЖНЫЕ ЛИНИИ

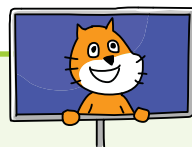
Для версии 3.0 нашей программы создадим две отдельных летающих линии вместо одной.

Щелкните правой кнопкой мыши по спрайту **Точка 3** и выберите команду **Дублировать** в контекстном меню, чтобы создать спрайт **Точка 4**. Затем измените код спрайта **Рисующая точка** так, как показано на следующем рисунке.



Новый код опускает перо вниз при перемещении между спрайтами **Точка 1** и **Точка 2**. Затем он поднимает перо вверх, идет к спрайту **Точка 3** и снова опускает перо, чтобы нарисовать линию от спрайта **Точка 3** до **Точка 4**.

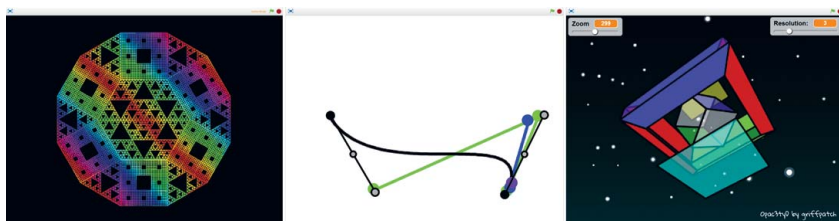
### КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Убедитесь, что вы видите две летающих радужных линии. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## ВЕРСИЯ 4.0: САМОСТОЯТЕЛЬНАЯ РАБОТА

Вы можете вносить свои собственные изменения и создавать новые программы. В той программе, что описана в этой главе, вы нарисовали линии. Но если вы знаете про кривые Безье, можно сделать линии в виде красивых кривых. Еще один математический способ создать красивые изображения – использовать фракталы (которые показаны на следующем рисунке). Перейдите на сайт Scratch и выполните поиск по словам *bezier*, *fractals* или *demoscene*, чтобы получить идеи для ваших собственных программ. Вы также можете увидеть блоки кода каждой найденной программы Scratch, нажав кнопку **See inside** на странице программы. Просмотрите некоторые примеры проектов на странице [www.nostarch.com/scratchplayground/](http://www.nostarch.com/scratchplayground/).



## ЗАКЛЮЧЕНИЕ

В этой главе вы создали проект, который:

- ▶ содержит пользовательский спрайт, созданный вами (пусть это даже просто точки);
- ▶ содержит блок **Выдать случайное**, определяющий случайное значение для спрайта;
- ▶ заставляет спрайты двигаться и отскакивать от краев сцены;
- ▶ содержит продублированные спрайты с кодом;
- ▶ содержит блоки **Поднять перо** и **Опустить перо** для рисования радужных линий.

Этот проект представляет собой демо, которое пользователи могут просматривать, но не могут управлять им. В главе 3 вы создадите игру-лабиринт, позволяющую игрокам взаимодействовать с программой при помощи клавиатуры, а не только наблюдать со стороны. Это будет первый реальный игровой проект в этой книге!



## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно посмотреть в конце книги.)

1. Что происходит в процессе движения спрайта после запуска блока **Опустить перо**?
2. Во время перемещения спрайта за ним не рисуется линия. Почему возникла такая ошибка?
3. Какой блок в программе **Радужные линии** отвечает за то, что линии выглядят радужными?
4. Какой блок кода нужно использовать, чтобы сделать радужные линии толще?
5. Как включается турбо-режим? А как выключается?
6. Как продублировать спрайт и его код?
7. В какую сторону направлен спрайт, если указано значение направления 90 градусов?
8. Какое значение направления надо указать, чтобы спрайт был направлен вверх?
9. Вам нужно, чтобы спрайт был направлен вниз и двигался в эту же сторону. Блоки какого цвета и категории нужны для этого?
10. Как выбрать новый фон из библиотеки фонов Scratch?
11. В области спрайтов есть спрайт с именем **Спрайт1**. Как его переименовать?



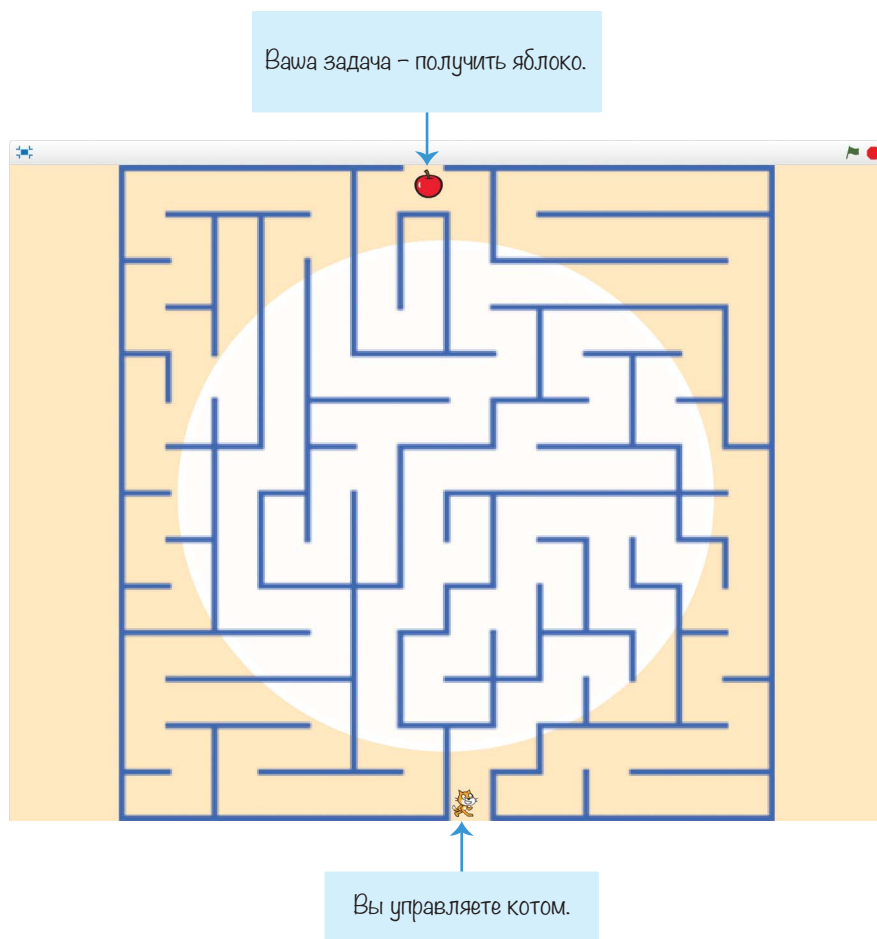
### 3

## БЕГУЩИЙ В ЛАБИРИНТЕ

**В**ероятно, вы уже играли в какую-нибудь игру-лабиринт. Но пробовали ли вы сделать такую игру сами? Лабиринты могут быть сложными для прохождения, но их легко программировать. В этой главе вы создадите игру, в которой игрок должен провести кота через лабиринт, в конце которого его ждет приз – вкусное яблоко!

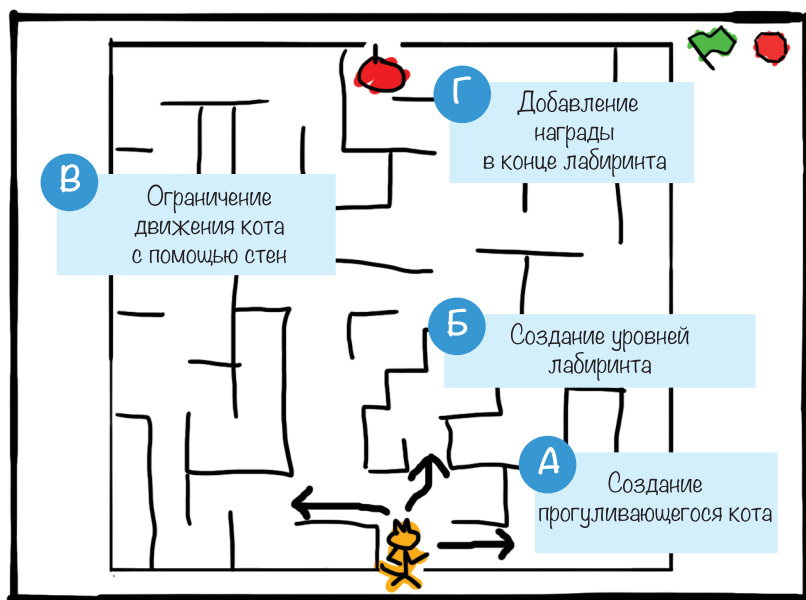
Вы узнаете, как управлять котом при помощи клавиатуры и как препятствовать движению, используя стены.

Перед тем как начать создавать код игры, взгляните на финальную программу. Перейдите по ссылке [scratch.mit.edu/studios/4188596/](https://scratch.mit.edu/studios/4188596/) и поиграйте в игру.



## ЭСКИЗ ПРОЕКТА

Для начала нарисуйте на бумаге то, как в вашем представлении должна выглядеть игра. Благодаря планированию вы можете сделать свою игру-лабиринт замечательной. Мой эскиз игры-лабиринта выглядит так, как показано на рисунке ниже.



Если вы хотите сэкономить время, можете начать с файла скелета проекта, который называется *Глава 03/Скелет\_проекта.sb2* и находится в запакованном файле с примерами. Перейдите по ссылке [https://eksmo.ru/files/Scratch\\_Sweigart.zip](https://eksmo.ru/files/Scratch_Sweigart.zip) и загрузите архивный файл на свой компьютер, щелкнув правой кнопкой мыши по ссылке и выбрав команду **Сохранить объект как**. Извлеките все файлы из архива. В файл скелета проекта уже загружены все спрайты, так что вам нужно всего лишь перетащить блоки кода в каждый спрайт. В редакторе Scratch выберите команду меню **Файл ► Загрузить с компьютера**, чтобы выбрать и загрузить файл *Глава 03/Скелет\_проекта.sb2*.

Даже если вы не используете скелет проекта, вам нужно загрузить этот запакованный файл. Он содержит изображения лабиринтов, которые будут использованы в этой главе.

Если вы хотите создать игру с нуля, выберите команду меню **Файл ► Новый**, чтобы запустить новый проект Scratch. В текстовом поле в верхнем левом углу переименуйте проект с *Untitled* на **Бегущий в лабиринте**.

## А СОЗДАНИЕ ПРОГУЛИВАЮЩЕГОСЯ КОТА

В игре «Бегущий в лабиринте» пользователь будет управлять спрайтом кота. В части А вы создадите код для управления котом клавишами со стрелками на клавиатуре.

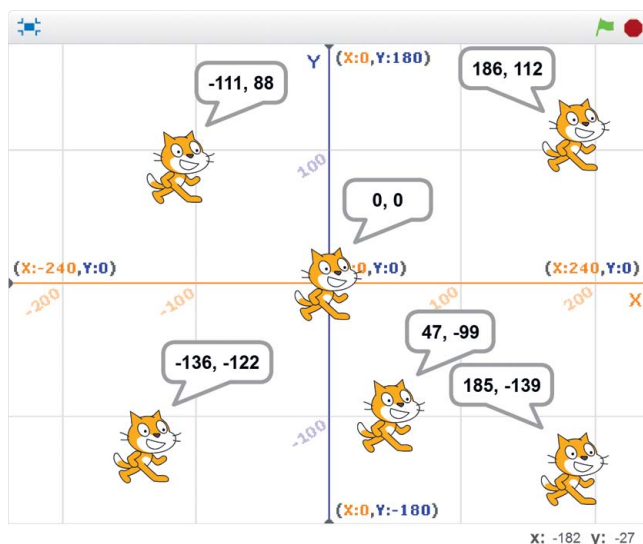


### ЭТО ИНТЕРЕСНО: КООРДИНАТЫ X И Y

Чтобы кот двигался по сцене, вам нужно использовать координаты. *Координатами* называются числа, которые указывают точное местоположение. Координата  $x$  представляет собой число, которое показывает, как далеко влево или вправо от центра сцены находится спрайт. Другими словами,  $x$  показывает *горизонтальное* положение спрайта. Координата  $y$  – это число, которое показывает, как далеко вверх или вниз от центра сцены находится спрайт. Таким образом, координата  $y$  отражает *вертикальное* положение спрайта.

Используемые вместе, координаты  $x$  и  $y$  показывают точное местоположение спрайта на сцене. Координата  $x$  всегда указывается первой, координата  $y$  – второй, и между ними ставится запятая. Например, если координата  $x$  равна 42, а координата  $y$  – 100, то в виде числовой записи они будут выглядеть следующим образом: (42, 100). Координаты точки в самом центре сцены составляют (0, 0). Эта точка называется *точкой начала*

отсчета. На следующем рисунке фоном представлена координатная сетка из библиотеки фонов Scratch. (Чтобы загрузить фон координатной сетки, нажмите кнопку **Выбрать фон из библиотеки** рядом с надписью **Новый фон** и выберите фон.) Я добавил несколько спрайтов котов, сообщающих свои координаты.



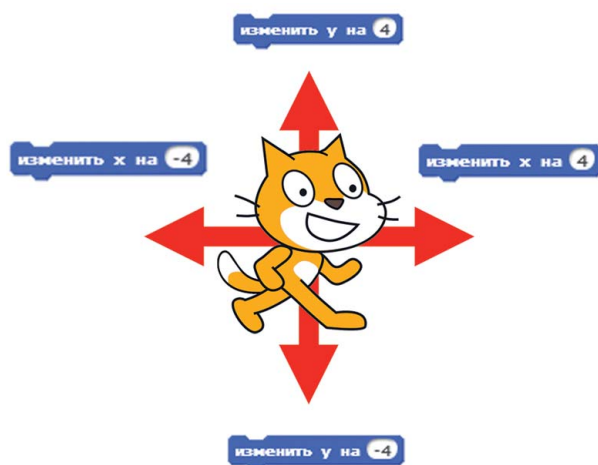
Крайняя правая точка на боковой стороне сцены имеет координату  $x$ , равную 240. Левее этой точки координаты  $x$  уменьшаются. В центре сцены координата  $x$  равна 0. Слева от центра координаты  $x$  становятся отрицательными числами. Крайняя левая точка на боковой стороне сцены имеет координату  $x$ , равную -240. Координаты  $y$  работают таким же образом: в верхней части сцены крайняя координата  $y$  имеет значение 180, центр - 0, а крайняя нижняя равна -180.

Координаты указателя мыши отображаются в правом нижнем углу сцены. На предыдущем рисунке указатель мыши находится в точке с координатами (-182, -27), что означает, что координата  $x$  составляет -182, а координата  $y$  -27.

Scratch отображает координаты выбранного спрайта в правом верхнем углу области скриптов. Когда вы изменяете координаты спрайтов, они двигаются по сцене, как показано в этой таблице:

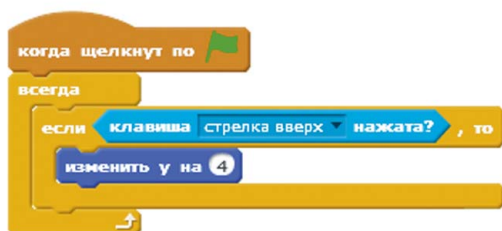
Чтобы спрайт переместился...	измените его...	на...
вправо	координату $x$	Положительное число
влево	координату $x$	Отрицательное число
вверх	координату $y$	Положительное число
вниз	координату $y$	Отрицательное число

Многие из блоков в темно-синей категории Движение меняют координаты  $x$  и  $y$  спрайта, например блоки **Изменить  $x$  на** и **Изменить  $y$  на**. Обратите внимание, что изменение координаты на отрицательное число — это то же самое, что вычесть из нее положительное число.



## 1. Добавление кода движения для спрайта игрока

Первый фрагмент кода, который вы добавите, будет отвечать за перемещение спрайта кота (который называется **Спрайт1**) с помощью клавиш со стрелками. Сначала нажмите кнопку **i** и переименуйте этот спрайт в **Рыжий кот**. Затем добавьте показанный ниже код. Вы найдете эти блоки в категориях **События**, **Управление**, **Сенсоры** и **Движение**.



Этот код циклично проверяет, нажата ли клавиша. Код дословно означает: «всегда проверять, нажата ли клавиша ↑, и если да, то изменить значение координаты y на 4».

Если клавиша ↑ не нажата, Scratch пропускает код внутри блока **Если, то**.

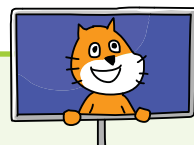
При нажатии клавиши ↑ спрайт **Рыжий кот** двигается вверх. Циклический блок **Всегда** подразумевает, что Scratch будет проверять снова и снова, нажата ли клавиша ↑. Проверка будет продолжаться до тех пор, пока вы не нажмете кнопку в виде красного знака остановки.



Блок **Всегда** необходим для работы этой программы. Без него Scratch бы проверил, нажата ли клавиша ↑, только один раз, а затем программа закончилась бы. Но нам нужно, чтобы Scratch продолжал проверять, нажата ли клавиша ↑, всегда. Если вам кажется, что ваша программа чего-то не делает, убедитесь, что вы не забыли добавить блок **Всегда**.



Когда вы пишете код самостоятельно, убедитесь, что вы используете блок **Изменить у на**, а не блоки **Изменить х на** или **Установить у на**. Если ваша программа не работает должным образом, проверьте, совпадает ли ваш код с кодом в этой книге.



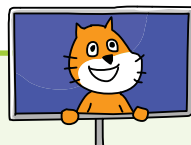
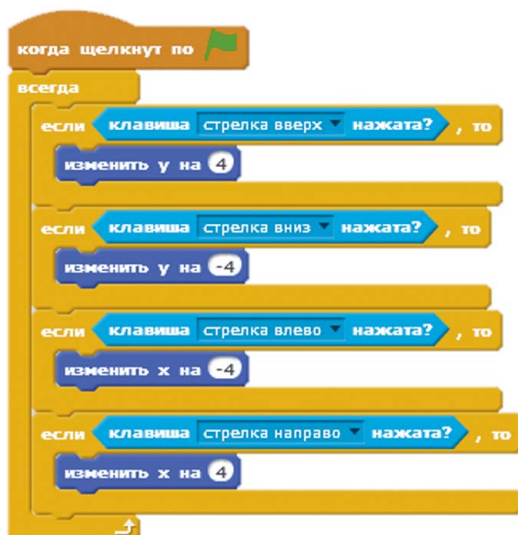
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага и попробуйте переместить кота, нажав клавишу  $\uparrow$ . Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## 2. Дублирование кода движения для спрайта кота

Теперь вы будете составлять код для трех оставшихся клавиш со стрелками:  $\leftarrow$ ,  $\downarrow$  и  $\rightarrow$ . Он похож на код для перемещения вверх спрайта **Рыжий кот**. Чтобы сэкономить время, вы можете щелкнуть правой кнопкой мыши по желтому блоку **Если, то** и выбрать пункт **Дублировать** в контекстном меню, чтобы создать копию блоков. Эти блоки будут идентичны, поэтому все, что вам нужно будет сделать – это изменить синие блоки движения на другие направления. Дублирование блоков часто позволяет писать программу быстрее, чем перетаскивание новых из области блоков.

Теперь Scratch будет проверять, нажата ли каждая из клавиш со стрелками, одну за другой. После проверки клавиши  $\rightarrow$ , Scratch возвращается в верхнюю часть цикла и снова проверяет клавишу  $\uparrow$ . Компьютер проверяет их так быстро, что кажется, будто все клавиши со стрелками проверяются одновременно!



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Кот должен двигаться вверх, вниз, влево или вправо при нажатии соответствующих клавиш со стрелками. Обратите внимание на то, что спрайт **Рыжий кот** достаточно мал и легко поместится в лабиринте, который вы создадите. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

Если ваша программа не работает и вы не знаете, как справиться с возникшей проблемой, можно начать все сначала с помощью файла проекта *лабиринт-часть-а.sb2*, который находится в архиве с примерами. В редакторе Scratch выберите команду меню **Файл ► Загрузить с компьютера**, чтобы загрузить файл *лабиринт-часть-а.sb2*, а затем переходите к части Б.

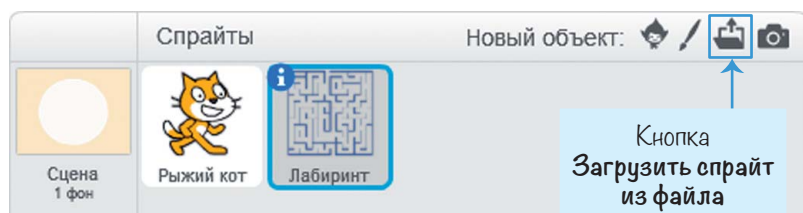
## 5 СОЗДАНИЕ УРОВНЕЙ ЛАБИРИНТА

Далее мы создадим спрайт лабиринта и установим фон. Играть в такую игру быстро надоедает, если все время проходить один и тот же лабиринт, поэтому мы сделаем в игре несколько уровней.

### 3. Загрузка изображений лабиринта

Вы можете сами нарисовать спрайт лабиринта или использовать изображения из архива с примерами. Одним из изображений лабиринта является файл *Лабиринт.sprite2*.

В редакторе Scratch нажмите кнопку **Загрузить спрайт из файла** и выберите *Лабиринт.sprite2*, чтобы загрузить его. У вас появится новый спрайт под названием **Лабиринт** с несколькими костюмами лабиринтов. Каждый спрайт в Scratch может иметь несколько костюмов, которые изменяют внешний вид. Вы можете увидеть их на вкладке **Костюмы**, которая часто используется для анимации спрайта. Ваша область спрайтов должна выглядеть следующим образом:

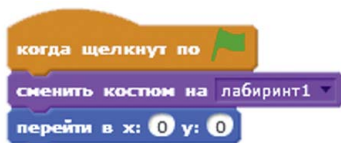


### 4. Изменение фона

Давайте добавим немного лоска в игру, поместив в качестве фона изображение. Вы можете использовать любой фон, который вам нравится. Изменить его можно, нажав кнопку **Выберите фон из библиотеки**, которая находится под надписью **Новый фон**, чтобы открыть окно **Библиотека фонов**. Выберите фон (я выбрал **Light**) и нажмите кнопку **ОК**.

## 5. Создание первого лабиринта

Добавьте код, показанный на следующем рисунке, в спрайт **Лабиринт**. Вы можете найти эти блоки в категориях **События**, **Внешность** и **Движение**.



Каждый из костюмов спрайта **Лабиринт** будет новым уровнем. Когда игрок нажимает кнопку в виде зеленого флага, чтобы запустить программу, игра начинается с первого костюма. Проследите, чтобы лабиринт находился в центре сцены. Код для перехода на следующий уровень мы добавим в шагах 8 и 9.

Обратите внимание, что область скриптов отображает блоки кода для выбранного спрайта. Убедитесь, что спрайт **Лабиринт** выбран в области спрайтов; если этого не сделать, вы можете случайно добавить код лабиринта к другому спрайту. Каждому спрайту для правильной работы нужен свой код. Если вы не видите пункт **Лабиринт1** в блоке **Сменить костюм на**, скорее всего, выбран спрайт **Рыжий кот**.

Если ваша программа Scratch не работает и вы не знаете, как это исправить, вы можете начать все сначала с помощью файла Scratch-проекта *лабиринт-часть-б. sb2*, который находится в архивном файле. В редакторе Scratch выберите команду меню **Файл ► Загрузить с компьютера**, чтобы загрузить файл *лабиринт-часть-б. sb2*, а затем переходите к части В.

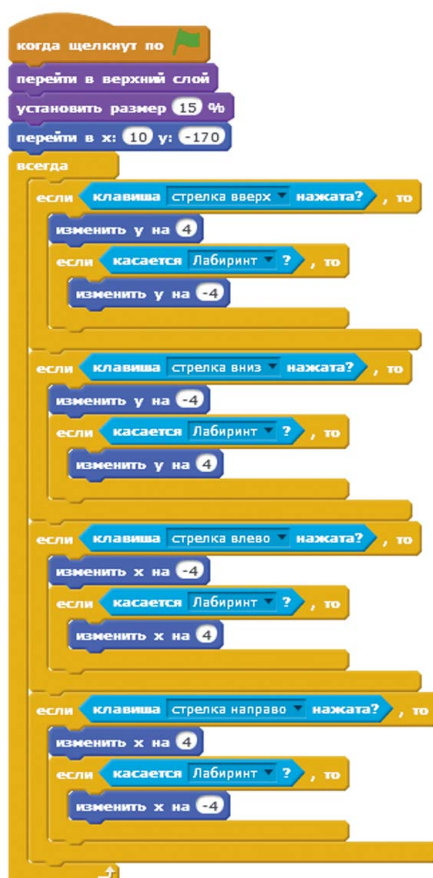
### **В** ОГРАНИЧЕНИЕ ДВИЖЕНИЯ КОТА ПРИ ПОМОЩИ СТЕН

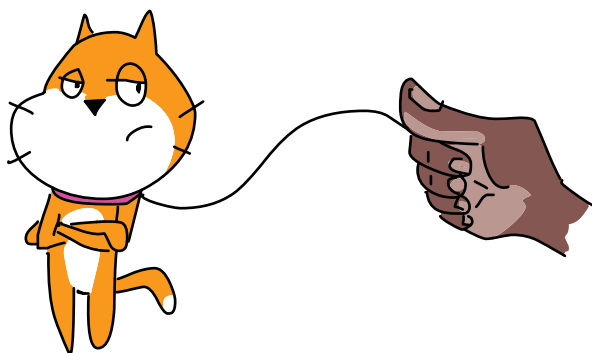
Теперь при нажатии кнопки с изображением зеленого флага вы сможете перемещать кота по лабиринту. Но вы также будете иметь возможность перемещать кота сквозь стены лабиринта, потому что ничто в программе не мешает этому.

Ваш код пока только выполняет: «при нажатии клавиши → переместить кота вправо». Кот движется независимо от того, есть ли на его пути стена или нет.

## 6. Проверим, касается ли кот стен

Давайте добавим код, который будет проверять, касается ли кот синих стен. Если да, то кот должен будет отходить назад. Поэтому, если кот движется вправо и касается стены, он должен автоматически передвинуться влево. Это устранил лишние движения игрока и сделает так, чтобы кот не перемещался сквозь стены. Выберите спрайт **Рыжий кот** в области спрайтов и измените код, чтобы он выглядел так, как показано ниже. Обратите внимание на то, что мы используем блок **Касается ?**, а не блок **Касается цвета ?**





Кроме того, вы, наверное, уже заметили, что по сравнению с размером лабиринта, спрайт **Рыжий кот** велик, как Годзилла, и выглядит нереалистично. Добавьте блок **Установить размер %** из категории **Внешность**, чтобы уменьшить спрайт **Рыжий кот**. Поскольку для прохождения лабиринта нужно, чтобы спрайт **Рыжий кот** всегда отображался поверх лабиринта, вам нужно добавить блок **Перейти в верхний слой**. Эти два блока находятся в верхней части скрипта.

## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить работу готового фрагмента кода. Убедитесь, что спрайт **Рыжий кот** не может пройти через стены лабиринта. Проверьте это для всех четырех направлений. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

Если ваша программа не работает и вы не знаете, как справиться с возникшей проблемой, начните сначала с помощью файла проекта *Scratch лабиринт-часть-в.sb2*, который находится в архиве с примерами. В редакторе Scratch выберите команду меню **Файл ► Загрузить с компьютера**, чтобы загрузить файл *лабиринт-часть-в. sb2*, а затем переходите к части Г.

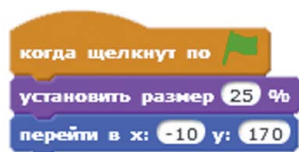
## Г ДОБАВЛЕНИЕ НАГРАДЫ В КОНЦЕ ЛАБИРИНТА

Пока не ясно, где конец лабиринта. Давайте добавим яблоко на выходе из него, чтобы сделать цель более очевидной для игрока.

### 7. Создание спрайта яблока

Нажмите кнопку **Выберите спрайт из библиотеки** (которая выглядит как изображение лица) рядом с надписью **Новый спрайт**. В появившемся окне **Библиотека спрайтов** выберите пункт **Apple** и нажмите кнопку **ОК**, чтобы добавить новый спрайт под названием **Apple** в область спрайтов. Для удобства вы можете переименовать его в **Яблоко**.

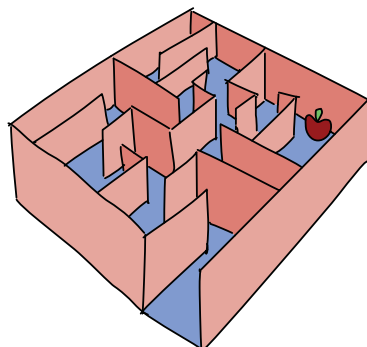
Когда начнется игра, спрайт **Яблоко** должен находиться на выходе из лабиринта в верхней части сцены. Спрайт **Яблоко** также должен быть достаточно мал, чтобы поместиться в лабиринте. Добавьте код, показанный на следующем рисунке, в спрайт **Яблоко**:



### 8. Выявление момента, когда игрок находит яблоко

Игра «Бегущий в лабиринте» уже содержит спрайт игрока, сам лабиринт и яблоко в конце. Теперь нужно написать код, позволяющий определить, когда игрок достигает конца лабиринта. Когда вы создадите этот код, ваша программа будет воспроизводить звук, а затем менять текущий костюм лабиринта на костюм следующего уровня. Но прежде чем добавить код, вам необходимо загрузить звук аплодисментов. В области спрайтов выберите **Рыжий кот**. Перейдите на вкладку **Звуки** в верхней части области бло-

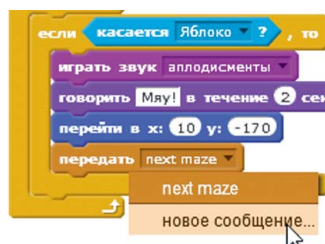
ков, а затем нажмите кнопку **Выбрать звук из библиотеки**. Эта кнопка выглядит как динамик и находится под надписью **Новый звук**.



В появившемся окне **Библиотека звуков** выберите пункт **Cheer**, а затем нажмите кнопку **ОК**, чтобы загрузить звук аплодисментов. Для удобства вы можете переименовать его, присвоив имя **Аплодисменты**. Теперь перейдите на вкладку **Скрипты**. Добавьте код, показанный на следующем рисунке, в спрайт **Рыжий кот**:

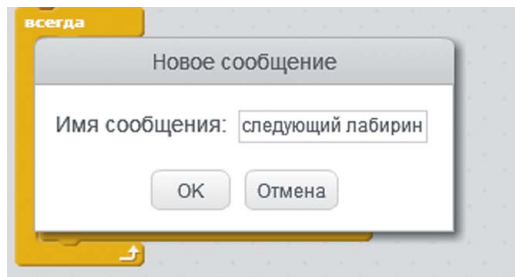


Чтобы блок **Передать** передавал сообщение **Следующий лабиринт**, щелкните мышью по черному треугольнику в блоке **Передать** и выберите пункт **Новое сообщение** в раскрывающемся списке.



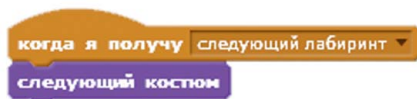
В появившемся окне введите текст **Следующий лабиринт** в качестве имени сообщения и нажмите кнопку **ОК**.





## 9. Добавление кода обработки сообщения в спрайт Лабиринт

В области спрайтов выберите спрайт Лабиринт и добавьте к его коду следующие блоки:



Этот код меняет уровень лабиринта, когда передается сообщение **Следующий лабиринт**.

### КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить код, который вы создали. Попробуйте сыграть в сделанную игру. Удостоверьтесь, что, когда кот касается яблока, уровень лабиринта меняется на следующий. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

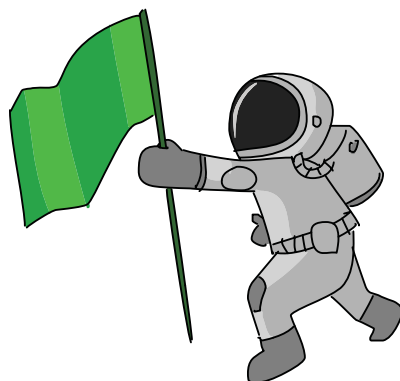
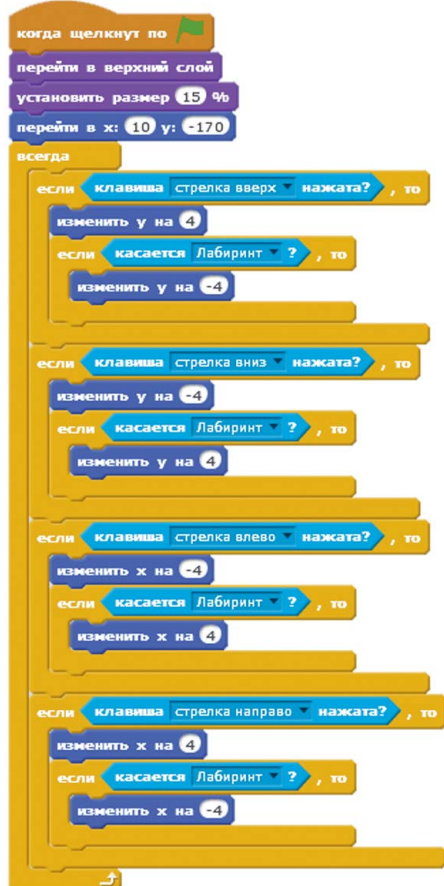
# ЗАКОНЧЕННАЯ ПРОГРАММА

Код, который вы увидите ниже, демонстрирует всю программу целиком. Если ваша программа работает неправильно, сверьте свой код с этой иллюстрацией. Полная программа находится также в файле *Лабиринт.sb2*.

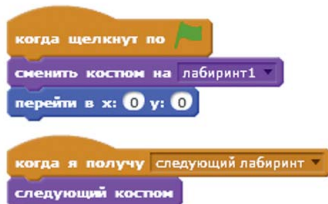
Я надеюсь, код этой программы-лабиринта не кажется вам слишком запутанным.



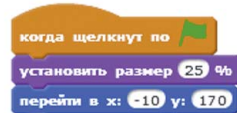
Рыжий кот



Лабиринт



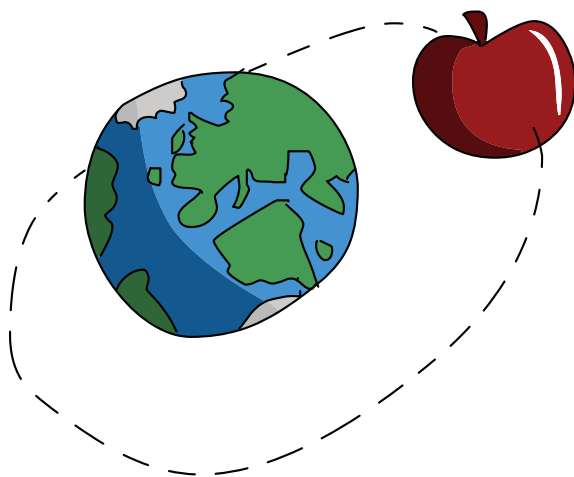
Яблоко



## ВЕРСИЯ 2.0: РЕЖИМ ДЛЯ ДВУХ ИГРОКОВ

Теперь, когда базовая версия игры-лабиринта готова, вы можете выполнить некоторую доработку и добавить к ней небольшие улучшения, по одному за раз. Это называется итеративная разработка – она помогает нам дополнять игру постепенно, чтобы не сделать ее настолько большой, что вы не сможете завершить начатое.

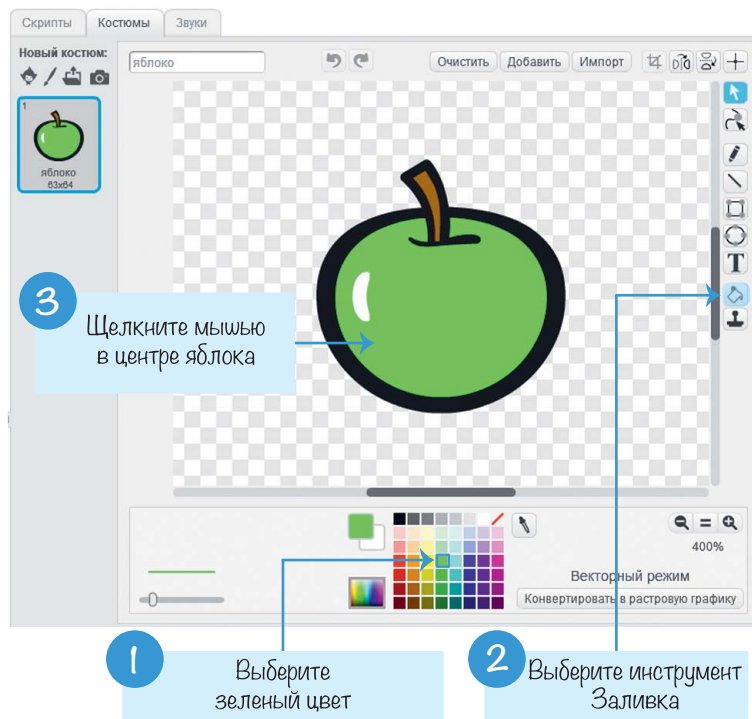
В версии 2.0 игры «Бегущий в лабиринте» вы добавите второго игрока. Два игрока будут играть друг против друга. Первый игрок начинает в нижней части лабиринта и будет двигаться вверх; второй игрок будет проходить лабиринт сверху вниз. Так как оба должны будут пройти по одному и тому же пути, расстояние для каждого из них будет одинаковым.



### Дублирование спрайта Яблоко

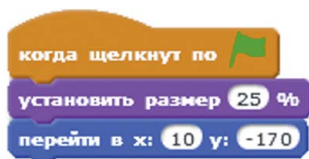
Второму игроку тоже нужна цель. Щелкните правой кнопкой мыши по спрайту **Яблоко** и выберите команду **Дублировать** в контекстном меню, чтобы создать копию яблока и его кода. Новому спрайту автоматически присваивается имя **Яблоко2**. Выберите спрайт **Яблоко2** и перейдите на

вкладку **Костюмы**. Выберите зеленый цвет в нижней части, а затем справа выберите инструмент **Заливка** (он выглядит как наклоненная чашка). Затем щелкните мышью по красной части яблока, чтобы перекрасить его в зеленый цвет. Когда вы сделаете это, **Яблоко2** будет выглядеть так, как показано на следующем рисунке.



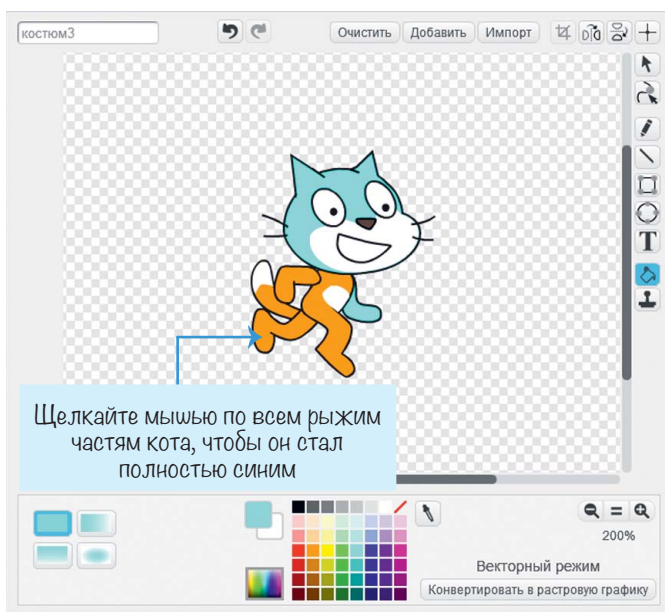
## Изменение кода спрайта **Яблоко2**

А теперь измените код спрайта **Яблоко2**, чтобы он выглядел так, как показано на следующем рисунке, и зеленое яблоко появлялось в нижней части лабиринта, а не сверху:



## Дублирование спрайта Рыжий кот

Теперь давайте добавим второй спрайт кота. Щелкните правой кнопкой мыши по спрайту **Рыжий кот** и выберите из меню **Дублировать**, чтобы сделать копию этого спрайта и его кода. Новый спрайт автоматически получает имя **Рыжий кот2**. Спрайты **Рыжий кот** и **Рыжий кот2** должны различаться, чтобы игроки их не перепутали. Как и в случае со спрайтом **Яблоко2**, перейдите на вкладку **Костюмы** и измените цвет спрайта **Рыжий кот2** с рыжего на синий.

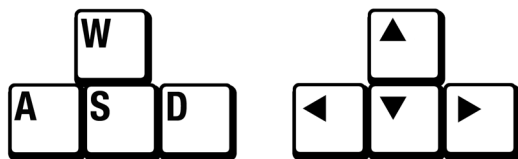


Нажмите кнопку **i** и переименуйте спрайт **Рыжий кот2** в **Синий кот**.

## Изменение кода спрайта Синий кот

На данный момент спрайт **Синий кот** имеет тот же код, что и спрайт **Рыжий кот**. Вам нужно изменить код; в противном случае клавиши со стрелками будут управлять обоими спрайтами сразу. Второй игрок будет управлять

спрайтом **Синий кот** с помощью клавиш **W**, **A**, **S** и **D**. Эти клавиши часто используются в качестве альтернативы клавишам  $\uparrow$ ,  $\leftarrow$ ,  $\downarrow$  и  $\rightarrow$  под левую руку.



В соответствии с кодом на рисунке ниже измените два блока **Перейти в x y** и **Клавиша нажата ?** для спрайта **Синий кот**. Кроме того, не забудьте изменить **Если касается Яблоко** на **Если касается Яблоко2** (См. рисунок на следующей странице)

## Возвращение в начальное положение

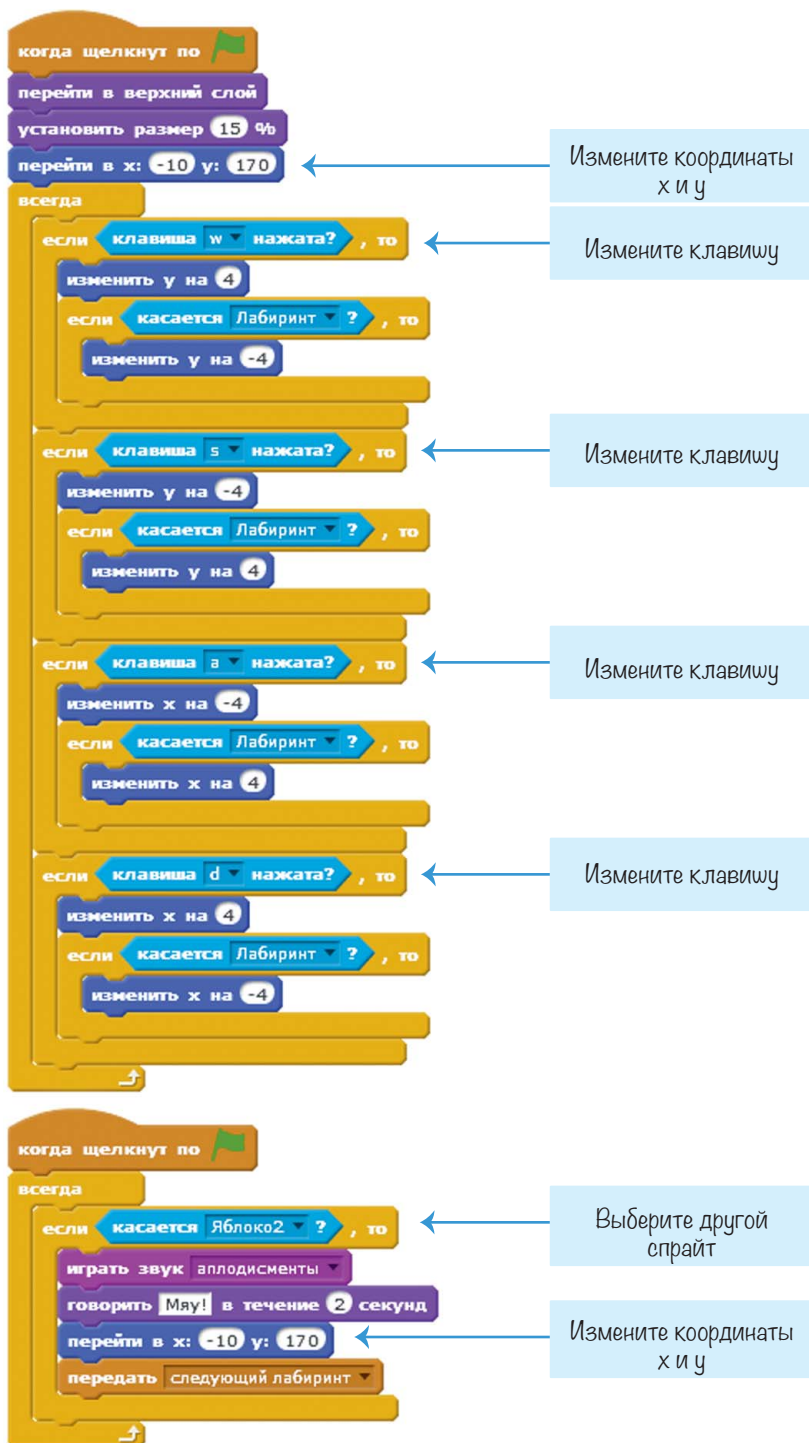
Спрайты котов вернутся в исходное положение, когда коснутся своих яблок. Они должны вернуться и в том случае, если выиграл один из игроков. Добавьте код, показанный на следующем рисунке, в спрайт **Рыжий кот**:



А затем добавьте код, показанный на следующем рисунке, в спрайт **Синий кот**:



Таким образом, при победе одного из соперников и выводе надписи «Следующий лабиринт», оба кота будут возвращаться в исходное положение.





## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Попробуйте передвинуть обоих игроков с помощью клавиш  $\uparrow$ ,  $\leftarrow$ ,  $\downarrow$ ,  $\rightarrow$  и **W**, **A**, **S**, **D**. Убедитесь, что каждая из восьми клавиш перемещает нужного кота, и только его одного. Попробуйте пройти лабиринт полностью. Убедитесь, что уровень лабиринта меняется на следующий, когда второй игрок касается зеленого яблока. Проверьте, что оба игрока находятся в исходных позициях, когда начинается следующий уровень. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

Вы только что обновили игру «Бегущий в лабиринте» до прохождения ее двумя игроками. Найдите друга для этой гонки. Игрок 1 использует клавиши  $\uparrow$ ,  $\leftarrow$ ,  $\downarrow$  и  $\rightarrow$ , а игрок 2 – клавиши **W**, **A**, **S** и **D**.

## ВЕРСИЯ 3.0: ЛОВУШКИ

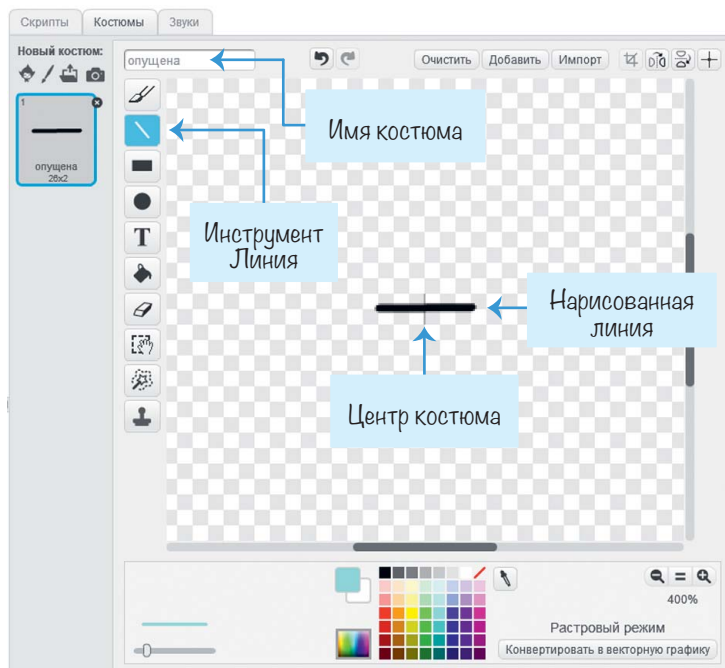
После того как вы несколько раз сыграли в эту игру против другого игрока, она может показаться слишком легкой. Давайте сделаем игру «Бегущий в лабиринте» немного сложнее, добавив в нее ловушки с шипами. Опасные ловушки будут поднимать и прятать шипы. Если игрок касается ловушки в то время, когда шипы выдвинуты, его движения будут замедляться. Это даст его сопернику преимущество во времени.

### Создание нового спрайта для ловушек

Нажмите кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**, чтобы создать новый спрайт. Для этого спрайта вы нарисуете два разных костюма: один – со втянутыми шипами, а другой – с шипами наружу. Не волнуйтесь, ловушки с шипами рисовать довольно легко. Выберите инструмент **Линия** и черный цвет, что-

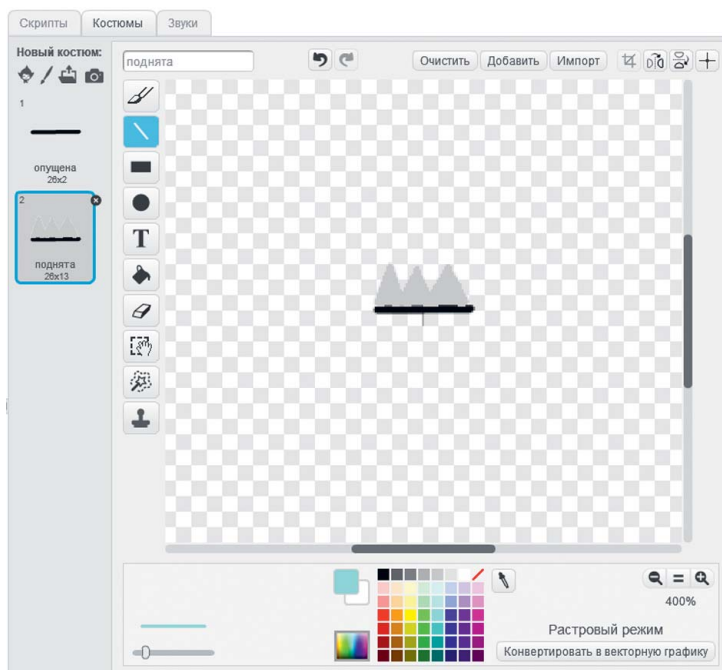


бы нарисовать простую линию. Так будет выглядеть ловушка, когда ее шипы убраны. Переименуйте этот костюм спрайта в **Ловушка в опущена**.

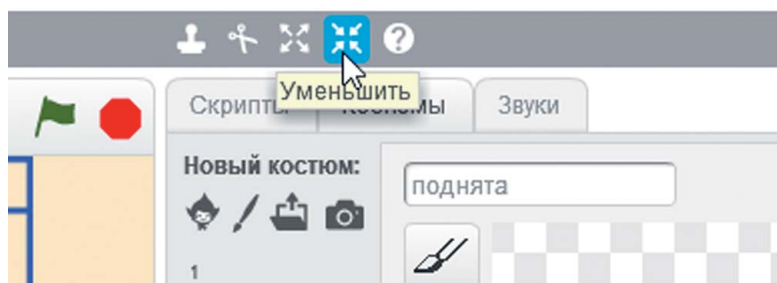


## Создание второго костюма для ловушек

Для спрайта **Ловушка** нужен второй костюм, который будет показывать ловушку с шипами наружу. Щелкните правой кнопкой мыши по костюму **Ловушка опущена** (одна линия, которую вы нарисовали) спрайта **Ловушка** и выберите команду **Дублировать** в контекстном меню для создания костюма **Ловушка опущена2** спрайта **Ловушка**. Для этого костюма выберите серый цвет и пририсуйте шипы, торчащие из линии. Переименуйте этот костюм в **Ловушка поднята**.



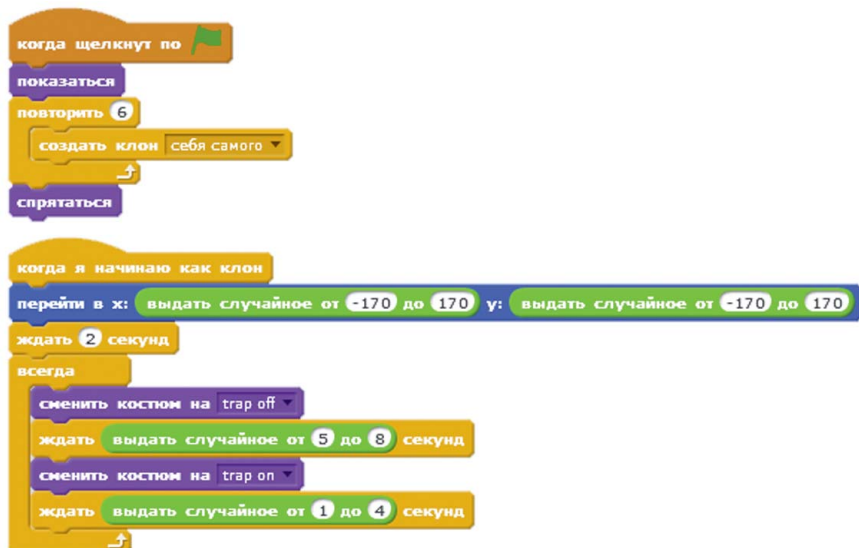
Убедитесь, что ловушка имеет подходящий размер, чтобы поместиться в лабиринте. Чтобы уменьшить его, нажмите кнопку **Уменьшить** (как показано на рисунке ниже), а затем щелкните мышью по спрайту на сцене. Нажмите кнопку **Уменьшить** столько раз, сколько необходимо, чтобы уменьшить спрайт до нужного размера.



Кроме того, нажмите кнопку **i** для спрайта, чтобы открыть его панель информации, и переименуйте спрайт в **Ловушка**.

## Добавление скопированного кода для ловушек

Для лабиринта вам потребуется несколько ловушек, поэтому вы можете продублировать спрайт **Ловушка**, но есть способ лучше. Вы можете использовать блоки клонирования, чтобы сделать клоны спрайта **Ловушка**. Для спрайта **Ловушка** создайте клон, который будет выглядеть следующим образом:

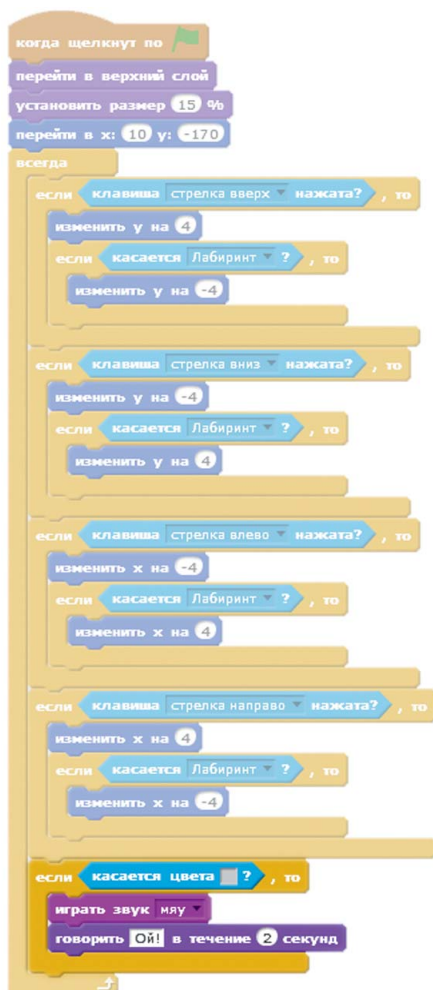


Цикл **Повторить 6** будет запускать блок **Создать клон себя самого** шесть раз, чтобы сделать шесть клонов. В дальнейшем эти клоны будут запускать код в блоке **Когда я начинаю как клон**. Сначала этот скрипт помещает спрайт **Ловушка** в случайное место в лабиринте. После того, как блок **Ждать 2 секунд** делает небольшую паузу, циклический блок **Всегда** начинает поочередно активировать костюмы **Ловушка опущена** и **Ловушка поднята**, отчего ловушка выпускает или втягивает шипы.



На данный момент коты не запрограммированы на какие-либо действия при соприкосновении с шипами ловушек. Выберите пункт **Рыжий кот** в области спрайтов и добавьте код, показанный на следующем рисунке, в *нижней части* уже существующего кода внутри блока **Всегда**. Для

того чтобы установить цвет в блоке **Касается цвета ?**, щелкните мышью по индикатору цвета внутри него. Следующий цвет, по которому вы щелкнете мышью в редакторе Scratch, станет выбранным цветом. Щелкните мышью по шипам спрайта **Ловушка** в области спрайтов, чтобы установить серый цвет в блоке **Касается цвета ?**. (Если на миниатюре спрайта не отображаются шипы, выберите спрайт **Ловушка в области спрайтов**, откройте вкладку **Костюмы**, а затем выберите пункт **Ловушка поднята**. В области спрайтов отобразится костюм с шипами.) Добавьте код, показанный на следующем рисунке, в скрипт с блоками **Клавиша нажата ?**:



Кот может беспрепятственно ходить по клонам ловушек, когда шипы убраны, так как на них отсутствует серый цвет. Но если шипы подняты, кот будет касаться серого цвета и останавливаться на 2 секунды, говоря: «Ой!»

## КОНТРОЛЬНАЯ ТОЧКА

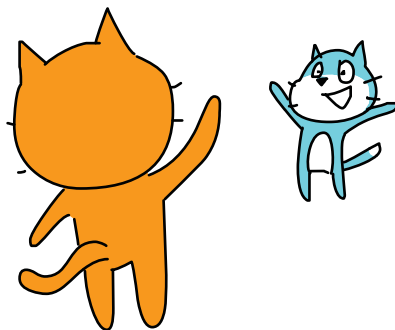


Нажмите кнопку в виде зеленого флага, чтобы проверить измененный код. Попробуйте поместить рыжего кота в ловушку, когда шипы убраны и когда они выдвинуты. Кот должен сказать: «Ой!» – только когда видны шипы. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## Копирование кода из спрайта **Рыжий кот** в **Синий кот**

Для спрайта **Синий кот** второго игрока тоже необходим код для определения цвета в блоке **Касается цвета ?**.

Вы можете воспользоваться следующим способом: нажав и удерживая кнопку мыши на желтом блоке **Если касается цвета ?** то спрайта **Рыжий кот**, перетащите блок и отпустите его поверх спрайта **Синий кот** в области спрайтов. Блок кода вернется на свое первоначальное место в скрипте **Рыжий кот**, а также продублируется в скрипт спрайта **Синий кот**. Этот метод, как правило, быстрее, чем перетаскивание новых блоков.



Выберите пункт **Синий кот** в области спрайтов. Переместите дублированный код в нижнюю часть цикла **Всегда**. Код выглядит точно так же, как в скрипте **Рыжий кот**. На следующем рисунке показано, как перетаскивать и дублировать эти блоки кода.

1 Перетащите желтый блок **Если, то**, чтобы вытащить его из скрипта

2 Отпустите кнопку мыши поверх спрайта **Синий кот** в области спрайтов

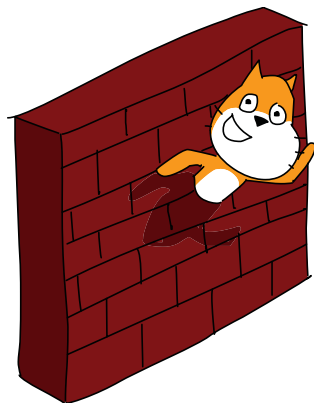
3 Код вернется обратно и встроится в первоначальный код, а также скопируется в спрайт **Синий кот**

Вы закончили версию 3.0 игры «Бегущий в лабиринте». С ловушками и двумя игроками она стала интереснее, чем изначальная версия 1.0. Вы всегда можете добавлять в свою игру новые возможности.

Теперь давайте добавим некоторые секретные коды, которые игроки могут использовать, чтобы взломать игру.

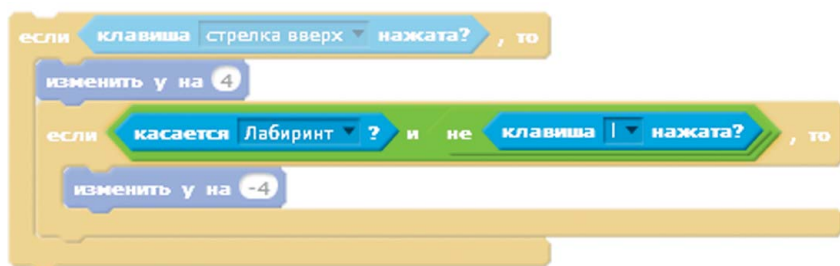
## ЧИТ-РЕЖИМ: УМЕНИЕ ПРОХОДИТЬ СКВОЗЬ СТЕНЫ

Телепортация – это крутой чит, но игроки не могут управлять тем, куда они телепортируются. Кроме того, мошенничество становится слишком очевидным, когда игрок внезапно перемещается по экрану через множество стен. Тем не менее вы можете добавить чит, который позволит коту двигаться сквозь стены, когда удерживается в нажатом положении специальная клавиша.



### Добавление кода для прохождения сквозь стены для рыжего кота

Для спрайта **Рыжий кот** измените код, отвечающий за его движение, следующим образом: блоки **Касается лабиринта ?** Замените на **касается лабиринта ? и не клавиша I нажата ?**. На рисунке ниже показан код для клавиши **I**. Замените блоки кода для всех четырех клавиш, управляющих движением кота.





Таким образом, если клавиша **L** *не нажата*, пройти сквозь стену невозможно. При *нажатой* клавише **L** фрагмент кода, блокирующий стены, пропускается, и игрок может пройти сквозь стену.

## Добавление кода для прохождения сквозь стены для синего кота

Создайте такие же изменения для прохождения стен в коде спрайта **Синий кот**. Только вместо кода **Клавиша l нажата ?**, укажите **Клавиша q нажата ?**. Теперь второй игрок может проходить сквозь стены, если клавиша **Q** удерживается в нажатом положении.



### КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить измененный код. Попробуйте пройти сквозь стены, удерживая нажатой клавишу **L** или **Q**. Убедитесь, что оба кота могут проходить сквозь стены, но только в том случае, когда удерживается соответствующая клавиша. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

С помощью этого чит-кода вы можете взломать игру и сделать вашего кота способным проходить сквозь стены. Этот пример демонстрирует, что в программах Scratch возможно все!

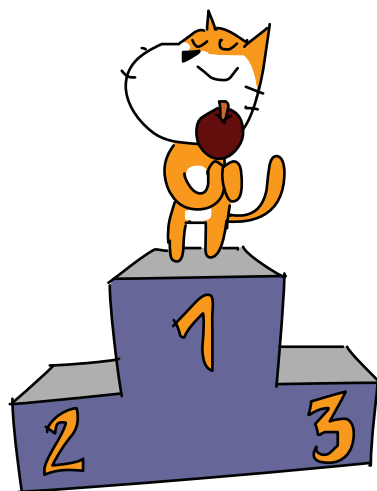
## ЗАКЛЮЧЕНИЕ

В этой главе вы создали игру, в которой:

- ▶ есть спрайт кота,двигающийся вверх, вниз, влево и вправо, когда игрок нажимает соответствующие клавиши;
- ▶ есть стены, сквозь которые спрайты не могут проходить;
- ▶ передаются сообщения от одного спрайта к другому;
- ▶ содержится спрайт лабиринта с восемью различными костюмами;
- ▶ поддерживается режим игры для двух игроков, использующих различные клавиши клавиатуры;
- ▶ добавлены задерживающие игроков ловушки, срабатывающие через случайные промежутки времени;
- ▶ содержится чит-режим, благодаря которому коты могут проходить сквозь стены.

Режим игры для двух игроков интереснее, чем одиночная игра. Теперь, вместо того, чтобы просто пройти лабиринт, вы соревнуетесь с другим игроком! Вы можете с гордостью показать вашу игру друзьям.

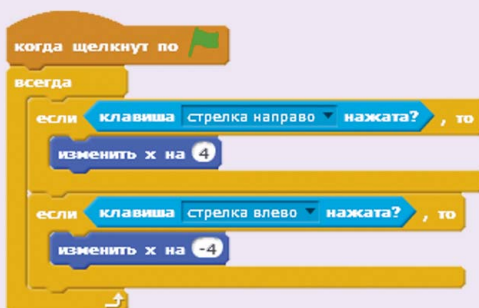
В главе 4 вы поиграете в баскетбол. В этой игре вид будет сбоку, в отличие от предыдущей игры, в которой используется вид сверху. Но это означает, что вы сможете добавлять прыжки и учитывать силу тяжести – возможности, которые часто используются во многих играх на основе Scratch.



## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно подсмотреть в конце книги.)

1. Какой блок влияет на размер спрайта?
2. Каким образом можно запрограммировать отправку сообщения от одного спрайта к другому с указанием, что делать?
3. Для чего на клавиатуре используются клавиши **W**, **A**, **S** и **D**?
4. Как скопировать отдельные блоки кода из одного спрайта в другой?
5. Что произойдет, если вы случайно используете блок кода **Изменить у на** вместо блока **Изменить x на**?
6. Если возникает необходимость в программе проиграть звук **Cheer**, как вы его загрузите?
7. Взгляните на код, показанный на следующем рисунке. Он позволяет игроку нажимать клавиши со стрелками для перемещения спрайта влево и вправо. Он работает, но что бы вы изменили, чтобы спрайт двигался быстрее?





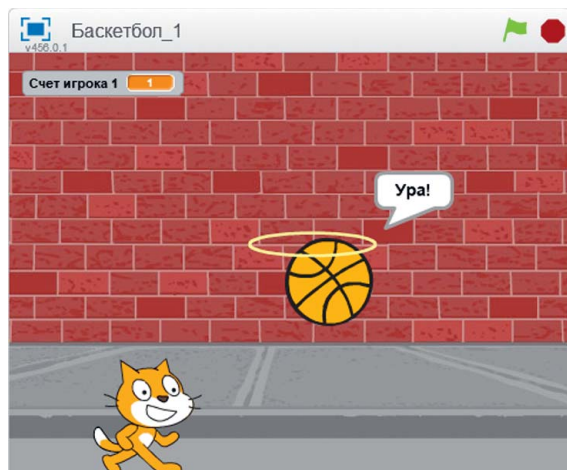
# 4

## БАСКЕТБОЛ С УЧЕТОМ СИЛЫ ТЯЖЕСТИ

**В**о многих играх-платформерах, таких как *Super Mario Bros.* и *Donkey Kong Country*, вы видите игру сбоку. Нижняя часть экрана представляет собой землю, а герои показаны с боковой стороны. В этих играх учтено действие *силы тяжести*: персонажи могут прыгать вверх, а затем опускаются вниз и приземляются.

В этой главе мы создадим игру в баскетбол, в которой будет действовать сила тяжести. Игрок будет прыгать и бросать мяч, а затем и баскетболист, и мяч будут приземляться.

Перед тем как начать программировать, взгляните на окончательную версию игры по ссылке [scratch.mit.edu/studios/4188596/](https://scratch.mit.edu/studios/4188596/).



## ЭСКИЗ ПРОЕКТА

Для начала давайте наметим то, что должно происходить в игре. Игрок управляет котом, который может перемещаться влево и вправо и подпрыгивать. Задача этой игры состоит в том, чтобы на сцене был кот, который бросает баскетбольный мяч в движущееся баскетбольное кольцо. Оно при этом будет перемещаться по сцене в случайном порядке.

Если вы хотите сэкономить время, вы можете начать с файла скелета проекта, под названием *Глава 04/Скелет\_проекта.sb2*, находящегося в запакованном файле с примерами, который вы скачали ранее по ссылке [https://eksmo.ru/files/Scratch\\_Sweigart.zip](https://eksmo.ru/files/Scratch_Sweigart.zip). В файл скелета проекта уже загружены все спрайты, так что вам нужно всего лишь перетащить блоки кода в каждый спрайт.



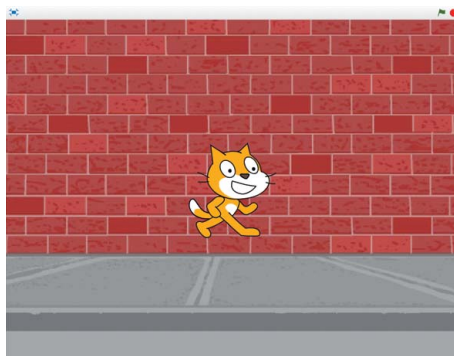
## А ОБУЧЕНИЕ КОТА ПОДПРЫГИВАНИЮ И ПРИЗЕМЛЕНИЮ

Давайте начнем с добавления силы тяжести, чтобы наш кот мог прыгать и приземляться.

### 1. Добавление кода силы тяжести к спрайту кота

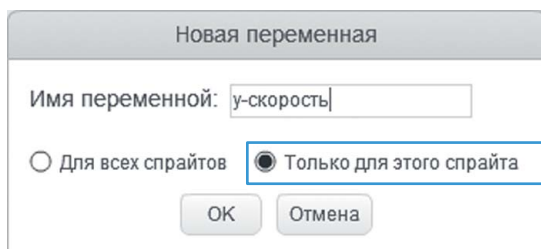
Нажмите кнопку **i** на миниатюре спрайта **Спрайт1**, чтобы открыть его панель информации. Измените имя спрайта на **Кот** и закройте панель информации. Далее в текстовом поле в верхней части редактора Scratch переименуйте программу *Untitled* в **Баскетбол**.

Нажмите кнопку **Выбрать фон из библиотеки** под надписью **Новый фон**, чтобы открыть окно библиотеки фонов. Выберите **Brick wall1** и нажмите кнопку **ОК**, чтобы изменить фон. Для удобства вы можете переименовать фон, присвоив ему имя **Кирпичная стена**. Сцена будет выглядеть следующим образом:



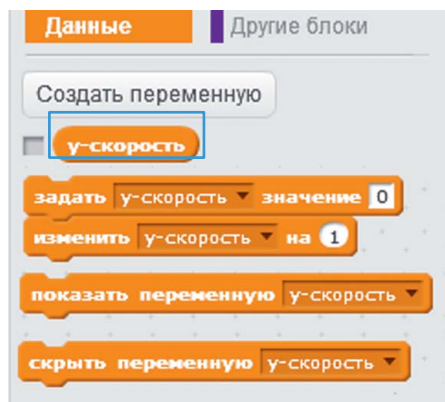
Для программирования силы тяжести требуются переменные. Переменную можно представить как коробочку, в которой хранятся числа или текст, которую можно открыть при необходимости и использовать ее содержимое в своей программе. Мы создадим числовую переменную, которая задает скорость падения кота.

Первым делом удостоверьтесь, что в области спрайтов выбран спрайт **Кот**, а затем перейдите на вкладку **Скрипты**. В категории **Данные** оранжевого цвета нажмите кнопку **Создать переменную**, в результате чего появится окно **Новая переменная**. Введите текст **у-скорость** в качестве имени переменной. (*Скорость* означает, как быстро какой-либо объект движется и в каком направлении. Когда значение переменной **у-скорость** представляет собой положительное число, кот движется вверх. Когда значение переменной **у-скорость** – отрицательное число, кот движется вниз. Установите переключатель в положение **Только для этого спрайта**. (Если вы видите только параметр **Для всех спрайтов**, то выбрана сцена, а не спрайт **Кот**.) Затем нажмите кнопку **ОК**.



Убедитесь, что выбран режим **Только для этого спрайта**!

В категории **Данные** появится несколько новых блоков, и одним из них будет скругленный блок переменной **у-скорость**, который можно увидеть на этом рисунке:



## ЭТО ИНТЕРЕСНО: РЕЖИМЫ «ДЛЯ ВСЕХ СПРАЙТОВ» И «ТОЛЬКО ДЛЯ ЭТОГО СПРАЙТА»

При создании переменной **у-скорость** вы должны установить переключатель в положение **Только для этого спрайта**. Этот параметр создает переменную, которая может использоваться только спрайтом **Кот**. Режим **Для всех спрайтов** создает переменную, которая может использоваться всеми спрайтами.

Чтобы определить тип созданной переменной, установите флажок в скругленном блоке переменной в области блоков, чтобы имя и значение переменной отобразились на сцене. Если вы выбрали режим **Только для этого спрайта**, имя спрайта будет отображаться перед именем переменной. А если вы выбрали режим **Для всех спрайтов**, то будет отображаться только имя переменной.





Имя спрайта **Кот** перед названием переменной **у-скорость** означает, что эта переменная действует только для этого спрайта

Убедитесь, что флажок установлен, чтобы переменная была видна на сцене

Если вы допустили ошибку и имя спрайта **Кот** не появилось перед именем переменной в вашей программе, щелкните правой кнопкой мыши по блоку **у-скорость** в оранжевой категории **Данные** и выберите в меню команду **Удалить переменную**. После этого снова создайте переменную **у-скорость** и удостоверьтесь, что вы выбрали режим **Только для этого спрайта**.

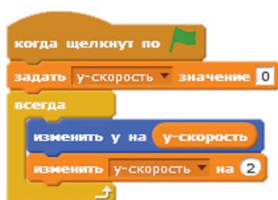
Как и любой другой блок, вы можете поместить блок **у-скорость** в любом месте, где вы обычно вводите число или текст. Блок кода будет использовать число или текст, указанный в качестве значения переменной. Когда вы используете переменные, их значения (число или текст) меняются *во время работы программы*.

Чтобы установить значение переменной, используется оранжевый блок **Задать значение**. Например, если вы создали переменную **Приветствие**, то могли бы использовать блок **Задать значение**, чтобы установить в нем значение **Привет!**. Затем вы можете использовать приветствие в блоке **Сказать**, который означает то же самое, что и ввод значения **Привет!** вручную. (Не добавляйте эти блоки и не создавайте переменную **Приветствие** в вашей баскетбольной программе, это всего лишь пример.)



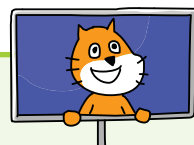
Если вы хотите изменить текст приветствия во время работы программы, добавьте еще один блок **Задать значение** в нее. Если переменная содержит число, вы можете прибавить или вычесть из этого числа, используя блок **Изменить на**.

Сила тяжести является причиной того, что объекты падают вниз с ускорением. В игре спрайт **Кот** должен двигаться вниз, и скорость, с которой он движется вниз, должна изменяться *в процессе работы программы*. Добавьте код, показанный на следующем рисунке, в спрайт **Кот**, чтобы в вашей программе появилась сила тяжести. Этот минимальный код нужен для того, чтобы кот в вашей игре приземлялся под действием силы тяжести. Этот код вы можете добавить в любой спрайт, и тот будет падать под действием силы тяжести.



При нажатии кнопки в виде зеленого флага переменная **у-скорость** имеет значение **0**, а затем скрипт запускает цикл **Всегда**. Положение по оси **у** (вертикальное положение) спрайта **Кот** изменяется с помощью переменной **у-скорость**, и значение переменной **у-скорость** меняется на **-2**. По мере прохождения этого программного цикла

положение по оси *y* будет меняться все быстрее и быстрее, что заставляет кота падать также быстрее и быстрее.

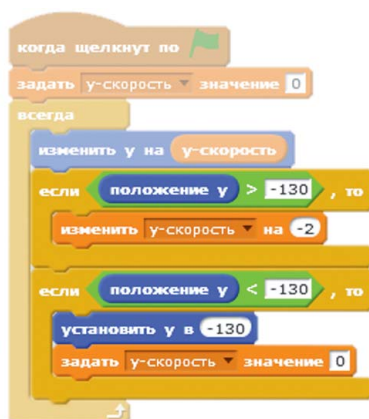


## КОНТРОЛЬНАЯ ТОЧКА

Прежде чем нажать кнопку в виде зеленого флага, перетащите спрайт **Кот** в верхнюю часть сцены. При нажатии кнопки в виде зеленого флага обратите внимание, что кот начинает падать. Если вы хотите, чтобы кот снова упал, нажмите кнопку в виде красного знака остановки, переместите спрайт **Кот** обратно в верхнюю часть сцены и снова нажмите кнопку в виде зеленого флага. Сохраните вашу программу.

## 2. Добавление кода уровня земли

В настоящий момент кот падает. Но мы хотим, чтобы, приземляясь, кот останавливался. Давайте добавим код к спрайту **Кот**, чтобы он выглядел следующим образом:



В этом коде мы устанавливаем положение по оси *y* уровня земли, введя значение  $-130$ . Если положение по оси *y* спрайта кота больше (выше) уровня земли, то **у-скорость** изменяется на  $-2$  и спрайт **Кот** будет падать. Наконец,

спрайт **Кот** будет падать ниже точки  $-130$ , а его положение по оси  $y$  будет меньше (ниже) уровня земли. Когда это произойдет, спрайт **Кот** будет переброшен на уровень земли  $-130$ , и **у-скорость** вернется обратно к  $0$ , чтобы остановить падение спрайта.



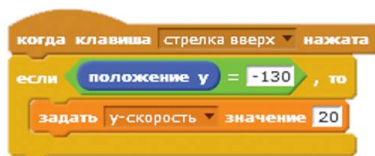
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Перетащите кота с помощью мыши, и отпустите. Убедитесь, что кот падает на землю, а не вылетает за край сцены. Вы можете экспериментировать с различным размещением уровня земли путем изменения значения  $-130$  на другое число. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

Когда вы закончите тестирование вашей программы, скройте переменную **у-скорость** со сцены, сбросив флажок рядом с ее именем в блоке в оранжевой категории **Данные**.

## 3. Добавление кода прыжков к спрайту **Кот**

После добавления кода силы тяжести в спрайт **Кот** будет легко «научить» кота подпрыгивать. Добавьте код, показанный на следующем рисунке, в спрайт **Кот**:



Теперь, когда вы нажимаете клавишу  $\uparrow$ , переменной **у-скорость** присваивается положительное значение  $20$ , что заставляет спрайт **Кот** подпрыгивать вверх. Но значение переменной **у-скорость** все равно будет меняться на  $-2$  при каждом прохождении цикла. Поэтому, хотя сна-

чала кот и подпрыгивает на 20, после прохождения цикла он будет на 18, затем 16 и так далее. Обратите внимание на то, что блок **Если то** проверяет, находится ли спрайт **Кот** на земле. Это нужно для того, чтобы кот мог подпрыгнуть только с земли и не мог этого сделать, находясь в воздухе!



Когда переменная **у-скорость** имеет значение 0, спрайт **Кот** находится в верхней точке прыжка. После этого за каждое прохождение цикла значение переменной **у-скорость** изменяется на -2, и спрайт продолжает падать, пока не упадет на землю. Попробуйте поэкспериментировать с различными числами для блоков **Задать у-скорость значение** и **Изменить у-скорость на**. Выясните, как сделать, чтобы кот прыгал выше или ниже (но всегда над землей), или сделать силу тяжести сильнее или слабее.



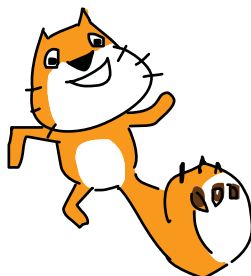
## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Нажмите клавишу  $\uparrow$  и убедитесь, что кот подпрыгивает вверх и падает вниз. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

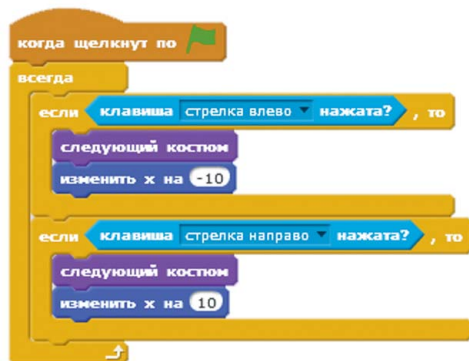
## 6 ОБУЧАЕМ КОТА ПЕРЕМЕЩЕНИЮ ВЛЕВО И ВПРАВО

Теперь давайте добавим код, который будет описывать то, как кот ходит, чтобы игрок мог управлять котом с помощью клавиатуры.



## 4. Добавление кода ходьбы к спрайту Кот

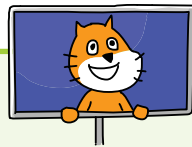
Добавьте код, показанный на следующем рисунке, в нижней части скрипта спрайта **Кот**:



Внутри цикла **Всегда** программа проверяет, нажата ли клавиша ← или →. Если да, спрайт **Кот** переключается на следующий костюм и меняет положение по оси x на -10 (перемещается влево) или 10 (движется вправо). Спрайт **Кот** содержит два костюма, которые можно увидеть, перейдя на вкладку **Костюмы** над областью блоков. Быстрое переключение между двумя костюмами с использованием блока **следующий костюм** позволяет создать впечатление, что кот движется.



### КОНТРОЛЬНАЯ ТОЧКА



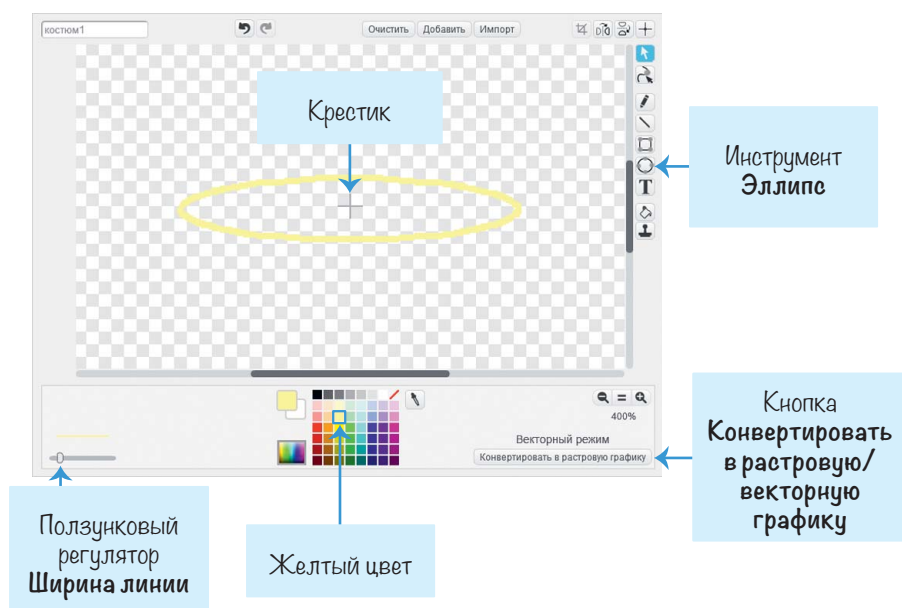
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Нажимайте клавиши ← и → и убедитесь, что кот движется в правильном направлении. Если при нажатии кнопки ← кот идет назад — это именно то, что нужно. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## В СОЗДАЕМ ЛЕТАЮЩЕЕ БАСКЕТБОЛЬНОЕ КОЛЬЦО

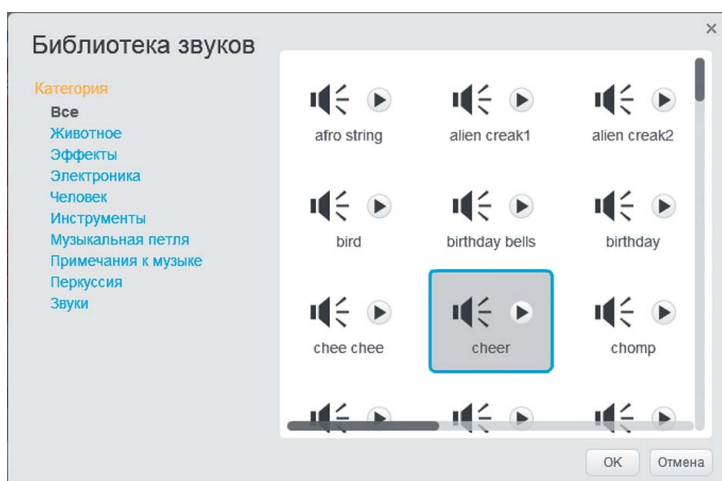
Теперь, когда спрайт **Кот** готов, давайте перейдем к следующему спрайту, необходимому в нашей игре: баскетбольному кольцу.

### 5. Создание спрайта кольца

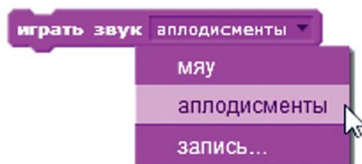
Нажмите кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**. Перед тем как начать рисовать, нажмите кнопку **Конвертировать в векторную графику** в правом нижнем углу. Кнопки инструментов для рисования отобразятся в правой части графического редактора. (Векторный режим позволяет рисовать формы целиком, а не отдельные пиксели изображения.) Выберите желтый цвет и, используя инструмент **Эллипс**, нарисуйте кольцо. Вы также можете двигать ползунковый регулятор **Ширина линии** в нижнем левом углу, чтобы сделать линию кольца толще. Убедитесь, что крестик графического редактора находится в центре кольца.



Присвойте созданному спрайту имя **Кольцо** на панели информации. Давайте сделаем так, чтобы спрайт **Кольцо** проигрывал звук аплодисментов, когда игрок попадает в кольцо, поэтому загрузите звук **Cheer**. Перейдите на вкладку **Звуки** в верхней части области блоков, а затем нажмите кнопку **Выбрать звук из библиотеки**, которая находится под надписью **Новый звук**. В открывшемся окне **Библиотека звуков** выберите звук **Cheer** и нажмите кнопку **ОК**. Для удобства вы можете переименовать звук, присвоив ему имя **Аплодисменты**.

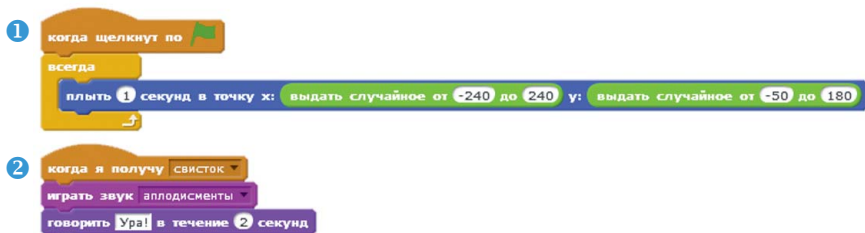


Звук **Аплодисменты** теперь появится в качестве варианта для воспроизведения в блоке **Играть звук**, который вы добавите к спрайту **Кольцо**.



Добавьте код, показанный на следующем рисунке, в спрайт **Кольцо**, чтобы сделать его летающим в верхней половине сцены. Вам нужно создать сообщение, выбрав в раскрывающемся списке блока **Когда я получу пункт Новое сообщение**. Назовите новое сообщение **Свисток**.





Скрипт 1 программирует кольцо на перемещение в новое положение каждую секунду. Играть в игру с движущимся кольцом намного сложнее! Скрипт 2 воспроизводит звук аплодисментов и отображает текст «Ура!», когда получено сообщение **Свисток**. Позже мы сделаем так, чтобы спрайт **Баскетбол** транслировал это сообщение, когда будет создаваться корзина.

## 6. Создание хитбокса

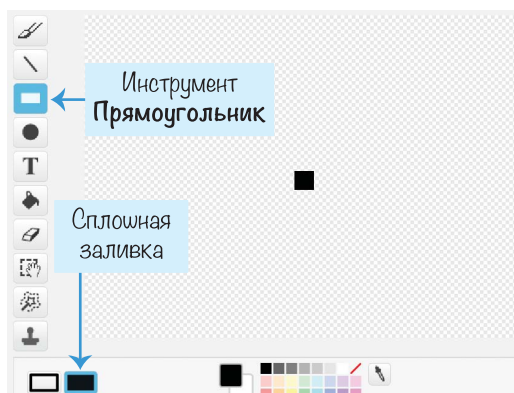
Теперь давайте подумаем о том, как создать код, который определяет, попал или нет игрок в кольцо. Мы могли бы написать программу, которая проверяет, *коснулся* ли мяч кольца или нет. Но поскольку кольцо довольно широкое, то в качестве попадания в кольцо придется рассматривать и касание кольца краем мяча. А нам нужно, чтобы бросок засчитывался только в том случае, если баскетбольный мяч проходит через середину кольца. Давайте подумаем, как решить эту задачу.

Если программа будет проверять, коснулся ли мяч кольца, бросок будет засчитан. Но в баскетболе не такие правила.

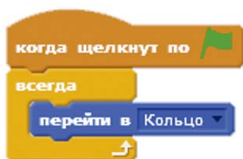


Вместо этого вы можете создать *хитбокс*. Термином «хитбокс» при создании игр обозначается прямоугольная область, определяющая, столкнулись ли два игровых объекта друг с другом. Мы сделаем спрайт **Хитбокс**. Создайте новый спрайт, нажав кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**. Нарисуйте маленький черный квадрат в середине крестика, используя инстру-

мент Прямоугольник и выбрав сплошной вариант заливки. Назовите этот спрайт **Хитбокс**. Спрайт **Хитбокс** будет выглядеть следующим образом:



Добавьте код, показанный на следующем рисунке, к спрайту **Хитбокс**:



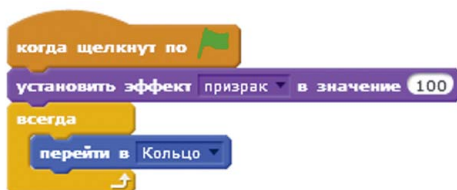
Спрайт **Хитбокс** теперь будет следовать за спрайтом **Кольцо**, независимо от того, куда он переместится.

На шаге 9 мы напишем программу, которая в качестве попадания будет учитывать только касание баскетбольного мяча со спрайтом **Хитбокс**, а не со спрайтом **Кольцо**. Мяч должен быть гораздо ближе к середине кольца, чтобы бросок считался успешным!



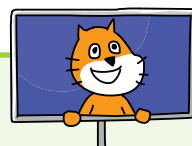
Черный квадратик в центре кольца выглядит довольно странно, поэтому давайте сделаем спрайт **Хитбокс** невидимым.

димым. Добавьте блок **Установить эффект призрак в значение** к спрайту **Хитбокс** и установите значение 100.



Существует различие между блоком **Спрятаться** и **Установить эффект призрак в значение 100**. Если вы используете блок **Спрятаться**, чтобы сделать спрайт **Хитбокс** невидимым, соприкасающиеся блоки никогда не обнаружат, что мяч коснулся спрайта **Хитбокс**, и игрок никогда не забросит мяч в кольцо. Блок **Установить эффект призрак в значение 100** делает спрайт невидимым, но этот блок позволяет соприкасающимся блокам обнаружить присутствие спрайта **Хитбокс**.

## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что кольцо скользит по сцене, и прямоугольный хитбокс всегда находится в центре кольца. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## Г ОБУЧАЕМ КОТА БРОСАТЬ МЯЧ В КОЛЬЦО

В этой части вы добавите в игру баскетбольный мяч, который кот будет бросать. Как и кот, мяч будет подчиняться закону тяготения и падать на землю.

## 7. Создание спрайта баскетбольного мяча

Нажмите кнопку **Выберите спрайт из библиотеки** рядом с надписью **Новый спрайт**, чтобы открыть окно **Библиотека спрайтов**. Выберите пункт **Basketball** и нажмите кнопку **ОК**. Для удобства вы можете переименовать спрайт, присвоив ему имя **Баскетбол**.

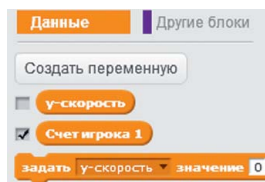


Затем перейдите на вкладку **Звуки** в верхней части области блоков и нажмите кнопку **Выбрать звук из библиотеки** рядом с надписью **Новый звук**, чтобы открыть окно **Библиотека звуков**. Выберите звук **Рор** и нажмите кнопку **ОК**. Для удобства вы можете переименовать его, присвоив имя **Стук**. Перейдите на вкладку **Скрипты** в верхней части области блоков, чтобы снова отобразить область скриптов.

Теперь переходим к оранжевой категории **Данные**. Вы создадите две переменные. Нажмите кнопку **Создать переменную**. Присвойте переменной имя **у-скорость** и прежде, чем нажмете кнопку **ОК**, убедитесь, что выбран вариант **Только для этого спрайта**. Поскольку эта переменная используется только для этого спрайта, переменная **у-скорость** спрайта баскетбольного мяча отделена от переменной **у-скорость** спрайта кота. Несмотря на то, что они имеют одинаковое имя, это две разные переменные.

Нажмите кнопку **Создать переменную** еще раз, чтобы создать еще одну переменную с именем **Счет игрока 1**, но на этот раз установите переключатель в положение **Для всех спрайтов**. (Имя переменной **Счет игрока 1** начина-

ется с прописной буквы, потому что ее будет видно на сцене. Сбросьте флажок напротив имени переменной **у-скорость**, чтобы ее не было видно на сцене.) В оранжевой категории **Данные** появятся новые блоки переменных.



## 8. Добавление кода для спрайта **Баскетбол**

После того как вы добавили звук **Стук** и две переменные, добавьте код, показанный на следующем рисунке, к спрайту **Баскетбол**:



Скрипт ① гарантирует, что игрок начинает, имея 0 очков, и скрывает спрайт **Баскетбол** в начале игры.

Скрипт ② использует код, аналогичный коду спрайта **Кот**. Когда игрок нажимает клавишу **Пробел**, баскетбольный мяч появляется перед котом и начинает двигаться вперед. Код присваивает переменной **у-скорость** спрайта

**Баскетбол** значение – положительное число, аналогично тому, как переменной **у-скорость** спрайта **Кот** присваивается положительное число, когда кот подпрыгивает. Таким образом, кот бросает мяч.

Блок **Повторять пока не положение**  $y < -130$  будет опускать спрайт **Баскетбол**, пока он не достигнет земли. Достигнув земли, баскетбольный мяч будет снова скрыт до того момента, когда в следующий раз игрок нажмет клавишу **Пробел**.



### КОНТРОЛЬНАЯ ТОЧКА

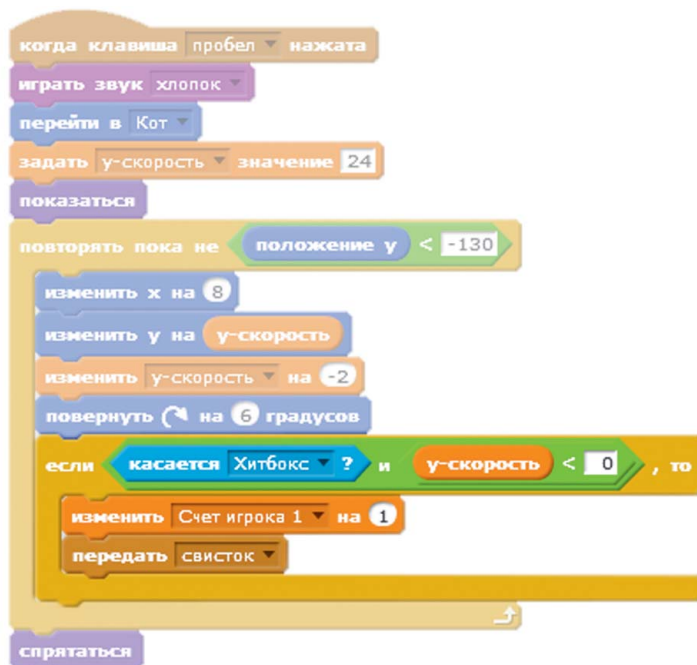
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Нажмите клавишу **Пробел**, чтобы заставить кота бросить баскетбольный мяч. Убедитесь, что мяч исчезает при соприкосновении с землей. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## 9. Учет успешных бросков

На этом этапе мы добавим код, который проверяет, коснулся ли спрайт **Баскетбол** спрайта **Хитбокс**. Он отвечает за то, чтобы счет игрока увеличивался на одно очко (увеличивается значение переменной **Счет игрока 1**) в случае, если мяч попал в кольцо. Тем не менее есть одна загвоздка: бросок не должен засчитываться, если баскетбольный мяч проходит через кольцо *снизу вверх*.

Имейте в виду, что если значение переменной **у-скорость** положительное, то блок **Изменить у на у-скорость** будет направлять движение спрайта **Баскетбол** **вверх**. Если значение переменной **у-скорость** равно 0, то спрайт **Баскетбол** не движется вверх или вниз. Но если значение переменной **у-скорость** отрицательное, то спрайт **Баскетбол** будет падать.

Таким образом, вы добавите дополнительное условие **Если, то** в код спрайта **Баскетбол**. Счет будет увеличиваться, только если мяч касается спрайта **Хитбокс** (применен блок **Касается Хитбокс ?**) и падает вниз (**у-скорость < 0**).



Блок **И** сочетает в себе два условия. В программе Scratch, чтобы запустить блоки кода внутри блока **Если, то**, *оба* эти условия должны быть истинными. Недостаточно только того, чтобы спрайт **Баскетбол** коснулся спрайта **Хитбокс** *или* значение переменной **у-скорость** было меньше 0. Чтобы бросок был засчитан, оба эти условия – и **Касается Хитбокс ?**, и **у-скорость < 0** – должны быть истинными. Если эти условия истинны, то значение переменной **Счет игрока 1** увеличится на 1, и появится сообщение **Свисток**.



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Попробуйте побросать мяч в кольцо. Значение переменной **Счет игрока 1** должно увеличиваться, только если баскетбольный мяч проходит через центр кольца и затем падает. Спрайт **Кольцо** должен также выводить текст «Ура!» и воспроизводить звук аплодисментов, когда бросок завершается успехом. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## 10. Исправление ошибки в счете

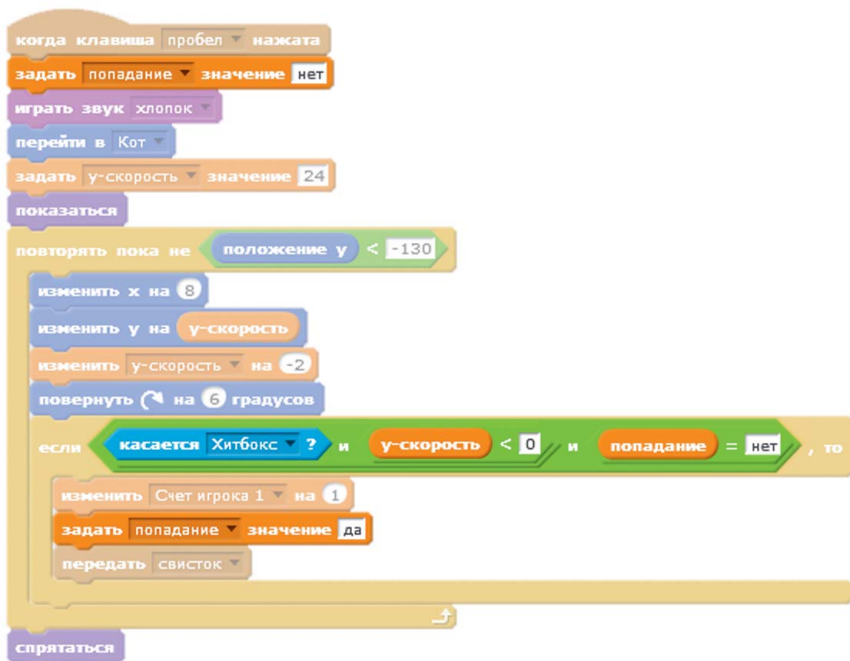
Вы заметили, что **Счет игрока 1** увеличивается на несколько очков за один бросок? Это *ошибка*, которая является проблемой. С ней программа может вести себя неожиданным образом. Нам нужно еще раз внимательно посмотреть на код, чтобы выяснить, почему это происходит.

Циклический блок **Повторять пока не** продолжает работать, пока мяч не упадет на землю. Поскольку весь блок кода используется для каждого броска, блок **Повторять пока не** несколько раз проверяет, коснулся ли спрайт **Баскетбол** спрайта **Хитбокс** и падает ли вниз. А правильно — чтобы **Счет игрока 1** увеличивался только в первый раз.

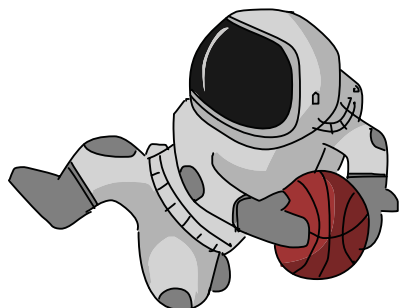
Эту ошибку нужно исправить, создав новую переменную, которая отслеживает первое попадание баскетбольного мяча в кольцо для каждого броска. После этого игрок будет получать очко только один раз за бросок.

Перейдите в оранжевую категорию **Данные** в верхней части области блоков, а затем нажмите кнопку **Создать переменную**. Присвойте переменной имя **Попадание** и установите переключатель в положение **Только для этого спрайта**. Затем измените код спрайта **Баскетбол** следующим образом:



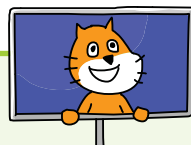


Переменной **Попадание** присваивается значение Нет, когда игрок нажимает клавишу «Пробел» в первый раз. Это правильно, так как игрок еще не попал в кольцо, когда мяч брошен и только летит. Также мы будем использовать блок **И**, чтобы добавить еще одно условие в код, которое проверяет, попал ли мяч в кольцо. Теперь попадание фиксируется и код в блоке **Если, то** запускается, когда истинны три условия:



1. Спрайт **Баскетбол** коснулся спрайта **Хитбокс**.
2. Значение переменной **у-скорость** отрицательно (баскетбольный мяч падает вниз).
3. Переменной **Попадание** присвоено значение **Нет**.

Когда спрайт **Баскетбол** впервые обнаруживает удачный бросок, он увеличивает значение переменной **Счет игрока 1** на 1 и присваивает переменной **Попадание** значение **Да**. При последующих проверках этого броска, значение переменной **Попадание** не будет равно **Нет**, поэтому бросок больше не будет учитываться. Значение переменной **Попадание** снова станет **Нет**, когда игрок нажмет клавишу **Пробел**, чтобы бросить мяч.



## КОНТРОЛЬНАЯ ТОЧКА

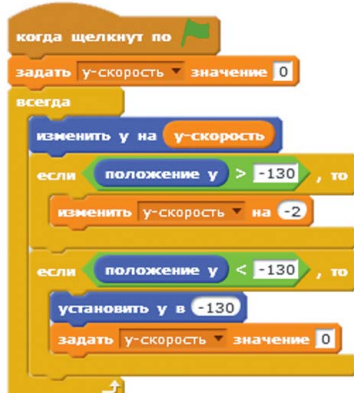
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Попробуйте сделать несколько бросков. Убедитесь, что значение переменной **Счет игрока 1** увеличивается только на 1 очко для каждого попадания в кольцо. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

# ЗАКОНЧЕННАЯ ПРОГРАММА

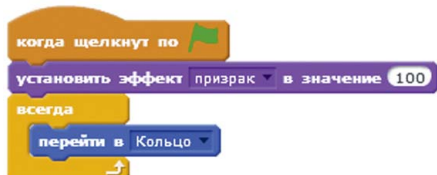
Ниже показана финальная версия кода. Если ваша программа не работает правильно, сверьте код вашей программы с кодом, приведенным ниже.



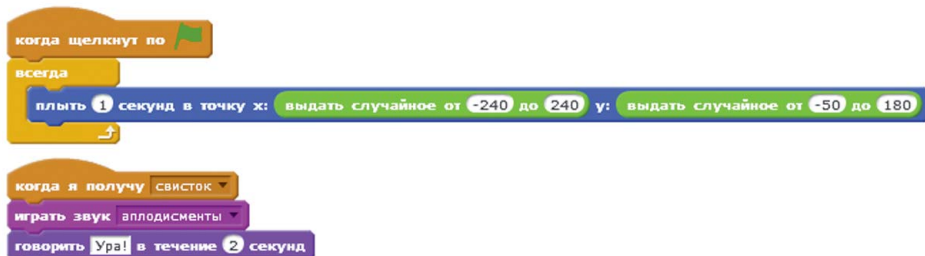
Кот



Хитбокс

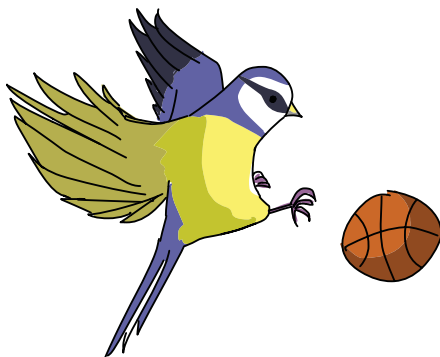
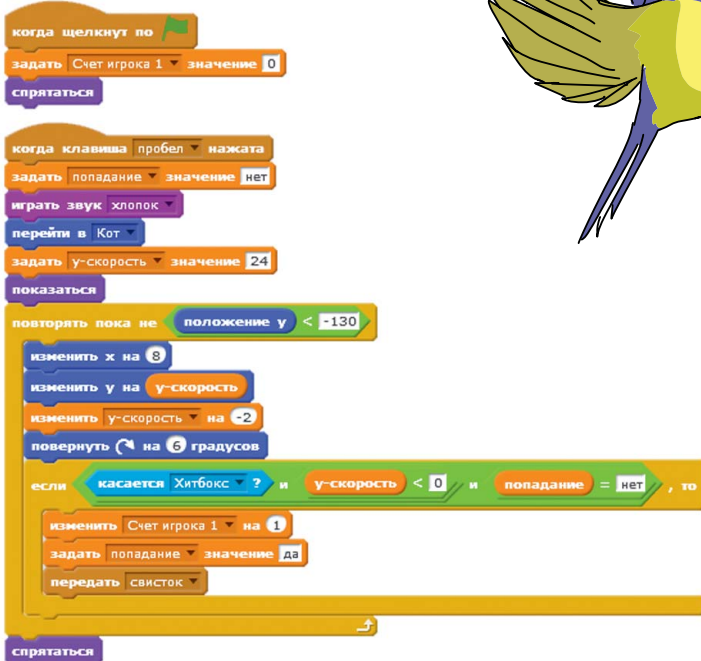


Кольцо





Баскетбол



## ВЕРСИЯ 2.0: РЕЖИМ ДЛЯ ДВУХ ИГРОКОВ

Давайте усовершенствуем игру «Баскетбол», добавив второго игрока. Это будет легко, так как код второго игрока будет почти идентичен коду первого игрока.

### Дублируем спрайты Кот и Баскетбол

Продублируйте спрайты **Кот** и **Баскетбол**, щелкнув правой кнопкой мыши по ним в области спрайтов и выбрав команду **Дублировать** в контекстном меню. Выберите новый спрайт **Кот2** и перейдите на вкладку **Костюмы** в верхней части области блоков. С помощью инструмента **Заливка** покрасьте кота в синий цвет.

Затем нажмите кнопку **Отразить слева направо** в правом углу графического редактора, чтобы второй игрок бросал мяч в кольцо с другой стороны.



## Меняем код спрайта **Кот2**

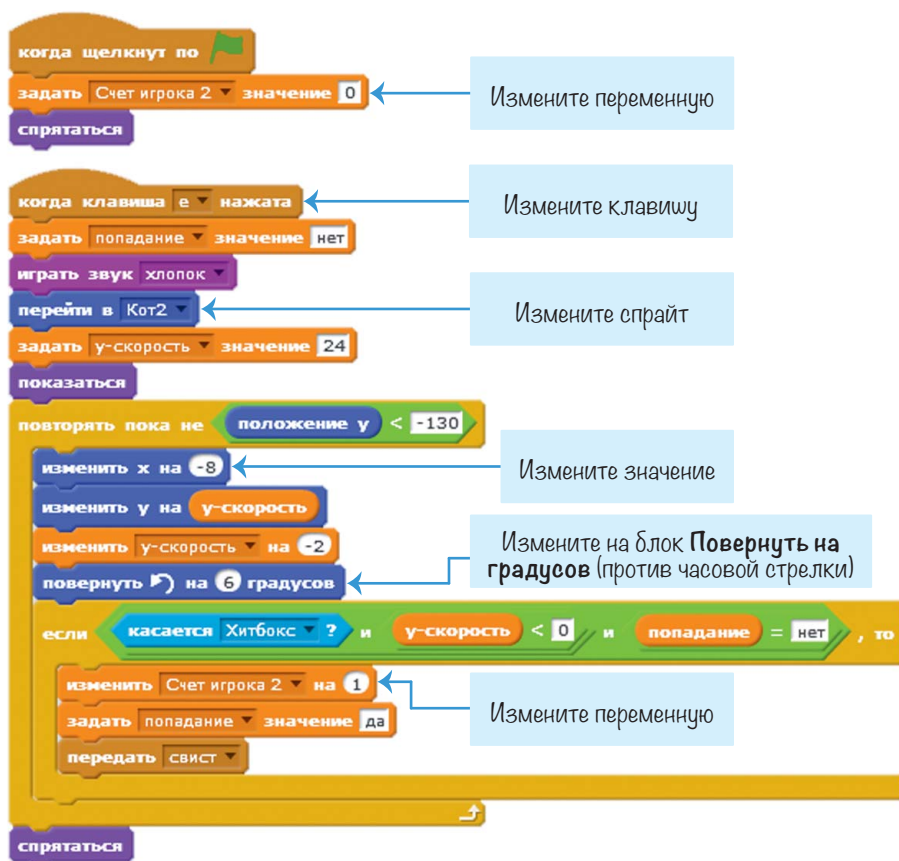
Измените код нового спрайта **Кот2**, чтобы он соответствовал коду на рисунке ниже:



Теперь два разных игрока смогут играть, используя одну клавиатуру. Каждый кот будет управляться разными клавишами.

## Меняем код спрайта Баскетбол2

Второму игроку нужна отдельная переменная для ведения счета. Перейдите в оранжевую категорию **Данные**, нажмите кнопку **Создать переменную**, присвойте этой переменной имя **Счет игрока 2** и установите переключатель в положение **Для всех спрайтов**. Измените код в новом спрайте **Баскетбол2** так, как показано на рисунке ниже.

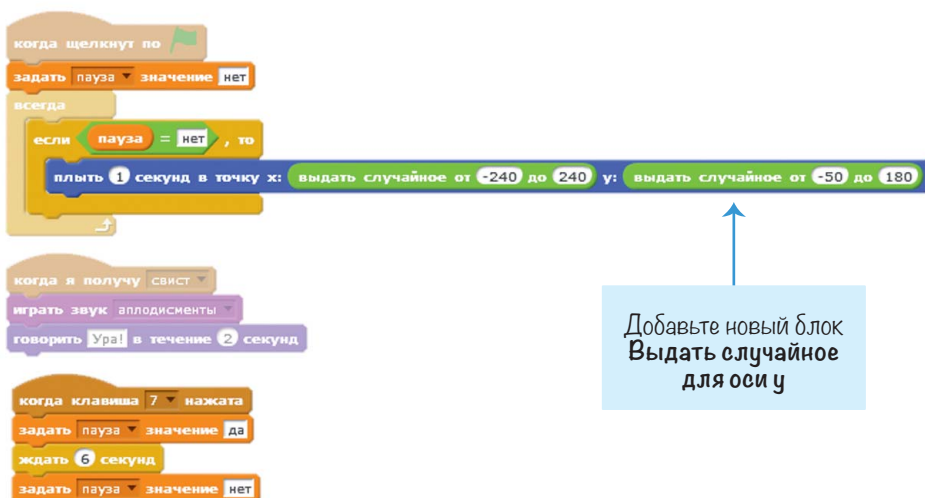


Этот код отвечает за то, чтобы брошенный вторым игроком баскетбольный мяч летел влево, и за изменение значения переменной **Счет игрока 2** для второго игрока. Вы можете переименовать переменную, перейдя в оранжевую категорию **Данные**, щелкнув правой кнопкой мыши по блоку переменной и выбрав пункт **Переименовать переменную**.

## ЧИТ-РЕЖИМ: ОСТАНОВКА КОЛЬЦА

Движущееся баскетбольное кольцо представляет собой довольно сложную мишень для попадания. Давайте добавим чит, который будет задерживать кольцо на месте в течение нескольких секунд, когда игрок нажимает клавишу 7.

Выберите спрайт **Кольцо** и нажмите кнопку **Создать переменную**, чтобы создать новую переменную с параметром **Только для этого спрайта** и именем **Пауза**. Затем измените код для спрайта **Кольцо** в соответствии с кодом, приведенным ниже.





## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Нажмите клавишу **7** и убедитесь, что кольцо не меняет своего положения в течение 6 секунд. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## ЗАКЛЮЧЕНИЕ

В этой главе вы создали игру, в которой:

- ▶ учитываются законы земного притяжения и реалистичные падения;
- ▶ используется боковая перспектива вместо вида сверху;
- ▶ используются переменные для отслеживания счета, скорости падения и попадания в кольцо;
- ▶ используется хитбокс, определяющий успешность броска.

Использовать силу тяжести в этой программе было довольно просто. К тому времени, когда вы доберетесь до главы 9, вы сможете создать продвинутую игру-платформер с более сложными прыжками и падениями. Но для начала потребуется еще попрактиковаться со Scratch. В главе 5 вы создадите игру с боковым режимом просмотра и используете функцию клонирования, чтобы дублировать спрайты десятки раз.



## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно посмотреть в конце книги.)

1. Чем игра с боковым режимом просмотра (например, «Баскетбол») отличается от игры с видом сверху (например, игра «Бегущий в лабиринте»)?
2. Что может хранить переменная?
3. В чем разница между режимами **Только для этого спрайта** и **Для всех спрайтов**?
4. Как сделать спрайт прыгающим?
5. Когда в игре «Баскетбол» прыгает кот, что удерживает его от бесконечного полета вверх?
6. В чем разница между блоками **Плыть** и **Перейти** в точку с определенными координатами  $x$  и  $y$ ?
7. Как запустить код внутри блока **Если, то**, когда истинны *два* условия?



5

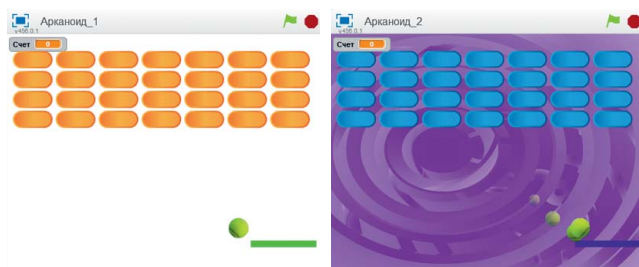
## АРКАНОИД

**В**ы когда-нибудь видели игру «Арканойд»? Игрок управляет платформой-ракеткой в нижней части экрана, отбивая шарик, который разбивает кирпичики в верхней части экрана. Игрок проигрывает, когда шарик пролетает мимо ракетки. Хотя эта игра достаточно проста для программирования, она быстро может наскучить.

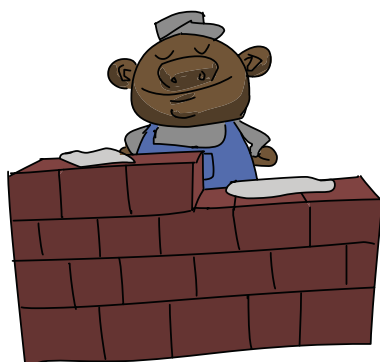
В этой главе вы узнаете несколько приемов, с помощью которых можно сделать игру более красочной и интересной, добавив анимацию и эффекты.

Вы будете использовать итеративный процесс: сначала сделаете базовую игру, а затем примените к ней небольшие улучшения. Результатом такого подхода будет профессионально выглядящая игра, которая удивит посетителей сайта Scratch.

На следующем рисунке показаны базовая версия игры «Арканоид» и улучшенная.

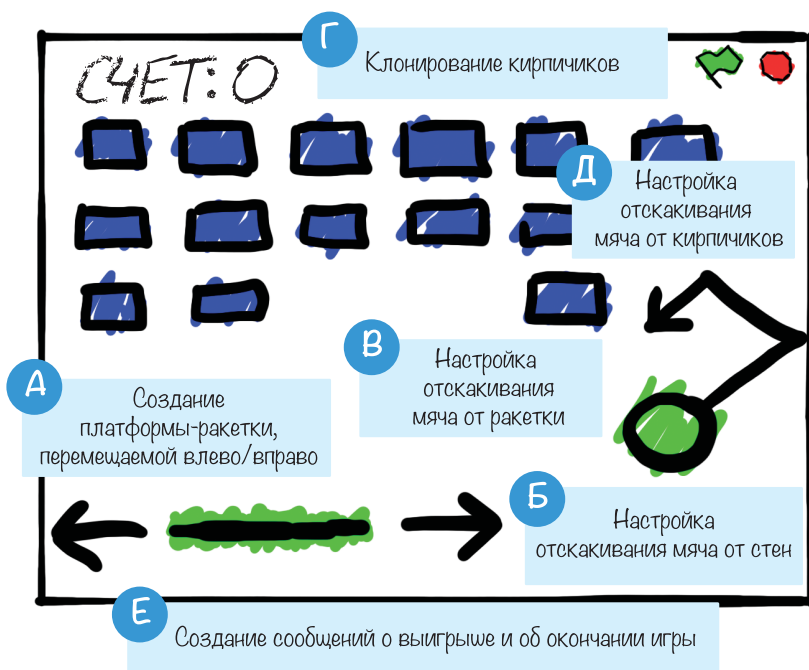


Перед тем как приступить к работе, взгляните на финальную версию игры, которая доступна по ссылке [scratch.mit.edu/studios/4188596/](https://scratch.mit.edu/studios/4188596/).



## ЭСКИЗ ПРОЕКТА

Перед началом работы нужно представить, как будет выглядеть окончательная версия игры, и нарисовать ее. Эскиз игры «Арканоид» должен выглядеть примерно так, как показано на следующем рисунке.



Если вы хотите сэкономить время, можете начать с файла скелета проекта, который называется *Глава 05/Скелет\_проекта.sb2* и находится в запакованном файле с примерами, который вы скачали ранее по ссылке [https://eksmo.ru/files/Scratch\\_Sweigart.zip](https://eksmo.ru/files/Scratch_Sweigart.zip). В файл скелета проекта уже загружены все спрайты, так что вам нужно всего лишь перетянуть блоки кода в каждый спрайт.

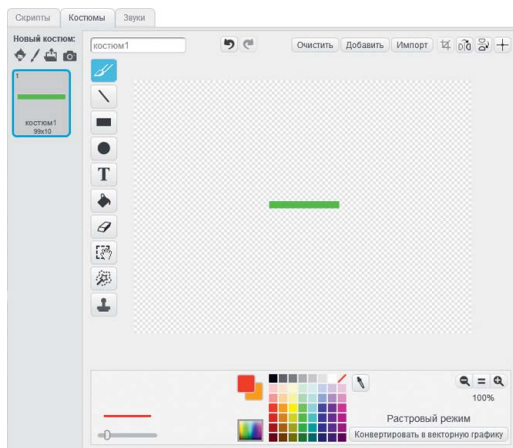
## А СОЗДАНИЕ ПЛАТФОРМЫ-РАКЕТКИ, ПЕРЕМЕЩАЕМОЙ ВЛЕВО/ВПРАВО

Игрок будет управлять платформой-ракеткой, двигая мышкой. Мяч отскакивает от ракетки к кирпичикам, но игрок проигрывает, если мяч летит мимо ракетки.

### 1. Создание спрайта платформы

В этой игре нам не нужен рыжий кот, так что щелкните правой кнопкой мыши по спрайту кота в области спрайтов и выберите команду **Удалить** в контекстном меню.

Затем нарисуйте спрайт, нажав кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**. Когда появится графический редактор, с помощью инструмента **Прямоугольник** нарисуйте прямоугольник, подобный тому, который изображен на рисунке.

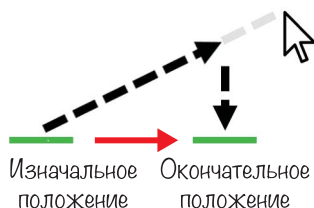


Для своей платформы-ракетки я выбрал зеленый цвет, а вы можете выбрать цвет по своему усмотрению. Чем короче ракетка, тем труднее поймать ею мяч. Позднее вы сможете экспериментировать с различными размерами ракетки, чтобы сделать игру легче или труднее. Нажмите кнопку **i**, чтобы открыть панель информации, и присвойте этому спрайту имя **Платформа**.

Затем добавьте код, показанный на следующем рисунке, чтобы запрограммировать спрайт **Платформа** на движение вслед за мышью вдоль нижней части сцены:



Спрайт **Платформа** постоянно движется на 10 шагов за мышью, при этом его положение по оси y остается равным -140.

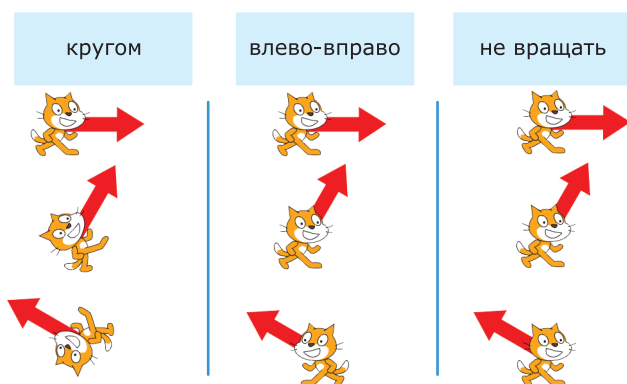


Горизонтальное движение платформы является результатом двух действий: следования за мышью на 10 шагов и установки положения платформы по оси  $x$  равным  $-140$

Спрайт **Платформа** будет двигаться только влево и вправо, потому что его положение по оси  $y$  не меняется и всегда определено в нижней части рабочей области ( $-140$ ).

## ЭТО ИНТЕРЕСНО: СТИЛЬ ВРАЩЕНИЯ

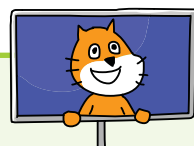
Стиль вращения определяет, как выглядит спрайт, когда меняет направление. Существует три типа поворотов: **Кругом**, **Влево-вправо** и **Не вращать**.



Когда спрайт настроен на вращение «кругом», он будет обращен именно в ту сторону, куда задано его направление. Однако такой стиль вращения не будет работать для игры с боковым режимом просмотра (например, для игры «Баскетбол» в главе 4), потому что спрайт перевернется вверх ногами, если его направление будет указывать налево. Для такого рода игр вы будете использовать стиль вращения «влево-вправо». Спрайт

будет обращен только на 90 градусов (вправо) или  $-90$  градусов (налево), в зависимости от того, какое положение ближе всего к направлению спрайта. Если вы не хотите, чтобы спрайт вращался даже при смене его направления, установите стиль вращения «не вращать».

Поскольку мы изменяем направление спрайта **Платформа**, нам необходимо установить стиль вращения спрайта с помощью блока **Стиль вращения**. Спрайт **Платформа** запрограммирован поворачиваться и следовать за мышью, но для нашей игры нужно, чтобы спрайт всегда выглядел плоским и горизонтальным, поэтому выбран стиль вращения «не вращать».



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Подвигайте мышью по сцене, и вы увидите, как спрайт Платформа следует за ней. Убедитесь, что платформа-ракетка все время остается в нижней части сцены. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

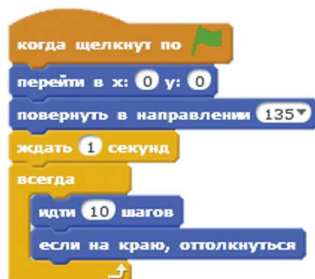
## 5 НАСТРОЙКА ОТСКАКИВАНИЯ МЯЧА ОТ СТЕН

В библиотеке Scratch есть несколько спрайтов мяча, которые можно бы использовать. Для этой игры мы применим спрайт **Tennis Ball**.

### 2. Создание спрайта мячика

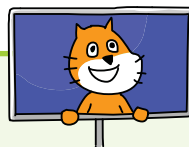
Нажмите кнопку **Выберите спрайт из библиотеки** рядом с надписью **Новый спрайт** и выберите спрайт **Tennis Ball**

в открывшемся окне **Библиотека спрайтов**. Для удобства вы можете переименовать его, присвоив имя **Мячик**. Добавьте код, показанный на следующем рисунке:



Когда начинается игра, спрайт **Мячик** появляется в положении (0, 0) в центре сцены; затем спрайт **Мячик** падает вниз и вправо по направлению к спрайту **Платформа**. Далее, в цикле **всегда**, спрайт **Мячик** начинает двигаться. Когда спрайт **Мячик** будет касаться края сцены, он будет отскакивать в новом направлении.

## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что спрайт **Мячик** движется по сцене и отскакивает от краев. Он не будет отскакивать от ракетки, поскольку этот код вы еще не добавили. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.



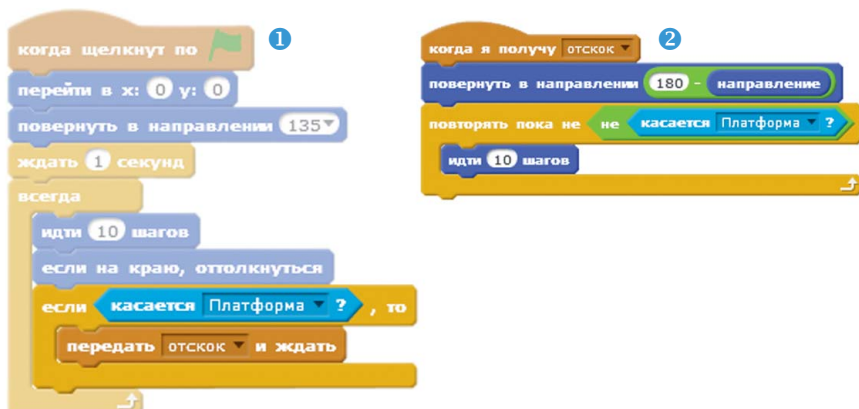
## В НАСТРОЙКА ОТСКАКИВАНИЯ МЯЧИКА ОТ ПЛАТФОРМЫ

На данном этапе создания игры спрайт **Мячик** отскакивает от краев сцены, но не отскакивает от спрайта **Платформа**. Давайте добавим этот код прямо сейчас.



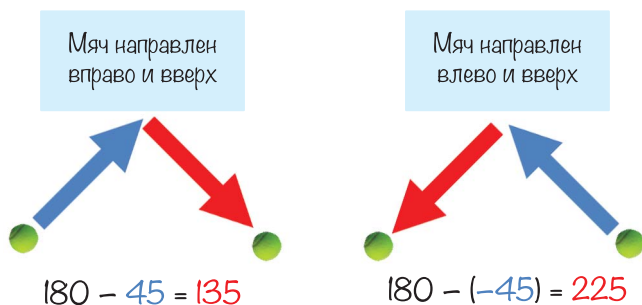
### 3. Добавление кода отскакивания к спрайту теннисного мяча

Добавьте код, показанный на следующем рисунке, к спрайту **Мячик**, чтобы он отскакивал от спрайта **Платформа**. Для этого вам необходимо создать новое сообщение – **Отскок**.



Вы будете использовать сообщение в скрипте ①, чтобы управлять тем, что происходит, когда мяч касается ракетки в скрипте ②.

Код **Повернуть в направлении 180** – направление в скрипте ② может показаться немного сложным, но это простое уравнение вычисляет направление, в котором мяч будет отскакивать, основываясь на текущем направлении движения мяча. Если мяч первоначально направлен вверх и вправо (45 градусов), то, когда он отскакивает от нижней части кирпичика, его новое направление будет вниз и вправо (135 градусов, потому что  $180 - 45 = 135$ ). Если мяч сначала движется вверх и влево (-45 градусов), то, когда он отскакивает от нижней части кирпичика, его новое направление будет вниз и влево (225 градусов, потому что  $180 - (-45) = 225$ ).



Это сообщение вы будете использовать в программе еще раз, когда добавите код, чтобы мяч отскакивал от кирпичиков.

## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что мячик отскакивает от ракетки. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.



## ЭТО ИНТЕРЕСНО: КЛОНИРОВАНИЕ

Блок **Создать клон себя самого** создает дубликат спрайта, который называется *клоном*. Эта функция удобна, когда вы хотите создать много копий одного объекта в вашей игре. Например таких, как множество враждебных персонажей, которые выглядят одинаково, груды монет для игрока, которые он будет собирать, или кирпичики в игре «Арканоид», которые вы должны будете разбить.

Рассмотрим, как работают клоны. Откройте Scratch в новой вкладке браузера и создайте новую программу. Добавьте код, показанный ниже, к спрайту **Кот**:



Скрипт ① программирует спрайт **Кот** отскакивать от границ сцены, точно так же как это делает спрайт **Мячик** в игре «Арканоид». В скрипте ② мы создаем клон снова и снова, каждые 2 секунды. В скрипте ③ используется блок **Когда я начинаю как клон**, чтобы управлять поведением клонированных спрайтов. Как вы думаете, что произойдет, когда вы запустите этот код? Запустите код сейчас, чтобы узнать, правильны ли ваши предположения.

Исходный спрайт **Кот** отскакивает от границ сцены. Каждые 2 секунды создается дубликат спрайта: это клоны. Каждый клон затем начнет вращаться, что запрограммировано в сценарии ③.

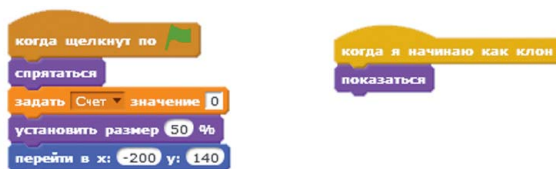
## Г КЛОНИРОВАНИЕ КИРПИЧИКОВ

Пришло время создать для нашей игры большое количество кирпичиков. Сначала вы создадите один спрайт кирпичика, а затем клонируете его с помощью блока Создать клон.

### 4. Создание спрайта кирпичика

Нажмите кнопку **Выберите спрайт из библиотеки** рядом с надписью **Новый спрайт** и выберите спрайт **Button 2** из открывшегося окна **Библиотека спрайтов**. Нажмите кнопку **i**, чтобы открыть панель информации, и присвойте этому спрайту имя **Кирпичик**.

Далее вам нужно создать новую переменную, выбрав оранжевую категорию **Данные** и нажав кнопку **Создать переменную**. Присвойте этой переменной имя **Счет** и установите переключатель в положение **Для всех спрайтов**. Затем добавьте код, показанный на следующем рисунке, к спрайту **Кирпичик**:



В начале игры значение переменной **Счет** равно 0, а это значит, что все очки, полученные в предыдущей игре, сбрасываются. Исходный спрайт скрывается блоком **Спрятаться**, сжимается в размере на 50 процентов и перемещается в верхний левый угол сцены в позицию с координатами (-200, 140). Клоны, которые мы создадим на следующем этапе, будут отображены с помощью блока **Показаться**.

### 5. Клонирование спрайта Кирпичик

Для игры «Арканоид» нужно создать много рядов кирпичиков. Чтобы их сделать, мы будем перемещать исходный спрайт в верхней части экрана, создавая след клонов. До-

бавьте код, показанный на следующем рисунке, к спрайту **Кирпичик**. (Убедитесь, что вы используете блок **Установить x в**, а не блок **Изменить x на**!)



Этот код будет создавать клоны спрайта **Кирпичик** для всех кирпичиков в игре, как показано ниже:

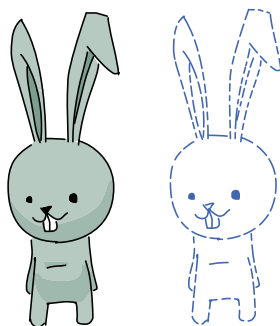


## 4. Спрайт создает четыре ряда клонов

Исходный спрайт перемещается в верхний левый угол сцены с координатами  $(-200, 140)$  ❶. Затем блок **Повторить 7 раз** перемещает спрайт на 65 шагов вправо, создавая клоны себя самого ❷, генерируя таким образом ряд из семи клонов кирпичиков ❸. Блок **Повторить 4** повторяет код, который создает ряд из семи кирпичиков, программируя создание четырех рядов клонов ❹. Семь клонов кирпичиков, умноженные на четыре ряда, в результате составляют 28 клонов. 29-й кирпичик на предыдущем рисунке – это исходный спрайт, не клон, и позже мы его спрячем.

После того как все клоны созданы, исходный спрайт скрывается. Теперь все кирпичики, которые находятся на сцене, – это клоны. Поэтому вам не нужно дублировать код под блоком **Когда я начинаю как клон** для исходного спрайта.

Представьте себе, что вам пришлось бы дублировать спрайты, вместо того чтобы клонировать их. Тогда вам пришлось бы изменить все 28 спрайтов **Кирпичик**, чтобы изменить код. Клонирование экономит много времени!



## А НАСТРОЙКА ОТСКАКИВАНИЯ МЯЧА ОТ КИРПИЧИКОВ

Спрайт **Мячик** отскакивает от краев сцены и спрайта **Платформа**. Теперь давайте сделаем так, чтобы он отскакивал от клонов кирпичиков.

## 6. Добавление кода отскакивания к спрайту Кирпичик

Обновите код для спрайта **Кирпичик**, чтобы он соответствовал следующему рисунку:



Когда спрайт **Мячик** ударяется о спрайт **Кирпичик**, спрайт **Кирпичик** передает сообщение **Отскок**, которое приносит в игру код спрайта **Мячик**. Направление движения мяча изменяется точно так же, как это происходит, когда он попадает на ракетку. Программа добавляет 1 очко к счету игрока, а затем клон удаляет сам себя.



## КОНТРОЛЬНАЯ ТОЧКА

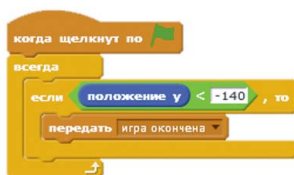
Нажмите кнопку в виде зеленого флага, чтобы проверить уже имеющийся фрагмент кода. Убедитесь, что верхняя часть сцены заполняется клонами кирпичиков и клоны исчезают, когда спрайт Мячик ударяется о них и отскакивает. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## Е СОЗДАНИЕ СООБЩЕНИЙ О ВЫИГРЫШЕ И ОБ ОКОНЧАНИИ ИГРЫ

Для этой игры вам нужно еще два спрайта, которые будут появляться в конце игры. Я создал их в графическом редакторе с помощью инструмента **Текст**. Если игрок разобьет все клоны спрайта **Кирпичик**, программа выведет на экран спрайт **Вы выиграли**. Если мячик пролетит мимо платформы, программа выведет на экран спрайт **Игра окончена**.

## 7. Изменение кода спрайта **Мячик**

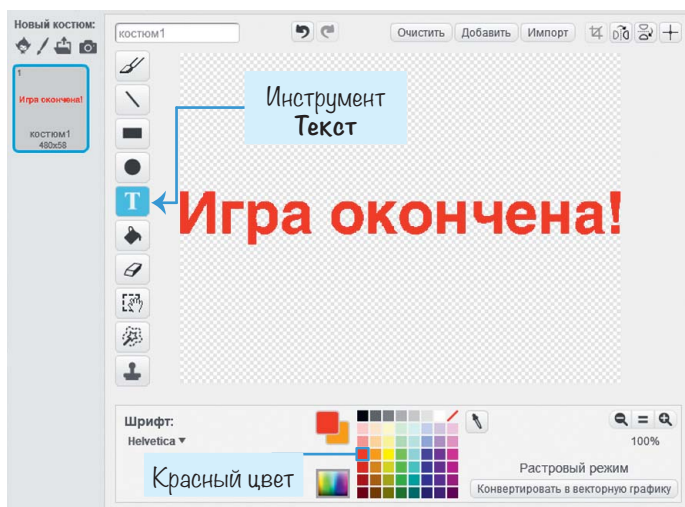
Когда спрайт **Мячик** пролетает мимо спрайта **Платформа** – другими словами, когда значение **у** позиции спрайта **Мячик** составляет менее чем  $-140$ , – игра окончена. После того как игра заканчивается, спрайт **Мячик** должен передать сообщение **Игра окончена**. Добавьте показанный ниже код к спрайту **Мячик**. Он необходим для вывода сообщения **Игра окончена**.



Передача сообщения **Игра окончена** означает, что нужно отобразить спрайт **Игра окончена**. Давайте приступим к следующему спрайту.


## 8. Создание спрайта **Игра окончена**

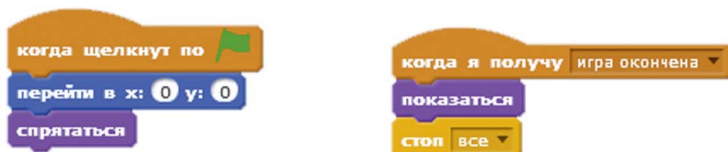
Нажмите кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**. Когда появится графический редактор, воспользуйтесь инструментом **Текст**, чтобы написать текст «Игра окончена!» красным шрифтом.





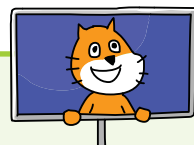
**ПРИМЕЧАНИЕ.** Обратите внимание, что русский текст не поддерживается напрямую в редакторе Scratch, поэтому потребуется воспользоваться сторонним приложением. Для этого перейдите на веб-страницу [scratch.mit.edu/projects/60521212/](https://scratch.mit.edu/projects/60521212/) и нажмите кнопку в виде зеленого флага или клавишу **Пробел**. Введите текст в появившееся поле и нажмите клавишу **Enter** или синюю кнопку справа от поля. Скопируйте текст из второй строки в виде символов типа `Àâñèëèè`, выделив его и нажав сочетание клавиш **Ctrl+C**. Перейдите в редактор Scratch и, выбрав инструмент **Текст** и шрифт **Helvetica**, вставьте скопированный текст с помощью сочетания клавиш **Ctrl+V**. Обратите внимание, что в редакторе Scratch для русских букв поддерживается только шрифт **Helvetica**.

Нажмите кнопку , чтобы открыть панель информации, и присвойте этому спрайту имя **Игра окончена**. Затем добавьте код, показанный на следующем рисунке, к спрайту **Игра окончена**:



Спрайт остается скрытым до тех пор, пока не получит сообщение **Игра окончена**. После этого блок **Стоп все** останавливает все спрайты.

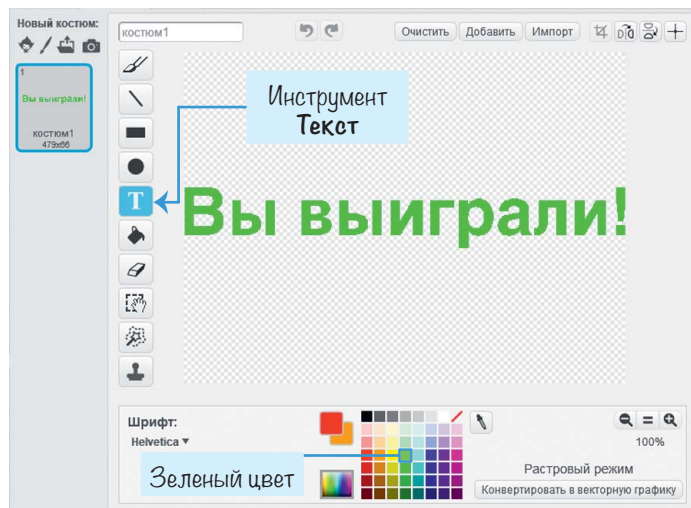
## КОНТРОЛЬНАЯ ТОЧКА



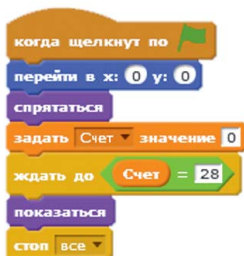
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Пусть мячик пролетит мимо ракетки. Вы увидите, что появится спрайт **Игра окончена** и программа остановится. Сохраните вашу программу.

## 9. Создание спрайта **Вы выиграли**

Нажмите кнопку **Нарисовать новый спрайт**, которая находится рядом с надписью **Новый спрайт**. В графическом редакторе используйте инструмент **Текст** и выберите зеленый цвет, чтобы сделать надпись **Вы выиграли!** Для корректного использования русских букв смотрите начало предыдущего шага (8).

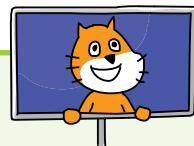


Нажмите кнопку **i**, чтобы открыть панель информации, и присвойте этому спрайту имя **Вы выиграли**. Добавьте код, показанный на следующем рисунке, к спрайту **Вы выиграли**:



Как и в случае со спрайтом **Игра окончена**, спрайт **Вы выиграли** будет скрыт, пока некоторое условие не будет выполнено. В этой игре игрок должен разбить все 28 кирпичиков, чтобы выиграть, так что условие будет **Счет = 28**.

После того как в игре отобразится спрайт **Вы выиграли**, программа прекращает все остальные спрайты с помощью блока **Стоп все**.



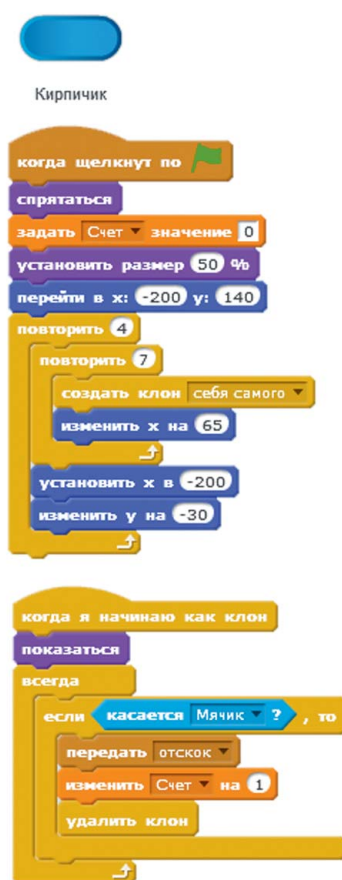
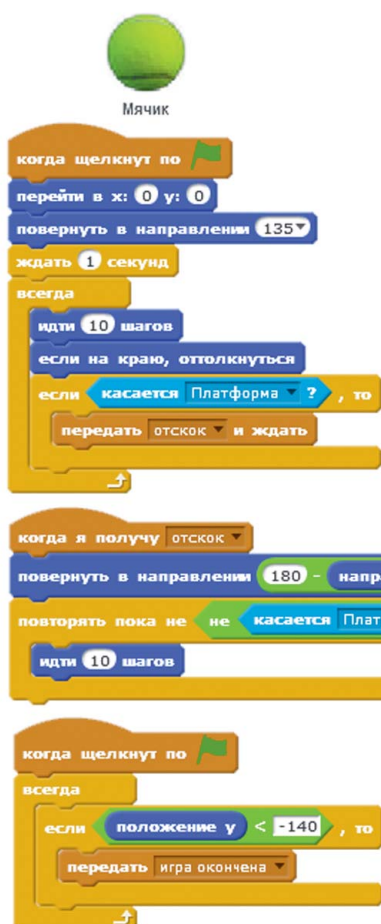
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить созданный фрагмент кода. Убедитесь, что спрайт **Вы выиграли** появляется после того, как все кирпичики разбиты и программа завершает работу. Чтобы выиграть игру быстрее, временно измените блок **Ждать до счет = 28** на **Ждать до счет = 1**. В этом случае вам нужно разбить только один кирпичик, чтобы выиграть. Измените блок обратно на **Ждать до счет = 28**. Сохраните вашу программу.

## ЗАКОНЧЕННАЯ ПРОГРАММА

Окончательный код для всей программы показан на рисунке ниже. Если ваша программа работает неправильно, сверьте свой код с тем, который приведен на рисунке.



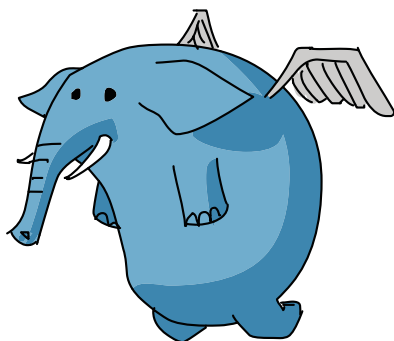


## ВЕРСИЯ 2.0: ПРИДАЕМ ИГРЕ ЛОСК

Ваша игра уже отлично работает. Теперь давайте придадим ей лоск. Множество идей для улучшения игры «Арканоид» пришли из текста «Придай лоск или проиграешь!» Мартина Йонассона и Петри Парго. Слово «лоск» в игровом дизайне означает улучшение деталей игры, чтобы сделать ее более живой и вызывающей эмоции у игроков. Эти приемы могут превратить игру из простой, базовой, в захватывающую и красочную. Улучшенная игра выглядит профессиональнее, чем простая. Вы можете

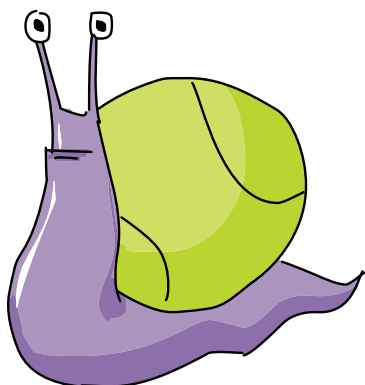
посмотреть упомянутую презентацию об улучшении игры «Арканоид» по ссылке по ссылке [www.nostarch.com/scratchplayground/](http://www.nostarch.com/scratchplayground/).

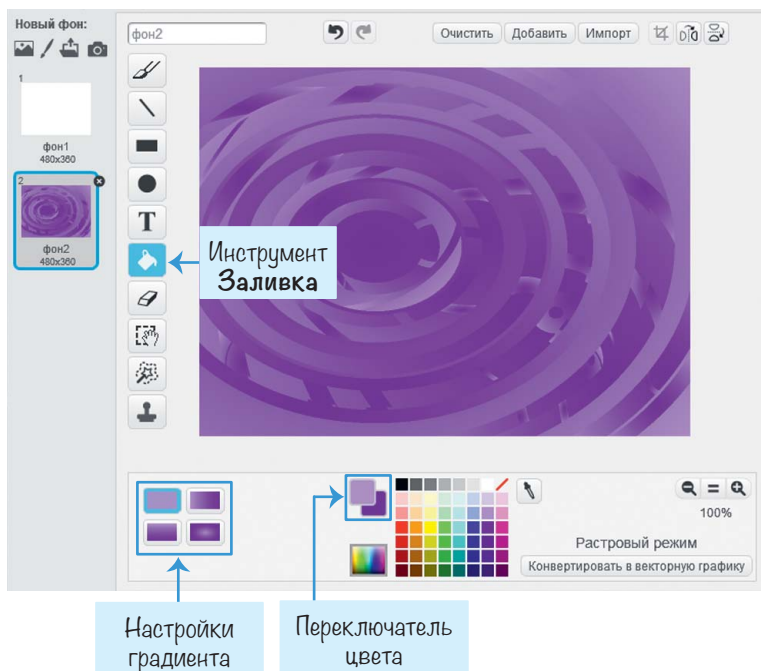
Вы можете добавить много красивых деталей к «Арканоиду», чтобы игра выглядела интересно и привлекательно. А лучше всего то, что вы можете использовать эти приемы улучшения в любой из ваших игр. Перед тем как заняться кодом, взгляните на полную версию игры на сайте [www.nostarch.com/scratchplayground/](http://www.nostarch.com/scratchplayground/).



## Создание классного фона

Чтобы игра выглядела круче, можно воспользоваться простым способом – нарисовать интересный фон. В области спрайтов нажмите кнопку **Сцена**, затем перейдите на вкладку **Фоны**, которая находится над областью блоков. В графическом редакторе выберите два различных фиолетовых оттенка. Сначала выберите светло-фиолетовый цвет; затем щелкните мышью по переключателю цвета и выберите темно-фиолетовый оттенок. Используя разные настройки градиента инструмента **Заливка**, создайте необычный, но интересный фон, как показано на следующем рисунке, щелкая мышью по фону случайным образом.





## Добавление музыки

Звуковое сопровождение создает настроение и оживляет игру. Выберите в области спрайтов пункт **Сцена** и перейдите на вкладку **Звуки** в верхней части области блоков. Нажмите кнопку **Выбрать звук из библиотеки** (она выглядит как динамик) под строкой **Новый звук**. Когда появится окно **Библиотека звуков**, выберите звук **Dance celebrate** и нажмите кнопку **ОК**. Для удобства вы можете переименовать звуковое сопровождение, присвоив имя **Танцевальная**.

Затем перейдите на вкладку **Скрипты** и добавьте код, показанный на следующем рисунке, в области скриптов сцены, чтобы в игре звучала фоновая музыка:



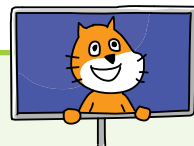
## Изменение цвета платформы при попадании мяча

На этом этапе мы окрасим платформу-ракетку различными цветами во время отскока от нее мяча. Добавьте код, показанный на следующем рисунке, к спрайту **Платформа**.



Сообщение **Отскок** передается, когда спрайт **Мячик** отскакивает от клона спрайта **Кирпичик**, а также от спрайта **Платформа**. Блок **Если расстояние до Мячик < 60, то** отвечает за изменение цвета платформы, когда спрайт **Мячик** отскакивает от спрайта **Платформа** (и означает, что мяч будет менее чем в 60 шагах).

### КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что платформа мигает разными цветами при отбивании мяча. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## Анимированное появление и исчезновение кирпичиков

Клоны кирпичиков в игре появляются довольно скучно. Они просто становятся видны, как только их клонировали. Для того чтобы анимировать их появление, измените код спрайта **Кирпичик**, чтобы он соответствовал рисунку ниже.

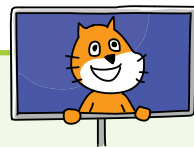


Присвойте блоку **Эффект призрака** значение **100**, а затем постепенно снижайте его. Этот эффект сделает так, чтобы клоны кирпичиков медленно исчезали, а затем мгновенно появлялись. Также этот код устанавливает появление клонов кирпичиков на 10 шагов ниже их конечного положения и медленно поднимает их, изменяя их положение по оси у в блоке **Повторить**. Благодаря этому фрагменту кода клоны кирпичиков как будто скользят на свои места перед началом игры.



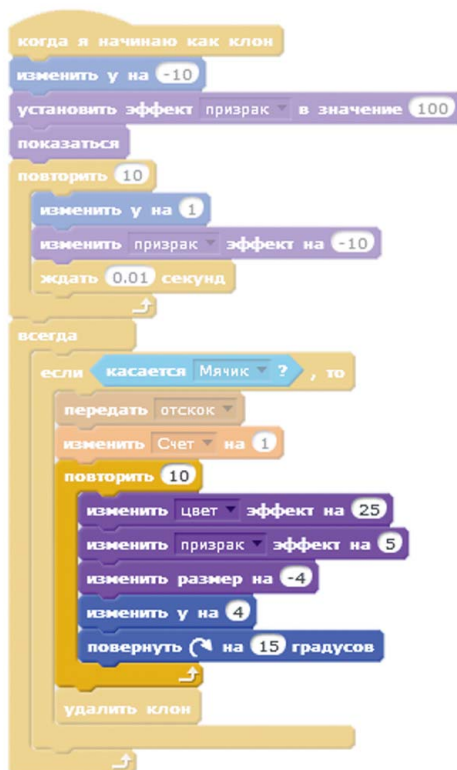


## КОНТРОЛЬНАЯ ТОЧКА



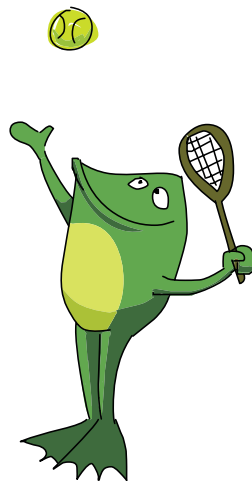
Нажмите кнопку в виде зеленого флага, чтобы проверить этот фрагмент кода. Убедитесь, что клоны кирпичиков исчезают из поля зрения, а потом мгновенно появляются. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

Теперь анимируем удаление клонов кирпичиков. Измените код спрайта **Кирпичик** так, чтобы исчезновение клонов кирпичиков было анимированным, вместо прежнего мгновенного исчезновения.



Теперь исчезновение клонов кирпичиков стало более захватывающим. Блоки **Изменить эффект на** внутри цикла **Повторить** сделают так, чтобы клоны кирпичи-

ков мигали разными цветами, и увеличат их эффект призрака так, чтобы они становились все более и более прозрачными. В то же время блок **Изменить размер на -4** делает так, чтобы клоны спрайта **Кирпичик** уменьшились в размере, блок **Изменить у на 4** поднимает их вверх, а блок **Повернуть по часовой стрелке на 15 градусов** поворачивает их. Такое анимированное исчезновение выглядит очень привлекательно.



## КОНТРОЛЬНАЯ ТОЧКА

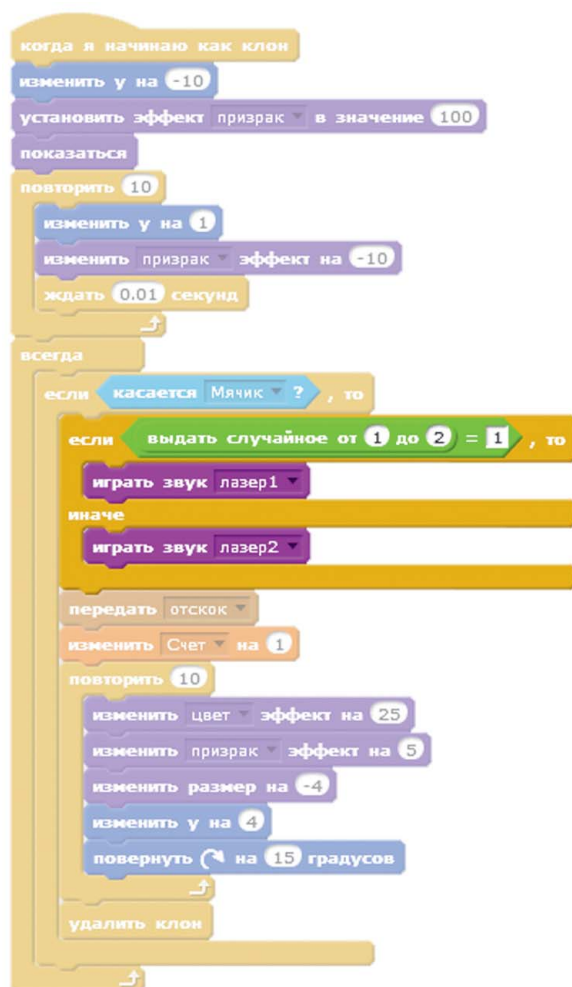


Нажмите кнопку в виде зеленого флага, чтобы проверить код, который уже готов. Ударьте по кирпичикам мячиком и убедитесь, что они вращаются и поднимаются вверх, а также что они медленно, а не мгновенно, исчезают из поля зрения. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## Звуковое сопровождение исчезновения кирпичиков

Давайте также сделаем, чтобы клоны кирпичиков проигрывали различные звуковые эффекты в момент своего исчезновения. Выберите спрайт **Кирпичик** в области спрайтов, а затем перейдите на вкладку **Звуки** над областью блоков. Нажмите кнопку **Выбрать звук из библиотеки** под надписью **Новый звук** и выберите звук **Laser1** из окна **Библиотека звуков**. Повторите этот шаг, чтобы добавить звук **Laser2**. Для удобства вы можете переименовать их, присвоив имена **Лазер1** и **Лазер2**.

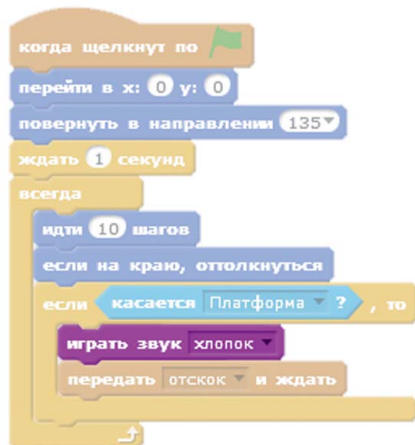
Измените код спрайта **Кирпичик** в соответствии с приведенным ниже рисунком:



После этого клоны кирпичиков будут воспроизводить случайный звуковой эффект при исчезновении. Блок **Если, то иначе** добавляет в программу некоторое разнообразие звуковых эффектов, случайно выбирая, какой звук будет проигрываться. Каждый раз, когда спрайт **Мячик** касается клона кирпичика, программа случайным образом выбирает значение 1 или 2 и в результате воспроизводит различные звуки.

## Звуковое сопровождение мячика

Теперь давайте добавим звуковой эффект при попадании спрайта **Мячик** на спрайт **Платформа**. Для каждого спрайта уже загружен звук «хлопок»; все, что вам нужно сделать, это обновить код спрайта **Мячик**, чтобы он соответствовал нижеприведенному рисунку:



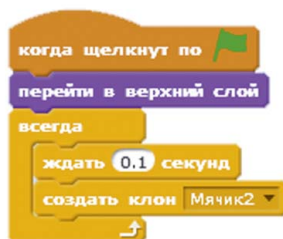
## Добавление хвоста к мячику

Добавление следа из клонов позади спрайта **Мячик**, когда он движется по сцене, создает впечатление, что у него есть хвост, как у кометы. Не стоит использовать клон спрайта **Мячик**, так как он будет реагировать на сообщение **Отскок** всякий раз, когда исходный спрайт **Мячик** совершает отскок. Вместо этого нажмите кнопку **Выбрать спрайт из библиотеки**, которая расположена рядом с надписью **Новый спрайт**. Затем в окне **Библиотека спрайтов** выберите спрайт **Tennis Ball** для создания другого спрайта под названием **Мячик2**. Клонировуйте этот второй мяч, чтобы создать след. В отличие от клона спрайта **Мячик**, клон спрайта **Мячик2** не содержит блок **Когда я получаю отскок**. Добавьте код, показанный на следующем рисунке, к спрайту **Мячик2**:



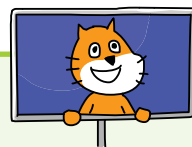
Единственное, что клон запрограммирован делать, это двигаться на текущее местоположение спрайта **Мячик**. Мячик продолжает двигаться, но клон остается на месте, уменьшается и становится все более прозрачным. В конце этой анимации, клон удаляется.

Вам также необходимо изменить код спрайта **Мячик**, используя следующий рисунок в качестве инструкции:



Этот скрипт создает новый клон спрайта **Мячик2** через 0,1 секунды ожидания, что создает след из мячиков.

## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить код, который готов к этому моменту. Убедитесь, что след из уменьшающихся клонов спрайта **Мячик2** следует за исходным спрайтом **Мячик**. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## Анимация появления спрайта **Игра окончена**

Когда игрок проигрывает, появляется текст «Игра окончена!». Анимированное появление надписи «Игра окончена!», подобно анимированному появлению клонов кирпичиков, выглядело бы интереснее. Измените код спрайта **Игра окончена**, чтобы он соответствовал коду, представленному на рисунке ниже. Сначала загрузите звуковой эффект **Gong**, нажав кнопку **Выбрать звук из библиотеки** на вкладке **Звуки**. Для удобства вы можете переименовать его, присвоив имя **Гонг**. Вы создадите новое сообщение под названием **Стоп игра**, которое будет сообщать спрайтам **Платформа** и **Мячик**, что нужно прекратить движение.



В начале игры спрайт **Игра окончена** скрывается, и значение его эффекта призрака равно 100. Когда в конце игры запускается блок **Показаться**, текст «Игра окончена!» по-прежнему полностью невидим. Потом текст «Игра окончена!» медленно проявляется за счет изменения значения эффекта призрака до -10, за что отвечает код анимации, находящийся внутри блока **Повторить 10**. Блоки **Повернуть по часовой стрелке на 15 градусов** и **Изменить размер на 12** поворачивают и увеличивают текст. После паузы в 4 секунды блок **Стоп все** завершает программу.

Для управления сообщением **Стоп игра** в спрайтах **Мячик** и **Платформа** добавьте код, показанный на следующем рисунке, в оба эти спрайта:

Добавьте этот код в спрайты  
Мячик и Платформа



Причина, по которой нужно использовать блок **Стоп другие скрипты спрайта** вместо блока **Стоп все**, в том, что программа должна продолжать работать во время анимации надписи «Игра окончена!».

Блок **Стоп другие скрипты спрайта** остановит движение спрайтов мячика и платформы, но другие спрайты в программе продолжают работать. Когда спрайт **Игра окончена** закончит появляться на экране, блок **Стоп все** завершит всю программу.

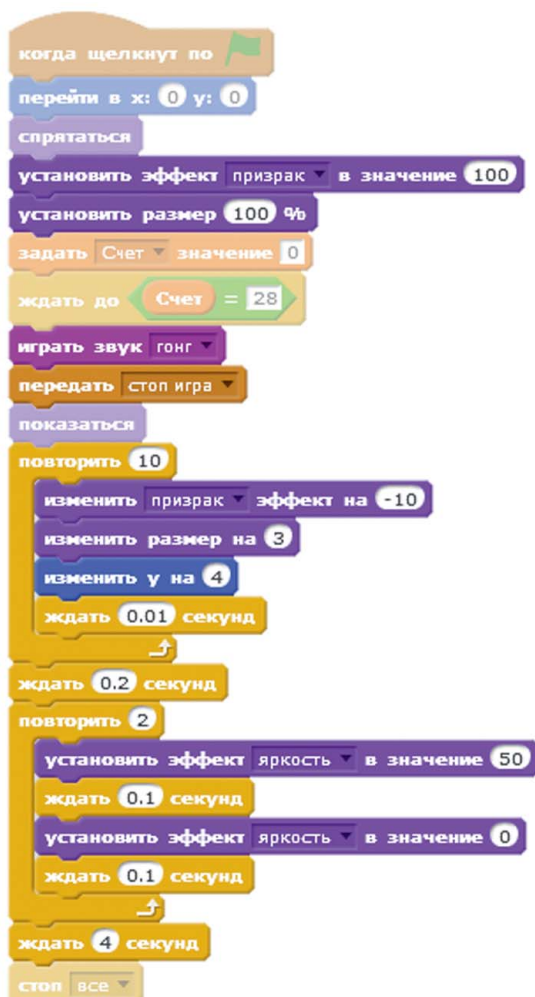


## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Начните играть в игру и проиграйте, чтобы проверить, что текст «Игра окончена!» появляется анимированно, а не просто мгновенно высвечивается на экране. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## Анимация появления спрайта **Вы выиграли**

Давайте также добавим анимированное появление для спрайта **Вы выиграли**. Обновите код спрайта **Вы выиграли**, чтобы он совпадал с тем, который приведен на следующем рисунке. Вам нужно будет загрузить звуковой эффект под названием **Gong**, нажав кнопку **Выбрать звук из библиотеки** на вкладке **Звуки**. Для удобства вы можете переименовать звук, присвоив ему имя **Гонг**.



В приведенном коде имеется два набора анимации. Один находится в блоке **Повторить 10**, а другой в блоке **Повторить 2**. Код в блоке **Повторить 10** отвечает за появление на экране спрайта **Вы выиграли**, увеличивает его и перемещает вверх. После этой короткой анимации, блок кода **Повторить 2** увеличивает яркость спрайта до 50, ждет десятую долю секунды, а затем снижает яркость до 0. На экране вы видите мигающую надпись. После четырехсекундной паузы блок **Стоп все** завершит всю программу.



## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Когда вы выиграете игру, убедитесь, что надпись «Вы выиграли!» появляется с анимацией, а не просто мгновенно вспыхивает на экране. Для того чтобы выиграть игру быстрее, можно временно изменить блок **Ждать пока счет = 28** на **Ждать пока счет = 1**. Теперь, чтобы выиграть, вам нужно разбить только один кирпичик. Сохраните вашу программу.

## ЗАКЛЮЧЕНИЕ

В этой главе вы создали игру, в которой:

- ▶ используются клоны для быстрого создания множества копий спрайтов **Кирпичик** и **Мячик2**;
- ▶ спрайт **Платформа** управляется с помощью мыши вместо клавиш ← и → на клавиатуре;
- ▶ отображаются сообщения «Игра окончена!» и «Вы выиграли!», которые были созданы с помощью инструмента **Текст** в графическом редакторе;
- ▶ используются спрайты с анимацией появления и исчезновения;
- ▶ используются звуковые эффекты и фоновая музыка, позволяющие оживить игру.



В процессе создания игры «Арканоид» вы познакомились с несколькими приемами, которые можно будет применять в ваших следующих играх. Вы можете добавлять анимацию при появлении надписи, цветные вспышки и звуковые эффекты для многих программ, чтобы сделать

их более красочными и интересным. Но это лучше делать после того, как вы убедитесь, что обычная, базовая версия вашей игры работает без сбоев. А затем уже можно начинать придавать игре яркость.

В этой главе также введен прием клонирования – полезный метод, который будет использоваться при создании игры «Зме-е-ейка!» в главе 6. По мере изучения этой книги игры, которые вы создаете, становятся все более сложными, но не волнуйтесь: вы просто должны внимательно следовать инструкциям шаг за шагом!

## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно подсмотреть в конце книги.)

1. Каким образом программа узнает, что спрайт **Мячик** пролетел мимо спрайта **Платформа**?
2. Какой блок создает клоны спрайта?
3. В каком блоке находится код, который запустится при создании клона?
4. Каковы три стиля вращения?
5. Почему спрайты **Вы выиграли** и **Игра окончена** скрываются после того, как вы нажали кнопку в виде зеленого флага?
6. Для чего используется блок **Ждать пока**?

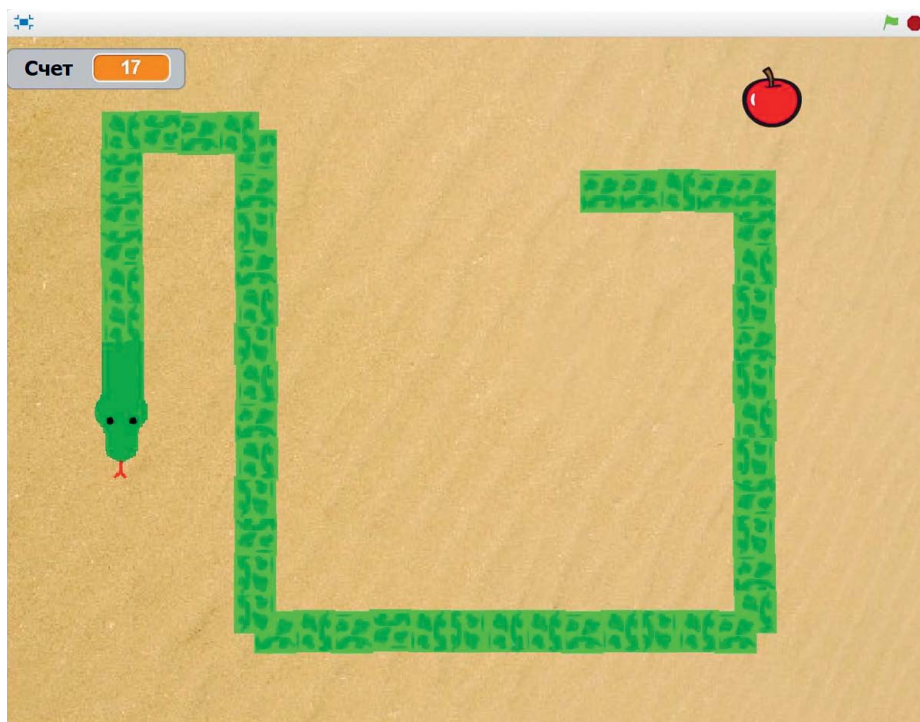
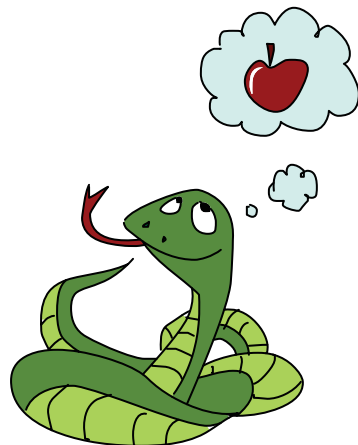


«Зме-е-ейка!» — это ремейк популярной игры, в которую многие играли на своих телефонах. Возможно, вы знаете эту игру под другим названием — Worm<sup>3</sup>. Для игры используются клавиши со стрелками, чтобы направлять постоянно движущуюся змею к яблокам, которые появляются на экране.

---

<sup>3</sup> В пер. с англ. «червяк». — *Примеч. ред.*

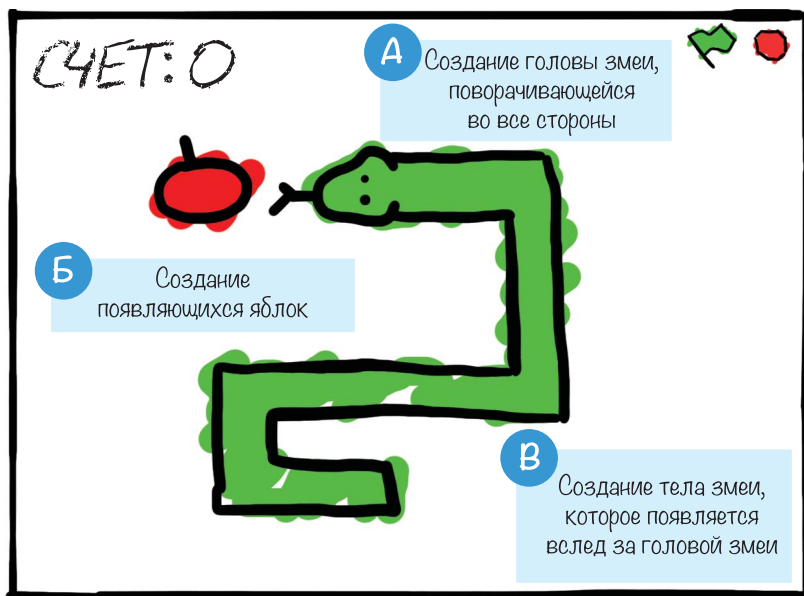
Чем больше яблок змея съест, тем длиннее она становится и тем сложнее уследить за тем, чтобы она не врезалась в самую себя или края сцены. Вы не можете замедлить движение змеи. Игра заканчивается, когда змея врезается в препятствие. Прежде чем приступить к созданию игры, взгляните на готовую программу на сайте [scratch.mit.edu/studios/4188596/](http://scratch.mit.edu/studios/4188596/).



Даже если змея сильно вырастет, вам все равно придется ее измерять в сантиметрах, потому что у змей нет ног!

## ЭСКИЗ ПРОЕКТА

Давайте набросаем эскиз — как должна выглядеть наша игра.



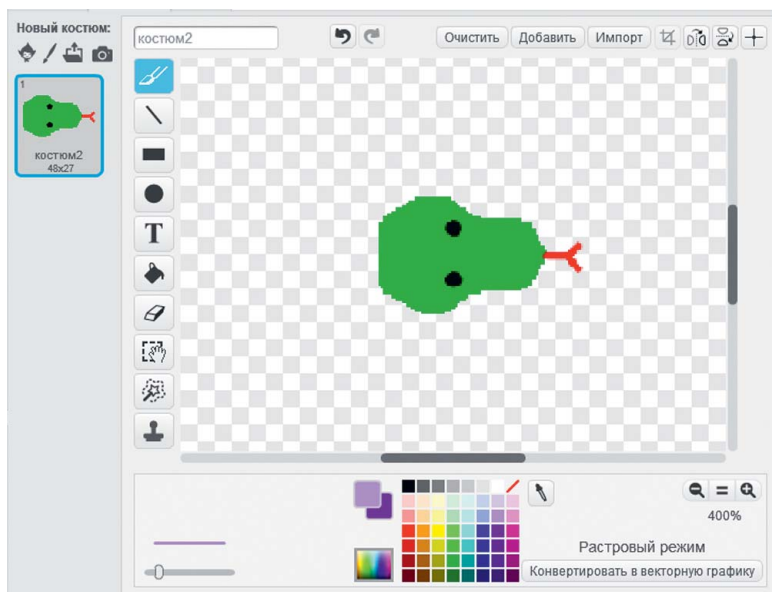
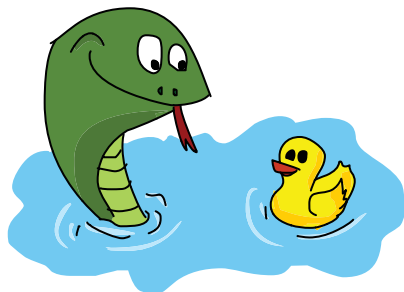
Если вы хотите сэкономить время, вы можете начать с файла скелета проекта с именем *Глава 06/Скелет\_проекта.sb2*, находящийся в запакованном файле с примерами, который вы скачали ранее по ссылке [https://eksmo.ru/files/Scratch\\_Sweigart.zip](https://eksmo.ru/files/Scratch_Sweigart.zip). В файл скелета проекта уже загружены все спрайты, так что вам нужно всего лишь перетащить блоки кода в каждый спрайт.

### А СОЗДАНИЕ ГОЛОВЫ ЗМЕИ, ПОВОРАЧИВАЮЩЕЙСЯ ВО ВСЕ СТОРОНЫ

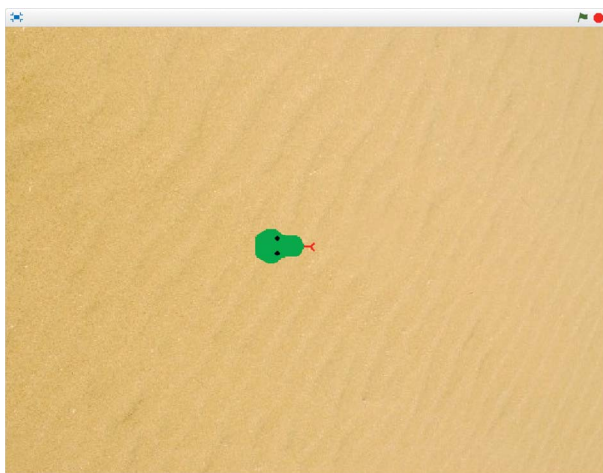
Мы начнем с создания головы змеи, которой игрок может управлять с помощью клавиатуры. Клавиши со стрелками изменяют направление головы змеи, но она всегда будет двигаться вперед. Позже мы запрограммируем тело змеи.

## 1. Создание спрайта головы

Для начала давайте сделаем фон более интересным. В области спрайтов нажмите кнопку **Сцена**, а затем перейдите на вкладку **Фоны** над областью блоков. Нажмите кнопку **Загрузить фон из файла** под надписью **Новый фон** и выберите файл *песок.jpg* из архивного файла с примерами.

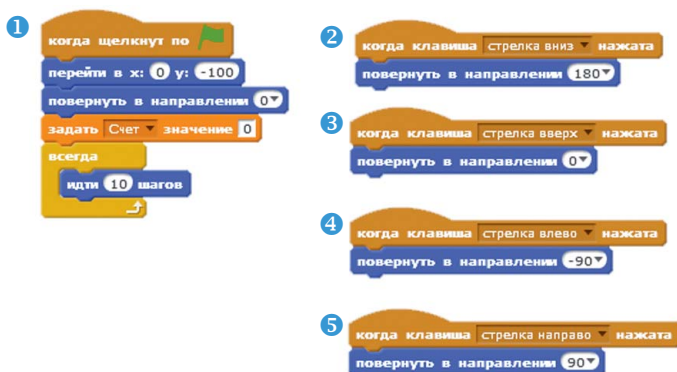


Затем нарисуйте голову змеи. Нажмите кнопку **Создать новый спрайт** рядом с надписью **Новый спрайт**. В графическом редакторе нарисуйте змеиную голову, вид сверху, чтобы она была обращена вправо. Это нужно сделать, поскольку все спрайты Scratch сначала повернуты на 90 градусов (то есть направо), вы должны сделать так же. После того как вы закончили рисовать, присвойте спрайту имя **Голова**.



Используйте инструменты **Уменьшить** или **Увеличить** в верхней части редактора Scratch, чтобы уменьшить или увеличить созданный спрайт **Голова**. Он должен быть примерно такого же размера, как показано на рисунке выше.

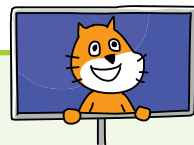
Добавьте показанный ниже код к спрайту **Голова**:



Скрипт ① задает для головы змеи начальные положение и направление (0 градусов или прямо вверх). Счет игрока должен быть установлен равным 0, для чего вам нужно создать переменную с именем **Счет** и в режиме **Для всех спрайтов**. Как и в предыдущих проектах, переменную можно создать, нажав кнопку **Создать переменную** в оранжевой категории **Данные**. Поскольку мы хотим, чтобы

змея *никогда* не переставала двигаться, программный код **1** содержит цикл **Всегда** с блоком **Идти 10 шагов**.

Скрипты **2**, **3**, **4** и **5** представляют собой короткие сценарии, которые отвечают за управление: направления соответствуют клавишам  $\uparrow$ ,  $\downarrow$ ,  $\leftarrow$  и  $\rightarrow$ .



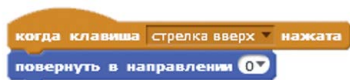
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что клавиши  $\uparrow$ ,  $\downarrow$ ,  $\leftarrow$  и  $\rightarrow$  правильно направляют голову змеи во всех четырех направлениях: вверх, вниз, влево и вправо. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

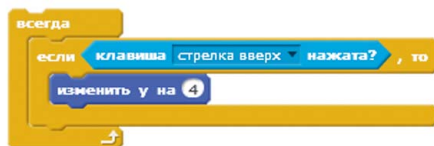
## ЭТО ИНТЕРЕСНО: РАЗНИЦА МЕЖДУ БЛОКАМИ «КОГДА КЛАВИША НАЖАТА» И «ЕСЛИ КЛАВИША НАЖАТА» ?

В программе «Зме-е-ейка!» блок кода **Когда клавиша нажата** перемещает игрока и используется, если мы ожидаем, чтобы клавиша была нажата *однократно*.

В игре «Бегущий в лабиринте» из главы 3, блок **Если, то** с блоком **Клавиша нажата ?** внутри цикла **Всегда** за-цикливает перемещение игрока. Используйте код **Если клавиша нажата ? то**, когда предполагается, что клавиша будет *нажата и удержана*.



Код игры «Зме-е-ейка!»



Код «Бегущий в лабиринте»



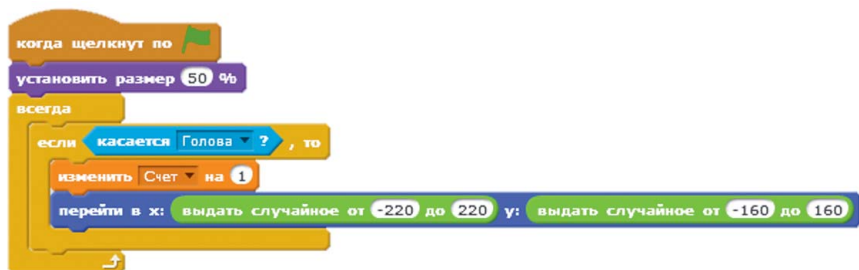
Эти два способа запрограммировать элементы управления игрой – не одно и то же. Убедитесь, что использовали соответствующие блоки кода для игры, которую вы хотите создать. Поскольку змея постоянно движется самостоятельно, игроку необходимо нажать клавишу только один раз, чтобы изменить направление движения змеи. Вот почему в игре «Зме-е-ейка!» используются блоки **Когда клавиша нажата**.

## 6 СОЗДАНИЕ ПОЯВЛЯЮЩИХСЯ ЯБЛОК

Базовые движения змеи готовы, теперь давайте добавим яблоко, которое змея будет пытаться съесть.

### 2. Добавление спрайта Яблоко

Нажмите кнопку **Выбрать спрайт из библиотеки** рядом с надписью **Новый спрайт** и выберите спрайт **Apple** из **Библиотека спрайтов**. Для удобства вы можете переименовать его, присвоив имя **Яблоко**. Добавьте код, показанный на следующем рисунке:



Этот код обеспечивает исчезновение спрайта **Яблоко**, когда его коснется змея, а затем его повторное появление в другом месте сцены. Новое положение выбирается на основе случайных чисел, как при броске игральных костей. Скрипт также добавляет 1 к значению переменной **Счет** каждый раз, когда спрайт **Голова** касается спрайта **Яблоко**.

## КОНТРОЛЬНАЯ ТОЧКА



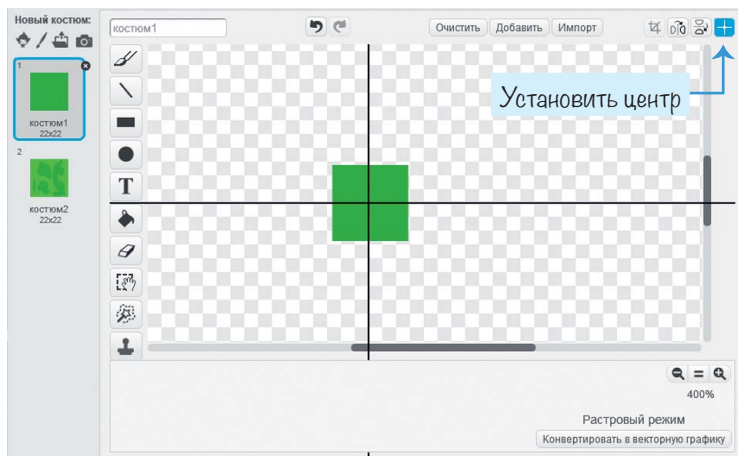
Нажмите кнопку в виде зеленого флага, чтобы проверить готовую часть кода. Направляйте змею к яблоку, чтобы съесть его. Когда голова змеи коснется яблока, удостоверьтесь, что яблоко переместилось в другое место. Значение переменной **Счет** должно увеличиваться на один каждый раз, когда голова змеи касается яблока. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## В СОЗДАНИЕ ТЕЛА ЗМЕИ

Теперь мы добавим тело змеи, которое будет увеличиваться в длину после каждого съеденного яблока.

### 3. Создание спрайта Тело

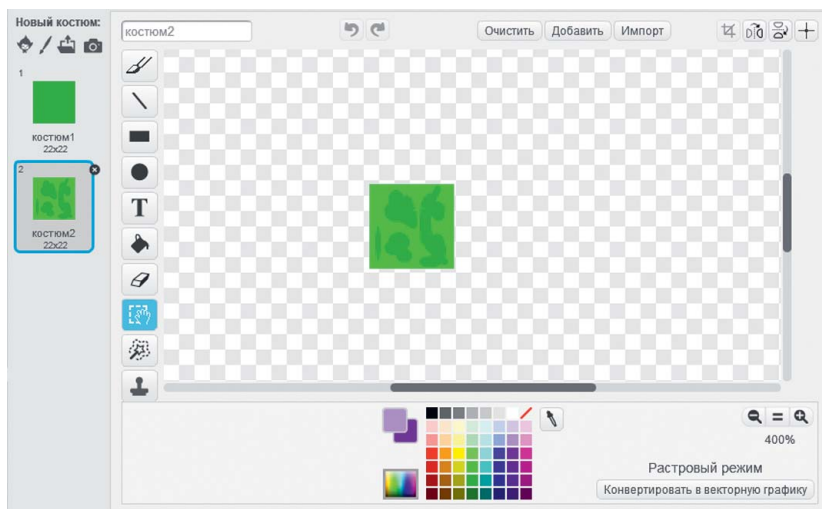
Нажмите кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**, чтобы создать новый спрайт. Нарисуйте небольшой квадрат, используя тот же цвет, что и для головы змеи. Убедитесь, что центр костюма находится посередине квадрата, нажав кнопку **Установить центр**, а затем щелкнув в центре квадрата.



Нажмите кнопку **i** на миниатюре спрайта, чтобы открыть его панель информации, и измените имя спрайта, указав слово **Тело**.

## 4. Создание второго костюма для спрайта **Тело**

Пока вы все еще находитесь в графическом редакторе, щелкните правой кнопкой мыши по костюму **Костюм1** и выберите в контекстном меню команду **Дублировать**. Используя инструмент **Заливка**, измените цвет квадрата созданного костюма на другой, например светло-зеленый цвет. (В моей версии игры используется темно-зеленый оттенок для головы змеи и светло-зеленый – для второго костюма тела.) Мы будем использовать светло-зеленый цвет, чтобы определить, врезалась ли змея в саму себя. На этом втором цвете вы можете добавить какие-нибудь узоры.

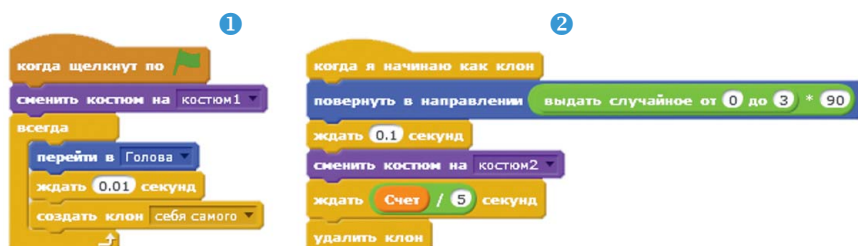


Если хотите, вы можете нарисовать тело змеи в одежде. Я бы порекомендовал галстук-бабочку, боа из перьев или что-нибудь еще. Это неважно.

Главное, убедитесь, что костюмы спрайта **Тело** имеют квадратную форму и что **Костюм2** окрашен в другой цвет, не такой, как **Костюм1** и спрайт **Голова**!

## 5. Добавление кода для спрайта Тело

Перейдите на вкладку **Скрипты** и добавьте показанный ниже код к спрайту **Тело**. Нам нужно, чтобы спрайт **Тело** всегда следовал за спрайтом **Голова** и генерировал клоны себя самого.



Исходный код спрайта **Тело** запускается после блока **Когда щелкнут по зеленому флагу** в сценарии 1. Во время движения спрайта **Голова** по сцене, спрайт **Тело** создает дорожку клонов самого себя.

Код клонов спрайта **Тело** срабатывает после блока **Когда я начинаю как клон** в скрипте 2. Когда клон спрайта **Тело** создается впервые, он повернут в случайном направлении. Блок **Выдать случайное от 0 до 3 \* 90** может определить направление клона **Тело** согласно значению 0, 90, 180 или 270 градусов. В разных положениях части тела змеи немного различаются.

В конечном счете клоны должны удалять себя со сцены, чтобы змея не росла в длину просто так. Таким образом, каждый клон ожидает короткое время, которое задается значением переменной **Счет** в блоке **Ждать Счет / 5 секунд** перед тем, как автоматически удалиться. Время ожидания одинаково для всех клонов, поэтому те клоны тела змейки, которые были созданы первыми, удаляются тоже первыми.

Поедание яблок увеличивает значение переменной **Счет**. По мере того, как значение переменной **Счет** увеличивается, время, которое клон тела ожидает, чтобы удалиться,

также увеличивается. Благодаря более продолжительному ожиданию змея становится длиннее, потому что на сцене оказывается больше клонов ее тела. Таким образом, чем больше яблок съест змейка, тем длиннее она станет.

Когда значение переменной **Счет** равно 0, ожидание будет составлять 0/5 секунд или 0 секунд. Когда значение переменной **Счет** равно 1, ожидание составляет 1/5, или 0,2 секунды. Когда это значение равно 2, ожидание составляет 2/5, или 0,4 секунды. Каждая единица, добавленная к значению переменной **Счет**, добавляет еще 0,2 секунды времени ожидания, в результате чего змея становится все длиннее и длиннее. И по мере того, как змея становится длиннее, сложность игры реально возрастает.

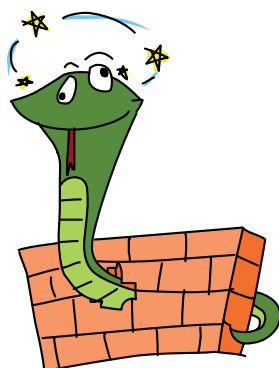
## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Проверьте, чтобы клоны тела змейки появлялись позади головы змеи, которая становится длиннее по мере поедания яблок. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## 6. Определим, врезается ли змейка в саму себя или в стену

Когда змея врезается в себя или в края сцены, нам нужно запустить блок **Когда я получу игра окончена**: спрайт **Голова** отобразит сообщение «Ой!» в течение 2 секунд и затем завершит программу. Вместо того, чтобы писать один и тот же код дважды, давайте поместим код для всех видов столкновений под блок **Когда я получу игра**



окончена, чтобы любой сбой мог передать сообщение **Игра окончена** и запустить этот код. Если вы хотите изменить код, вам просто нужно изменить его в одном месте — в скрипте **Когда я получу игра окончена**.

Добавьте код, показанный на следующем рисунке, к спрайту **Голова**:



Используя блок **Касается цвета ?**, первый цикл **Если** проверяет условие, когда змея касается своего тела: убедитесь, что для этого условия вы указали цвет, в который окрашен **Костюм2**. Следующий цикл **Если** проверяет горизонтальную и вертикальную границы сцены. Когда змея пересекает их, посылается то же сообщение **Игра окончена**. Будем надеяться, что игрок достаточно быстр, чтобы избежать столкновений; в противном случае эта змея станет ис-с-с-торией!

Вы заметили блок **Ждать 0,01 секунду** в спрайте **Тело**, который заставляет клоны спрайта **Тело** немного подождать, прежде чем менять костюм? На рисунке ниже снова показан код для спрайта **Тело**:



Второй костюм для спрайта **Тело** светлее, чем тот, в который окрашен спрайт **Голова**, чтобы легче было понять, врезалась ли змея в саму себя. Поскольку клоны **Тело** создаются в том же месте, что и спрайт **Голова**, они ее касаются когда появляются впервые. Вот почему нам нужно приостановить обнаружение столкновения, когда создан клон. Без этой небольшой задержки спрайт **Голова** считал бы, что врезался в клон **Тело**, который только что был создан.



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Врежьтесь в стену и в тело змеи, чтобы убедиться, что функция обнаружения столкновений работает. Если ваша змея проигрывает, даже если она не касается самой себя или краев сцены, попробуйте увеличить время ожидания с 0,01 до 0,02 или еще больше. Также убедитесь, что код **Игра окончена** не срабатывает, если змея не врезается во что-либо. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

# ЗАКОНЧЕННАЯ ПРОГРАММА

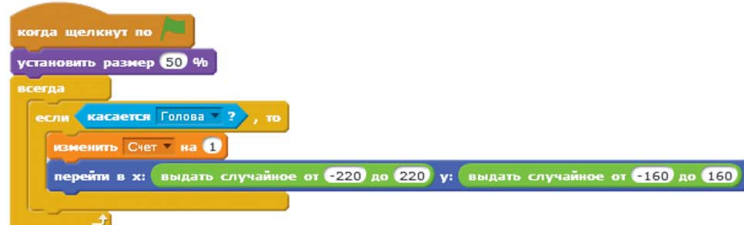
На рисунке ниже показан итоговый код всей программы. Если ваша программа работает неправильно, сверьте свой код с кодом, приведенным ниже.



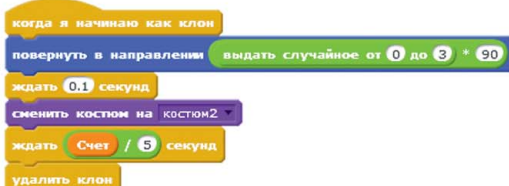
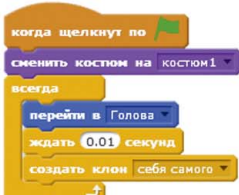
Голова



Яблоко



Тело

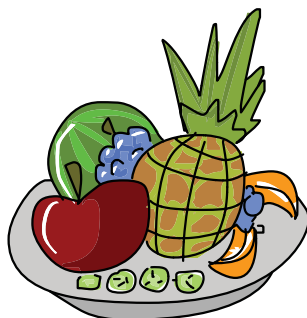




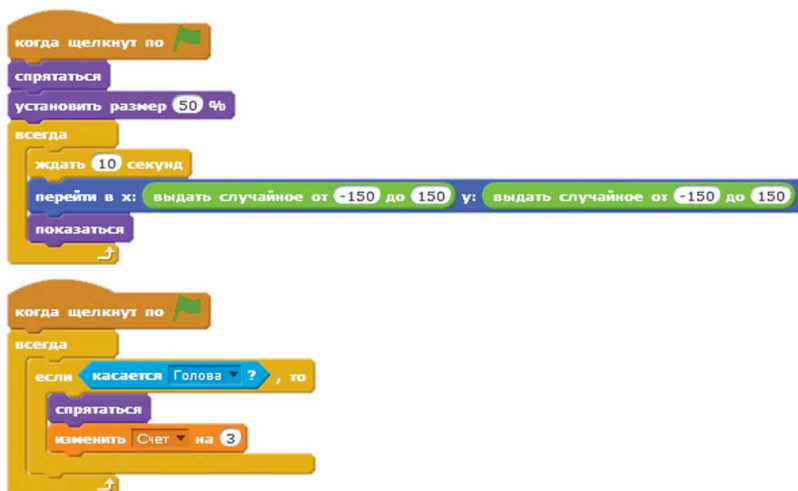
## ВЕРСИЯ 2.0: ДОБАВЛЕНИЕ БОНУСНЫХ ФРУКТОВ

Выполнение однотипных заданий очень быстро надоедает, даже если это игра. Давайте добавим немного бонусных фруктов!

Мы добавим второй тип фруктов, который увеличит счет на 3 очка, когда змея его съест. Нажмите кнопку **Выбрать спрайт из библиотеки**. В окне **Библиотека спрайтов** выберите пункт **Fruit Platter** и нажмите кнопку **ОК**. Для удобства вы можете переименовать добавленный спрайт, присвоив имя **Салатик**. (Или, быть может, **Десерт**.)



Добавьте код, показанный на следующем рисунке, к новому спрайту:



Код спрайта **Салатик** практически идентичен коду спрайта **Яблоко**, за исключением того, что здесь используется 10-секундная пауза перед тем, как появиться где-то еще на сцене после того, как змея коснется его. С этими бонусными очками игровой счет будет действительно быстро увеличиваться!



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Через 10 секунд должен появиться спрайт **Салатик**. Убедитесь, что значение переменной **Счет** увеличивается на 3 очка, когда змея касается спрайта **Салатик**, и что спрайт после этого исчезает. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## ЧИТ-РЕЖИМ: НЕПОБЕДИМОСТЬ

Было бы здорово сделать змею очень-очень длинной! Давайте добавим чит-режим, чтобы змея могла расти, не врезаясь в себя. Мы также добавим змее эффект радуги, чтобы игрок мог видеть, когда включен чит-режим.

### Изменение головы

Измените код спрайта **Голова**, чтобы он соответствовал приведенному на рисунке ниже. Вы добавите скрипт, который писали ранее, а также совершенно новый скрипт. Для

1

когда щелкнул по

здать чит-режим значение **выкл**

перейти в х: 0 у: -100

повернуть в направлении 0°

здать Счет значение 0

всегда

идти 10 шагов

если касается цвета ? и **чит-режим = выкл**, то

передать игра окончена и ждать

если касается край ? , то

передать игра окончена и ждать

если **чит-режим = вкл**, то

изменить цвет эффект на 5

иначе

установить эффект цвет в значение 0

2

когда клавиша **пробел** нажата

если **чит-режим = выкл**, то

здать чит-режим значение **вкл**

иначе

здать чит-режим значение **выкл**

Добавлен новый режим

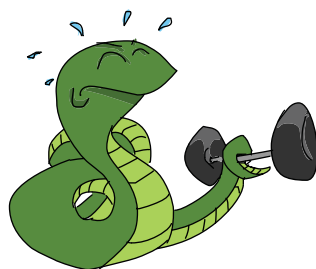
обоих скриптов необходимо создать новую переменную с именем **Чит-режим**. Создайте эту переменную в режиме **Для всех спрайтов**.

Скрипт ② управляет чит-режимом. Нажатие клавиши **Пробел** переключает у переменной **Чит-режим** значения **Вкл** и **Выкл**. Когда присвоено значение **Вкл**, сообщение **Игра окончена** не будет отображаться, даже если голова змеи коснется ее тела. Причина в том, что в скрипте ① вы добавили условие **Чит-режим = выкл** в блок **Если, то** который проверяет, врезалась ли змея в себя. Если переменной **Чит-режим** присвоено значение **Вкл**, то не имеет значения, врезалась ли змея в себя, потому что оба условия – **Касается цвета ?** и **Чит-режим = выкл** – должны соблюдаться для запуска блока **Передать игра окончена и ждать**.

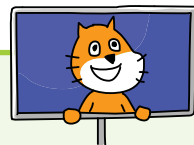
Скрипт ① также содержит блок **Если, то иначе**, который придает голове змеи радужный эффект, когда включен чит-режим. Когда чит-режим отключен, цветовой эффект сбрасывается до 0, вследствие чего спрайту **Голова** возвращается его исходный цвет.

## Изменение кода спрайта **Тело**

Теперь нам нужно добавить радужный эффект в клоны спрайта **Тело**. Измените код спрайта **Тело**, чтобы он соответствовал следующему рисунку:



Когда переменной **Чит-режим** присвоено значение **Вкл**, спрайт **Тело** постепенно меняет свой цвет, создавая раду-гу. В случае, если переменной **Чит-режим** присвоено зна-чение **Выкл**, спрайт **Тело** сбрасывает значение своего цве-тового эффекта на 0.



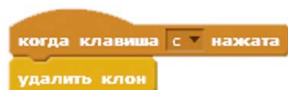
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы прове-рить готовый фрагмент кода. Нажмите клавишу Пробел и попытайтесь врезаться головой змейки в ее тело. Игра не должна заканчиваться. Затем вновь нажмите клави-шу Пробел и попытайтесь врезаться в себя. Теперь иг-ра должна закончиться. Убедитесь, что эффект радуги появляется после нажатия клавиши Пробел. Затем на-жмите кнопку в виде красного знака остановки и сохра-ните программу.

## Чит-режим: отрезание хвоста!

Игра перестает быть сложной, если змея непобедима. Но было бы удобно, если бы вы могли временно избавиться от тела змеи, если существует опасность врезаться в него. Да-вайте добавим чит, который мгновенно удалит тело.

Добавьте код, показанный на следующем рисунке, к спрайту **Тело**:



Когда игрок нажимает клавишу **C**, все клоны удаляют-ся. (С исходным спрайтом ничего не происходит, потому что это не клон.)

## ЗАКЛЮЧЕНИЕ

В этой главе вы создали игру, в которой:

- ▶ используется спрайт для генерации следа, состоящего из клонов, по мере передвижения спрайта по сцене;
- ▶ используются костюмы головы и тела змеи, которые вы нарисовали в графическом редакторе;
- ▶ определяется, когда голова змеи касается тела, проверяя, какого цвета на теле коснулась голова;
- ▶ используется переменная для отслеживания активности чит-режима;
- ▶ игрок может менять направление движения змеи с помощью клавиатуры, но не влияет на скорость;
- ▶ нет определенной цели игры. Игрок может играть так, как может.

Управление игрой «Зме-е-ейка!» с клавиатуры аналогично управлению в игре «Бегущий по лабиринту», которую мы создавали в главе 3. Пользователь видит игровую область сверху и перемещается вверх, вниз, влево и вправо. Однако игра «Зме-е-ейка!» отличается тем, что игровой персонаж движется *всегда*. Игрок может выбирать только направление, в котором движется змейка. Вы можете использовать этот стиль движения в собственных играх с быстрым темпом перемещения.

В главе 7 вы узнаете, как использовать мышь в ваших программах, создавая клон игры Fruit Ninja. После этого вы сможете использовать в своих программах и клавиатуру, и мышь!

## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно посмотреть в конце книги.)

1. В чем разница между блоками **Когда клавиша нажата** и **если клавиша нажата**?
2. Что делает блок **Перейти в х: выдать случайное от -220 до 220 у: выдать случайное от -160 к 160**?
3. В чем разница между блоками **Плыть** и **Перейти в**?
4. Почему нарисованная голова змеи должна быть повернута вправо?
5. Почему центр костюма спрайта **Голова** должен быть центрирован?



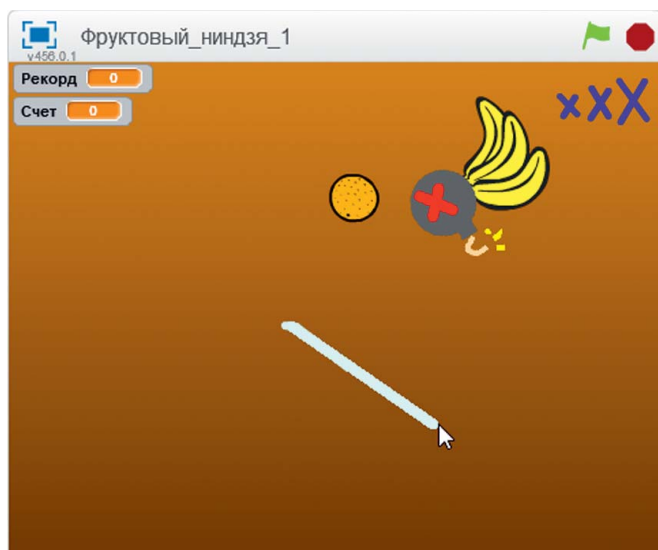
7

## ФРУКТОВЫЙ НИНДЗЯ

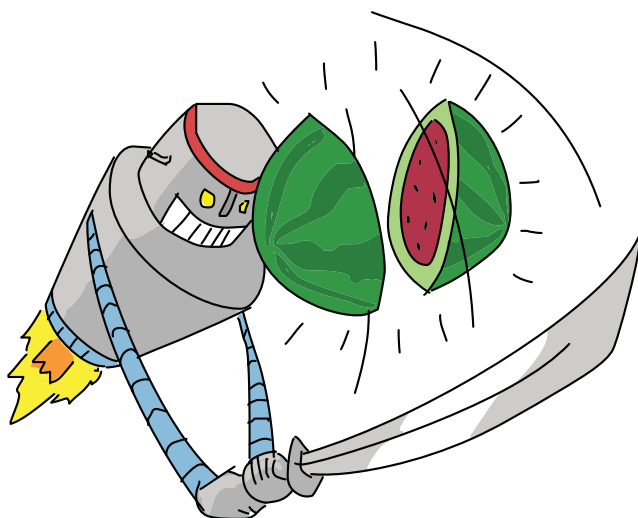
**Ф**ruit Ninja — популярная игра для смартфонов и планшетов, выпущенная в 2010 году. Идея игры заключается в том, чтобы успеть разрезать подброшенные в воздух фрукты до того, как они упадут. В этой главе вы сделаете игру «Фруктовый ниндзя», имитирующую механику игры Fruit Ninja.

В процессе программирования этой игры мы будем использовать некоторые совершенно новые функции Scratch, так что приготовьтесь, будет интересно!

Прежде чем начать создание собственной программы, посмотрите уже готовую программу на странице [scratch.mit.edu/studios/4188596/](https://scratch.mit.edu/studios/4188596/).



В игре, которую мы собираемся сделать, игрок разрезает фрукты с помощью мышки.





## ЭСКИЗ ПРОЕКТА

Давайте нарисуем, как должна выглядеть игра. Эскиз игры «Фруктовый ниндзя» должен выглядеть примерно так:



Если вы хотите сэкономить время, вы можете начать с файла скелета проекта с именем *Глава 07/Скелет\_проекта.sb2*, находящийся в запакованном файле с примерами, который вы скачали ранее по ссылке [https://eksmo.ru/files/Scratch\\_Sweigart.zip](https://eksmo.ru/files/Scratch_Sweigart.zip). В файл скелета проекта уже загружены все спрайты, так что вам нужно всего лишь перетащить блоки кода в каждый спрайт.

### **А** СОЗДАНИЕ НАЧАЛЬНОЙ ЭКРАННОЙ ЗАСТАВКИ

Игра «Фруктовый ниндзя» будет содержать *начальную экранную заставку*. Когда игрок нажимает кнопку в виде зеленого флага, на начальном экране отображается на-

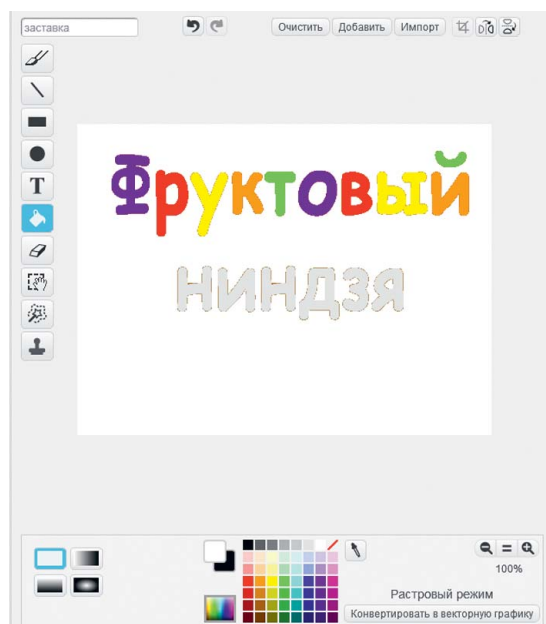
звание игры. Когда игрок проиграет, игра вернется к начальному экрану. Начальный экран – это подходящее место для размещения вашего имени или логотипа, когда вы показываете свои игры другим игрокам. Действительно, может, именно сейчас стоит придумать название вашей будущей игровой студии и использовать его в ваших программах!

Создайте новый проект в редакторе Scratch и введите имя «Фруктовый ниндзя».

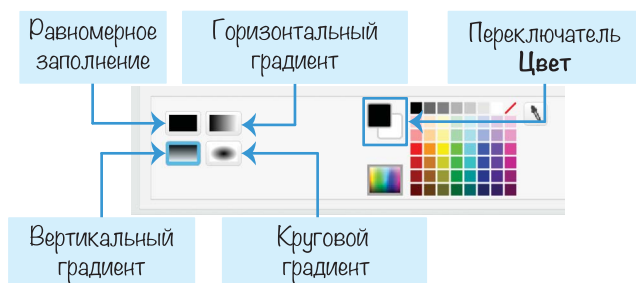
## 1. Рисование фона

Выберите пункт **Сцена** в области спрайтов, а затем перейдите на вкладку **Фоны** в верхней части области блоков. (Когда выбран пункт **Сцена**, вкладка **Костюмы** меняется на **Фоны**.) Выберите понравившийся цвет, а затем с помощью инструмента **Линия** нарисуйте буквы для названия игры «Фруктовый ниндзя». Затем используйте инструмент **Заливка** для заливки букв цветом.

Когда будет готово название игры, замените белый фон на темно-коричневый градиент. (При выборе инструмента **Заливка** рядом с цветными кнопками появляются четы-

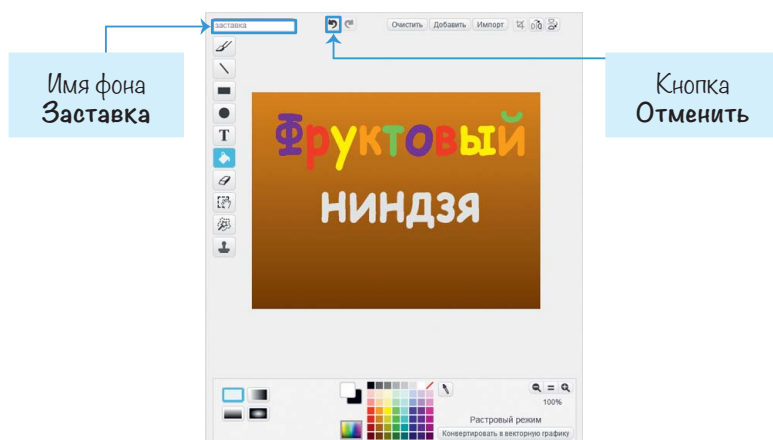


ре кнопки градиента.) Сначала щелкните мышью по переключателю **Цвет** и выберите светло-коричневый цвет. Затем выберите тип градиента, нажав соответствующую кнопку. Я использовал вертикальный градиент.



Щелкните мышью по холсту в графическом редакторе, чтобы заполнить фон градиентом. Если вы хотите поэкспериментировать, нажмите кнопку **Отменить** в верхней части графического редактора, прежде чем использовать другой градиент. Не забудьте также заполнить отверстия в буквах, в которых они есть, например в буквах *Р* или *Ф*.

Когда все будет готово, присвойте фону имя **Заставка**. Теперь нам нужно создать простой фон. Когда игра начинается, программа должна скрыть заставку с названием игры «Фруктовый ниндзя» и переключиться на этот фон. Нажмите кнопку **Нарисовать новый фон** в разделе **Новый фон**, чтобы создать второй фон. Затем нарисуйте коричневый градиент для этого фона, например так:



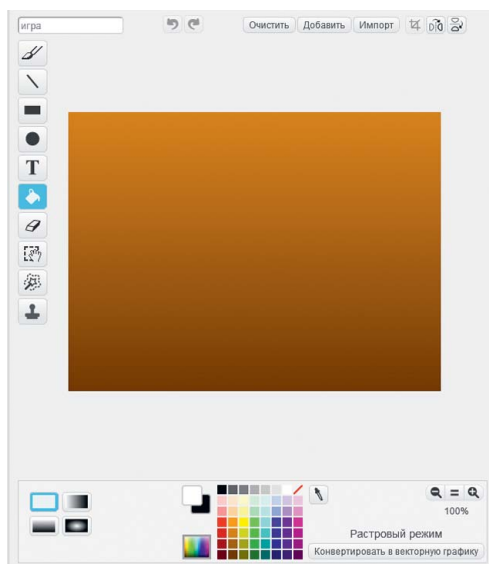
Переименуйте созданный второй фон, присвоив ему имя **Игра**. Когда начнется игра, мы переключимся с заставки на игровой фон.

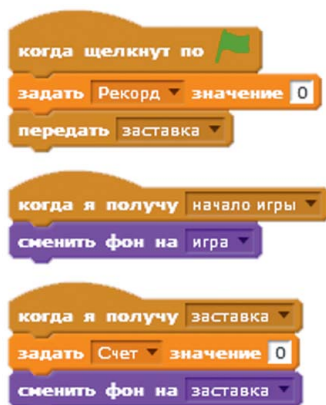
## 2. Добавление кода сцены

Теперь запрограммируем сцену. Выберите пункт **Сцена** в области спрайтов и перейдите на вкладку **Скрипты** в верхней части области блоков. Создайте две переменные, **Счет** и **Рекорд**, нажав кнопку **Создать переменную в оранжевой категории Данные**. Для обеих переменных укажите режим **Для всех спрайтов**.

**ПРИМЕЧАНИЕ.** На сцене используются переменные только в одном режиме – *Для всех спрайтов*, поэтому параметр *Только для этого спрайта* даже не будет доступен!

Вам также необходимо создать два новых сообщения, **Заставка** и **Начало игры**. Сделайте это, щелкнув по черному треугольнику в блоке **Сообщение** или **Когда я получу**, и выберите команду **Новое сообщение** в раскрывающемся списке. Затем добавьте следующие скрипты на сцену.





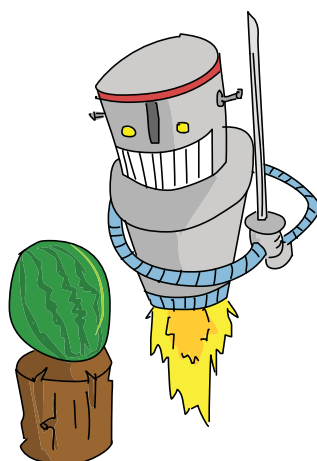
Этот код сбрасывает значения переменных **Рекорд** и **Счет** на 0 при запуске программы. Сообщение **Заставка** передается в начале программы и когда пользователь проигрывает игру. Значение переменной **Счет** сбрасывается до 0, и блок **Сменить фон на заставка** снова выводит начальную заставку, когда пользователь проигрывает игру.

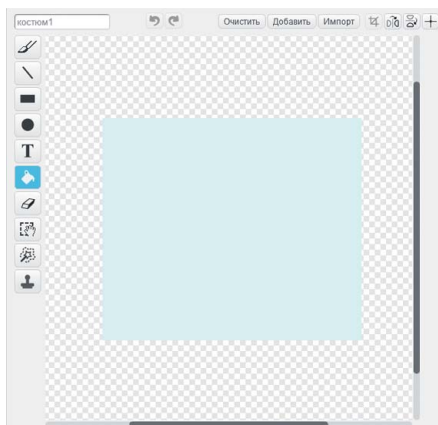
## Б СОЗДАНИЕ СЛЕДА ОТ РАЗРЕЗАНИЯ ФРУКТОВ

В игре «Фруктовый ниндзя» не используется спрайт игрового персонажа. Вместо этого игрок нажимает кнопку мыши и двигает курсор по экрану, отчего на экране ненадолго возникает след, похожий на эффект разрезания фруктов.

### 3. Создание спрайта Ломтик

Нажмите кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**. Откройте панель информации и переименуйте спрайт, присвоив ему имя **Ломтик**. В графическом редакторе нарисуйте голубой прямоугольник, который выглядит показанным ниже на рисунке образом.



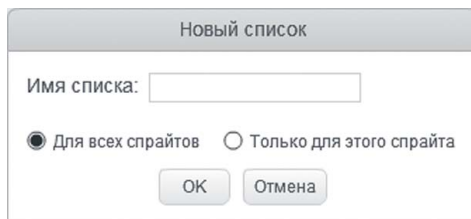


Спрайт **Ломтик** не должен совпадать по цвету с любым из оттенков, используемых на рисунках фруктов. В противном случае некоторые фрукты будут блокировать разрезание других фруктов.

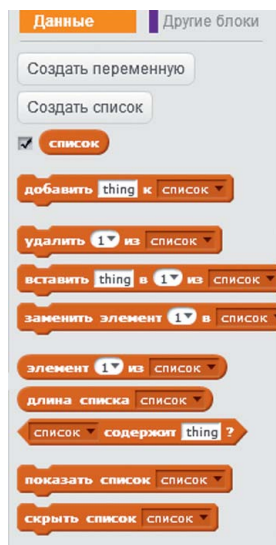
## ЭТО ИНТЕРЕСНО: СОЗДАНИЕ СПИСКОВ

Теперь вы узнаете еще об одной продвинутой функции Scratch – списках, новом способе хранения данных. *Список* похож на переменную, отличаясь тем, что он может хранить множество значений, а не только одно. Откройте новую вкладку веб-браузера и запустите копию редактора Scratch. Списки мы изучим в отдельном проекте Scratch.

Выберите оранжевую категорию **Данные** и нажмите кнопку **Создать список**, чтобы открыть окно **Новый список**. Присвойте списку имя **список** и установите переключатель в положение **Для всех спрайтов**.



Ниже кнопки **Создать список** появятся новые темно-оранжевые блоки.



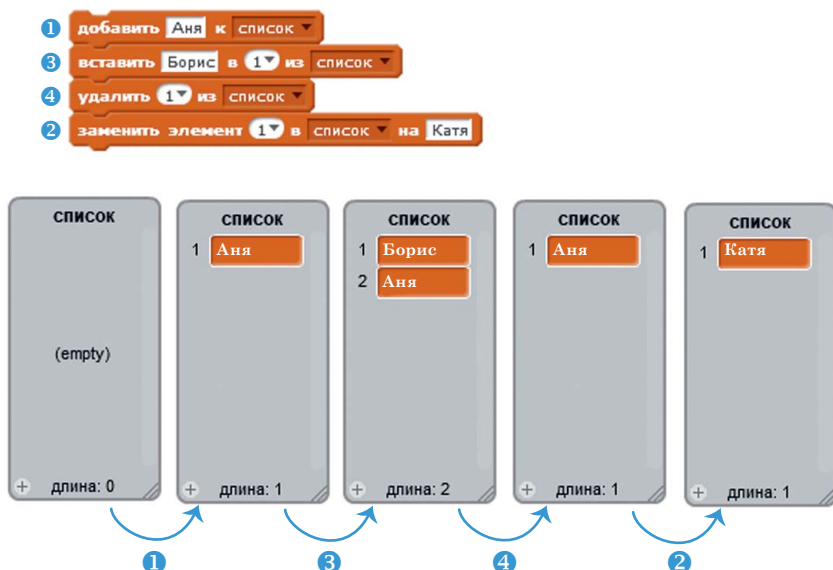
Списки имеют больше блоков, а не только **Задать значение** и **Изменить на**. Списки содержат значения, а каждое значение — *позицию*, поэтому значениям нужны разные типы блоков. Позиции в списке определяются числами; число 1 — первое значение в списке.

Управлять списками позволяют следующие блоки:

- ▶ блок **Добавить к** добавляет значение в конец списка;
- ▶ блок **Удалить из** удаляет значение из определенной позиции;
- ▶ блок **Вставить в из** похож на блок **Добавить к**, но здесь вы можете выбрать позицию, куда необходимо вставить значение;
- ▶ блок **Заменить элемент в** похож на сочетание **Удалить из** и **Вставить в из**. Он заменяет значение в определенной позиции новым значением.

Добавьте код, показанный на следующем рисунке, а затем дважды щелкните на нем, чтобы запустить.

Взгляните на то, как влияют эти блоки кода на список с именем **Список**.



Напоминаем, что вы можете использовать значение переменной в скругленном блоке. Такие блоки списка работают так же, но вам нужно указать позицию, чтобы скругленный блок определил, какое значение в списке он должен получить. Скрипты на следующем рисунке, как слева, так и справа, заставляют кота приветствовать вас, но в коде, в котором используется список (слева), необходимо указать позицию извлекаемого значения:

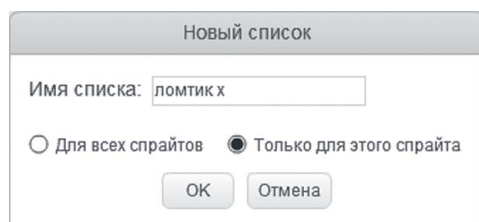




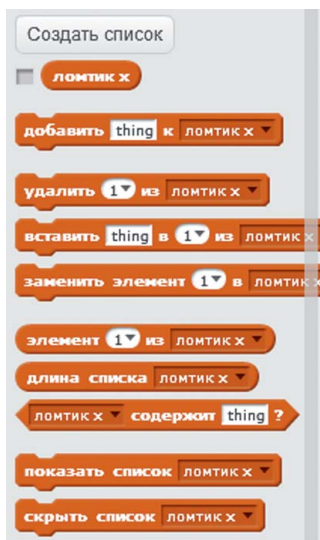
## 4. Создание списков и переменных для спрайта Ломтик

Для игры «Фруктовый ниндзя» вы создадите список, чтобы отслеживать, в какой позиции на экране игрок «разрезает» фрукты. Поскольку разрез тянется между множеством позиций с разными координатами  $x$  и  $y$  с помощью блоков категории **Перо**, использование списка подходит идеально.

Перейдите на вкладку **Скрипты** в верхней части области блоков и выберите оранжевую категорию Данные. Нажмите кнопку **Создать список**, чтобы открыть окно **Новый список**. Присвойте списку имя **Ломтик x** и выберите режим **Только для этого спрайта**.



Под кнопкой **Создать список** появятся новые темно-оранжевые блоки.

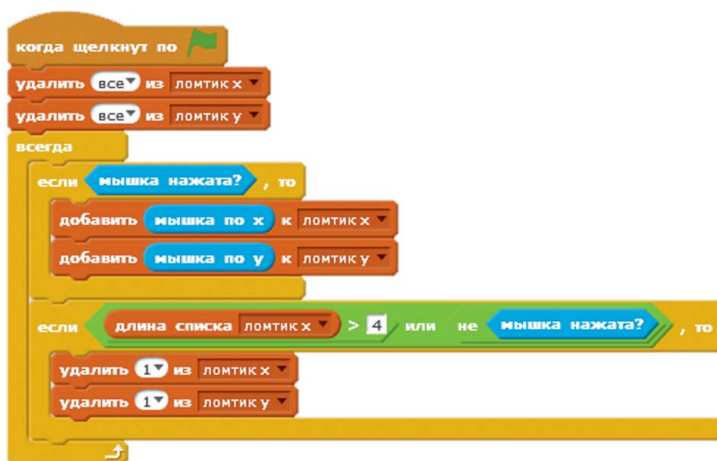


В оранжевой категории **Данные** нажмите кнопку **Создать список**, чтобы создать второй список для игры «Фруктовый ниндзя». Присвойте списку имя **Ломтик у** и выберите режим **Только для этого спрайта**.

Затем нажмите кнопку **Создать переменную**, создайте переменную с именем **i** и выберите режим **Только для этого спрайта**. Вы будете использовать эту переменную позднее. Спрайт **Ломтик** будет использовать два списка и переменную **i**, чтобы рисовать светло-голубой след.

## 5. Запись перемещений мыши

Вы будете использовать блоки **Перо** для рисования линии разреза. Но сначала вам нужно узнать, где рисовать линии. Добавьте код, показанный на следующем рисунке, в спрайт **Ломтик**:



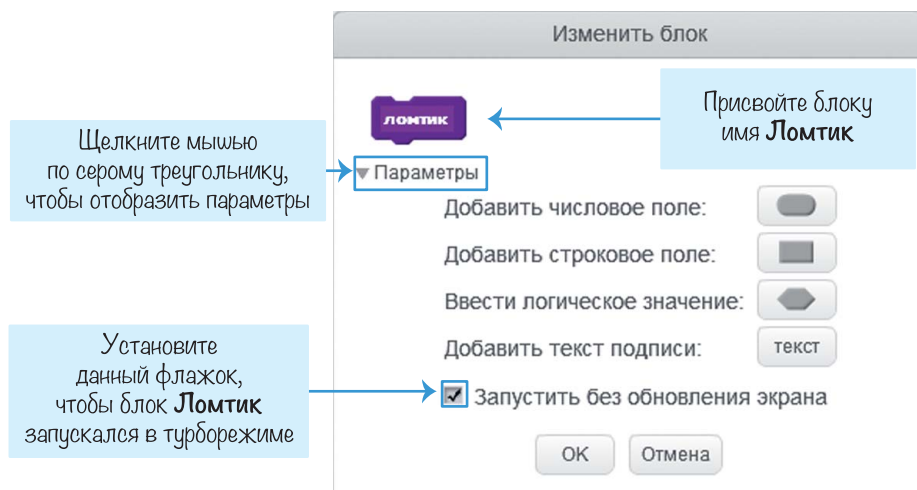
Когда игрок нажимает кнопку мыши и удерживает ее нажатой, координаты **x** и **y** мыши добавляются в конец списков **Ломтик x** и **Ломтик y**. Если в списках указано более четырех значений, значение в начале каждого списка будет удалено. То есть программа записывает последние четыре координаты **x** и **y** мыши. Первые значения также удаляются, когда кнопка мыши отпущена, поэтому след укорачивается, когда игрок отпускает кнопку мыши.

Этот скрипт передает правильные координаты для списков **Ломтик x** и **Ломтик y**. Следующий сценарий будет фактически рисовать след разрезания фрукта.

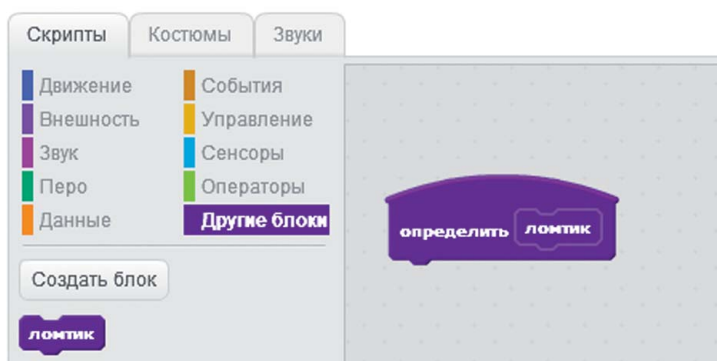
## 6. Создание пользовательского блока для рисования разреза

Выберите фиолетовую категорию **Другие блоки** и нажмите кнопку **Создать блок**, чтобы открыть окно **Новый блок**. Фиолетовые цвет имеют блоки, созданные вами самостоятельно в редакторе Scratch.

Укажите имя **Ломтик** для вашего нового фиолетового блока. Нажмите серый треугольник, чтобы развернуть группу элементов управления **Параметры**, и установите флажок **Запустить без обновления экрана**. Если этот флажок установлен, Scratch будет запускать код в вашем настраиваемом фиолетовом блоке в турборежиме, даже если игрок не включил этот режим нажатием кнопки в виде зеленого флага с зажатой клавишей **Shift**. Нажмите кнопку **ОК**, чтобы создать пользовательский блок.

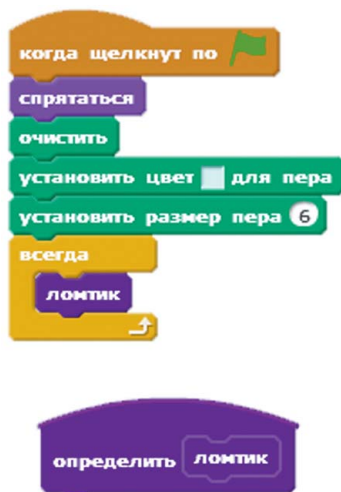


Новый блок **Ломтик** появится в фиолетовой категории **Другие блоки**.

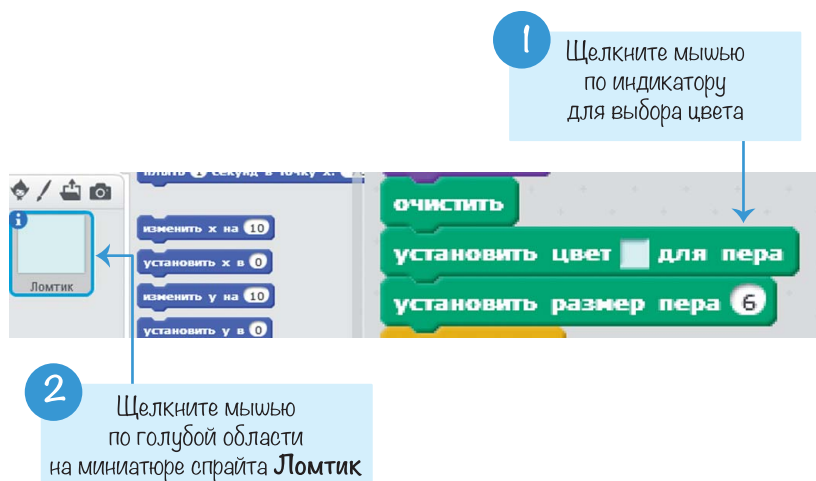


Этот новый блок с изогнутым верхом в области скриптов является *определяющим* для блока **Ломтик**. Когда вы хотите использовать этот новый определяющий блок, вы также должны перетащить *вызывающий* блок **Ломтик** из области блоков в код. Всякий раз, когда Scratch запускает этот вызывающий блок, запускается код в определяющем блоке.

Добавьте код, показанный на следующем рисунке, в спрайт **Ломтик**:

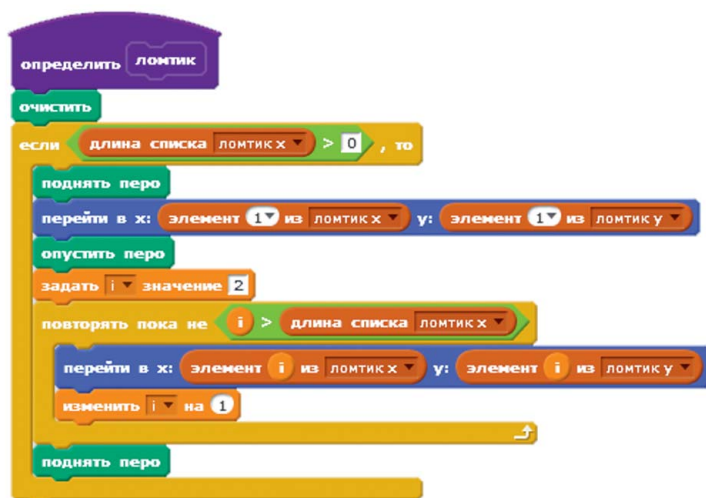


Индикатор в блоке **Установить цвет для пера** должен быть того же цвета, что и голубой прямоугольник спрайта **Ломтик**. Щелкните мышью по индикатору цвета в блоке, а затем выберите голубой цвет на миниатюре спрайта **Ломтик**.



Этот скрипт скрывает спрайт **Ломтик** и задает правильные размер и цвет пера. Последующий цикл **Всегда** за циклено запускает блок **Ломтик**.

Затем вам необходимо указать блок **Определить ломтик**. Вам нужно, чтобы блок **Ломтик** рисовал линию, начинающуюся с первых координат: первое значение в списке **Ломтик x** (для позиции по оси  $x$ ) и первое в списке **Ломтик y** (для позиции по оси  $y$ ). Эта линия продолжается в следующих позициях по осям  $x$  и  $y$  в списках **Ломтик x** и **Ломтик y**, затем в следующих значениях и так далее, пока не будет достигнута последняя пара значений. Наш код гарантирует, что длины списков **Ломтик x** и **Ломтик y** всегда одинаковы. Добавьте код, показанный на следующем рисунке, в блок **Определить ломтик**. Обратите внимание, что длина блока **Ломтик x** задается блоком темно-оранжевого цвета и относится к категории **Данные**, а не к зеленой категории **Операторы**.



Этот код поднимает перо, прежде чем перенести его в первые значения координат *x* и *y* в списках **Ломтик x** и **Ломтик y**, а затем опускает перо. В цикле **Повторять пока не** используется переменная с именем *i*, чтобы отслеживать, куда перо должно двигаться дальше.

После первой итерации цикла **Повторять пока не** спрайт переходит к значению 2 в списках **Ломтик x** и **Ломтик y**. Перо рисует линию, когда движется спрайт. Потом значение переменной *i* увеличивается на один во время следующей итерации цикла, и спрайт переходит к значению 3 в списках **Ломтик x** и **Ломтик y**. Цикл продолжает запускаться до тех пор, пока значение переменной *i* не превысит число значений в списке (т. е. длину списка).

**ПРИМЕЧАНИЕ.** *Весь код под блоком **Определить ломтик** можно было бы поместить в другой цикл **Всегда**, но тогда он не работал бы в турборежиме. Параметр **Запустить без обновления экрана**, который вы выбрали при создании блока **Ломтик**, разрешает турборежим для кода в блоке **Определить ломтик**. В противном случае рисование разреза будет слишком медленным для игры.*



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Удерживая нажатой кнопку мыши, переместите указатель мыши по сцене. Убедитесь, что вслед за указателем мыши появляется светлый голубой след и быстро исчезает, если отпустить кнопку мыши. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

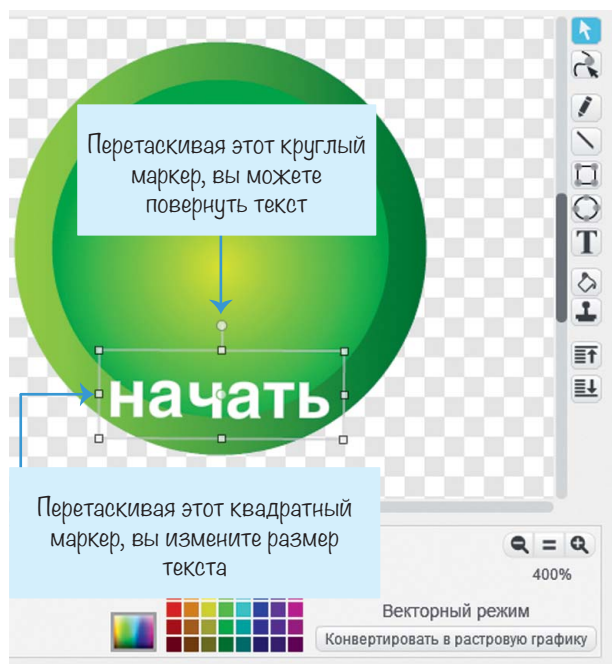
## В СОЗДАНИЕ КНОПКИ «НАЧАТЬ»

Игра «Фруктовый ниндзя» требует, чтобы у игрока была быстрая реакция. Давайте добавим кнопку на начальный экран, чтобы предоставить игроку шанс подготовиться к игре. Разрезание кнопки «Начать» с помощью мыши будет запускать игру.

### 7. Создание спрайта для кнопки «Начать»

Вам не нужно рисовать кнопку «Начать», потому что в Scratch уже есть готовая. Нажмите кнопку **Выбрать спрайт из библиотеки** рядом с надписью **Новый объект**. Выберите пункт **Button1** в окне **Библиотека спрайтов** и нажмите кнопку **ОК**. Далее переименуйте спрайт кнопки на панели информации, присвоив имя **Кнопка**.

Перейдите на вкладку **Костюмы** в верхней части области блоков. Выберите белый цвет; затем, используя инструмент **Текст**, введите слово «Начать» и щелкните в любой позиции на экране, но не по тексту. Затем измените размер текста и перетащите его так, чтобы он располагался в нижней части зеленой кнопки.

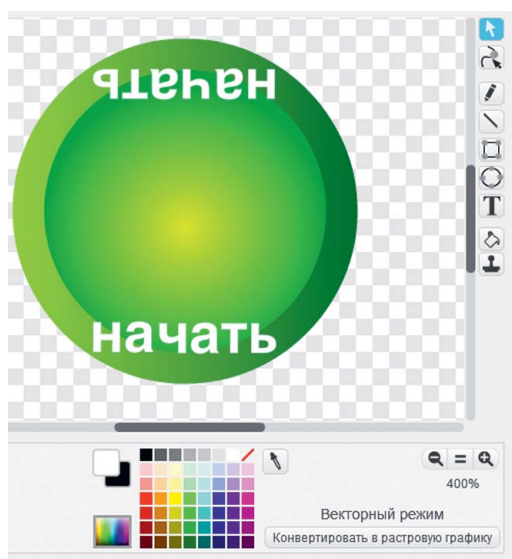


**ПРИМЕЧАНИЕ.** Обратите внимание, что русский текст не поддерживается напрямую в редакторе Scratch, поэтому потребуется воспользоваться сторонним приложением. Для этого перейдите на сайт [scratch.mit.edu/projects/60521212/](http://scratch.mit.edu/projects/60521212/) и нажмите кнопку в виде зеленого флага или клавишу **Пробел**. Введите текст в появившееся поле и нажмите клавишу **Enter** или синюю кнопку справа от поля. Скопируйте текст из второй строки в виде символов типа `Âàñëëëéé`, выделив его и нажав сочетание клавиш **Ctrl+C**. Перейдите в редактор Scratch и, выбрав инструмент **Текст** и шрифт **Helvetica**, вставьте скопированный текст с помощью сочетания клавиш **Ctrl+V**. Обратите внимание, что в редакторе Scratch для русских букв поддерживается только шрифт **Helvetica**.

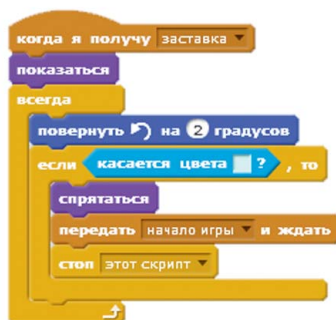
Создайте надпись «Начать», вновь используя инструмент **Текст**. И снова щелкните где-нибудь на экране, но не на тексте. Нажав и удерживая кнопку мыши на верхнем маркере текстовой рамки, поверните текст так, чтобы он



стал перевернутым. Затем поместите его в верхней части зеленой кнопки.

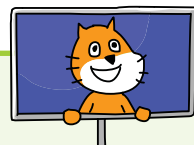


Добавьте код, показанный на следующем рисунке, в спрайт **Кнопка**:



Для блока **Касается цвета ?** установите тот же цвет, что и у миниатюры спрайта **Ломтик** в области спрайтов. Щелкните мышью по цветному индикатору в блоке и выберите голубой цвет так же, как вы это делали в шаге 6. Кнопка «Начать» будет медленно поворачиваться на экране, пока игрок ее не разрежет. Программа поймет, что кнопка была разрезана, когда ее коснется голубой цвет разреза. В этот момент она передаст сообщение о начале игры всем остальным спрайтам.

## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Установите указатель мыши на кнопку «Начать». Нажав и удерживая кнопку мыши, разрежьте кнопку, чтобы убедиться, что она исчезает, а фон меняется. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

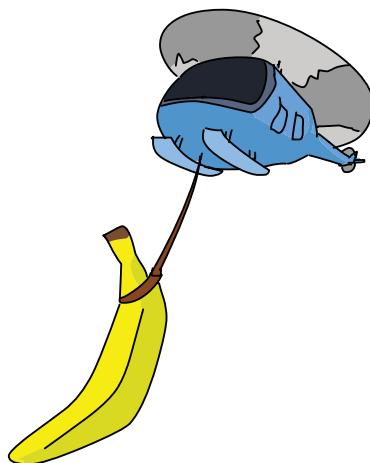
## Г СОЗДАНИЕ ДВИЖУЩИХСЯ ФРУКТОВ И БОМБ

Поскольку фрукты в игре выглядят по-разному, но делают одно и то же, вы будете использовать один спрайт с тем же кодом, но с разными костюмами.

### 8. Создание спрайта фруктов

Нажмите кнопку **Выбрать спрайт из библиотеки** рядом с надписью **Новый спрайт** и выберите пункт **Apple** в окне **Библиотека спрайтов**. Затем нажмите кнопку **ОК**. Щелкните мышью по значку **i** созданного спрайта, чтобы открыть панель информации, и присвойте спрайту имя **Фрукт**.

Перейдите на вкладку **Костюмы** спрайта **Фрукт**. Нажмите кнопку **Выбрать костюм из библиотеки** рядом с текстом **Новый костюм**. Добавьте костюм **Bananas**. Нажмите кнопку еще раз и добавьте костюм **Orange**. Затем снова нажмите

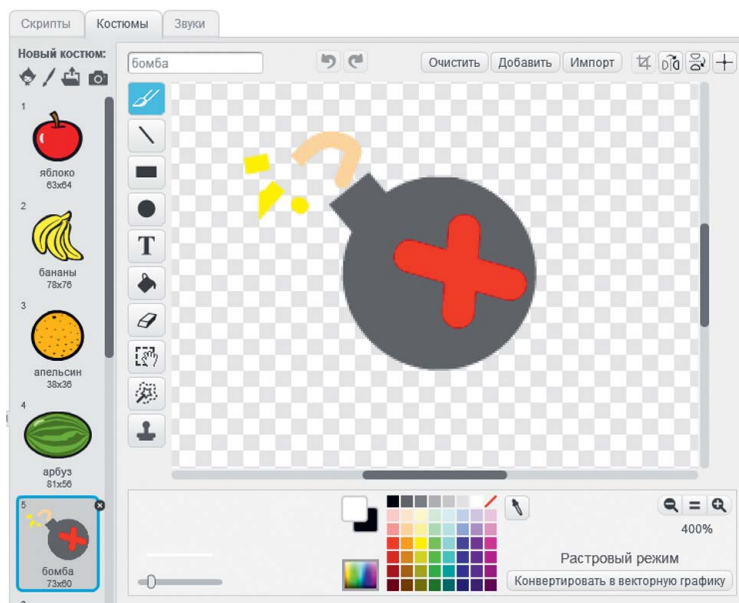


кнопку **Выбрать костюм из библиотеки**, чтобы добавить костюм **Watermelon-a**. Для удобства вы можете переименовать костюмы, присвоив им имена **Яблоко**, **Бананы**, **Апельсин** и **Арбуз**, соответственно. Убедитесь, что порядок костюмов такой же, как показано на следующем рисунке. Если это не так, перетащите костюмы в правильном порядке.



В библиотеке нет костюма для бомб, поэтому вам придется нарисовать его для игры «Фруктовый ниндзя». Если игрок случайно разрезает бомбу, он проигрывает игру. В графическом редакторе нажмите кнопку **Нарисовать новый костюм** рядом с надписью **Новый костюм**.

Нарисуйте бомбу с красным крестиком на ней. Используйте инструмент **Круг**, чтобы нарисовать основную часть бомбы и инструмент **Линия**, чтобы нарисовать верхнюю часть и крестик. Чтобы сделать линии более толстыми или, наоборот, узкими, перемещайте ползунковый регулятор **Ширина линии**. Присвойте этому костюму имя **Бомба**.



## 9. Создание костюмов разрезанных фруктов

Спрайт **Фрукт** будет содержать костюмы для четырех фруктов, бомбы и разрезанных фруктов. Этот шаг прост, потому что вам не придется ничего рисовать. Вы просто продублируете костюмы и разделите их на части, чтобы они выглядели аккуратно нарезанными.

В графическом редакторе щелкните правой кнопкой мыши по костюму **Яблоко** и выберите команду **Дублировать** в контекстном меню, чтобы создать второй костюм для яблока.



Также продублируйте костюмы **Бананы**, **Апельсин**, **Арбуз** и **Бомба**. Убедитесь, что они находятся в том же порядке, что и исходные костюмы:

1. Яблоко
2. Бананы
3. Апельсин
4. Арбуз
5. Бомба
6. Яблоко2
7. Бананы2
8. Апельсин2
9. Арбуз2
10. Бомба2

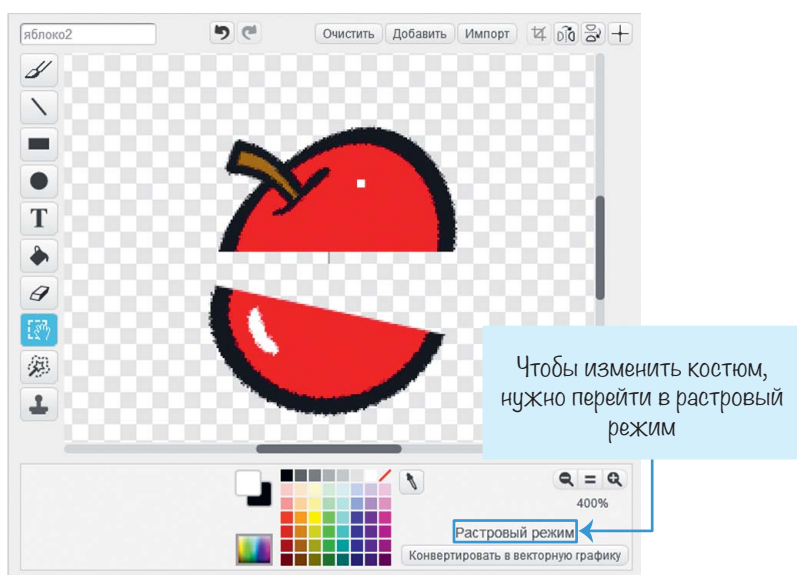
**ПРИМЕЧАНИЕ.** *Дважды проверьте порядок расположения костюмов! В дальнейшем исправить порядок, в котором расположены костюмы, будет трудно. Проявив сейчас внимательность, вы избавите себя от проблем в будущем.*

Чтобы код работал с каждым фруктом, порядковый номер костюма нарезанного фрукта будет на пять единиц больше, чем номер костюма целого фрукта. Таким образом, костюм яблока сохранен под номером 1, а костюм нарезанного яблока – под номером 6. Костюм бананов имеет номер 2, а костюм нарезанных бананов – номер 7. Вот почему порядок костюмов так важен. Кроме того, вам нужен второй костюм бомбы под номером 10. Продублируйте костюм **Бомба**, но рисовать разрезанный вариант не нужно.

Костюмы для фруктов были нарисованы в Scratch в векторном режиме, т.е. в виде набора фигур, а не пикселей. И хотя векторные изображения выглядят лучше растровых (пиксельных), векторные изображения нельзя разделить пополам в графическом редакторе. Поэтому сначала

вам нужно конвертировать костюмы в формат растровой графики.

Для каждого продублированного костюма фруктов (костюмы 6, 7, 8 и 9) выберите костюм и нажмите кнопку **Конвертировать в растровую графику** в правом нижнем углу графического редактора. Затем используйте инструмент **Выбрать** и растяните прямоугольник выделения на половину костюма. Теперь немного передвиньте выделенную половину и немного ее поверните, чтобы фрукт выглядел так, как будто он разрезан пополам.



Вы также можете выделить и немного повернуть вторую половину. Повторите эту последовательность действий для всех дублированных костюмов.



## Ю. Добавление кода в спрайт **Фрукт**

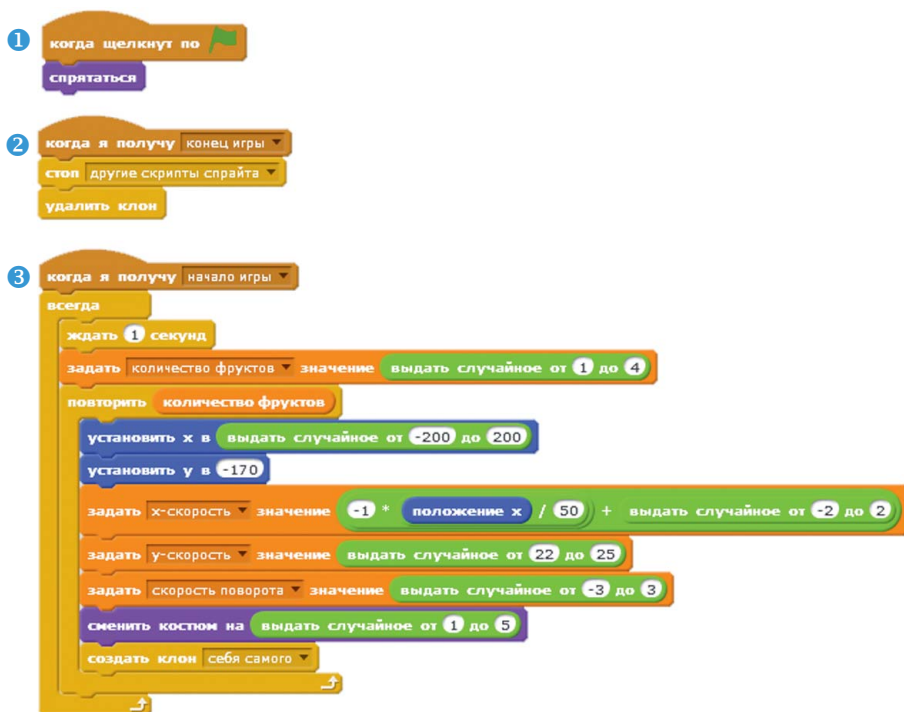
Спрайт **Фрукт** теперь содержит все необходимые костюмы. Далее добавим код, который будет определять его поведение. Спрайт **Фрукт** создает клоны себя самого, и каждый клон будет случайным образом выбирать костюм 1, 2, 3, 4 или 5. Затем клон сам себя подбрасывает в воздух. Если клон обнаружит, что был разрезан, то переключится на разрезанный костюм. Или если клон случайно выбрал костюм бомбы, а игрок его разрежет, игра закончится.

Создайте переменные **х-скорость**, **у-скорость**, **Скорость поворота** и **Количество фруктов**, выбрав для всех режим **Только для этого спрайта**. Все клоны будут использовать эти переменные, чтобы определить, как они подбрасываются и падают. Убедитесь, что вы выбрали режим **Только для этого спрайта**. В этой программе режим **Для всех спрайтов** используется только для переменных **Счет** и **Рекорд**.

Если бомба была разрезана, игра оповестит пользователя звуковым уведомлением. Перейдите на вкладку **Звуки** и нажмите кнопку **Выбрать звук из библиотеки** под строкой **Новый звук**. Выберите файл **Alien squeak2** и нажмите кнопку **ОК**. Для удобства вы можете переименовать звуковой файл, присвоив ему имя **Вжик**.

Определив переменные и настроив звуковое сопровождение, теперь вы можете добавить код, позволяющий исходному спрайту **Фрукт** создавать по несколько своих клонов каждую секунду. Код для клонов вы добавите в шаге 11.

Добавьте код, показанный на следующем рисунке, в спрайт **Фрукт**.



Скрипт ① скрывает фрукты в начале игры. При получении сообщения **Конец игры**, то есть когда игра заканчивается, скрипт ② очищает сцену, удаляя клоны. Скрипт ③ запускается, когда игрок разрезает кнопку «Начать» и начинает создавать клоны.

Давайте подробнее рассмотрим, как работает скрипт ③. После паузы в 1 секунду значение переменной **Количество фруктов** устанавливается равным 1, 2, 3 или 4. Код в цикле **Повторить количество фруктов** создает новый клон при каждой итерации. Внутри цикла исходный спрайт **Фрукт** определяет свое положение со случайной позицией по оси x и позицией по оси y равной -170, размещая исходный спрайт **Фрукт** в нижней части сцены.

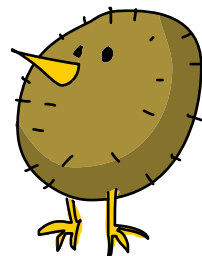
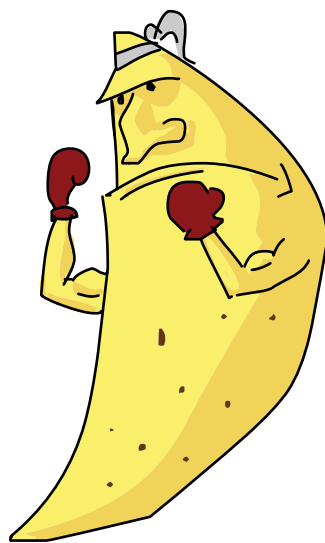
Затем переменным **x-скорость**, **y-скорость** и **Скорость поворота** присваиваются произвольные значения



(вот почему клонированные фрукты будут выбрасываться случайным образом). Костюм спрайта выбирается также случайным образом из костюмов 1, 2, 3, 4 или 5. Весь этот код позволяет спрайту **Фрукт** создавать клоны себя самого. Положение, переменные и костюм клона копируются из исходного спрайта **Фрукт**, что означает, что исходник готов случайным образом настроить себя для следующего клона.

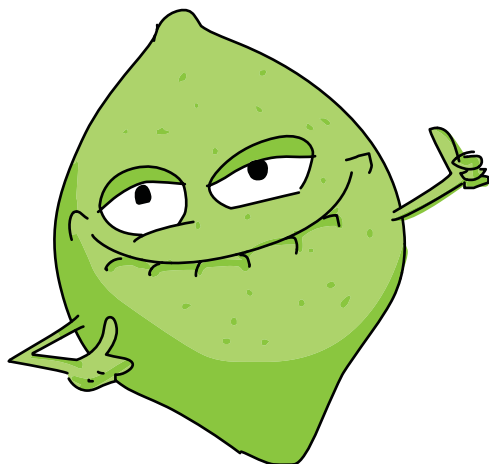
Блок **Установить x-скорость** на использует сложное уравнение, присваивающее значение  $-1 * \text{положение } x / 50 + \text{выдать случайное от } -2 \text{ до } 2$  переменной **x-скорость**. Значение переменной **x-скорость** определяет, как далеко слева или справа выбрасывается фрукт. Значение должно быть небольшим, поэтому **Положение x** спрайта (случайное число от -200 до 200) делится на 50, что соответствует числу в диапазоне от -4 до 4.

Нам нужно, чтобы фрукт был брошен в центр сцены. Это означает, что, если спрайт **Фрукт** находится на левой стороне сцены, его положение по оси *x* будет отрицательным, а клон должен быть выброшен вправо. Другими словами, переменная **x-скорость** должна иметь положительное значение. Если спрайт **Фрукт** находится справа (с положительным значением положения по оси *x*), его следует выбросить влево. Это означает, что значение **Положение x / 50** должно быть умножено на -1. Поэтому, если позиция по оси *x* положительна, значение переменной **x-скорость** будет отрицательным, и наоборот. Умножение значения переменной **x-скорость** на -1 гарантирует, что фрукт с левой стороны сцены будет брошен направо, а фрукт с правой стороны будет выброшен влево.



Чтобы внести некоторые изменения в броски, добавляется случайное число в диапазоне от  $-2$  до  $2$ . Вы увидите, как клоны используют значение переменной **х-скорость** в шаге 11.

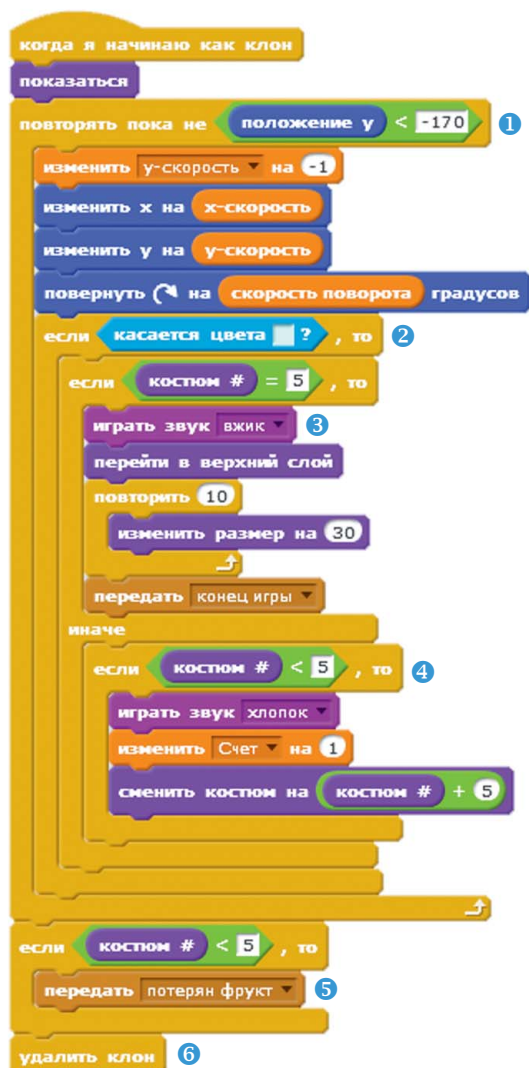
Значения переменных **х-скорость** и **у-скорость** используются вместе, поэтому траектория полета фрукта имеет изогнутую форму, называемую *параболой*. Она используется во многих научных и инженерных приложениях, но вам не нужно детально изучать параболы, чтобы использовать их в ваших играх. Не беспокойтесь, если вы не понимаете математику, зашифрованную в этих блоках кода. Если вы добавляете блоки точно так, как показано в этой книге, ваши фрукты будут вылетать правильно.



## II. Добавление кода для клонов спрайта **Фрукт**

Когда исходный спрайт **Фрукт** создает клон себя самого, клон запускает собственный код, который отвечает за подбрасывание этого клона и определение, разрезан ли он.

Сначала создайте сообщение **Потерян фрукт**, щелкнув мышью по черному треугольнику в блоке **передать**, выбрав в раскрывающемся списке пункт **Новое сообщение** и введя название **Потерян фрукт**. Затем добавьте код, показанный на следующем рисунке, в спрайт **Фрукт**.



Этот скрипт управляет клонами фруктов и выглядит довольно сложным, поэтому давайте разберем его шаг за шагом. Исходный спрайт **Фрукт** скрыт, поэтому первое, что делает клон, — это отображает себя.

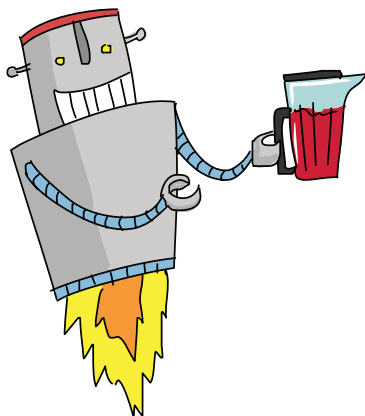
Затем код для определения гравитации и разрезания повторяется в цикле ① так долго, пока клон находится в воздухе. Когда значение положения по оси *y* клона становится меньше  $-170$ , клон падает за нижний край сцены.

Первая часть блока **Повторять пока не** заставляет клон падать быстрее под действием силы тяжести, изменяя значение переменной **у-скорость** на  $-1$ . Затем положение по оси  $x$ , положение по оси  $y$  и направление движения клона заменяются значениями переменных **х-скорость**, **у-скорость** и **Скорость поворота**, соответственно, перемещая фрукт по траектории параболы в воздухе. Форма параболы получается, когда горизонтальная скорость (в данном случае значение переменной **х-скорость**) не изменяется, но вертикальная скорость (в данном случае значение переменной **у-скорость**) уменьшается с течением времени.

Цвет индикатора в блоке **Касается цвета ?** должен соответствовать цвету спрайта **Ломтик** ②. Клон определяет, был ли он разрезан, проверив, касается ли он цвета следа спрайта **Ломтик** (см. шаг 6). Если он прикасается к этому цвету, блок **Если то иначе** проверяет, выбран ли номер костюма клона 5 (это номер костюма бомбы). Если бомба была разрезана, клон **Фрукт** воспроизводит звук **Вжик** ③, увеличивается на экране, запустив блок **Изменить размер на 30**, а затем передает сообщение **Конец игры**.

В том случае, если номер костюма разрезанного клона не является 5 (то есть если это не костюм **Бомба**), то запускается код в части **Иначе** блока **Если то иначе** ④. Блоки **Если то** проверяют, какой из четырех костюмов фруктов выбран клоном, заменяют костюм на разрезанный костюм того же фрукта и добавляют 1 к значению переменной **Счет**.

После блока **Повторять пока не** положение  $y < -170$ , если по-прежнему выбран один из цельных костюмов (то есть костюмы с 1 по 4), выводится сообщение **Потерян фрукт** ⑤ (мы рассмотрим код, который получает это сообще-



ние, на этапе 12). Независимо от того, является ли этот костюм разрезанным или цельным, на этом этапе клон удаляется ⑥.

## КОНТРОЛЬНАЯ ТОЧКА



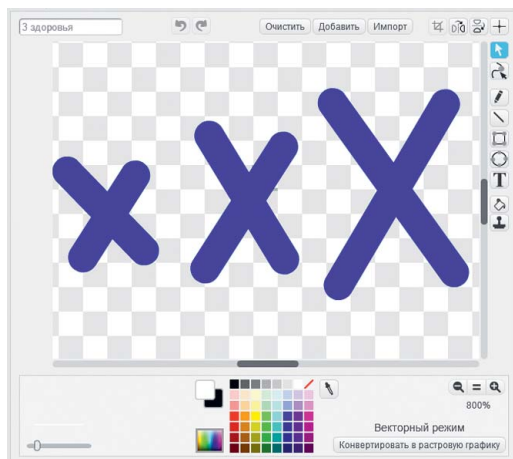
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Попробуйте разрезать фрукты. Убедитесь, что внешний вид фруктов изменяется, они становятся разрезанными, а значение переменной **Счет** увеличивается на 1. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## А СОЗДАНИЕ СПРАЙТА ЗДОРОВЬЯ

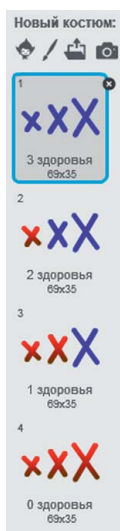
В этой игре не стоит делать разрез через весь экран, потому что можно случайно наткнуться на бомбу. Но чтобы игрок не *слишком* осторожничал, разрешим ему пропустить три фрукта, прежде чем он проиграет.

### 12. Создание спрайта здоровья

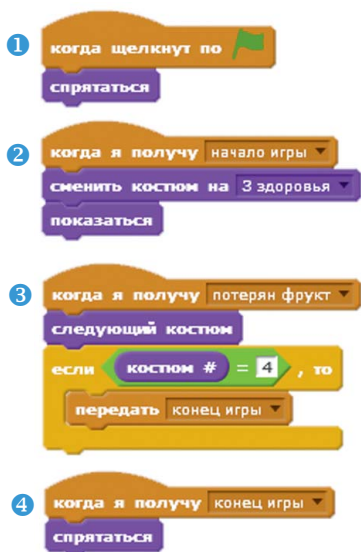
Спрайт **Здоровье** показывает, сколько еще фруктов игрок может пропустить, прежде чем проиграет. Каждый раз, когда игрок пропускает фрукт, спрайт переключается на следующий костюм. Нажмите кнопку **Нарисовать новый спрайт** рядом с названием раздела **Новый спрайт**. Откройте панель информации нового спрайта и присвойте ему имя **Здоровье**. В графическом редакторе используйте инструмент **Линия**, чтобы нарисовать три синих буквы «X», и дублируйте этот костюм еще три раза. Я увеличил размер букв «X» слева направо, но вы можете выбрать такой стиль, какой вам нравится.



Переименуйте **Костюм1** в **3 здоровья**, **Костюм2** в **2 здоровья**, **Костюм3** в **1 здоровье**, и **Костюм4** в **0 здоровья**. Оставьте синими все три буквы «X» в костюме **3 здоровья**. Для других костюмов используйте инструмент **Заливка**, выберите вертикальный градиент, а затем темный и светлый оттенки красного цвета, чтобы раскрасить красным градиентом буквы «X». Костюм **2 здоровья** будет иметь одну красную букву «X», **1 здоровье** – две, а костюм **0 здоровья** – три. Убедитесь, что ваши костюмы находятся в том же порядке, что и на рисунке, показанном ниже:



Спрайт **Здоровье** будет переключать костюмы, ориентируясь на сообщения, которые он получает. Добавьте код, показанный на следующем рисунке, в спрайт **Здоровье**. На этом этапе все сообщения должны уже существовать, поэтому создавать их не нужно.



Когда игрок находится на начальном экране и нажимает кнопку в виде зеленого флага, спрайт **Здоровье** прячется, что вы можете увидеть в скрипте ①.

Когда игрок разрезает спрайт **Кнопка**, передается сообщение о начале игры, в скрипте ② спрайта **Здоровье** определено отображение костюма **3 здоровья**. Каждый раз, когда неразрезанный фрукт падает за нижнюю границу сцены, он передает сообщение **Потерян фрукт**, в результате чего спрайт **Здоровье** переходит к следующему костюму в скрипте ③.

Этот следующий костюм будет содержать одну красную букву «Х». Если спрайт **Здоровье** переходит к костюму **0 здоровья**, все три буквы «Х» становятся кра-



сного цвета и передается сообщение **Конец игры**. Когда спрайт **Здоровье** получает это сообщение, он скрывается с помощью очень короткого скрипта 4.



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить код, который уже готов. Пусть некоторые фрукты упадут неразрезанными. Удостоверьтесь, что значки здоровья «X» становятся красными при каждом пропущенном фрукте. Начните новую игру и убедитесь, что значки «X» снова синего цвета. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## Е ПОДГОТОВКА КОНЦОВКИ ИГРЫ

В этой игре проиграть можно двумя разными способами: пропустить три фрукта или разрезать бомбу. В любом из этих случаев передается сообщение **Конец игры**, в результате чего происходит несколько действий. Клоны спрайта **Фрукт** удаляют себя, спрайт **Здоровье** скрывается, а сцена окрашивается в белый свет. Мы уже добавили код для клонов и спрайта **Здоровье**. Теперь давайте добавим код, чтобы сцена затухала в белый экран.

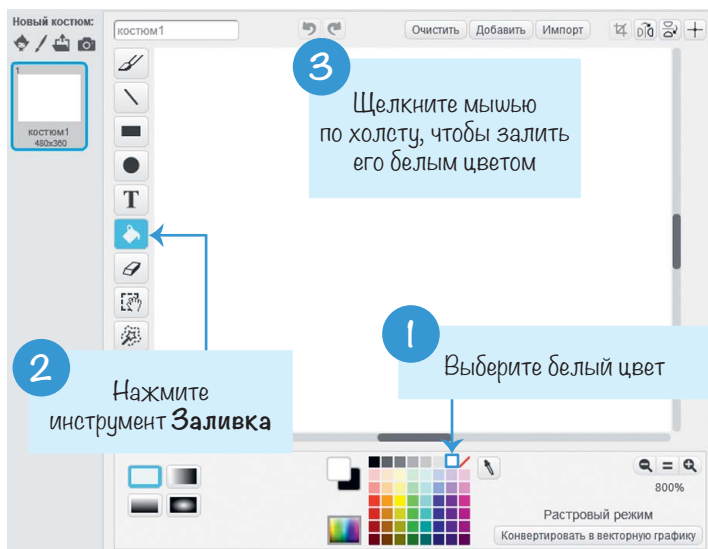


### 13. Создание спрайта Затухание

Мы применим эффект **Призрак**, чтобы сцена исчезала. Нажмите кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**. На панели информации присвойте



созданному спрайту имя **Затухание**. В графическом редакторе используйте инструмент **Заливка**, чтобы заполнить весь холст белым. На рабочей области не должно быть ни одного пикселя другого цвета.



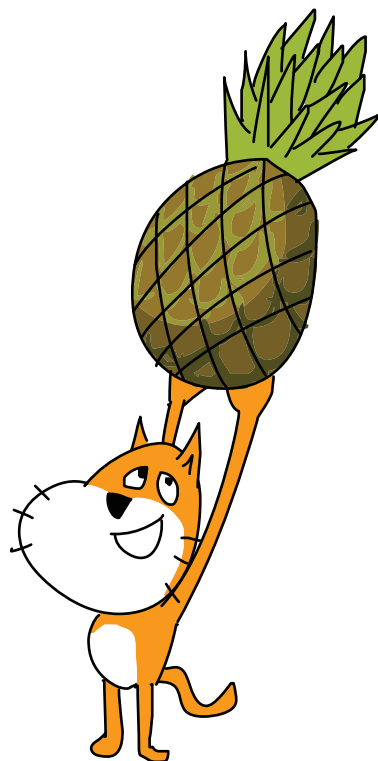
Спрайт **Затухание** блокирует большую часть сцены. Когда запускается программа, спрайт **Затухание** скрывается и становится видимым только тогда, когда передается сообщение **Конец игры**. Добавьте код, показанный на следующем рисунке, в спрайт **Затухание**:



Когда спрайт **Затухание** получает сообщение **Конец игры**, он перемещается в начало координат  $x$  и  $y$  (0, 0) ❶. Поскольку спрайт имеет тот же размер, что и сцена (480 пикселей в ширину, 360 пикселей в высоту), размещение его в начале координат полностью перекрывает сцену. Блок **Перейти в верхний слой**, в свою очередь, помещает его поверх любого другого спрайта. Это гарантирует, что все остальные спрайты находятся позади него.

Прежде чем он станет видимым, спрайт **Затухание** устанавливает эффект **Призрак** со значением 100 ❷, становясь полностью невидимым. Затем цикл **Повторить** уменьшает эффект **Призрак** на 10 ❸. Спрайт **Затухание** постепенно становится более заметным, и сцена постепенно погружается в белый цвет.

Если значение переменной **Счет** больше переменной **Рекорд**, значение переменной **Рекорд** обновляется ❹. После короткой, в 1 секунду, задержки спрайт **Затухание** передает сообщение **Заставка**. Спрайт **Затухание** скрывается, на сцене появляется начальная заставка и спрайт **Кнопка** ❺. Игра выглядит так, как если бы игрок нажал кнопку в виде зеленого флага, но переменной **Рекорд** было присвоено значение, равное самому высокому счету, достигнутому к этому моменту.





## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Запустите игру и убедитесь, что, когда вы разрезаете бомбу или когда три фрукта падают неразрезанными, спрайт **Затухание** постепенно становится видимым и заново запускает игру. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Код для этой программы слишком велик, чтобы полностью показать его в этой книге. Полностью готовый код вы можете просмотреть в архиве с примерами, открыв файл *Фруктовый\_ниндзя\_3.sb2*.

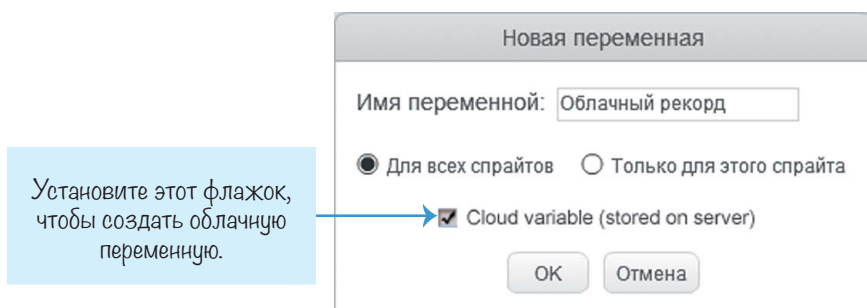
## ВЕРСИЯ 2.0: ИГРОВОЙ СЧЕТ

Scratch позволяет хранить переменные на своем сайте, используя *облачные переменные*. Облачные переменные Scratch действуют так же, как и обычные, за исключением того, что их значения сохраняются, даже если веб-браузер был закрыт. Каждый пользователь Scratch может получить доступ к облачным переменным с помощью программы Scratch. Вы можете устроить соревнование за рекорд, используя облачные переменные для всех, кто играет в вашу игру!

Поскольку облачные переменные требуют передачи существенного объема данных через сайт Scratch, есть некоторые ограничения по их использованию:

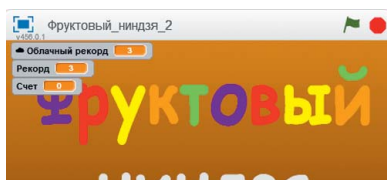
- ▶ облачные переменные хранят только числа, а не текст;
- ▶ облачные переменные нельзя использовать для создания чатов или многопользовательских игр;
- ▶ требуется второе или оба значения облачной переменной для его обновления;
- ▶ новые пользователи Scratch, не использовавшие еще сайт, не могут использовать облачные переменные.

Значение переменной **Рекорд** отображает наибольшее количество очков, которое набрал игрок. Использование облачной переменной позволяет показать рекордный счет каждого игрока на сайте Scratch. Выберите спрайт **Затухание** и нажмите кнопку **Создать переменную** в оранжевой категории **Данные**. Введите имя **Облачный рекорд** и установите переключатель в положение **Для всех спрайтов** (так же, как **Рекорд**). Затем установите флажок **Cloud variable (stored on server)**.

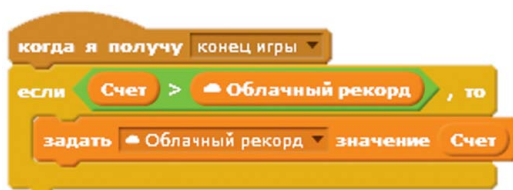


**ПРИМЕЧАНИЕ.** Если флажок *Cloud variable* не отображается, это означает, что у вас все еще статус новичка на сайте Scratch и вы сможете использовать облачные переменные только через некоторое время. Создавайте больше проектов, обсуждайте на дискуссионных форумах и комментируйте проекты других пользователей Scratch. Затем подождите несколько дней, чтобы узнать, изменился ли ваш статус и стала ли вам доступна возможность использования облачных переменных.

Теперь, когда вы создали переменную **Облачный рекорд**, сделайте так, чтобы она отображалась на сцене выше обычной переменной **Рекорд**.



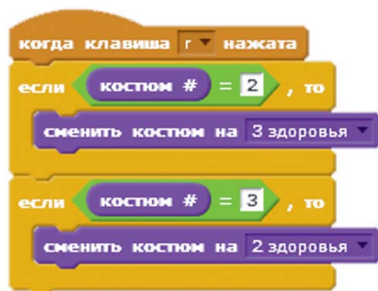
Затем добавьте код, показанный на следующем рисунке, в спрайт **Затухание**:



Блок **Задать Облачный рекорд значение Счет** изменяет значение переменной **Облачный рекорд** для текущего, а также любого другого игрока на веб-сайте Scratch. Теперь у вас есть система рекордов для всех игроков!

## ЧИТ-РЕЖИМ: ВОССТАНОВЛЕНИЕ ЗДОРОВЬЯ

Вы можете добавить секретную кнопку, которая позволит игроку восстановить потерянное здоровье. Конечно, это не защитит игрока от случайно разрезанной бомбы, но это поможет ему избежать потери здоровья от большого количества неразрезанных фруктов. Нажатие клавиши **R** будет восстанавливать один пункт здоровья после того, как вы добавите код, показанный на следующем рисунке, в спрайт **Здоровье**. Во время игры вы можете нажимать клавишу **R** столько раз, сколько будет нужно. Добавьте код, показанный на следующем рисунке, в спрайт **Здоровье**:





## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. После того как вы уронили целый фрукт, нажмите клавишу **R** и убедитесь, что одно очко здоровья восстановлено. Вы увидите, что цвет значка здоровья «X» сменился с красного на синий. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## ЗАКЛЮЧЕНИЕ

В этой главе вы создали игру, в которой:

- ▶ есть игровая заставка, с которой начинается игра, вместо того, чтобы запускать игру нажатием зеленого флажка;
- ▶ используются градиенты для заполнения фонов и костюмов;
- ▶ используются списки для отслеживания нескольких значений;
- ▶ отслеживается положение мыши с помощью блоков **Мышка по x** и **Мышка по y**;
- ▶ используются костюмы, чтобы клоны одного спрайта выглядели как разные фрукты;
- ▶ происходит не просто завершение программы, а возврат на начальный экран;
- ▶ сохраняется наивысший счет игрока.

Эта глава была намного длиннее предыдущих глав, потому что игра «Фруктовый ниндзя» более сложная, чем предыдущие игры. Но я думаю, что на данный момент это одна из самых забавных игр в этой книге. По мере того, как вы будете больше узнавать о программировании, вы

сможете делать все более продвинутые игры, так что готовьтесь!

В главе 8 вы создадите космическую игру, в которой игроку нужно будет сбивать прилетающие астероиды. Давайте взрывать!

## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно подсмотреть в конце книги.)

1. В какой категории можно создавать пользовательские блоки?
2. В чем разница между вызывающим и определяющим блоками для пользовательского блока?
3. За что отвечает параметр **Запуск без обновления экрана** пользовательских блоков?
4. Для чего используется блок **Перейти в верхний слой**?
5. В чем разница между списком и переменной?
6. Чем облачная переменная отличается от обычной переменной?
7. Почему флажок **Cloud variable** может не отображаться?
8. Могут ли облачные переменные хранить текст?
9. В области спрайтов есть спрайт с именем **Спрайт1**. Как его переименовать?



8

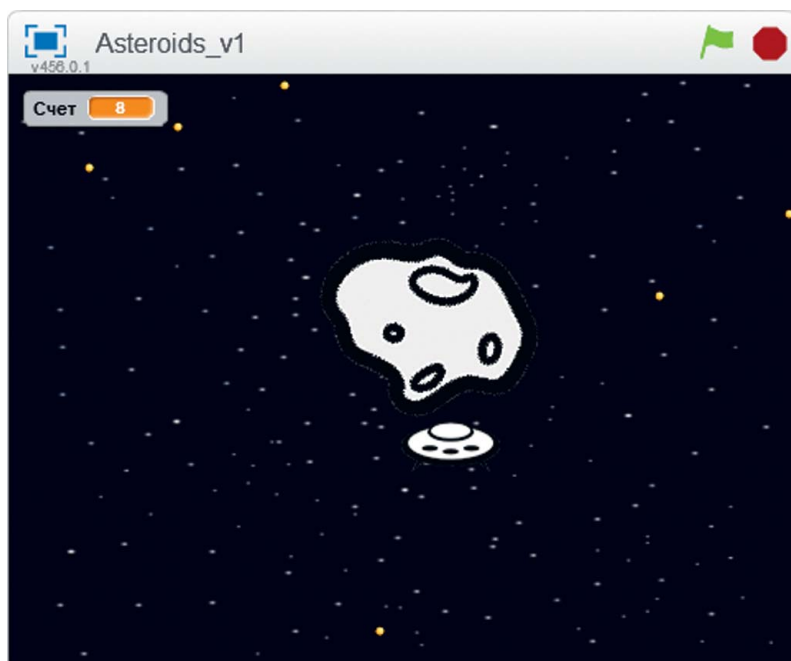
## УНИЧТОЖИТЕЛЬ АСТЕРОИДОВ... В КОСМОСЕ!

**A**steroids — классическая игра, разработанная в 1979 году компанией Atari. С тех пор многие программисты переделывали эту игру, являющуюся также отличным проектом для программирования в Scratch. Игрок пилотирует космолет, который должен уничтожить космические астероиды, избегая при этом столкновения с обломками... в космосе! (Хорошо известно, что добавление «... в космосе!» делает любую игру круче.)



Вместо того чтобы непосредственно управлять движением космолета, игрок *толкает* космолет, подобно хоккейной шайбе на льду, и, поскольку космолет игрока обладает инерцией, он скользит по сцене. Чтобы замедлить космолет, игроки должны толкнуть его в противоположном направлении. Тут требуется умение передвигать космолет без потери управления, но это только половина веселья. Другая половина – взрывать астероиды.

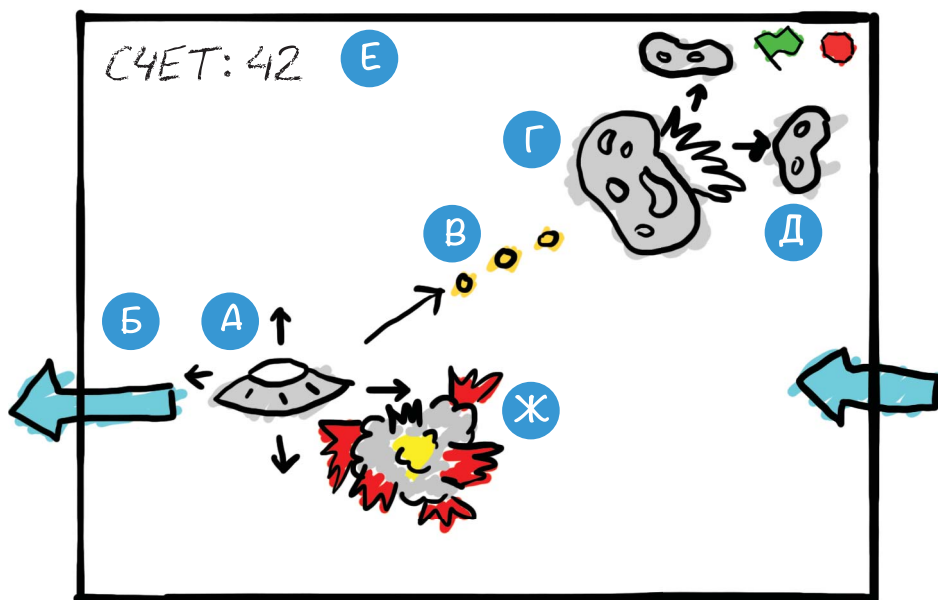
Прежде чем приступить к написанию кода, посмотрите окончательную версию игры на сайте [scratch.mit.edu/studios/4188596/](https://scratch.mit.edu/studios/4188596/).



## ЭСКИЗ ПРОЕКТА

Нарисуем на бумаге, как должна выглядеть игра. В нашей версии игры игрок управляет космолетом клавишами **W**, **A**, **S** и **D** и целится в летящие астероиды с помощью мыши.

Вот как выглядит мой эскиз:



И вот что мы будем делать в каждой части:

- А. Создание движущегося космолета
- Б. Выход космолета за края сцены
- В. Прицеливание с помощью мыши и стрельба клавишей **Пробел**
- Г. Создание астероидов
- Д. Создание астероидов, раскалывающихся надвое при попадании
- Е. Ведение счета и создание таймера
- Ж. Взрыв космолета при столкновении с астероидом

Если вы хотите сэкономить время, вы можете начать с файла скелета проекта с именем *Глава 08/Скелет\_проекта.sb2*, находящийся в запакованном файле с примерами, который вы скачали ранее по ссылке [https://eksmo.ru/files/Scratch\\_Sweigart.zip](https://eksmo.ru/files/Scratch_Sweigart.zip). В файл скелета проекта уже загружены все спрайты, так что вам нужно всего лишь перетаскивать блоки кода в каждый спрайт.

## **А** СОЗДАНИЕ ДВИЖУЩЕГОСЯ КОСМОЛЕТА

Прежде чем создавать код для игры, нам нужно настроить фон и создать спрайт. Мы сделаем эту игру космической, добавив звездный фон и спрайт космолета. Создайте новый проект в редакторе Scratch и введите имя **Астероиды** в качестве названия проекта. Нажмите кнопку **Выбрать фон из библиотеки** в разделе **Новый фон**, выберите фон **Stars** и нажмите кнопку **ОК**. Для удобства вы можете переименовать фон, присвоив имя **Звезды**.

Мы не будем использовать спрайт кота, который помещен изначально в каждый проект Scratch, поэтому, прежде чем продолжить, щелкните правой кнопкой мыши по этому спрайту и выберите в контекстном меню команду **Удалить**.

### 1. Создание спрайта **Космолет**

В качестве космолета мы будем использовать изображение летающей тарелки, которое вы найдете в архивном файле с примерами.

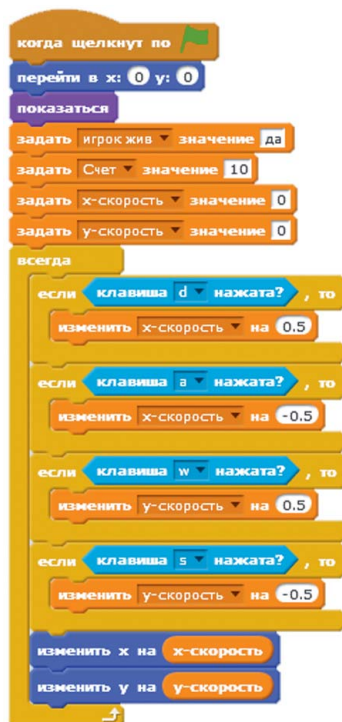
Нажмите кнопку **Загрузить спрайт из файла** рядом с надписью **Новый спрайт**. Затем выберите файл изображение *космолет.png* из запакованного ZIP-файла с примерами.

В оранжевой категории **Данные** нажмите кнопку **Создать переменную** и создайте переменную с именем **х-скорость** и режимом **Только для этого спрайта**. Повторите предыдущие шаги, чтобы создать переменную с именем **у-скорость**.

**ПРИМЕЧАНИЕ.** Если параметр **Только для этого спрайта** недоступен, значит, вместо спрайта **Космолет** выбрана сама сцена. Закройте окно **Новая переменная**, выберите спрайт **Космолет** и снова нажмите кнопку **Создать переменную**.

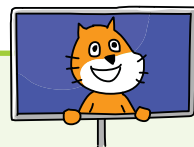
Вам также необходимо создать две переменные с именами **Счет** и **Игрок жив**, но с параметром **Для всех спрайтов**.

Затем добавьте код, показанный на следующем рисунке, в спрайт **Космолет**. Код определяет начальное положение космолета и первоначальные значения переменных. Он также содержит алгоритм, который определяет управление игрой.



Вы можете удивиться, почему мы не использовали код из тех предыдущих игр, в которых применялись блоки **Изменить x на** или **Изменить y на**. Сейчас объясним. В этой программе нажатие одной из клавиш – **W**, **A**, **S** или **D** – добавляет или вычитает из значений переменных **х-скорость** и **у-скорость**. Затем код в нижней части сценария изменяет позицию спрайта **Космолет** по осям **x** и **y**, используя значения этих переменных. Даже после того, как игрок отпускает клавишу, переменные по-прежнему отражают обновленную позицию, поэтому космолет продолжает двигаться.

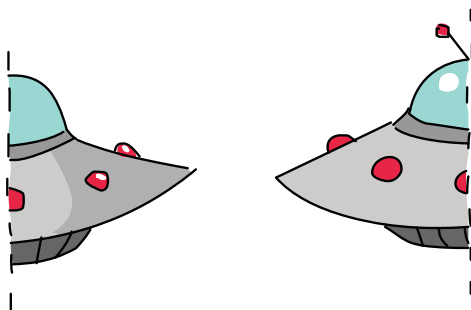
## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Нажмите клавиши **W**, **A**, **S** или **D**, чтобы увидеть, как космолет реагирует на нажатие той или иной клавиши. Убедитесь, что все четыре клавиши **W**, **A**, **S** и **D** перемещают космолет в правильном направлении. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

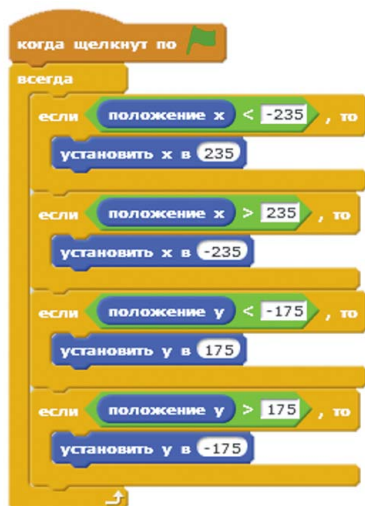
## Б ВЫХОД КОСМОЛЕТА ЗА КРАЯ СЦЕНЫ

Когда вы проверяли код, заметили ли вы, что спрайт **Космолет** мгновенно останавливается, когда врежется в край сцены? Причина в том, что Scratch предотвращает перемещение спрайтов за пределы сцены, что полезно для большинства программ на языке Scratch. Но в программе «Уничтожитель астероидов... в космосе!» мы хотим, чтобы спрайты перемещались за пределы сцены и *появлялись* на другой стороне.



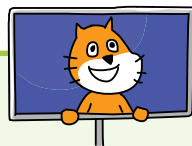
### 2. Добавление необходимого кода в спрайт Космолет

Код, показанный на следующем рисунке, позволит космолету перемещаться на другую сторону сцены, когда он достигает края. Добавьте код, показанный на рисунке ниже.



Левый и правый края сцены находятся в точках с координатами по оси  $x$   $-240$  и  $240$  соответственно, а нижний и верхний края сцены – в точках с координатами по оси  $y$   $-180$  и  $180$  соответственно. Мы используем эти значения для написания кода, который изменяет положение спрайта **Космолет**, когда он проходит через эти четыре координаты. Каждый раз, когда положение по оси  $x$  или  $y$  спрайта **Космолет** оказывается в 5 шагах от этих краев, новый код перемещает спрайт **Космолет** на другую сторону сцены. Так как переменные  **$x$ -скорость** и  **$y$ -скорость** будут перемещать спрайт **Космолет** с той же скоростью и в одном направлении, спрайт **Космолет** будет выглядеть так, как будто он постоянно перемещается со сцены/на сцену.

## КОНТРОЛЬНАЯ ТОЧКА

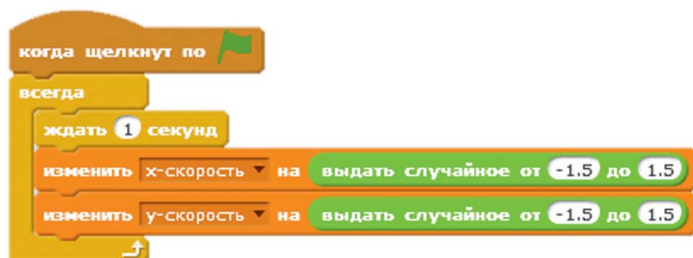


Нажмите кнопку в виде зеленого флага, чтобы проверить уже готовый код. Убедитесь, что при достижении любого из четырех краев сцены спрайт **Космолет** перемещается на другую сторону. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

### 3. Добавление кода случайных движений в спрайт Космолет

Управление в этой игре довольно проблематичное, но давайте усложним игру еще сильнее. Мы добавим случайные небольшие движения в спрайт **Космолет**, чтобы игрок не мог просто оставаться в центре, не двигаясь вообще.

Добавьте код, показанный на следующем рисунке, в спрайт **Космолет**, чтобы случайные движения происходили каждую секунду:



Внутри цикла **Всегда** значения переменных **х-скорость** и **у-скорость** случайным образом немного изменяются после паузы в 1 секунду. Это означает, что каждую секунду космолет совершает случайное движение.

#### КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Не нажимайте клавиши **W**, **A**, **S** и **D**. Подождите, когда космолет станет понемногу двигаться сам по себе. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## В ПРИЦЕЛИВАНИЕ С ПОМОЩЬЮ МЫШИ И СТРЕЛЬБА КЛАВИШЕЙ ПРОБЕЛ

Код спрайта **Космолет** завершен, теперь давайте добавим мощное оружие – бластер. Шары, которыми он стреляет, разрушают опасные астероиды! В космосе!

### 4. Создание мощного бластера

В библиотеке Scratch есть спрайт, который мы можем использовать для создания в своей программе мощных шаров бластера. Нажмите кнопку **Выбрать спрайт из библиотеки** рядом с надписью **Новый спрайт**. Выберите спрайт **Ball** и нажмите **ОК**. Откройте панель информации этого спрайта, нажав кнопку **i**, и присвойте спрайту имя **Шар бластера**.

Нам нужно, чтобы спрайт **Шар бластера** издавал звук лазера, когда спрайт **Космолет** запускает его. Перейдите на вкладку **Звуки** над областью блоков. Затем нажмите кнопку **Выбрать звук из библиотеки** в разделе **Новый звук**. Выберите звук **Laser1** и нажмите кнопку **ОК**. Для удобства вы можете переименовать звуковой фрагмент, присвоив ему имя **Лазер**.

Мы создадим клоны спрайта **Шар бластера**, но клоны и исходный спрайт будут запускаться разными кодами. Существует только один спрайт **Шар бластера**, но игроку нужна возможность стрелять сразу несколькими энергетическими шарами. Мы создадим клоны исходного спрайта **Шар бластера**, так что на сцене может быть более одного энергетического шара. Исходный спрайт останется скрытым; все спрайты **Шар бластера**, которые появляются на сцене, будут клонами.

Мы применим переменную с именем **Я клон**, чтобы отслеживать, какой спрайт является исходным, а какие – клоны. Перейдите на вкладку **Скрипты**, чтобы вернуться в область скриптов. В оранжевой категории **Данные** нажмите кнопку **Создать переменную**. Присвойте пере-

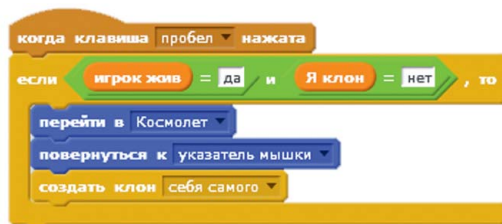


менной имя **Я клон** и установите переключатель в положение **Только для этого спрайта**. Для исходного спрайта **Шар бластера** присвойте этой переменной значение **Нет**, а для клонов – значение **Да**. Добавьте код, показанный на следующем рисунке, в спрайт **Шар бластера**:



В начале игры исходный спрайт скрывается и остается скрытым всю игру. На сцене появляются только его клоны. Кроме того, спрайт, который мы используем, слишком велик для игры, поэтому установите его размер равным 10 процентам от исходного, чтобы сделать его меньше.

Теперь добавьте код, показанный на следующем рисунке, к спрайту **Шар бластера**. Игрок стреляет из бластера мощными энергетическими шарами, нажимая клавишу **Пробел**. Исходный спрайт **Шар бластера** создает видимые для игрока клоны, которые двигаются вслед за мышью. Это позволяет игроку перемещать мышь, чтобы прицеливаться и направлять энергетические шары.

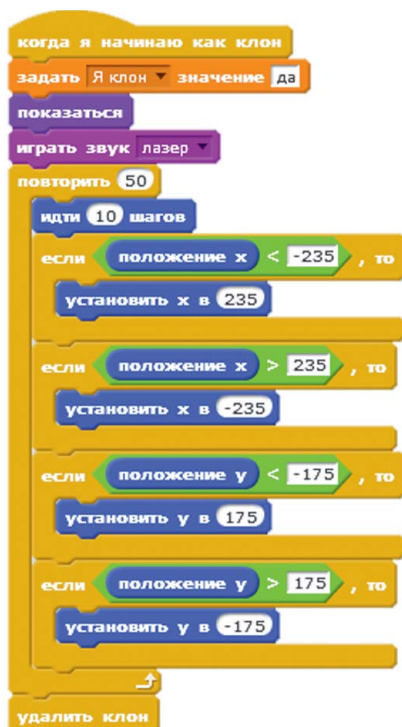


Код, находящийся под блоком **Когда клавиша Пробел нажата**, работоспособен и для исходного спрайта, и для клонов, если мы не дадим инструкции, что этот код должен выполняться только для исходного спрайта. Но нам не нужно, чтобы существующие клоны создавали новые клоны. Поэтому блок **Если то** проверяет, присвоено ли значение **Нет** переменной **Я клон**, чтобы данный код

запускался только для исходного спрайта **Шар бластера** и только он мог создавать клоны.

Разумеется, спрайт **Космолет** может запускать клоны спрайта **Шар бластера** только в том случае, если игрок жив, поэтому код также проверяет, что переменная **Игрок жив** имеет значение **Да**.

Затем добавьте код, показанный на следующем рисунке, в спрайт **Шар бластера**, который отвечает за то, чтобы клоны двигались вслед за мышью.



Теперь клон может появляться и двигаться вперед. Клоны должны выходить за края сцены так же, как и спрайт **Космолет**, поэтому мы используем аналогичный код, чтобы получить нужный эффект.

Обратите внимание, что клоны самостоятельно присваивают переменной **Я клон** значение **Да**. Это делается для того, чтобы клоны не запускали код скрипта **Когда клавиша Пробел нажата**. Блок **Если то** в этом скрипте запускает код, только если переменная **Я клон** имеет значение **Нет**.

Клоны движутся вперед 50 раз – по 10 шагов за раз. Это означает, что клоны спрайта **Шар бластера** имеют ограничения и не могут двигаться вечно. После того как цикл завершается 50 раз, клон удаляет себя и исчезает со сцены.



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Прицельтесь с помощью мыши и нажмите клавишу **Пробел**. Убедитесь, что клоны спрайта **Шар бластера** появляются у спрайта **Космолет** и двигаются по направлению к мыши. Клоны должны выходить за пределы сцены и в конечном итоге исчезать. После проверки нажмите кнопку в виде красного знака остановки и сохраните программу.

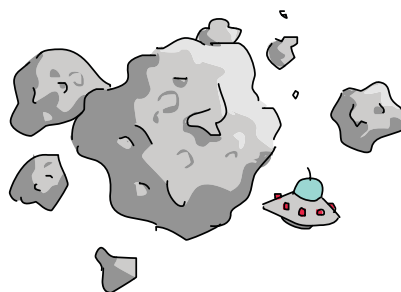
## Г СОЗДАНИЕ АСТЕРОИДОВ

Теперь нам нужно создать цели, по которым игрок будет стрелять. В этой игре по сцене летают астероиды, пока в них не попадет клон **Шар бластера**. Они будут многократно раскалываться на два меньших обломка (астероида), пока не станут достаточно маленькими, чтобы испариться.

### 5. Создание спрайта **Астероид**

Добавьте новый спрайт, нажав кнопку **Загрузить спрайт из файла** и выбрав файл *Астероид.png*. Этот файл находится в архиве с примерами.

Выберите оранжевую категорию **Данные**, а затем нажмите кнопку **Создать переменную**. Установите переключатель в положение **Только для этого спрайта** и присвойте переменной имя **Попадания**. Повторите эти шаги, чтобы создать переменные с именами **х-скорость**, **у-скорость** и **Вращение**.



Для всех этих переменных выбирайте параметр **Только для этого спрайта**. Мы будем использовать переменную попадания в шаге 6, чтобы отслеживать, сколько попаданий получил астероид, а также размер астероида.

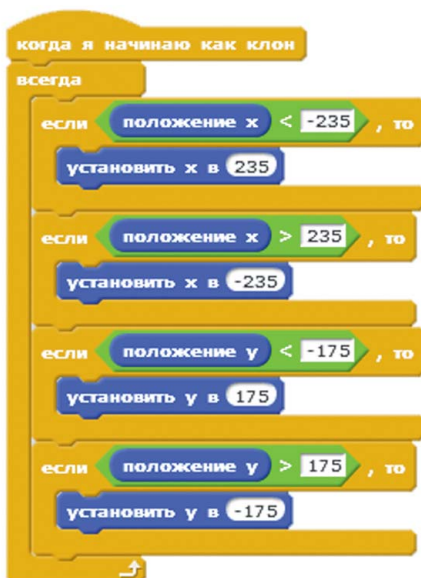
Давайте напишем код, с помощью которого спрайт **Астероид** будет создавать новые клоны. Каждый клон будет иметь случайную скорость и вращение. Результатом будет непредсказуемый рой астероидов... в космосе!

Добавьте код, показанный на следующем рисунке, в спрайт **Астероид**.



Как и спрайт **Шар бластера**, исходный спрайт **Астероид** будет скрыт и будет генерировать клоны. Новые клоны создаются каждые 8–12 секунд. Когда клон создается, он отображает себя, присваивает случайную скорость и вращение и начинает движение.

Так же как спрайты **Космолет** и **Шар бластера**, астероиды будут вылетать за пределы сцены. Добавьте код, показанный на следующем рисунке, к спрайту **Астероид**.



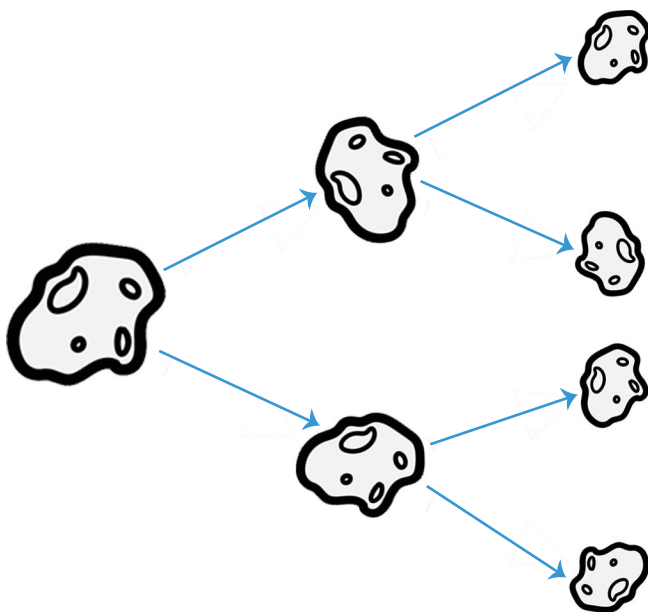
## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить код, который вы только что добавили. Убедитесь, что спрайт **Астероид** медленно перемещается и вращается и что он вылетает за пределы сцены. Кроме того, убедитесь, что новые клоны появляются каждые 8–12 секунд. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## А СОЗДАНИЕ АСТЕРОИДОВ, РАСКАЛЫВАЮЩИХСЯ НАДВОЕ ПРИ ПОПАДАНИИ

Когда мощный энергетический заряд попадает в астероид, который создает два новых более мелких клона себя самого. На сцене это выглядит так, будто он раскололся на два обломка.

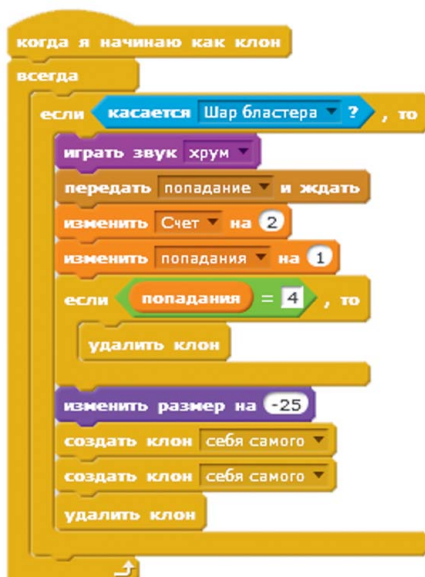


### 6. Добавление кода раскалывания астероида

На вкладке **Звуки** нажмите кнопку **Выбрать звук из библиотеки**. Затем выберите пункт **Chomp** и нажмите кнопку **ОК**. Для удобства вы можете переименовать звуковой файл, присвоив ему имя **Хрум**.

Звук **Хрум** будет воспроизводиться всякий раз, когда клон спрайта **Шар бластера** попадает в астероид.

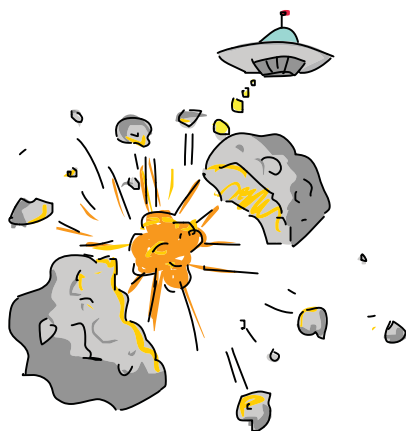
Добавьте код, показанный на следующем рисунке, к спрайту **Астероид**. Вам нужно будет создать новое сообщение – **Попадание**.



Когда клон спрайта **Шар бластера** попадет в клон спрайта **Астероид**, последний воспроизведет звуковой эффект **Хрум** и передаст сообщение **Попадание**.

Затем **Астероид** прибавит 2 к значению переменной **Счет** и 1 к значению переменной **Попадания**. «Подстреленный» астероид немного уменьшится, изменив размер на -25, поэтому, когда он («родитель») дважды клонирует себя, его «дочерние» клоны будут меньшего размера. Наконец, клон спрайта **Астероид** удалит сам себя.

Значения переменной **Попадания** двух маленьких «дочерних» клонов будут на единицу больше, чем значение переменной «родителя». Когда астероид получит четвертое попадание, а значение его переменной Попадания станет равно 4, код уничтожит этот клон вместо того, чтобы создать два новых. (Код после блока **Удалить**

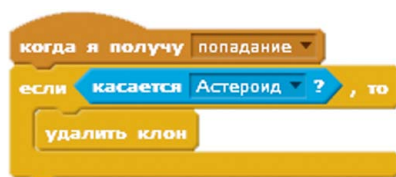


**ЭТОТ КЛОН** не запускается, так как клон больше не существует.) Это предотвращает превращение одного спрайта **Астероид** в бесконечное количество астероидов, когда один спрайт разбивается на 2 осколка, затем на 4, 8, 16, 32 и так далее по нарастающей до бесконечности.

Однако если вы *хотите* экспоненциально увеличить количество спрайтов для астероидов, увеличьте их число в блоках **если попадания = 4**.

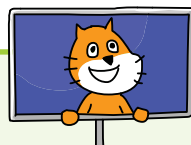
## 7. Добавление сообщения «попадание» в спрайт Шар бластера

Выберите спрайт **Шар бластера** в области спрайтов и добавьте в него код, показанный на следующем рисунке:



Все клоны спрайта **Шар бластера** получают сообщение **Попадание**, но только те, которые в настоящее время касаются спрайта **Астероид**, будут удалены. (Один достигнет астероида, потому что сообщение передается только с помощью спрайта **Астероид**, когда он касается спрайта **Шар бластера**.) На экране это будет выглядеть так, словно энергетический шар исчезает после попадания в астероид.

### КОНТРОЛЬНАЯ ТОЧКА

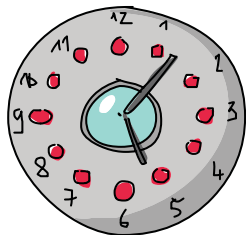


Нажмите кнопку в виде зеленого флага, чтобы проверить эту часть кода. Попробуйте взорвать несколько астероидов. Убедитесь, что энергетический шар исчезает, а спрайт **Астероид** заменяется двумя меньшими клонами. Когда заряд попадает в астероид в четвертый раз, астероид должен исчезнуть. После этого нажмите кнопку в виде красного знака остановки и сохраните программу.



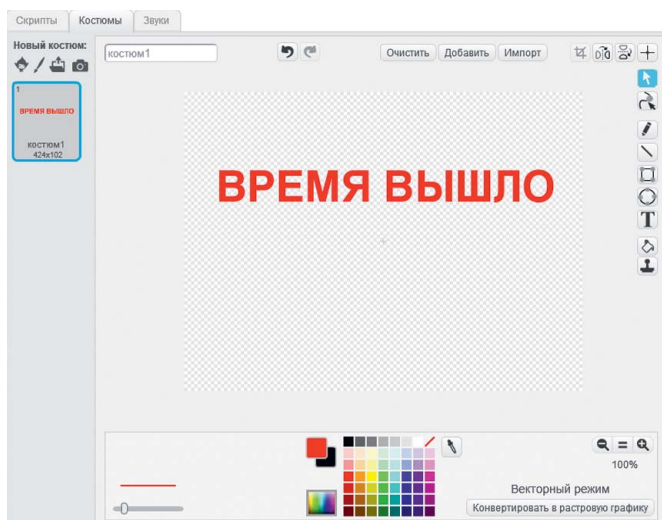
## ВЕДЕНИЕ СЧЕТА И СОЗДАНИЕ ТАЙМЕРА

Игра «Уничтожитель астероидов... в космосе!» быстро усложняется, когда астероиды превращаются в тучу крошечных осколков, летающих по всей сцене. Успешная стратегия игры заключается в том, чтобы стрелять по астероидам медленно и осторожно, уничтожая мелкие осколки, прежде чем стрелять в более крупные. Но нам нужно, чтобы игрок не расслаблялся, поэтому давайте сделаем, чтобы значение переменной **Счет** уменьшалось на один каждую секунду и игра заканчивалась, когда значение становилось равным нулю. Таким образом, игроку придется поддерживать быстрый темп игры, стреляя по астероидам.



### 8. Создание спрайта **Время вышло**

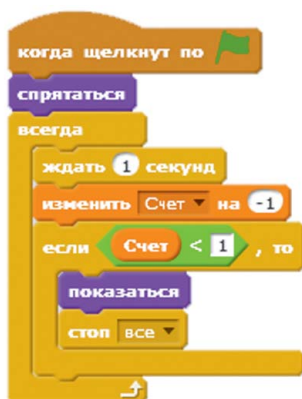
Нажмите кнопку **Нарисовать новый спрайт** рядом с надписью **Новый спрайт**. В графическом редакторе выберите инструмент **Текст** и напишите фразу «ВРЕМЯ ВЫШЛО» красными прописными буквами.



В области спрайтов нажмите кнопку **i** созданного спрайта, чтобы открыть его панель информации. Переименуйте спрайт, присвоив ему имя **Время вышло**.

**ПРИМЕЧАНИЕ.** Обратите внимание, что русский текст не поддерживается напрямую в редакторе Scratch, поэтому потребуется воспользоваться сторонним приложением. Для этого перейдите на веб-страницу [scratch.mit.edu/projects/60521212/](http://scratch.mit.edu/projects/60521212/) и нажмите кнопку в виде зеленого флага или клавишу **Пробел**. Введите текст в появившееся поле и нажмите клавишу **Enter** или синюю кнопку справа от поля. Скопируйте текст из второй строки в виде символов типа `Àâñëëëé`, выделив его и нажав сочетание клавиш **Ctrl+C**. Перейдите в редактор Scratch и, выбрав инструмент **Текст** и шрифт **Helvetica**, вставьте скопированный текст с помощью сочетания клавиш **Ctrl+V**. Обратите внимание, что в редакторе Scratch для русских букв поддерживается только шрифт **Helvetica**.

Добавьте код, показанный на следующем рисунке, в спрайт **Время вышло**.



Этот код в начале игры скрывает спрайт **Время вышло** и уменьшает значение переменной **Счет** на единицу после паузы в 1 секунду. Когда значение переменной **Счет** достигает **0**, код отображает спрайт **Время вышло**.

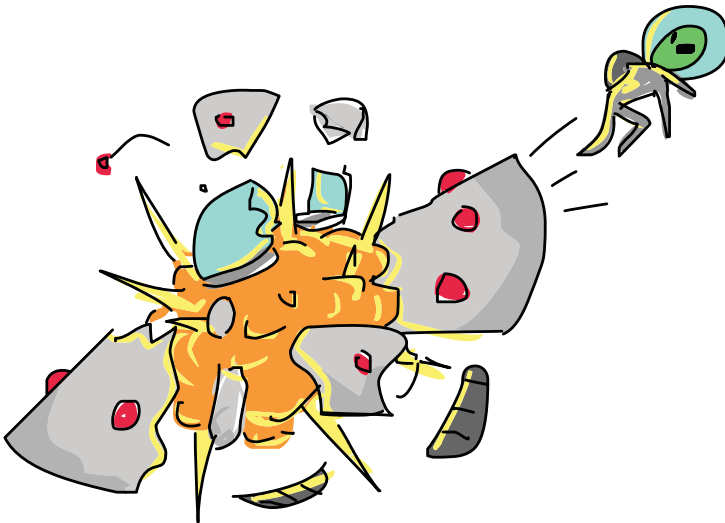


## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что значение переменной **Счет** уменьшается на единицу каждую секунду. Если значение переменной **Счет** равно нулю, должен отобразиться спрайт **Время вышло**. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## Ж ВЗРЫВ КОСМОЛЕТА ПРИ СТОЛКНОВЕНИИ С АСТЕРОИДОМ

Игрок может проиграть, если не будет взрывать астероиды достаточно быстро, чтобы значение переменной **Счет** не достигло нуля. Проиграть можно и в том случае, если астероид попадет в космолет. Давайте добавим код, чтобы определить это поведение и для отображения анимации взрыва.



## 9. Загрузка спрайта Взрыв

Для создания анимации взрыва доступно восемь изображений. Эти костюмы находятся в файле *Взрыв.sprite2* в архивном файле.

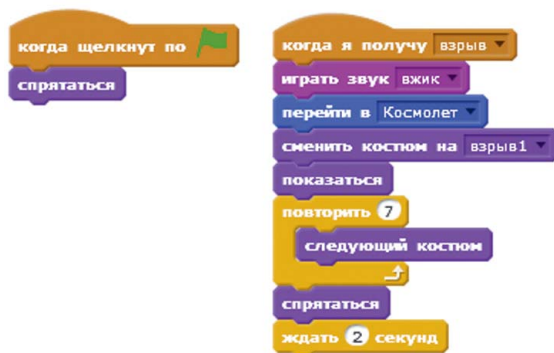
В редакторе Scratch нажмите кнопку **Загрузить спрайт из файла** рядом с надписью **Новый спрайт**. Выберите файл *Взрыв.sprite2* и нажмите кнопку **ОК**. Восемь костюмов для анимации взрыва появятся на вкладке **Костюмы** спрайта **Взрыв**.

## 10. Добавление кода спрайта Взрыв

Для спрайта **Взрыв** вы создадите новое сообщение — **Взрыв**. Когда спрайт **Взрыв** получит это сообщение, он последовательно покажет свои костюмы, создав таким образом анимацию взрыва.

В момент взрыва спрайт **Взрыв** будет воспроизводить звуковой эффект. Загрузите соответствующий звук, для этого перейдите на вкладку **Звуки** над областью блоков. Затем нажмите кнопку **Выбрать звук из библиотеки** в разделе **Новый звук**. Выберите звук **Alien creak2** и нажмите кнопку **ОК**. Для удобства вы можете переименовать его, присвоив имя **Вжик**.

Добавьте код, показанный на следующем рисунке, в спрайт **Взрыв**:



Спрайт **Взрыв** остается скрытым до тех пор, пока он не получит сообщение **Взрыв**. Затем он запускает воспроизведение звука, переходит в положение на сцене, где находится

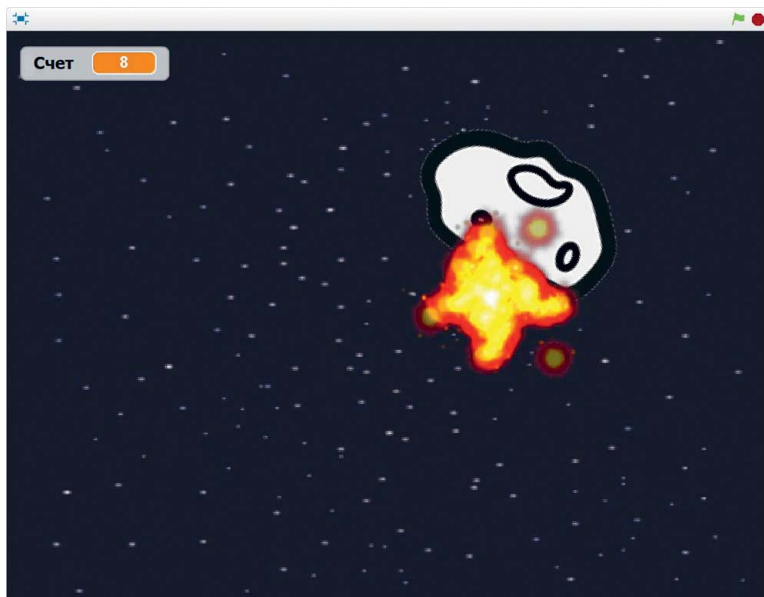
космолет, и переключает костюмы семь раз, чтобы создать зрелищную анимацию взрыва.

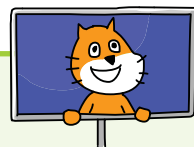
## II. Добавление кода взрыва в спрайт Космолет

Спрайт **Космолет** передаст сообщение **Взрыв**, когда он коснется одного из клонов астероидов. Добавьте следующий сценарий к спрайту **Космолет**:



Анимация взрыва создается путем поочередного показа каждого костюма в течение короткого времени. Это похоже на покадровую анимацию, которая используется в мультфильмах. Каждый костюм – это кадр. Код быстро меняет костюмы, чтобы взрыв выглядел реалистичным.





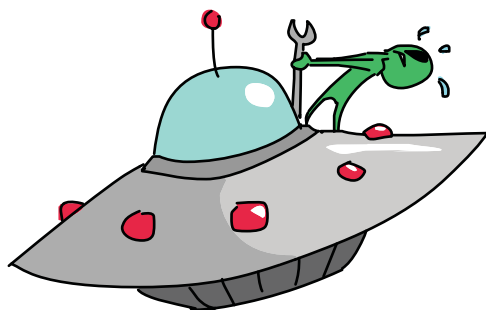
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что в начале игры спрайт **Взрыв** скрыт. Врежьтесь в астероид и убедитесь, что спрайт **Взрыв** появляется там, где находится спрайт **Космолет**. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Код для этой программы слишком велик, чтобы приводить его здесь полностью, но вы можете просмотреть его в архиве с примерами. Имя файла, в котором находится код, – *Астероиды\_1.sb2*.

## ВЕРСИЯ 2.0: ОГРАНИЧЕНИЕ БОЕЗАПАСА

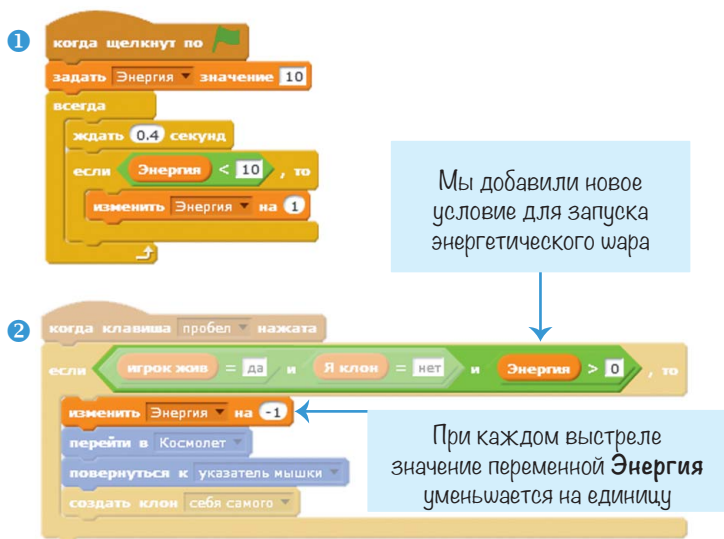
Как только вы освоите игру, она может стать слишком легкой для вас. Одна особенность, которая упрощает игру, состоит в том, что вы можете стрелять так быстро, как нажимаете клавишу **Пробел**. Это действие позволяет игроку стрелять без разбора вместо того, чтобы тщательно целиться в астероиды. Но мы можем изменить это, добавив новую переменную – **Энергия**. Каждый выстрел энергетическим шаром уменьшает значение этой переменной на единицу. Когда значение переменной **Энергия** становится равным 0, космолет не может стрелять. С течением времени значение переменной **Энергия** будет медленно увеличиваться, что заставит игрока тщательнее прицеливаться и не тратить заряды.



Вам потребуется переменная для отслеживания уровня энергии бластера космолета. Выберите оранжевую категорию **Данные** и нажмите кнопку **Создать переменную**. Создайте переменную с именем **Энергия** и установите переключатель в положение **Для всех спрайтов**. В области блоков установите флажок **Энергия** (как и флажок **Счет**), чтобы этот блок появился на сцене.

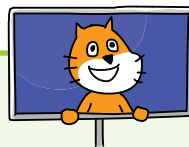
Значение переменной **Энергия** будет составлять 10 в начале игры, а затем будет уменьшаться на единицу каждый раз, когда игрок выпускает энергетический шар из бластера. Игрок сможет стрелять, только если значение переменной **Энергия** выше нуля.

Измените код спрайта **Шар бластера**, чтобы он соответствовал показанному ниже рисунку.



Скрипт ① – совершенно новый, он присваивает значение 10 переменной **Энергия**, прежде чем перейти к циклу **Всегда**. Код цикла, если значение переменной **Энергия** меньше 10, ждет 0,4 секунды, а затем увеличивает значение переменной **Энергия** на один. Таким образом, значение переменной **Энергия** никогда не будет выше 10. Скрипт ② немного изменен, чтобы значение переменной

**Энергия** было больше нуля, для того чтобы игрок мог стрелять. Когда энергетический шар запущен, блок **Изменить Энергия на -1** уменьшает значение переменной **Энергия**.



## КОНТРОЛЬНАЯ ТОЧКА

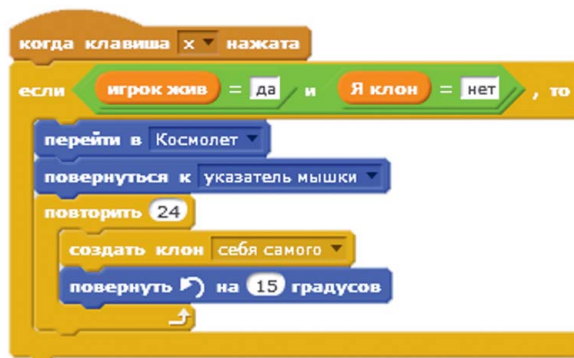
Нажмите кнопку в виде зеленого флага, чтобы проверить эту часть кода. Убедитесь, что переменная **Энергия** отображается на сцене рядом с переменной **Счет**. В начале игры значение переменной **Энергия** устанавливается равным 10 и уменьшается на 1 каждый раз, когда игрок нажимает клавишу **Пробел**. Если значение переменной **Энергия** равно 0, нажатие клавиши **Пробел** не должно приводить к появлению новых клонов спрайта **Шар бластера**. Также убедитесь, что значение переменной **Энергия** увеличивается на 1 примерно каждые полсекунды. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## ЧИТ-РЕЖИМ: ЗВЕЗДНАЯ БОМБА

Ограничение энергии, которое мы ввели в версии 2.0 игры «Уничтожитель астероидов... в космосе!», создает трудности для игрока, но давайте добавим секретный чит, чтобы обойти это. Чит, позволяющий иметь неограниченную энергию, сделает игру слишком скучной. Вместо этого мы добавим специальную энергетическую бомбу, взрыв которой будет похож на звезду, расширяющуюся в разные стороны от космолета. Выстрел звездной бомбой будет осуществляться при нажатии клавиши **X**.

Добавьте код, похожий на обычную стрельбу (когда игрок нажимает клавишу **Пробел**), показанный на рисунке ниже, в спрайт **Шар бластера**:





Как и в скрипте **Когда клавиша Пробел нажата**, этот сценарий проверяет, жив ли игрок и не является ли спрайт клоном. Данный код должен запускаться только исходным спрайтом, а не клоном.

Внутри блока **Если, то** спрайт **Шар бластера** переходит к космолету и поворачивается в сторону указателя мыши. Затем спрайт клонирует себя 24 раза. После создания каждого клона спрайт меняет направление на 15 градусов против часовой стрелки. Так получается звезда, состоящая из энергетических зарядов, летящих во всех направлениях.



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить эту часть кода. Нажмите клавишу **X** и наблюдайте, как клоны энергетических зарядов вылетают из космолета в разные стороны. Поскольку этот чит секретный, убедитесь, что игрок может использовать его независимо от уровня энергии. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## ЗАКЛЮЧЕНИЕ

В этой главе вы создали игру, в которой предусмотрено следующее:

- ▶ использование инерционного движения для управления космолетом;
- ▶ переменные **х-скорость** и **у-скорость**, позволяющие отслеживать, насколько быстро перемещается спрайт **Космолет**;
- ▶ спрайты могут вылетать за пределы сцены;
- ▶ клоны спрайта **Астероид**, которые могут создавать два клона самих себя меньшего размера;
- ▶ переменные **Счет** и **Энергия**, значения которых постоянно уменьшаются и увеличиваются с течением времени;
- ▶ покадровая анимация взрыва.

Эта игра предлагает пользователям настоящее испытание, и ради этого, нам, программистам, пришлось добавлять все функции по очереди. Игрок не управляет космолетом непосредственно, а лишь подталкивает его. Если бы мы закончили написание кода спрайта **Космолет** на этом этапе, игрок мог бы просто спрятаться в углу сцены и таким образом защититься от астероидов, поэтому мы сделали все спрайты способными вылетать за пределы сцены. Даже с этим дополнением игрок мог бы попытаться спрятаться в центре сцены. Поэтому мы добавили случайные небольшие движения космолету.

В игре достаточно сложно уворачиваться от большого количества мелких астероидов, поэтому игрок может попробовать аккуратно и неторопливо уничтожить мелкие обломки, прежде чем стрелять в крупные астероиды. На этот случай мы сделали переменную **Счет**, значение которой уменьшается с течением времени, чтобы игрок стрелял быстрее. Игрок также может попробовать стрелять наугад, без прицеливания, поэтому мы создали переменную **Энер-**

гия, ограничивающую скорость, с которой игрок может стрелять.

Каждый раз, когда вы добавляете какую-нибудь функцию в свою игру, вы должны представлять, как она повлияет на игровой процесс. Слишком сложная игра раздражает, а слишком простая быстро наскучивает. Суть в том, чтобы найти баланс.

Следующая игра – самая продвинутая программа в этой книге – это платформер в духе Super Mario Bros. или Super Meat Boy. В ней будут не только прыжки и гравитация, как в игре «Баскетбол», но и возможность разрабатывать пользовательские уровни для нее, не меняя код.

## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно посмотреть в конце книги.)

1. Как работает код, который отвечает за выход объекта за пределы сцены и появление его с другой стороны?
2. Для чего спрайту **Шар бластера** требуется переменная **Я клон**?
3. Что мешает клону спрайта **Астероид** бесконечно раскалываться на множество обломков?
4. Как код спрайта **Взрыв** создает анимированный эффект взрыва космолета?



## 9

## ПРОДВИНУТЫЙ ПЛАТФОРМЕР

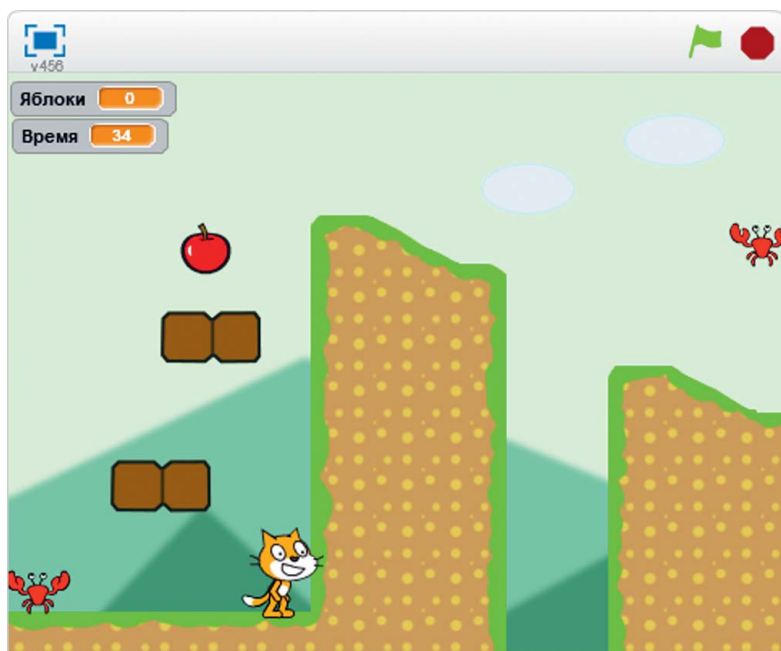
**П**ервая видеоигра из серии Super Mario Bros. была представлена в 1985 году и стала лучшей игровой франшизой<sup>4</sup> компании Nintendo, а также одной из самых узнаваемых игр всех времен. Игра, в которой персонаж бегает, прыгает и перепрыгивает с платформы на платформу, называется *платформер*.

---

<sup>4</sup> Включает, помимо изначальной игры, ее продолжения, а также выпуск сувенирной и другой продукции. — *Примеч. ред.*

В этой главе мы создадим с помощью Scratch такую игру. Роль Марио/Луиджи будет играть кот. Игроку нужно управлять прыгающим котом, собирая яблоки и избегая крабов, которые будут к нему подкрадываться. Игра ограничена во времени: игроку дается всего 45 секунд, чтобы собрать как можно больше яблок и не попасться крабам!

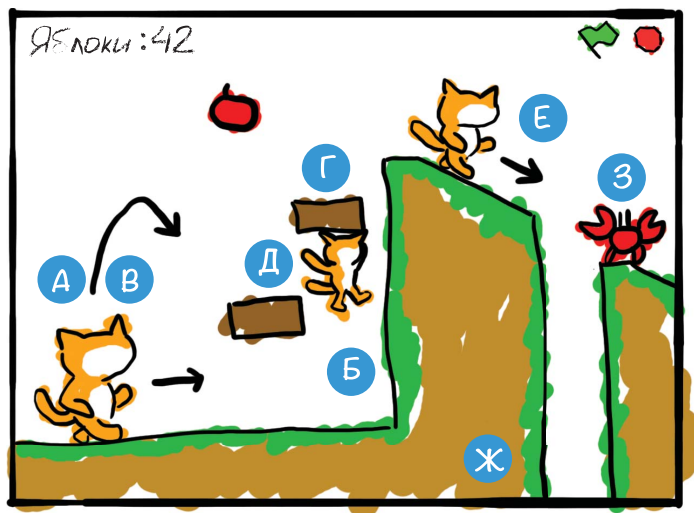
Прежде чем начать писать код игры, посмотрите готовую программу по ссылке [scratch.mit.edu/studios/4188596/](https://scratch.mit.edu/studios/4188596/).



Приготовьтесь к программированию самой сложной игры в этой книге!

## ЭСКИЗ ПРОЕКТА

Давайте нарисуем на бумаге, как должна выглядеть игра. Игрок управляет котом, который прыгает по сцене, собирая яблоки, которые появляются случайным образом. Крабы ходят и прыгают по платформам в случайном порядке.



Вот что мы будем делать в каждом разделе этой главы:

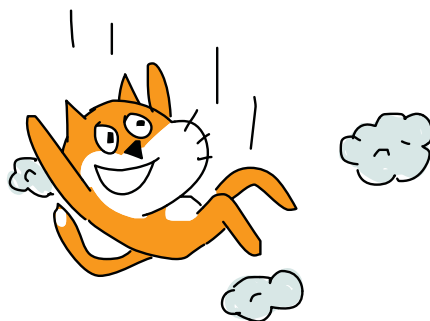
- А. Имитация гравитации, падения и приземления
- Б. Использование крутых склонов и стен
- В. Обучение кота высоким и низким прыжкам
- Г. Добавление возможности обнаружения препятствий сверху
- Д. Использование хитбокса для спрайта **Кот**
- Е. Улучшение анимации ходьбы
- Ж. Создание уровня
- З. Добавление крабов и яблок

Эта игра-платформер – самая продвинутая в моей книге, но каждый может ее создать, если будет следовать инструкциям в этой главе. Давайте программировать каждую часть по шагам.

Если вы хотите сэкономить время, вы можете начать с файла скелета проекта с именем *Глава 09/Скелет\_проекта.sb2*, находящегося в запакованном файле с примерами, который вы скачали ранее по ссылке [https://eksmo.ru/files/Scratch\\_Sweigart.zip](https://eksmo.ru/files/Scratch_Sweigart.zip). В файл скелета проекта уже загружены все спрайты, так что вам нужно всего лишь перетащить блоки кода в каждый спрайт.

## А ИМИТАЦИЯ ГРАВИТАЦИИ, ПАДЕНИЯ И ПРИЗЕМЛЕНИЯ

В первой части мы добавим код для имитации гравитации, падения и приземления персонажей, как в игре «Баскетбол» в главе 4. Важное отличие состоит в том, что в платформере кот приземляется, когда касается спрайта земли, а не нижнего края сцены. Программировать такое поведение немного сложнее, потому что нам нужно, чтобы земля была холмистая, а уровень содержал платформы.



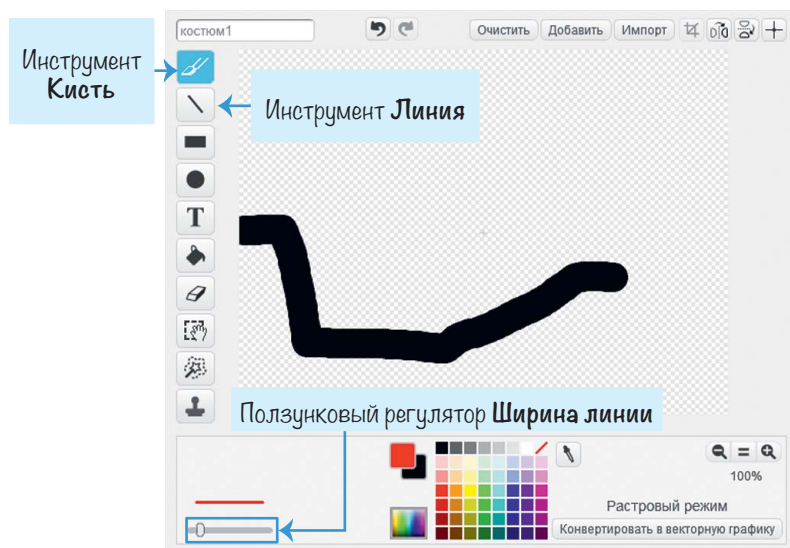
Для начала присвойте проекту имя **Платформер**, указав его в текстовом поле в левом верхнем углу редактора Scratch.

### 1. Создание спрайта **Земля**

Давайте используем простую фигуру в качестве земли в первых нескольких скриптах, просто чтобы изучить, как будет работать код.

Нажмите кнопку **Создать новый спрайт** в разделе **Новый спрайт**, чтобы создать временный спрайт земли, пока вы знакомитесь с кодом платформера. В графическом редакторе используйте инструмент **Кисть** или **Линия**, чтобы нарисовать землю. Вы можете сделать линии более толстыми с помощью ползункового регулятора **Ширина линии** в нижнем левом углу графического редактора. Обязательно нарисуйте пологий склон справа и крутой склон слева.

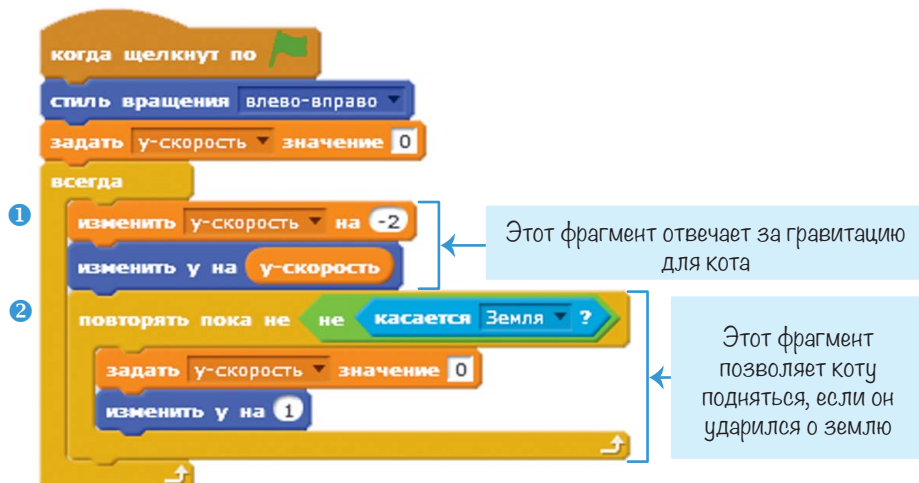
Откройте панель информации созданного спрайта и присвойте ему имя **Земля**. Кроме того, переименуйте **Спрайт1** в спрайт **Кот**.



## 2. Добавление кода гравитации и приземления

Теперь, когда у нас есть спрайт земли, нам нужно, чтобы кот приземлялся на него при падении.

Выберите спрайт **Кот**. В оранжевой категории **Данные** нажмите кнопку **Создать переменную** и установите переключатель в положение **Только для этого спрайта**. В качестве имени переменной укажите значение **у-скорость**. Затем добавьте код, показанный на следующем рисунке, в спрайт **Кот**:





Этот код в своем цикле **Всегда** выполняет два действия: он заставляет спрайт **Кот** падать, пока тот не коснется спрайта **Земля** ❶, а затем приподнимает спрайт **Кот**, если он оказался слишком глубоко в земле ❷.

Эти два фрагмента кода отвечают за падение кота, удар о землю и, при необходимости, поднимают кота из толщи земли и помещают его на поверхность.



Код, отвечающий за падение ❶, вычитает 2 из значения переменной **у-скорость**, а затем перемещает спрайт **Кот** по оси **y** в позицию со значением переменной **у-скорость**, заставляя кота падать все быстрее и быстрее. Если вы программировали игру «Баскетбол» в главе 4, код падения должен быть вам знаком.

Но блок **Повторять пока не** ❷ будет запускать цикл до тех пор, пока спрайт **Кот** не перестанет касаться спрайта **Земля**. (Если кот все еще находится в воздухе и падает, он не коснется земли, поэтому код в цикле будет пропущен.) Внутри этого цикла переменной **у-скорость** присваивается значение 0, так что падение кота прекратится. Блок **Изменить y на 1** немного поднимет спрайт **Кот**. Блок **Повторять пока не касается Земля** продолжит поднимать погруженный в землю спрайт **Кот** до тех пор, пока он не окажется на поверхности. Это нужно для того, чтобы кот оставался на поверхности земли, независимо от формы этой поверхности.



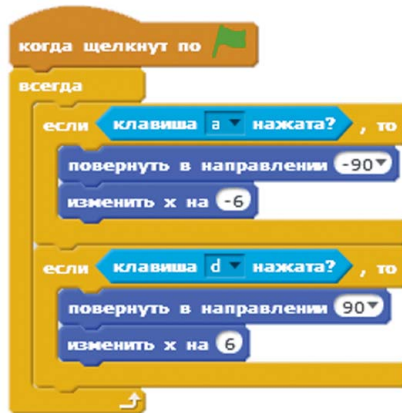


## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Перетащите кота с помощью мыши и отпустите. Удостоверьтесь, что кот падает и немного опускается в землю, а затем медленно поднимается из нее. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

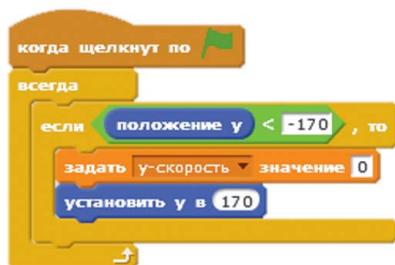
### 3. Обучение кота ходьбе и способности пересекать края сцены

Наш кот также должен ходить влево и вправо с помощью клавиш **A** и **D**, поэтому добавьте код, показанный на следующем рисунке, к спрайту **Кот**:

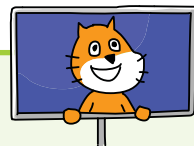


Этот код очень прост: нажатие клавиши **A** указывает коту направление влево ( $-90$ ) и перемещает его в позицию по оси  $x$  на  $-6$  (влево); нажатие клавиши **D** указывает коту направление вправо и перемещает его в позицию по оси  $x$  на  $6$  (вправо).

Затем добавьте код, показанный на следующем рисунке, чтобы спрайт **Кот** мог перемещаться за край сцены и появляться с другой стороны, если упадет за пределы нижней части сцены.



Этот код очень похож на код вылета за края сцены, который мы написали в игре «Уничтожитель астероидов... в космосе!» в главе 8. Код для перемещения влево и вправо напишем позже.



## КОНТРОЛЬНАЯ ТОЧКА

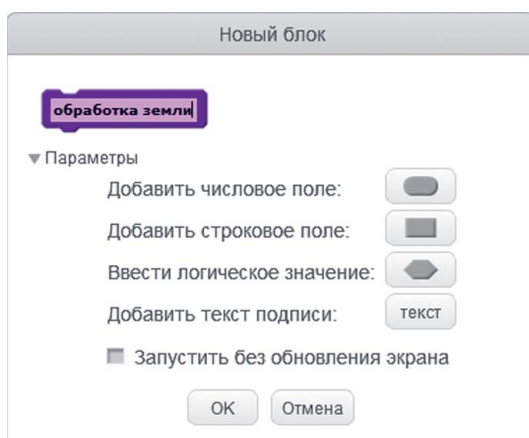
Нажмите кнопку в виде зеленого флага, чтобы проверить уже готовый фрагмент кода. Нажимайте клавиши **A** и **D**, чтобы перемещать кота по склонам. Если кот сорвется с края спрайта **Земля** и упадет за пределы нижней части сцены, то он должен появиться сверху. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Этот платформер содержит много скриптов, поэтому вы можете легко запутаться. Если ваша программа не работает и вы не можете понять, почему, загрузите файл проекта *Платформер\_1.sb2* из архива с примерами. В редакторе Scratch выберите команду меню **Файл ► Загрузить** с компьютера, чтобы загрузить файл, и продолжите чтение с этого момента.

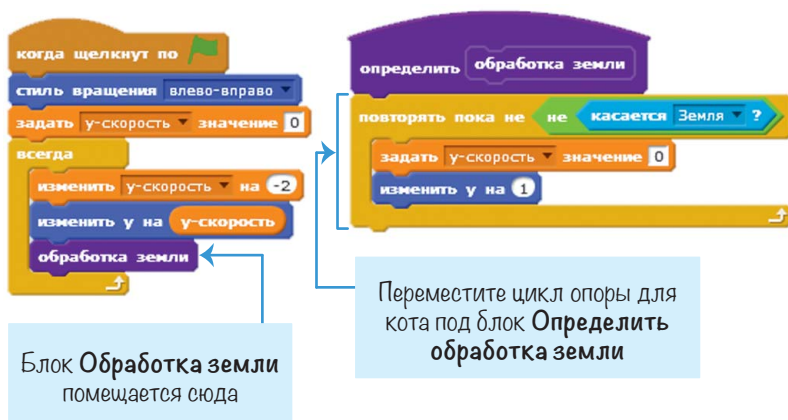
## 4. Удаление задержки подъема из земли

На данный момент самая большая проблема с кодом заключается в том, что спрайт **Кот** поднимается из толщи земли очень медленно. Этот код должен работать так быстро, чтобы игрок видел только спрайт, находящийся на поверхности, но не в земле.

В этом нам помогут темно-фиолетовые пользовательские блоки. Перейдите в категорию **Другие блоки** и нажмите кнопку **Создать блок**. Присвойте блоку имя **Обработка земли** и нажмите серый треугольник, чтобы развернуть список в разделе **Параметры**. Установите флажок **Запустить без обновления экрана**.



Теперь в области **Скрипты** должен появиться блок **Определить обработка земли**. Теперь нужно изменить код спрайта **Кот**, чтобы он мог использовать блок **Обработка земли**. Блок **Обработка земли** помещается на место блока **Повторять пока не касается Земля**, и этот цикл перемещается под блок **Определить обработка земли**.



Этот код работает точно так же, как и раньше, но теперь блок **Обработка земли** содержит обновленный параметр **Запустить без обновления экрана**, поэтому код цикла работает в турборежиме. Подъем кота происходит мгновенно, поэтому кот никогда не погружается в толщу земли.



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить уже готовый код. Двигайте кота, используя клавиши, или используйте мышь, чтобы сбросить кота с верхней части сцены вниз. Теперь спрайт **Кот** никогда не должен погружаться в землю. Затем нажмите кнопку в виде красного знака остановки и сохраните программу. Если у вас возникли проблемы, откройте файл *Платформер\_2.sb* из архива с примерами и продолжайте чтение с этого момента.

## 6 ИСПОЛЬЗОВАНИЕ КРУТЫХ СКЛОНОВ И СТЕН

Спрайт **Земля** имеет возвышенности и склоны, по которым может ходить кот, и вы можете в графическом редакторе придать спрайту **Земля** практически любую форму. Это существенное улучшение по сравнению с хождением по ровной поверхности нижней части сцены, как в игре «Баскетбол». Но теперь проблема в том, что кот может подниматься по крутому склону слева так же легко, как по пологому склону справа. Это не соответствует реальности. Мы хотим, чтобы по крутому склону кот поднимался медленнее. Чтобы сделать это, мы внесем небольшие изменения в блоки кода, отвечающего за ходьбу.

На данный момент спрайты заполнены большим количеством разных скриптов. Поэтому щелкните правой кноп-

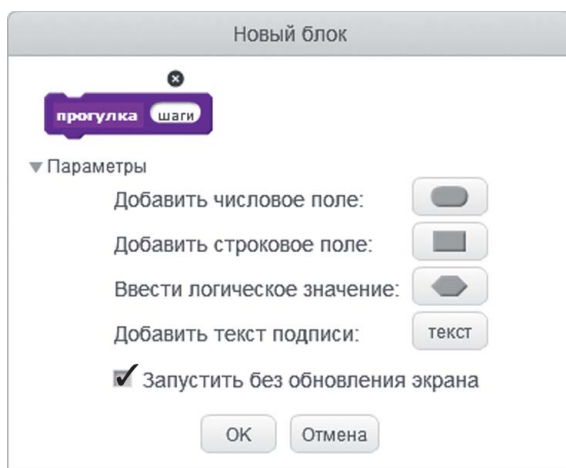
кой мыши в области скриптов и выберите **Очистить**, чтобы выстроить скрипты в аккуратные ряды.

## 5. Добавление кода для крутого склона

Теперь нам нужно отредактировать код ходьбы для спрайта **Кот**, а также добавить новый код. Вместо простого присваивания определенного значения позиции *x* мы будем использовать новый пользовательский блок. Давайте присвоим ему имя **Прогулка** и дадим этому новому пользовательскому блоку *вход* с именем **Шаги**. Вход похож на переменную, но его можно использовать только в пользовательском блоке **Определить**.



Чтобы создать блок **Прогулка**, нажмите кнопку **Создать блок** в темно-фиолетовой категории **Другие блоки**. Нажмите кнопку **Добавить числовое поле**, чтобы сделать вход **Шаги**. Когда мы хотим вызвать новый блок **Прогулка**, нам нужно определить этот вход с помощью некоторого значения **Шаги**. Убедитесь, что установлен флажок **Запустить без обновления экрана**!



Для работы этого кода нужно, чтобы вы создали переменную с именем **Склон** (в режиме **Только для этого спрайта**) для спрайта **Кот**. Мы будем использовать эту переменную для того, чтобы определить, не слишком ли крутой склон для того, чтобы кот мог подняться. Этот код немного сложен, но мы разберем его шаг за шагом. Пока что код спрайта **Кот** выглядит следующим образом:



Нам нужно, чтобы кот прошел шесть единиц, как мы делали это раньше, поэтому мы используем значения **-6** и **6** в скрипте ходьбы при вызове блока **Прогулка**. В блоке **Определить прогулка**, блок ввода **Шаги** используется в блоках

**Изменить x на.** Так код становится более компактным, потому что мы можем использовать один и тот же скрипт для перемещения кота влево (с помощью блока **Прогулка -6**) и вправо (с помощью блока **Прогулка 6**).

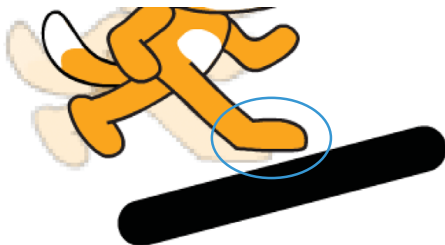
Код в цикле **Повторять пока не** использует значение переменной **Склон**, чтобы определить, является ли склон проходимым для кота или он представляет собой стену, которая должна блокировать движение спрайта **Кот**. Значение переменной **Склон** начинается с 0 и изменяется на 1 каждый раз, когда цикл **Повторять пока не** увеличивает положение спрайта **Кот** по оси y на 1. Этот цикл продолжает работать до тех пор, пока спрайт **Кот** больше не касается земли или значение переменной **Склон** не становится равным 8.

Если значение переменной **Склон** меньше 8, то наклон не очень крутой. Спрайт **Кот** может подняться по склону, поэтому скрипт **Определить прогулка** ничего больше делать не будет.

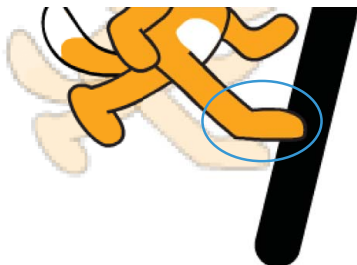
Но если значение переменной **Склон = 8**, цикл **Повторять пока не** перестанет запускаться. Этот код, в переводе на человеческий язык, будет звучать так: «Спрайт был поднят на 8 единиц, но он все еще касается спрайта **Земля**, значит это крутой склон».

В этом случае нам нужно запретить подъем и ходьбу. Блоки **Изменить y на 8** и **Изменить x на -1 \* шаги** блокируют движение спрайта **Кот**. Умножение входа **Прогулка** на -1 дает противоположное значение входа и переменной.

Пологий наклон: Спрайт Кот поднимается на 8 единиц, и больше не касается земли. Спрайт Кот может подняться по этому склону

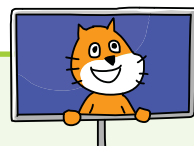


Крутой наклон: Спрайт Кот поднимается на 8 единиц, и все еще касается земли. По этому склону спрайт Кот не может подняться





Этот код в точности похож на код в игре «Бегущий по лабиринту» в главе 3, который запрещает игроку проходить сквозь стены.



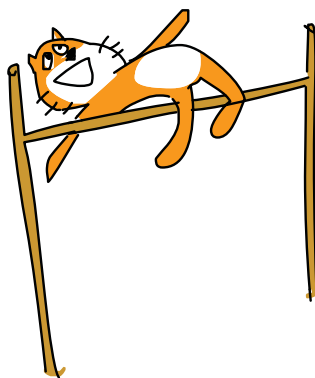
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Используйте клавиши **A** и **D**, чтобы заставить кота ходить. Кот должен подняться по пологому склону справа, но крутой склон слева должен остановить кота. Нажмите красную кнопку остановки и сохраните программу.

Если вы запутались, откройте файл *Платформер\_3.sb* из архива с примерами и продолжайте чтение с этого момента.

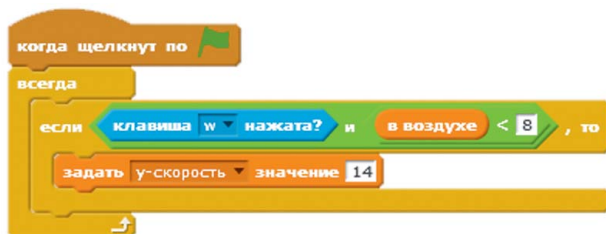
## В ОБУЧЕНИЕ КОТА ВЫСОКИМ И НИЗКИМ ПРЫЖКАМ

Теперь, когда код ходьбы готов, давайте добавим возможность прыжка с помощью клавиши **W**. В игре «Баскетбол» мы присваивали переменной **Падение** положительное значение. Это означало, что игрок каждый раз подпрыгивал на высоту, которая указывается значением этой переменной. Но во многих платформах игрок может делать низкий прыжок, быстро нажимая и отпуская кнопку прыжка или прыгать выше, удерживая нажатой кнопку прыжка. В этом платформере мы будем использовать высокие и низкие прыжки, поэтому нам придется придумать что-то более продвинутое, чем код прыжка для игры «Баскетбол».



## 6. Добавление кода прыжка

Сначала давайте создадим переменную в режиме **Только для этого спрайта** с именем **В воздухе**. Переменная будет иметь значение 0 всякий раз, когда спрайт **Кот** находится на земле. Когда спрайт **Кот** прыгает или падает, значение переменной **В воздухе** начнет увеличиваться. Чем больше значение переменной **В воздухе**, тем дольше будет находиться кот не на земле, а в воздухе. Добавьте код, показанный на следующем рисунке, в спрайт **Кот**:



Цикл **Всегда** продолжает проверять, удерживается ли нажатой клавиша **W**. Если это так, то спрайт Кот получает значение скорости 14, то есть кот будет двигаться вверх. Но обратите внимание, что есть два условия, чтобы кот продолжал двигаться вверх – игрок должен удерживать клавишу **W** и значение переменной **В воздухе** должно быть меньше 8.

Давайте отредактируем два существующих скрипта спрайта **Кот**, чтобы добавить переменную **В воздухе**, которая ограничивает высоту прыжка кота.



Если игрок сначала удерживает клавишу **W**, чтобы кот сделал прыжок, значение переменной **у-скорость** устанавливается равным 14. За это отвечает код в цикле **Всегда** ❶. Он изменяет положение по оси **y** спрайта **Кот** на положительное значение переменной **у-скорость**, перемещая спрайт вверх. В начале прыжка значение переменной **В** воздухе увеличивается, но все равно остается меньше 8. Поэтому, если игрок продолжает удерживать клавишу **W**, значение переменной **у-скорость** продолжает устанавливаться равным 14 вместо того, чтобы уменьшаться. Это происходит из-за блока **Изменить у-скорость на -2**. В этом и состоит причина того, что спрайт совершает прыжок вверх дольше, чем если бы игрок удерживал клавишу **W** всего на одну итерацию в цикле. Но в конечном итоге значение переменной **В** воздухе станет равным или превысит 8, поэтому не будет разницы, нажата ли клавиша **W**. Помните, что оба условия – **Клавиша W нажата** и **В воздухе < 8** – должны быть истинными для кода внутри блока **Если, то**, чтобы блок был запущен.

В этот момент значение переменной **у-скорость** будет уменьшаться, как и ожидалось, и в конечном итоге спрайт **Кот** упадет. В скрипте ❷, когда кот находится на земле, значение переменной **В** воздухе сбрасывается на 0.

## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Нажмите клавишу **W**, чтобы прыгнуть. При быстром нажатии прыжок будет небольшим. При удерживании клавиши **W** прыжок будет выше. Удостоверьтесь, что кот может прыгать, только когда он стоит на земле, и не может делать двойные прыжки. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

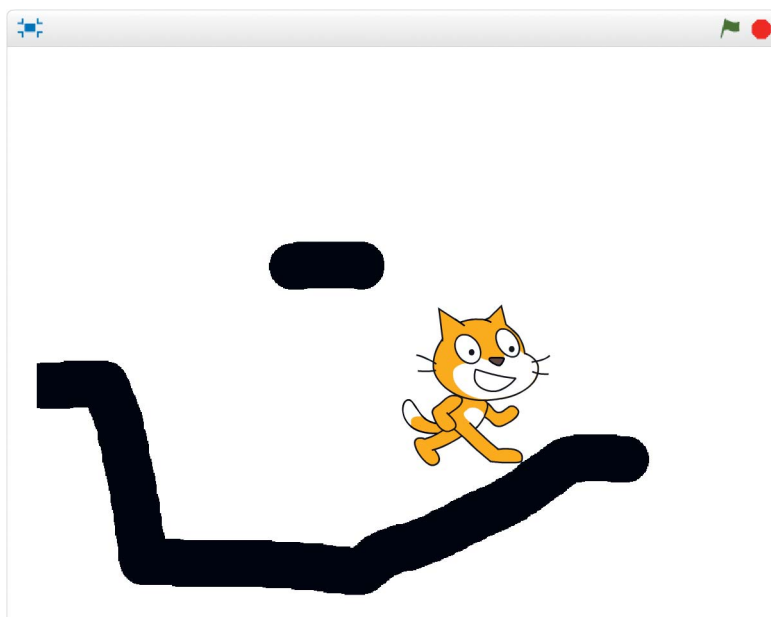
Если у вас возникли проблемы, откройте файл *Платформер\_4.sb* из архива с примерами и продолжайте чтение с этого момента.

## Г ДОБАВЛЕНИЕ ОБНАРУЖЕНИЯ ПРЕПЯТСТВИЙ СВЕРХУ

Теперь кот может ходить по земле, и стены будут препятствовать ему двигаться сквозь них. Но если игрок подпрыгивает и ударяется головой кота снизу о платформу, то спрайт **Кот** поднимается над ней! Чтобы решить эту проблему, нам нужно внести некоторые изменения в код подъема и добавить обнаружение препятствий сверху.

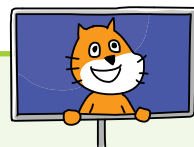
### 7. Добавление низкой платформы к спрайту Земля

Добавьте в костюм **Земля** короткую низкую платформу, как показано на следующем рисунке. Удостоверьтесь, что она расположена достаточно высоко для того, чтобы кот проходил под ней, и достаточно низко, чтобы он мог коснуться ее или запрыгнуть на нее.



Эта платформа должна быть достаточно низкой, чтобы кот мог задеть ее головой. Если это не удастся, перерисуйте платформу немного ниже.

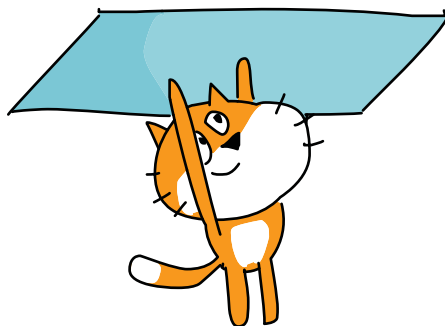
## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Попрыгайте котом под низкой платформой. Обратите внимание, что, когда спрайт **Кот** касается платформы, он оказывается на платформе. Это ошибка, которую нам нужно исправить. Нажмите кнопку в виде красного знака остановки.

## 8. Добавление кода обнаружения препятствия сверху

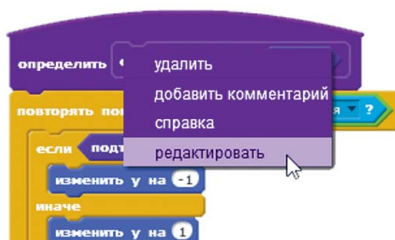
Проблема с кодом находится в пользовательском блоке **Обработка земли**. Этот код предполагает, что спрайт **Кот** всегда падает сверху и, если спрайт **Кот** касается спрайта, он должен быть поднят над ним. Спрайт **Земля** – это некий твердый объект, сквозь который кот не может пройти, что также справедливо и для платформы. Нам нужно изменить код, чтобы, когда спрайт **Кот** подпрыгивает и касается спрайта **Земля**, он *переставал* подниматься, потому что ударяется головой. Мы знаем, что спрайт **Кот** движется вверх, когда значение его переменной **у-скорость** выше 0. Итак, давайте отредактируем пользовательский блок **Обработка земли**, чтобы добавить новый логический вход с именем **Подъем**.



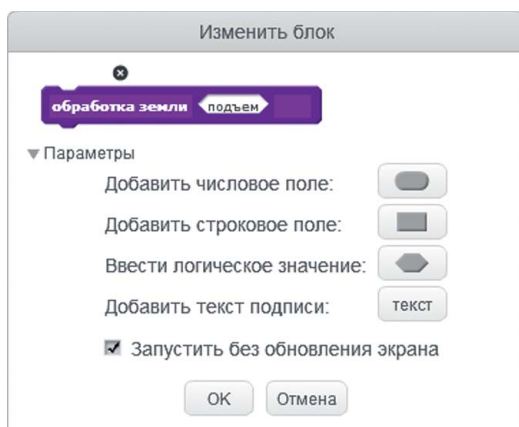
*Логическое (булево)* значение может быть только истинным или ложным. Мы используем логический вход, потому что нам нужно знать, больше ли нуля значение переменной **у-скорость**, когда происходит первое обращение к бло-

ку **Обработка земли**. Это истинное или ложное значение хранится во входе **Подъем** точно так же, как хранилось бы в переменной. Если мы поместим код **у-скорость > 0** в блок **Если, то** вместо **Подъем**, то кот окажется сверху на потолке вместо того, чтобы наткнуться на него.

Щелкните правой кнопкой мыши по блоку **Определить обработка земли** и выберите в меню команду **Редактировать**.



Щелкните мышью по серому треугольнику, чтобы раскрыть раздел **Параметры**, и нажмите кнопку **Ввести логическое значение**. Присвойте этому новому полю ввода **Подъем** и нажмите кнопку **ОК**.



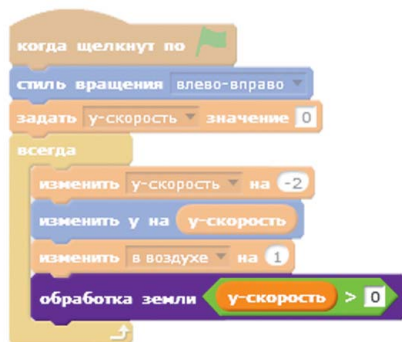
Так вы добавите новый блок **Подъем**, который можете перетащить из блока **Определить обработка земли** точно так же, как вы это делаете с блоками из области блоков. Этот блок **Подъем** будет использоваться в новом блоке **Если, то иначе**. Измените код блока **Определить обра-**

**ботка земли** так, чтобы он соответствовал коду на рисунке ниже.



Если кот движется вверх, то из-за блока **Изменить у на -1** кажется, будто кот ударяется головой. В противном случае скрипт ведет себя так, как он делал это ранее, поднимая кота, чтобы он находился на земле.

Далее в этом скрипте мы должны отредактировать вызов **Обработка земли**. Мы добавим логическое условие, чтобы определить, движется ли спрайт вверх, то есть значение переменной **у-скорость > 0**.



Блок **Обработка земли** **у-скорость > 0** определяет вход **Подъем** как истинный, если значение переменной **у-скорость** больше 0 (то есть если спрайт прыгает и перемещается вверх). Если значение переменной **у-скорость** не превышает 0, то спрайт либо падает, либо стоит, что приводит к тому, что вход **Подъем** определяется как ложный.

Таким образом, блок **Определить обработка земли** определяет, должен ли он запустить блок **Изменить у на -1** (чтобы спрайт **Кот** не мог подняться сквозь потолок) или запустить код **Изменить у на 1** (чтобы спрайт **Кот** был поднят из-под земли). В любом случае, если спрайт **Кот** касается спрайта **Земля** (если запущен код внутри блока **Повторять пока не касается Земля**), переменной **у-скорость** должно быть присвоено значение **0**, чтобы спрайт **Кот** перестал падать или прыгать.



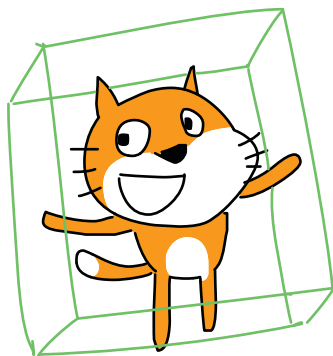
## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить код, готовый к этому моменту. Пройдите котом под низкую платформу и подпрыгните. Кот должен врезаться головой в платформу, а не оказаться сверху на этой платформе. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

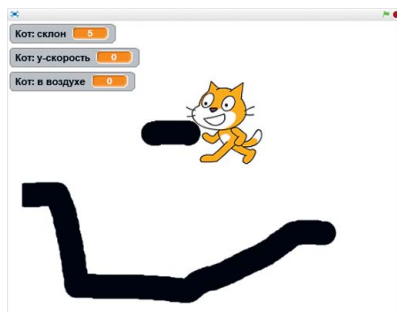
Если у вас возникли проблемы, откройте файл *Платформер\_5.sb* из архива с примерами и продолжайте чтение с этого момента.

## А ИСПОЛЬЗОВАНИЕ ХИТБОКСА ДЛЯ СПРАЙТА КОТ

В этой игре есть еще одна проблема. Поскольку код ориентируется на спрайт **Кот**, касающийся спрайта **Земля**, любая часть спрайта **Кот** может «стоять» на земле, даже усы или щека кота! На этом рисунке кот не падает, потому что его щека «приземлилась» на платформу, что не очень правдоподобно.







К счастью, мы можем исправить эту проблему, используя концепцию «хитбокс», которую применяли в игре «Баскетбол».

## 9. Добавление костюма **Хитбокс** к спрайту **Кот**

Перейдите на вкладку **Костюмы** спрайта **Кот**. Затем нажмите кнопку **Нарисовать новый костюм** и нарисуйте черный прямоугольник, который покрывает большую часть (но не все) площади двух других костюмов. На следующем рисунке показан полупрозрачный первый костюм **Кот**, чтобы вы могли видеть, сколько площади перекрывает черный прямоугольник.

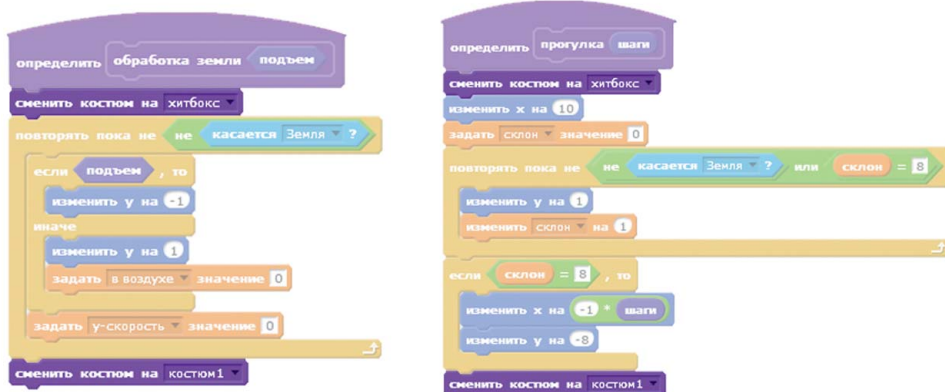


Присвойте этому костюму имя **Хитбокс**. Всякий раз, когда код платформера проверяет, касается ли спрайт **Кот** спрайта **Земля**, мы перед проверкой переключаем костюм на черный прямоугольник костюма **Хитбокс**, а после проверки возвращаемся к обычному костюму. Совершая эти действия, Хитбокс определяет, касается ли кот земли.

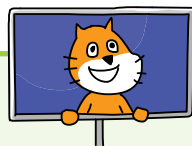
Эти переключения костюма будут закодированы темно-фиолетовыми пользовательскими блоками с параметром **Запускать без обновления экрана**, поэтому костюм **Хитбокс** никогда не отобразится на экране.

## 10. Добавление кода хитбокса

Мы добавим блоки **Сменить костюм на** в начале и конце обоих темно-фиолетовых пользовательских блоков. Измените код спрайта **Кот** так, чтобы он соответствовал коду на следующем рисунке:



Эти блоки из фиолетовой категории **Внешность** будут изменять костюм на **Хитбокс**. Поскольку **Хитбокс** представляет собой простой прямоугольник, не имеющий выступающих частей, которые могут «цепляться» за платформы, как, например, голова или усы кота, игра будет больше соответствовать реальности.



### КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить измененную часть кода. Подпрыгните своим котом и убедитесь, что он не будет повисать на платформе, зацепившись за нее щекой или хвостом. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Если у вас возникли проблемы, откройте файл *Платформер\_6.sb* из архива с примерами и продолжайте чтение с этого момента.

## Е УЛУЧШЕНИЕ АНИМАЦИИ ХОДЬБЫ

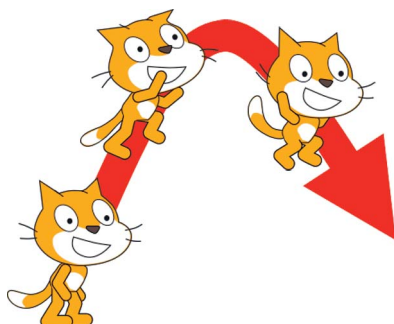
Спрайт **Кот**, с которого начинается любой проект Scratch, изначально содержит два костюма с именами **Костюм1** и **Костюм2**.



Конечно, вы можете сделать простую анимацию ходьбы, переключаясь только между этими двумя костюмами. Но пользователь Scratch с ником grifffpatch создал целую серию костюмов ходьбы для кота.



Этот пользователь также сделал костюмы для стоящего, прыгающего и падающего кота.



Использование этих костюмов сделает игру более привлекательной, чем если бы вы применяли только те два стандартных костюма, которые есть у спрайта **Кот**. Нам просто нужно добавить код анимации, который в нуж-

ное время будет переключаться между этими костюмами. Пользователь griffpatch создал несколько классных программ Scratch с использованием этих костюмов – см. сайт [scratch.mit.edu/users/griffpatch/](http://scratch.mit.edu/users/griffpatch/).

## II. Добавление новых костюмов к спрайту Кот

Чтобы добавить новые костюмы, вы должны загрузить файлы костюмов в свой проект Scratch. Вы найдете восемь рисунков ходьбы, а также рисунки стоящего кота, прыгающего и падающего в архиве с примерами. Имена файлов для этих изображений – *Прогулка1.svg*, *Прогулка2.svg* и так далее вплоть до *Прогулка8.svg*, а также *Остановка.svg*, *Прыжок.svg* и *Падение.svg*.

Затем в редакторе Scratch перейдите на вкладку **Костюмы** спрайта **Кот**. Нажмите кнопку **Загрузить костюм из файла** рядом с названием раздела **Новый спрайт** и выберите документ *Остановка.svg*, чтобы загрузить файл. Так будет создан новый костюм с именем *Остановка.svg*.

Удалите исходные костюмы **Костюм1** и **Костюм2**, но сохраните костюм **Хитбокс**. Поместите костюмы в следующем порядке (важно, чтобы этот порядок вы соблюдали точно) и со следующими именами:

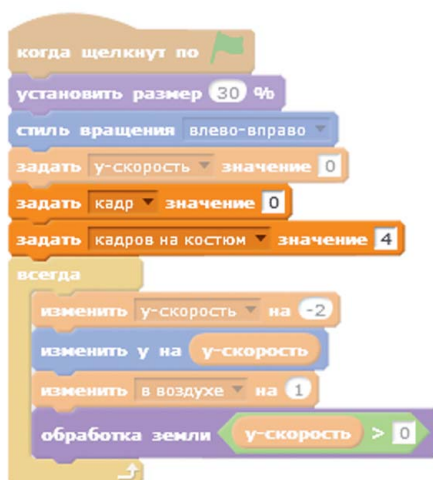
- |                |              |               |
|----------------|--------------|---------------|
| 1. Остановка_1 | 5. Прогулка2 | 9. Прогулка6  |
| 2. Прыжок_1    | 6. Прогулка3 | 10. Прогулка7 |
| 3. Падение_1   | 7. Прогулка4 | 11. Прогулка8 |
| 4. Прогулка1   | 8. Прогулка5 | 12. Хитбокс   |

Каждый костюм имеет не только имя (например, **Прогулка1**, **Прыжок\_1** или **Падение\_1**), но также и число. Номер костюма присваивается в соответствии с порядком расположения костюма на вкладке **Костюмы**. Например, верхний костюм называется **Остановка\_1**, но также он является костюмом 1. Костюм под ним называется **Прыжок\_1**, но он также известен как костюм 2. Код, который мы добавим на следующем шаге, будет обращаться к костюмам по их именам и номерам.

## 12. Создание набора правильных блоков костюмов

Поскольку мы используем много разных костюмов, будет немного сложно определить, какой рисунок нам нужно показать и когда. Мы будем использовать идею анимационных кадров: несколько кадров, быстро показанных друг за другом, создают движущееся изображение.

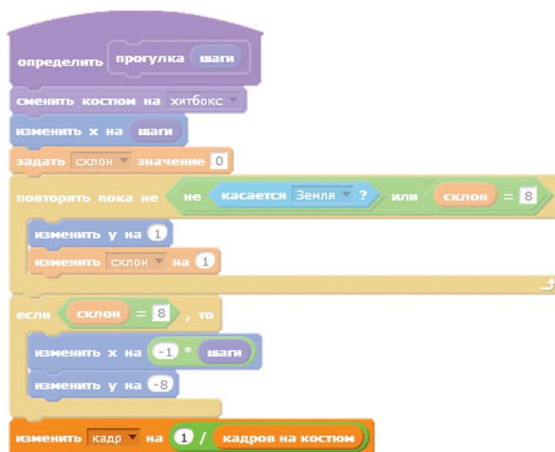
Чтобы отслеживать кадры, создайте две переменные в режиме **Только для этого спрайта** и с именами **Кадр** и **Кадров на костюм**. Затем добавьте два блока **Задать для этих исходных переменных** в скрипт **Когда щелкнут по зеленому флагу спрайта Кот**.



Теперь установка завершена.

Когда игрок перемещает спрайт **Кот** влево или вправо, нужно, чтобы значение переменной **Кадр** увеличивалось. Переменная **Кадров на костюм** отслеживает, насколько быстро или медленно выполняется анимация.

Давайте изменим код в пользовательском блоке **Определить прогулка**, чтобы увеличить значение переменной **Кадр** на сумму, рассчитанную переменной **Кадров на костюм**.

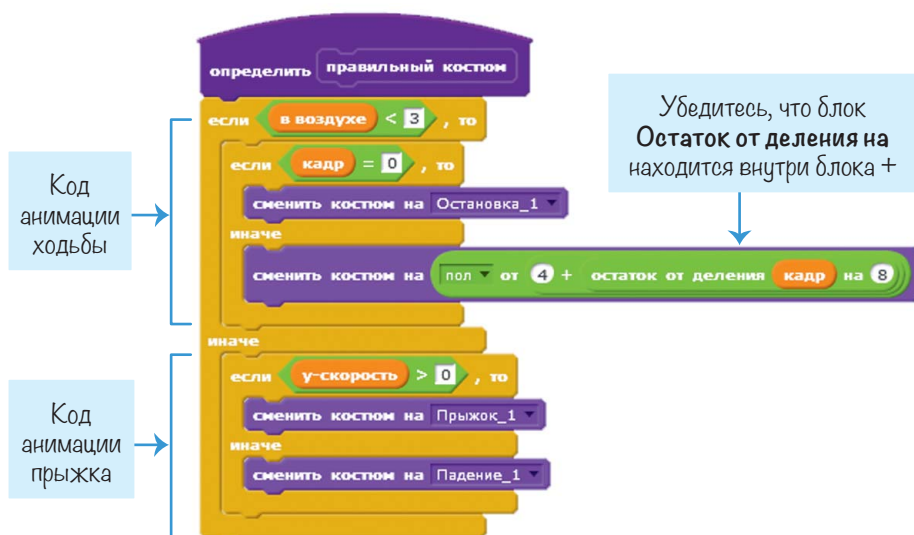


Когда кот стоит на месте (то есть не двигается влево или вправо), значение переменной **Кадр** должно быть сброшено на 0. Измените существующий код **Когда щелкнут по зеленому флагу спрайта Кот**, чтобы добавить третий блок **Если, то**, который сбрасывает значение переменной **Кадр**.



Теперь давайте напишем код, который будет определять, какой показывать костюм. Мы будем использовать этот код в нескольких местах в скриптах, которые мы написали, поэтому давайте создадим пользовательский блок.

В темно-фиолетовой категории **Другие блоки** нажмите кнопку **Создать блок** и присвойте блоку имя **Правильный костюм**. Нажмите серый треугольник со списком **Параметров**, установите флажок «Запускать без обновления экрана» и затем нажмите кнопку **ОК**. Добавьте следующие блоки к спрайту **Кот**, начиная с нового блока **Определить правильный костюм**.



Если спрайт **Кот** находится на земле (или только что начал прыгать или падать, при этом значение переменной **В воздухе** меньше 3), нам нужно, чтобы программа показывала либо костюм **Остановка\_1**, либо один из костюмов **Прогулка**. Помните, что скрипт **Когда щелкнут по зеленому флагу** сохраняет значение переменной **Кадр** равным 0, если игрок не нажимает клавишу **A** или **D**. Поэтому, когда значение переменной **Кадр** равно 0, блок **Сменить костюм на Остановка\_1** отображает костюм **Остановка\_1**. В противном случае нужно вычислить, какой из восьми костюмов **Прогулка** показать. Этот расчет отправляет к



костюмам по их номерам, которые присваиваются согласно порядку их расположения на вкладке **Костюмы**.

Решение о том, какой костюм показать, принимает блок **Сменить костюм на пол от 4 + остаток от деления кадр на 8**. Ничего себе, этот блок выглядит сложным! Давайте разберем его, чтобы лучше понять каждую часть.

Блок **Остаток от деления на** выполняет математическую операцию деление по модулю, результатом которой является остаток от деления целого числа на другое целое число. Например, результат деления по модулю выражения  $7/3$  будет 1, так как  $7/3 = 2 + \text{остаток от деления} - 1$ . Мы будем использовать **Остаток от деления на** для расчета номера костюма, который нужно показать.

Значение переменной **Кадр** продолжает увеличиваться, хотя у нас есть только восемь костюмов **Прогулка**. Когда значение переменной **Кадр** находится в диапазоне от 0 до 7, нужно, чтобы отображались костюмы с 4 по 11. Вот почему наш код имеет блоки **4 + остаток от деления кадр на**. Но когда значение переменной **Кадр** увеличивается до 8, нам необходимо вернуться к костюму 4, а не к костюму 12.

Блок **Остаток от деления на** поможет нам совершить это возвращение с номерами костюмов. Мы можем управлять, какой костюм отображается с помощью математического трюка: поскольку остаток от деления  $8/8$  будет равен 0, значение переменной **Кадр** равное 8 отобразит первый костюм **Прогулка**! К этому числу мы должны добавить 4, потому что первый в списке костюм **Прогулка** – это фактически костюм 4. (Как вы помните, костюм 1, костюм 2 и костюм 3 – это костюмы кота, который стоит, прыгает и падает соответственно). Эта сумма затем используется блоком **Пол**. Это термин программирования, означающий «округление к меньшему». Иногда значение переменной **Кадр** будет числом вроде 4,25 или 4,5, поэтому **4 + остаток от деления кадр на** будет составлять 8,25 или 8,5, но мы просто хотим округлить число к меньшему, чтобы показать костюм 8.

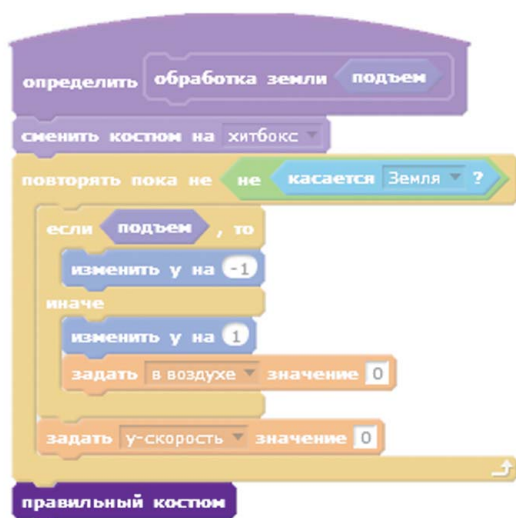


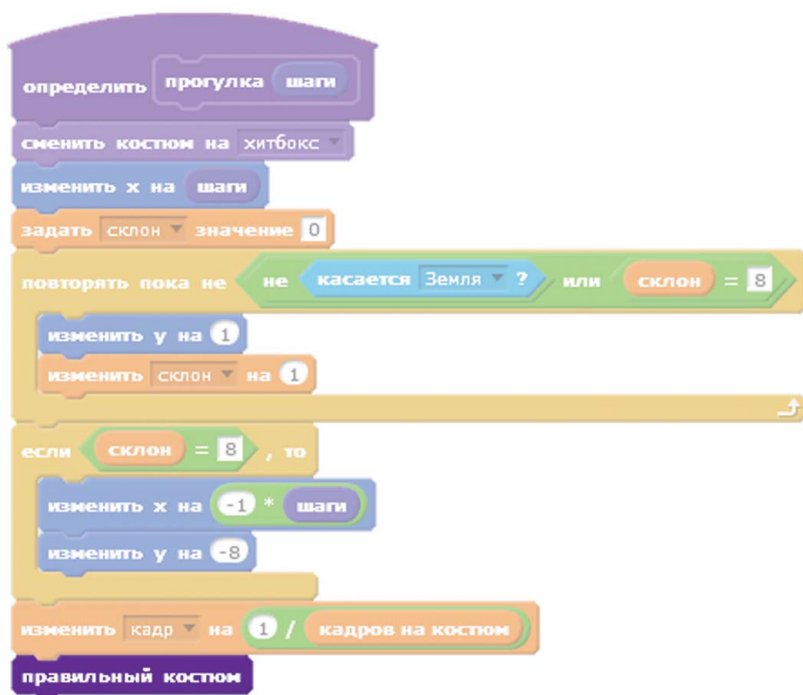
Вот так! Это самая сложная математика, которой вы воспользовались в этой книге, но, когда все разложено по полочкам, ее становится легче понять.



Код в части **Иначе** блока **Если, то иначе** отвечает за то, что происходит, если значение переменной в воздухе больше или равно 3. Мы проверяем значение переменной **у-скорость**, чтобы увидеть, падает ли кот (то есть если значение переменной **у-скорость** меньше или равно 0) или подпрыгивает (то есть если значение переменной **у-скорость** больше 0) и переключиться на правильный костюм. На этом заканчивается код **Определить правильный костюм**.

Замените блок **Сменить костюм на костюм1** в блоках **Определить обработка земли** и **Определить прогулка** на новые блоки **Правильный костюм**. Кроме того, добавьте блоки **Изменить кадр на 1 / кадров на костюм**, чтобы значение переменной **Кадр** все время увеличивалось, как показано на рисунке ниже.





## КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Управляя котом с помощью клавиш, заставьте его прогуляться по сцене и убедитесь, что анимация ходьбы отображается правильно. Кроме того, убедитесь, что костюмы **Остановка\_1**, **Прыжок\_1** и **Падение\_1** отображаются в нужное время. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Если у вас возникли проблемы, откройте файл *Платформер\_7.sb* из архива с примерами и продолжайте чтение с этого момента.

## Ж СОЗДАНИЕ УРОВНЯ

Новая анимация ходьбы делает наш платформер более привлекательным. Теперь давайте изменим простой белый фон на реальный уровень. Что хорошо в коде, который мы написали для спрайта **Кот**, так это то, что ходить, прыгать и падать он будет со спрайтом **Земля** любой формы или цвета. Так что если мы изменим костюм спрайта **Земля** (скажем, для разных уровней), нам не придется перепрограммировать спрайт **Кот**!

### 13. Загрузка и добавление фона для сцены

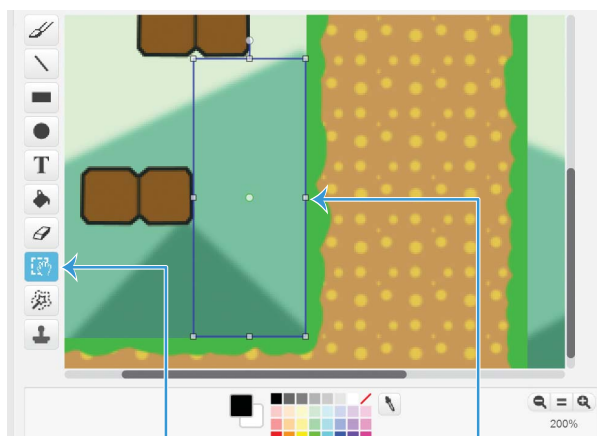
Перейдите на вкладку **Костюмы** спрайта **Земля**. Нажмите кнопку **Загрузить костюм из файла** и выберите файл *ФонПлатформера.png*, который находится в архиве с примерами. После того как этот костюм загружен, вы можете удалить предыдущий костюм.

Недостаточно добавить файл *ФонПлатформера.png* в качестве костюма для спрайта **Земля**. Вы также должны загрузить его в качестве фона сцены. Нажмите кнопку **Загрузить фон из файла** под названием раздела **Новый фон** и выберите *ФонПлатформера.png*, чтобы загрузить его. Нам нужно загрузить файл в обе позиции, потому что в следующем шаге мы будем стирать все «фоновые элементы» из спрайта **Земля**. Нам нужен только спрайт **Земля**, чтобы указать, по каким его частям спрайт **Кот** может ходить. Фон будет изображением, которое отображается на сцене.

### 14. Создание хитбокса для спрайта **Земля**

Код игры-платформера основан на том, что спрайт **Кот** коснулся спрайта **Земля**. Костюм спрайта **Земля** – это хитбокс, поэтому, если костюм спрайта **Земля** – это прямоугольник, который занимает всю сцену, он будет взаимодействовать со сценой так, как будто вся сцена – сплошная земля. Нам нужно удалить области костюма спрайта **Земля**, которые являются частью фона, а не платформ.

Самый простой способ сделать это – использовать инструмент **Выбрать** в графическом редакторе. Перетащите прямоугольник области выделения на область костюма, которую вы хотите удалить. Выбрав область, нажмите клавишу **Del**, чтобы удалить эту часть.

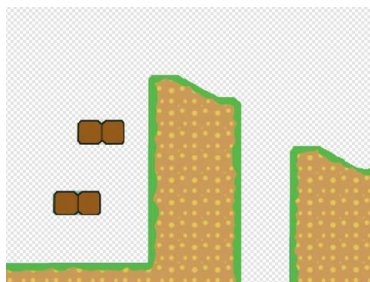


1 Выберите инструмент **Выбрать**

2 Перетащите прямоугольник области выделения на область, которую нужно удалить, и нажмите клавишу **Del**

Используйте инструмент **Ластик** для стирания непрямоугольных областей. Если вы допустили ошибку, нажмите кнопку **Отменить** в верхней части графического редактора для отмены удаления.

Продолжайте удалять фоновые области костюма, пока не останутся только изображения платформ.

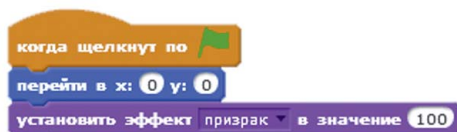


Если при создании этого костюма у вас возникли проблемы, вы можете использовать готовый файл *ФонПлатформера\_хитбокс.png* из архива с примерами. Фоновые

области изображения уже удалены, поэтому, чтобы его добавить, вам просто нужно нажать кнопку **Загрузить костюм из файла** на вкладке **Костюмы**.

## 15. Добавление кода спрайта **Земля**

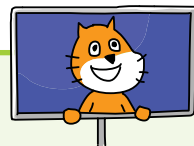
Задник сцены нужен для того, чтобы разместить на нем изображения платформ и фона. Спрайт **Земля** нужен для того, чтобы определить, какие части являются твердыми, чтобы спрайт **Кот** мог по ним пройти. Добавьте код, показанный на следующем рисунке, в область скриптов спрайта **Земля**:



Костюм спрайта **Земля** должен располагаться точно поверх задника сцены, чтобы они совпадали. Поскольку фон сцены и костюм спрайта **Земля** были получены из одного и того же файла изображения, вы можете сделать это, переместив спрайт **Земля** в координаты (0, 0). В противном случае костюм спрайта **Земля** не будет идеально совпадать с фоном.

*Рисунок* костюма спрайта **Земля** не имеет такого значения, как *форма* костюма. Поскольку спрайт **Земля** идеально совпадает с фоном, мы можем назначить эффект **Призрак** со значением 100, и костюм земли и фон будут выровнены. Фон показывает, как выглядит уровень, в то время как спрайт **Земля** действует как его хитбокс.

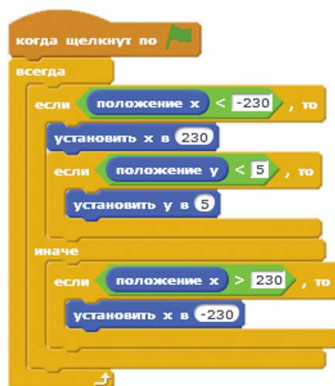
### КОНТРОЛЬНАЯ ТОЧКА



Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что кот может передвигаться по сцене. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

## 16. Добавление дополнительного кода в спрайт Кот

Обратите внимание, что на уровне есть пара висящих в воздухе платформ и холм с провалом посередине. Когда кот падает в провал, он вылетает за пределы сцены и появляется сверху. Давайте добавим код пересечения края для левого и правого краев сцены. Добавьте код, показанный на следующем рисунке, в спрайт **Кот**.

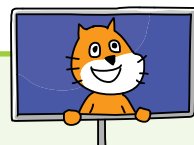


Когда кот подходит к левому краю сцены, ее **Положение x** будет меньше  $-230$ . В этом случае мы делаем его выходящим за пределы сцены и появляющимся справа, устанавливая **Положение x** равным  $230$ .

Кроме того, если кот двигается с левой стороны сцены, его **Положение y** будет меньше  $5$ . При пересечении края он окажется в грунте на правом краю сцены, поэтому блок **Если, то** проверяет это условие и устанавливает **Положение y** равным  $5$ .



Другой блок **Если, то** переносит кота на левый край сцены, когда он находится на правом краю (то есть его **Положение x** больше 230).



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Удостоверьтесь, что кот может уйти с левого края сцены и показаться справа и наоборот. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Если у вас возникли проблемы, откройте файл *Платформер\_8.sb* из архива с примерами и продолжайте чтение с этого момента.

## 3 ДОБАВЛЕНИЕ КРАБОВ И ЯБЛОК

Настройка целой *игры-платформера* завершена! Игрок может заставить кота ходить, прыгать, падать и стоять на платформах. У спрайта **Кот** есть несколько классных анимаций, и фон выглядит как настоящая видеоигра.

Теперь все, что нам нужно сделать, это собрать игру, используя те части, которые у нас есть. Мы добавим яблоко, которое будет появляться на сцене случайным образом (как мы делали это в игре «Зме-е-ейка!» в главе 6), и нескольких врагов, которые будут пытаться коснуться спрайта **Кот** и украсть яблоки.

### 17. Добавление спрайта **Яблоко** и кода для него

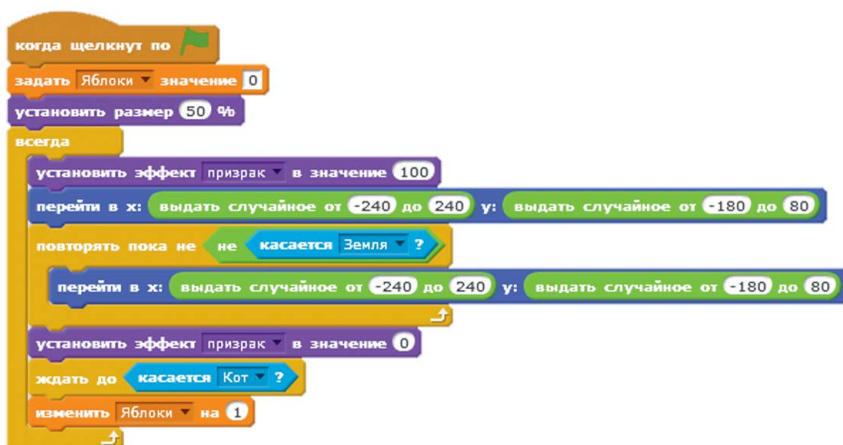
Нажмите кнопку **Выбрать спрайт из библиотеки** и в появившемся окне **Библиотека спрайтов** выберите спрайт **Apple**, а затем нажмите кнопку **ОК**. Для удобства вы можете переименовать спрайт, присвоив ему имя **Яблоко**.

Как и в предыдущих играх, мы будем использовать переменную для отслеживания счета игры. Нажмите оранжевую категорию **Данные**, затем нажмите кнопку **Создать**



переменную, чтобы создать переменную в режиме **Для всех спрайтов** и с именем **Яблоки**. Эта переменная будет отслеживать счет игрока.

В области скриптов спрайта **Яблоко** добавьте код, показанный на следующем рисунке.



В начале игры, когда игрок нажимает кнопку в виде зеленого флага, счет – значение переменной **Яблоки** – равен нулю. Кроме того, поскольку спрайт **Яблоко** слишком велик, мы уменьшим его размер на 50 процентов.

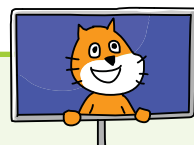
Во время игры спрайт **Яблоко** должен появляться на сцене случайным образом в разных местах. Мы делаем спрайт **Яблоко** невидимым, используя блок **Установить эффект призрака в значение 100**. Затем он перемещается по сцене в случайное место.

Но любое случайное место на сцене для нашей игры не подходит. Нам нужно убедиться, что спрайт **Яблоко** находится *не внутри* спрайта **Земля**, потому что игроку будет невозможно получить его. Чтобы не допустить появление спрайта **Яблоко** в каком-либо месте спрайта **Земля**, цикл продолжает генерировать новые случайные места, пока спрайт **Яблоко** не перестанет касаться спрайта **Земля**. Игрок не будет видеть перемещения яблока, поскольку эффект **Призрака** по-прежнему имеет значение 100. Когда спрайт **Яблоко** обнаруживает место, которое не касается



спрайта **Земля**, он снова становится видимым с помощью кода **Установить эффект призрак в значение 0**.

Затем спрайт **Яблоко** ждет, пока спрайт **Кот** его коснется. Когда это происходит, значение переменной **Яблоки** увеличивается на один, а циклы спрайта **Яблоко**ходят на сцене новое случайное место.



## КОНТРОЛЬНАЯ ТОЧКА

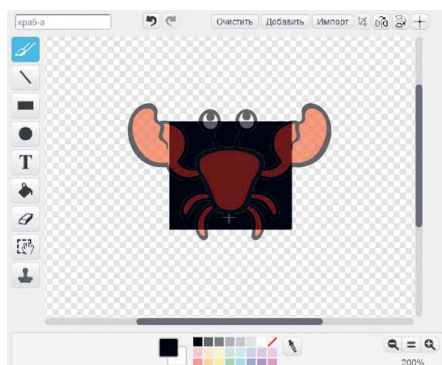
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Проверьте, чтобы спрайт **Яблоко** никогда не появлялся внутри земли. Когда кот касается яблока, значение переменной **Яблоки** увеличивается на 1, и спрайт **Яблоко** должен появиться в новом случайном месте. Нажмите красную кнопку остановки и сохраните программу.

## 18. Создание спрайта **Краб**

Игра будет очень простой, если все, что нужно делать, — это прыгать и собирать яблоки. Давайте добавим врагов, которых игрок должен будет избегать.

Щелкните правой кнопкой мыши по спрайту **Кот** и выберите команду **Дублировать** в контекстном меню. Для врагов мы будем использовать тот же код, что и для спрайта **Кот**, чтобы они могли прыгать и падать на платформы. (Мы удалим код, который отвечает за управление спрайтом клавишами клавиатуры, и заменим его кодом, который будет отвечать за случайное передвижение крабов.) Переименуйте этот дублированный спрайт, присвоив ему имя **Краб**. Затем на вкладке **Костюмы** нажмите кнопку **Выбрать костюм из библиотеки**, выберите костюм **Crab-a** и нажмите кнопку **ОК**. Затем снова откройте библиотеку костюмов и выберите костюм **Crab-b**. Для удобства вы можете переименовать эти костюмы, присвоив им имена **Краб-а** и **Краб-б** соответственно.

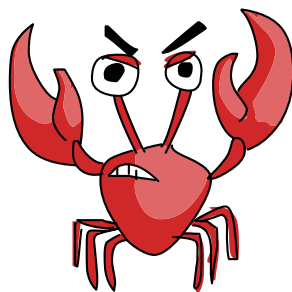
Спрайт **Краб** по-прежнему обладает всеми костюмами спрайта **Кот** (**Остановка\_1**, **Падение\_1**, **Прогулка1** и т. д.). Удалите эти костюмы кота, включая **Хитбокс**. Создайте новый костюм **Хитбокс**, который подходит для краба по размеру. Ниже показано, как должен выглядеть костюм **Хитбокс** (костюм краба показан поверх него, чтобы вы могли видеть относительный размер, но костюмом является только черный прямоугольник).



## 19. Разработка искусственного интеллекта врага

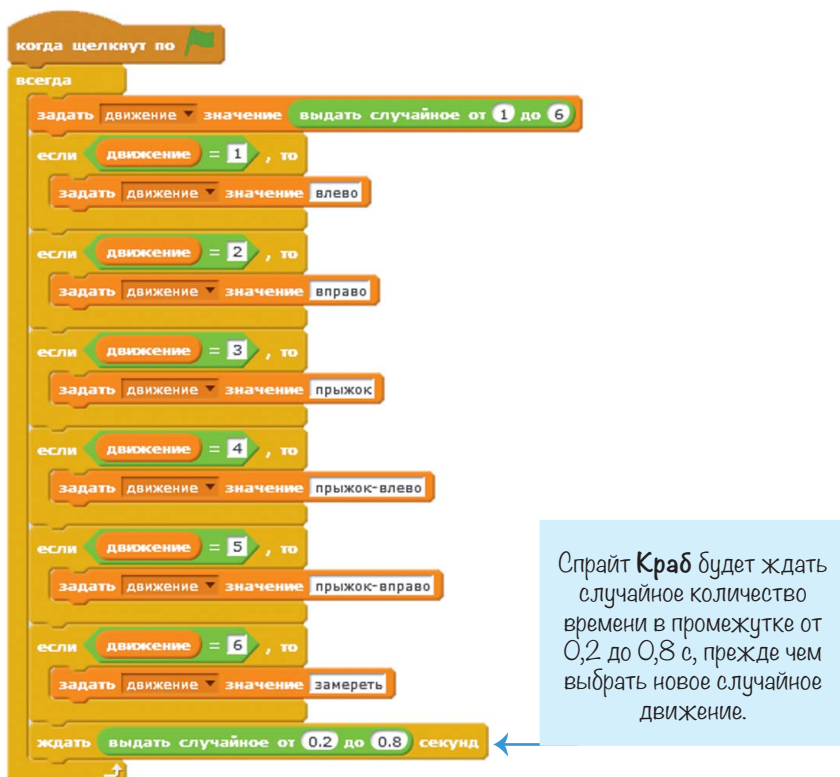
В играх понятие *искусственный интеллект (ИИ)* относится к коду, который управляет движениями врагов и их реакцией на игрока. В нашем платформере крабы не особенно блещут интеллектом: они будут просто двигаться случайным образом.

В оранжевой категории **Данные** нажмите кнопку **Создать переменную** и установите переключатель в положение **Только для этого спрайта**. Присвойте переменной имя **Движение**. Значение переменной **Движение** будет содержать число, отражающее движения краба:



- |                 |                   |
|-----------------|-------------------|
| 1. Идет влево   | 4. Прыжок влево   |
| 2. Идет вправо  | 5. Прыжок вправо  |
| 3. Прыжок вверх | 6. Стоит на месте |

Движения спрайта **Краб** будут определяться произвольно и часто меняться. Добавьте код, показанный на следующем рисунке, в спрайт **Краб**.



В самом начале переменной **Движение** задается случайное число в диапазоне от 1 до 6, которое определяет, какое движение сделает краб.

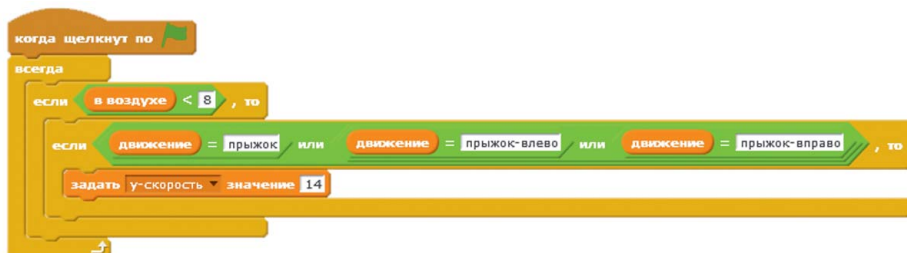
Остальной код спрайта **Краб** определяет эти движения. Найдите код из спрайта **Кот**, где использовался блок **Если клавиша нажата**, и замените его блоками, которые проверяют значение переменной **Движение**. (Если вы запустите программу прямо сейчас, вы увидите, что клавиши клавиатуры будут управлять спрайтами *и Кот и Краб*, потому что они имеют одинаковый код!)

Измените скрипт спрайта **Краб**, который проверяет, нажимает ли игрок клавиши **A** или **D**, чтобы он соответствовал коду, показанному на рисунке ниже.



Как и в случае со спрайтом **Кот**, этот код позволяет спрайту **Краб** ходить влево и вправо. Измените значения в блоке **Прогулка** на  $-4$  и  $4$ , чтобы краб двигался медленнее, чем игрок.

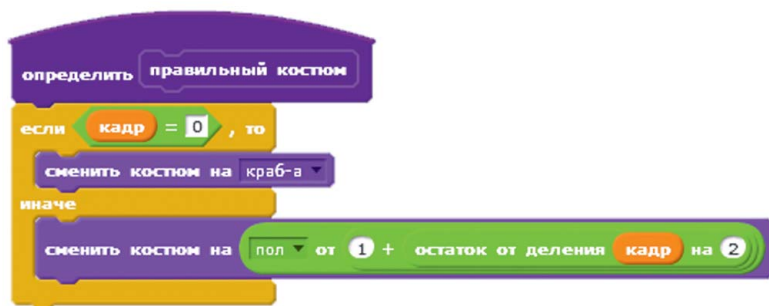
Затем измените сценарий, который отвечает за прыжок персонажа, при нажатии клавиши **W**, чтобы он соответствовал коду, показанному на рисунке ниже.



Этот код позволяет спрайту **Краб** прыгать вверх, влево и вправо.

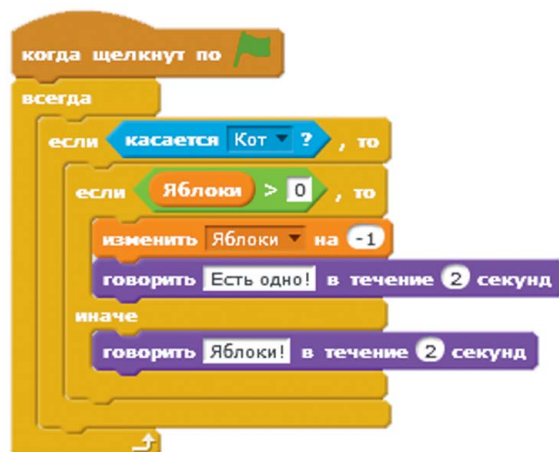
Теперь давайте анимируем движения краба. Спрайт **Краб** содержит только два костюма: **Краб-а** и **Краб-б**. Мы будем переключаться между этими двумя костюмами, чтобы казалось, что краб ходит. Мы можем несколько упростить блок **Правильный костюм** для спрайта **Краб**.

Измените код блока **Правильный костюм** так, чтобы он выглядел следующим образом:



Обратите внимание, что числа в полях блоков **Пол от 1 + остаток от деления кадр на 2** также изменились. Первый – это костюм 1, а у краба только два костюма, поэтому номера в этих блоках были изменены на 1 и 2.

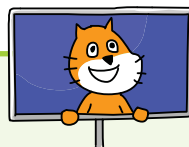
Наконец, нам нужно создать новый скрипт, чтобы крабы могли украсть яблоки у игрока. Добавьте код, показанный на рисунке ниже, в спрайт **Краб**.



Когда спрайт **Краб** касается игрока, он уменьшает на единицу значение переменной **Яблоки** и говорит: «Есть одно!» Если у игрока нет яблок, спрайт **Краб** скажет «Яблоки!» – и значение переменной **Яблоки** не уменьшится.

С двумя крабами играть будет интереснее, поэтому щелкните правой кнопкой мыши по спрайту **Краб** в области спрайтов и выберите в контекстном меню пункт **Дублировать**.

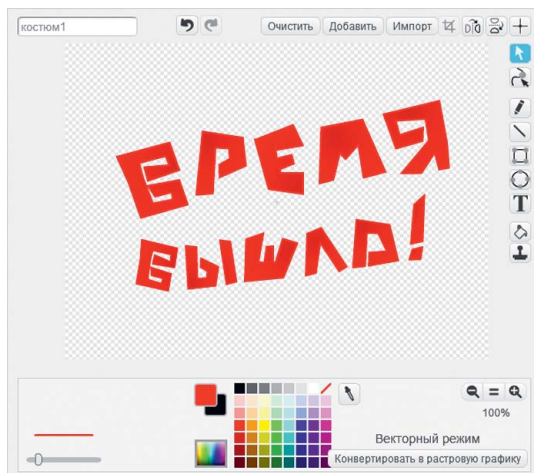
## КОНТРОЛЬНАЯ ТОЧКА



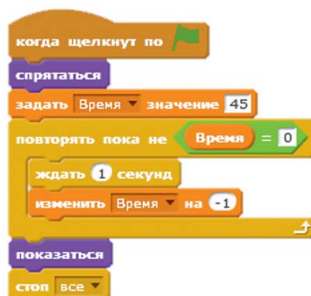
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что на сцене движутся два краба. Когда один из них коснется кота, краб должен сказать: «Есть одно!» – и одно из ваших яблок должно пропасть. Если же у кота нет яблок, краб просто должен сказать: «Яблоки!» Проверив это, нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

## 20. Добавление спрайта **Время вышло**

Мы почти закончили! Последнее, что нам нужно добавить в игру – это таймер. Ограниченное количество времени будет стимулировать игрока собирать яблоки как можно быстрее, вместо того, чтобы играть медленно и аккуратно. Нажмите кнопку **Создать новый спрайт** и нарисуйте текст «Время вышло» в графическом редакторе. Мой рисунок выглядит следующим образом:



Присвойте спрайту имя **Время вышло**. Затем создайте переменную в режиме **Для всех спрайтов** с именем **Время** и добавьте код, показанный на следующем рисунке.



Этот код дает игроку 45 секунд, чтобы собрать как можно больше яблок, по возможности избегая крабов, которые их крадут. Когда значение переменной **Время** достигает 0, появляется спрайт **Время вышло**, и игра заканчивается.

Теперь ваша игра «Платформер» готова к финальному тестированию!



## КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить код игры. Ходите и прыгайте, собирая яблоки и пытаясь избежать крабов. Убедитесь, что, когда таймер достигает 0, игра заканчивается. Затем нажмите кнопку в виде красного знака остановки и сохраните программу. Код для этой программы слишком велик, чтобы полностью привести его в этой книге. Поэтому вы можете просмотреть полный код в архиве с примерами. Имя файла – *Платформер\_улучшенный.sb2*.

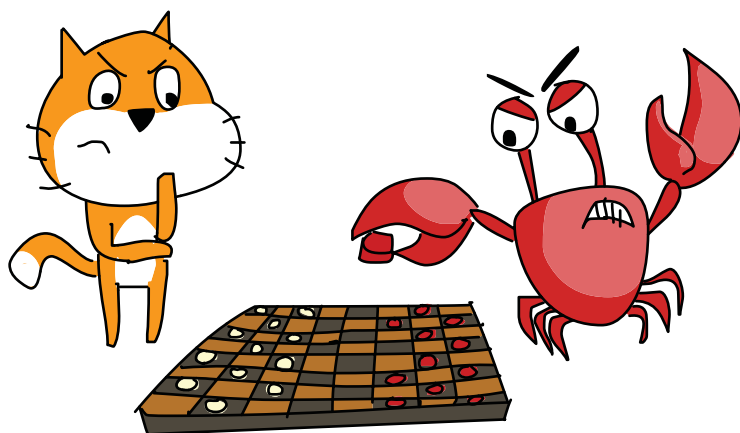
## ЗАКЛЮЧЕНИЕ

Вы сделали это! Вас можно назвать мастером Scratch! Продвинутая игра «Платформер» – самый сложный и объемный проект в этой книге. Вы объединили и использовали множество различных концепций, чтобы сделать эту игру, так что эту главу стоит прочитать еще несколько раз.

В этой главе вы создали игру, в которой:

- ▶ используется спрайт земли, на котором стоит игрок;
- ▶ используются темно-фиолетовые пользовательские блоки с параметром **Запуск без обновления экрана**;
- ▶ игрок может ходить вверх и вниз по наклонным поверхностям;
- ▶ поддерживается распознавание потолка, поэтому игрок ударяется головой о низкие платформы;
- ▶ реализована подробная анимация ходьбы, прыжков и падения;
- ▶ реализован искусственный интеллект врагов, поэтому они могут передвигаться самостоятельно.

Эта книга подошла к концу, но это не мешает вам продолжать эксперименты в программировании. Вы всегда можете посмотреть проекты других скретчеров, чтобы вдохновиться какой-то идеей. Найдите игру, которая вам понравится, и попробуйте создать ее с нуля. Самое замечательное в Scratch – это то, что вам предоставляются неограниченные возможности для создания игр. Вы можете создавать клоны популярных классических игр, таких как *Pac-Man* или *Flappy Bird*. Или вы можете создавать уникальные игры, используя свои собственные проекты. Удачи!





## ОБЗОРНЫЕ ВОПРОСЫ

Попробуйте ответить на следующие практические вопросы, чтобы проверить свои знания. Возможно, вы пока не знаете все ответы, зато вы всегда можете лучше узнать Scratch и выяснить недостающее. (Ответы также можно посмотреть в конце книги.)

1. Темно-фиолетовые пользовательские блоки помогают избежать дублирования кода. Почему это хорошо?
2. В чем сходство ввода темно-фиолетового пользовательского блока и переменной?
3. Где можно использовать вход темно-фиолетового пользовательского блока?
4. Что в математике означает понятие *деление по модулю*?
5. Что в программировании означает слово **Пол**?

# ДОПОЛНИТЕЛЬНЫЕ РЕСУРСЫ

Готовы к большему? Существует множество ресурсов Scratch, чтобы не дать вам заскучать.

- ▶ **Scratch Programming Playground Studio** ([www.inventwithscratch.com/studio/](http://www.inventwithscratch.com/studio/)) – ресурс, на котором вы можете делиться своими проектами, в том числе основанными на играх, описанных в этой книге. Вы можете сравнить свои проекты с играми других читателей.
- ▶ **Форумы Scratch** ([scratch.mit.edu/discuss/](http://scratch.mit.edu/discuss/)) – на этой дискуссионной площадке скретчеры делятся идеями, а также задают вопросы и отвечают на них.
- ▶ **ScratchEd** ([scratched.gse.harvard.edu](http://scratched.gse.harvard.edu)) – онлайн-сообщество, созданное для преподавателей, которые используют Scratch для обучения. Здесь можно написать свою историю успеха, обменяться ресурсами Scratch, задать вопрос и многое другое.

Доступно много забавных игр и анимаций, которые вы можете создать с помощью Scratch, но есть и некоторые ограничения. Ваши программы Scratch могут выглядеть не как «настоящие» игры, в которые вы играете на компьютере, игровой консоли, планшете или смартфоне. Поэтому вполне естественно, что вы захотите научиться писать код

на профессиональном языке программирования. Существует много языков на выбор, но я рекомендую Python или JavaScript. Python – это самый простой язык для изучения (помимо Scratch), но он по-прежнему используется профессиональными разработчиками программного обеспечения. JavaScript сложнее; этот язык, используется для приложений, которые работают в веб-браузере.

Если вы хотите изучить Python, я рекомендую книгу, которую написал сам: «Создаем компьютерные игры на Python». Эта книга будет для вас очередной ступенькой на пути к тому, чтобы стать мастером-программистом!

# ОТВЕТЫ НА ВОПРОСЫ

## ГЛАВА 2

1. Когда спрайт перемещается после запуска блока **Опустить перо**, он тянет за собой линию по ходу движения.
2. Не был запущен блок **Опустить перо** или был запущен блок **Поднять перо**. В обоих случаях прекращается рисование линий.
3. За радужность линий отвечает блок **Изменить цвет пера на**.
4. За толщину линий отвечает бирюзовый блок **Установить размер пера**.
5. Щелкните мышью по кнопке в виде зеленого флажка, удерживая клавишу **Shift**, чтобы включить/отключить турборежим.
6. Щелкните правой кнопкой мыши по спрайту в списке и выберите команду **Дублировать** в контекстном меню.
7. Спрайт будет направлен вправо, если его значение составляет 90 градусов.
8. Следует указать 0 градусов – значение направления вверх.
9. В темно-синей категории **Движение** расположены блоки для направления спрайта вниз и перемещения.
10. Нажмите кнопку **Выбрать фон из библиотеки** рядом с меткой **Новый фон**.
11. Щелкните мышью по бело-голубой кнопке **i** на миниатюре спрайта, чтобы открыть панель информации, и введите новое имя в текстовое поле.

## ГЛАВА 3

1. Изменить размер спрайта позволяет блок **Установить размер %**.
2. За отправку сообщения другому спрайту отвечает блок **Передать**.
3. Клавиши **W**, **A**, **S** и **D** на клавиатуре могут использоваться как альтернатива клавишам  $\uparrow$ ,  $\leftarrow$ ,  $\downarrow$  и  $\rightarrow$ .
4. Перетащите блоки кода на миниатюру спрайта в области спрайтов для дублирования кода.
5. Спрайт будет перемещаться вверх/вниз вместо влево/вправо.
6. На вкладке **Звуки** нажмите кнопку **Выбрать звук из библиотеки** и выберите пункт **Cheer**.
7. Чтобы спрайт двигался быстрее, замените значение 4 в темно-синих блоках на большее число.

## ГЛАВА 4

1. Игра с боковым режимом просмотра позволяет имитировать гравитацию, так как верхняя часть сцены находится в воздухе, а основание — на земле.
2. Переменная может хранить фрагмент текста или число.
3. В режиме **Для всех спрайтов** данную переменную могут использовать и изменять все спрайты, а в режиме **Только для этого спрайта** — только текущий спрайт.
4. Вы можете создать прыгающий спрайт, применив силу тяжести с помощью переменной **у-скорость**.
5. Когда кот находится в воздухе, числовое значение переменной **у-скорость** уменьшается. Когда значение переменной **у-скорость** становится отрицательным числом, кот начинает опускаться.

6. Блок **Перейти** перемещает спрайт в позицию с указанными координатами  $x$  и  $y$  мгновенно, в то время как блок **Плыть** перемещает спрайт в течение указанного периода времени.
7. Поместите зеленый блок **И** внутрь блока **Если** в соответствии с двумя условиями внутри него.

## ГЛАВА 5

1. Спрайт **Мячик** определяет, что пролетел мимо спрайта **Платформа**, когда его положение по оси  $y$  становится меньше -140.
2. Для создания клонов используется блок **Создать клон**.
3. Блок **Когда я начинаю как клон** содержит код, запускающий созданные клоны.
4. Три стиля вращения – **Влево-вправо**, **Кругом** и **Не вращать**.
5. Они скрываются при нажатии кнопки в виде зеленого флага, чтобы их не было видно в начале игры.
6. Блок **Ждать пока** приостанавливает выполнение кода до тех пор, пока не будет выполнено указанное условие.

## ГЛАВА 6

1. Блок **Когда клавиша нажата** срабатывает при каждом нажатии клавиши. Блок **Если клавиша нажата** срабатывает при нажатии и удерживании клавиши.
2. Он отвечает за мгновенное перемещение спрайта в случайную позицию на сцене.
3. Блок **Перейти** перемещает спрайт в позицию с указанными координатами  $x$  и  $y$  мгновенно, а блок **Плыть** – в течение определенного периода времени.

4. Потому что все спрайты в Scratch появляются повернутыми вправо.
5. Центр костюма должен находиться в центре спрайта **Голова**, чтобы он вращался вокруг своего центра.

## ГЛАВА 7

1. Темно-фиолетовая категория **Другие блоки** позволяет создавать пользовательские блоки.
2. В блоке **Определить** указывается код пользовательского блока. Вызывающий блок инициирует запуск этого кода.
3. Параметр **Запуск без обновления экрана** необходим для выполнения кода пользовательского блока в турборежиме.
4. Блок **Перейти в верхний слой** помещает спрайт поверх всех остальных спрайтов на сцене.
5. Переменная может хранить только одно значение, а список — несколько.
6. Облачная переменная разрешает доступ к своему значению всем пользователям сайта Scratch.
7. Если вы не видите флажок **Cloud variable**, значит, ваша учетная запись недавно зарегистрирована и у вас нет доступа к облачным переменным.
8. Облачные переменные могут хранить только цифры, но не текст.
9. Щелкните мышью по бело-голубому значку **i** на миниатюре спрайта, чтобы открыть панель информации, и введите новое имя в текстовое поле.

## ГЛАВА 8

1. Код выхода за пределы сцены проверяет, находится ли спрайт рядом с краем сцены, и при соблюдении

данного условия мгновенно перемещает его на другую сторону рабочей области.

2. Переменная **Я клон** оповещает скрипт, выполняется ли он исходным спрайтом или его клоном.
3. Клоны прекращают создавать дополнительные копии, если значение их переменной **Попадания** превышает 4.
4. Спрайт **Взрыв** содержит несколько различных костюмов, которые он сменяет друг за другом, генерируя покадровую анимацию.

## ГЛАВА 9

1. Отсутствие дублирующегося кода означает, что вам нужно вносить изменения только в одном месте скрипта, если требуется исправить ошибки или добавить новый код.
2. Пользовательский блок хранит значение, которое может быть помещено внутрь блоков кода, например в переменную.
3. Вход пользовательского блока можно использовать только в скрипте пользовательского блока.
4. «Деление по модулю» означает определение остатка от деления целых чисел.
5. «Пол» означает округление числа к меньшему.



# ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

## A

Asteroids, 16, 215

## F

Fruit Ninja, 16, 171, 173

## I

info

команда, 41

## L

Linux, 21

## M

macOS, 21

## R

Raspberry Pi, 21

## S

Scratch

автономный редактор, 22

версии, 22

запуск, 21

зеленый флажок

запуска, 23

интерфейс, 22

красный сигнал

остановки, 23

название проекта, 23

область блоков, 23

область скриптов, 23

область спрайтов, 23

панель меню, 23

помощь, 31

скачивание, 22

сцена, 23

## W

Windows, 21

## A

Арканоид, 16, 119–120, 128–129,  
137, 150

выигрыш, 132

клонирование кирпичиков,  
129

окончание игры, 132

отскакивание мячика от  
кирпичиков, 131

отскакивание мячика от  
платформы, 126

отскакивание мячика от стен,  
124

создание кирпичиков 129

создание платформы 121

улучшение игры 138

эскиз проекта 120

## Б

Баскетбол, 16, 89

броски в кольцо, 104

остановка кольца, 116

перемещение влево и вправо,  
98

прыжки и приземление, 91

режим для двух игроков, 113

создание парящего кольца,  
100

чит-режим, 116

эскиз проекта, 90

Бегущий в лабиринте, 16, 118,  
158

добавление ловушек, 77

ограничение движения кота,  
65

режим для двух игроков, 72

- создание прогуливающегося кота, 58
- создание уровней, 64
- чит-режим, 85
- эскиз проекта, 56
- Библиотека фонов
  - окно, 39, 64
- Библиотека звуков
  - окно, 69, 101, 105, 139, 143
- Библиотека спрайтов
  - окно, 68, 105, 125, 129, 145, 159, 167, 189, 192, 278
- Блок
  - вставить в из, 181
  - выдать случайное, 28, 43, 52, 116, 162, 172, 199
  - добавить к, 181
  - если, то, 61–62, 98, 108, 110, 118, 144, 158, 169, 202, 224, 226, 240, 258, 261, 269, 272, 277–278
  - если, 231
  - если клавиша нажата, 158, 172, 282, 293
  - если расстояние до, 140
  - если то иначе, 202, 272
  - ждать, 27, 80, 136, 150–151, 162, 164
  - задать значение, 94–95, 181, 211
  - заменить элемент в, 181
  - запустить без обновления экрана, 185, 188, 251–253
  - и, 83, 110, 269, 293
  - изменить, 47, 88, 95, 107, 130, 143, 202, 239, 248, 258, 262–263, 291
  - изменить у на, 60, 88, 107, 143, 219, 262–263
  - изменить x на, 60, 130, 255
  - изменить на, 95, 181
  - изменить размер на, 143, 147, 202
  - изменить цвет пера на, 47, 291
  - изменить эффект на, 142
  - касается цвета?, 66, 82–83, 164, 169, 191, 202
  - клавиша нажата?, 75, 82, 85–86, 172
  - когда клавиша нажата, 158–159, 172, 224, 226, 240, 293
  - когда щелкнут по зеленому флагу, 29, 46, 162, 268, 269–270
  - когда я начинаю как клон, 80, 128, 131, 162, 293
  - когда я получу, 101, 163–164, 178
  - определить, 187–188, 251, 253–254, 263, 268, 294
  - опустить перо, 47
  - остаток от деления на, 270–271
  - очистить, 47, 253
  - передать, 69, 200
  - перейти в, 46, 50, 75, 118, 172, 196, 293
  - перейти в x: у:, 46
  - перейти в верхний слой, 67, 208, 213, 294
  - поднять перо, 47, 52, 291
  - повернуть в направлении, 43, 127
  - повернуть по часовой стрелке на, 143, 147
  - повторить, 131, 141–142, 149
  - повторять пока не, 107, 109, 188, 202, 248, 251, 255, 263
  - показаться, 18, 77, 127, 129, 147, 278
  - пользовательский, 253, 260, 269
  - правильный костюм, 270, 272, 283
  - сменить костюм на, 65, 265, 270–272

создание пользовательского,  
185, 251, 253, 270  
создать клон себя самого, 80,  
128  
спрятаться, 104, 129, 241  
стоп, 134, 147–149  
удалить из, 181  
установить размер %, 67, 104,  
199, 292  
установить x в, 130  
установить размер пера, 47,  
291  
установить цвет для пера, 187  
эффект, 104, 141, 279–280  
добавление, 27  
удаление, 28  
булево  
значение, 260

## В

Внешность, 23, 44, 65, 67, 265  
Всегда  
цикл, 28–29, 37, 42, 58, 61, 67,  
80–81, 84, 95, 98–99, 104,  
155, 158, 162, 171,  
187–188, 222, 238, 248,  
257–258, 260, 287  
Выбрать  
инструмент, 38, 59, 91,  
123–125, 145, 159, 167,  
189, 192–193, 218, 223,  
275, 278, 280, 291

## Г

Гравитация, 201, 245–247  
Градусы, 43  
Графический редактор, 24

## Д

Данные 24, 92–94, 97, 105–106,  
109, 115–116, 129, 157,  
178, 180, 183–184, 187,  
210, 218, 223, 227, 238,  
247, 278, 281

Две радужные линии, 51  
Движение, 23, 42, 44, 46, 60–61,  
65, 291  
Демо, 36  
Демосцена, 35  
Другие блоки, 24, 185–186, 251,  
253, 270, 294  
Дублировать  
команда, 45, 51, 62, 72, 74, 78,  
84, 113, 117, 131, 161,  
194, 280, 284, 291

## З

Звук, 24, 101, 229  
выбор из библиотеки 69, 101,  
105, 139, 143, 147–148,  
197, 223, 229, 235, 292  
Зме-е-ейка, 16, 278  
создание головы змеи, 155  
создание тела змеи, 160  
создание яблок, 159  
чит-режим, 168  
эскиз проекта, 155  
Значение  
логическое, 261  
указатель мышки, 46

Загрузить  
с компьютера  
команда, 57, 63, 65, 67, 250  
спрайт из файла  
команда, 64, 218, 226, 235  
фон из файла  
команда, 156, 274

Заливка  
инструмент, 73, 113–114,  
138–139, 161, 176, 204,  
207

Запуск без обновления экрана  
параметр, 213, 287, 294  
режим, 185, 188, 251–253

## И

Игра  
управление мышью, 97, 150,  
216–217, 223, 226, 249

управление с клавиатуры, 75,  
77, 81, 88, 216, 220, 222,  
292

## К

Код

удаление, 28

Костюм

создание нового, 25, 193, 264

Костюмы

вкладка, 24–25, 41, 64, 73–74,  
82, 99, 113, 176, 189, 192,  
195, 235, 264, 267, 271,  
274, 276, 280

Кисть

инструмент, 41, 246

Координаты

изменение, 60

Координаты x и y, 58

Конвертировать в растровую  
графику

команда, 196

## Л

Ластик

инструмент, 275

Линия

инструмент, 77, 176, 193, 203,  
246

## Н

Направление, 43

Нарисовать новый фон

кнопка, 177

Новый

блок

окно, 50, 185–186  
команда, 57

список

окно, 180, 183

Новая переменная

окно, 92, 218

Новое сообщение

команда, 69, 101, 126, 147,  
178, 200, 235

## О

Окно подсказок, 31

Операторы, 24, 42, 187

Отменить

кнопка, 26, 177, 275

Отразить слева направо

команда, 113–114

## П

Падение

имитация, 92, 117, 245–246,  
248, 287

Панель информации, 41–42, 79,

91, 122, 129, 135, 161,  
179, 192, 203, 223, 233,  
246, 291, 294

Переменная

для всех спрайтов, 92–93, 105,  
115, 118, 129, 157, 169,  
219, 238, 279, 285, 292

облачная, 210, 213, 294

создать, 92

только для этого спрайта,  
92–94, 105, 109, 116, 118,  
178, 183–184, 197, 218,  
224, 227, 247, 254, 257,  
268, 281, 292

удаление, 94

Перо, 24, 46, 183–184, 188

Пипетка

инструмент, 26

Платформер, 16, 117, 242, 245,  
250, 274

анимация ходьбы, 266

крабы и яблоки, 278

крутизна склонов, 252

прыжки разной высоты, 256

создание уровня, 274

хитбокс для кота, 263

хитбокс краба, 278

эскиз проекта, 244

Поделиться

команда, 30

- Пользовательский блок
  - параметры, 261
  - редактирование, 261
- Правка
  - меню, 28
- Приземление
  - имитация, 245–247
- Программа
  - демонстрация, 30
  - запуск, 29
  - публикация, 30
- Проект
  - сохранение, 23
  - эскиз, 36, 56, 90, 120, 155, 175, 216, 244
- Прямоугольник
  - инструмент, 103, 122

**Р**

- Радужные линии, 15, 35, 38, 44, 49, 54
  - создание движущихся точек, 40
  - создание фона, 38
  - эскиз проекта, 36
- Радужные треугольники, 50
- Редактор
  - графический, 24
- Режим для двух игроков, 72, 113

**С**

- Сенсоры, 24, 61
- Скрипты, 24, 42, 69, 92, 105, 139, 162, 178, 182–183, 223, 251
- События, 24, 42, 44, 46, 61, 65
- Создать переменную
  - команда, 92, 105, 109, 115–116, 129, 157, 178, 184, 210
- Создавай
  - раздел, 22–23, 38
- Сообщение
  - передача, 75, 132, 191, 202, 205, 208

- создание, 200
- Сохранить локальный файл
  - команда, 23, 26
- Сохранить объект как
  - команда, 57
- Спрайт, 23
  - создание нового, 24, 40, 77, 100, 102, 122, 133, 135, 156, 160, 179, 203, 206, 232, 246, 285
- Список
  - для всех спрайтов, 180
  - создание, 180–181, 183–184
- Стиль вращения
  - параметр, 123
- Создать переменную
  - команда, 218, 223, 227, 238, 247, 279, 281
- Сцена, 139, 176, 178
  - выбор из библиотеки, 68, 105, 124, 129
  - выход за пределы, 217, 220, 225, 250
  - дублирование, 45, 72, 74
  - настройка, 38
  - очистка, 38
  - переименование, 41

**Т**

- Турборежим
  - запуск, 49
- Текст
  - ввод русских букв, 134, 190, 233
  - инструмент, 132–135, 150, 189–190, 232–233

**У**

- Увеличить
  - инструмент, 157
- Удалить переменную
  - команда, 94
- Уменьшить
  - инструмент, 157

Уничтожитель астероидов, 16,  
215, 220, 232, 239, 250  
ограничение боезапаса, 237  
прицеливание и стрельба, 223  
создание астероидов, 226  
создание космолета, 218  
счет, 232  
таймер, 232  
чит-режим, 239  
эскиз проекта, 216  
эффект взрыва, 234  
Управление, 24, 42, 44, 46, 61,  
171, 222  
Установить центр  
кнопка, 160

## Ф

Файл  
меню, 17, 23, 57, 63, 65, 67,  
250

Фон  
выбор из библиотеки, 64  
создание, 156, 274

Фруктовый ниндзя, 16, 173,  
175–177, 179, 183–184,  
189, 193, 212  
заставка, 175  
игровой счет, 209  
окончание игры, 206  
создание бомб, 192  
создание фруктов, 192  
уровень здоровья, 203  
чит-режим, 211  
эскиз проекта, 175

## Х

Хитбокс, 102–104, 107–110, 117,  
264–265, 267, 274–276,  
281

## Ц

Цикл  
если, 164  
повторить, 198, 208  
повторять пока не, 255

## Ч

Чит-режим 85, 116, 168, 170,  
211, 239

## Ш

Ширина линии  
ползунковый регулятор, 100,  
193, 246

## Э

Эллипс  
инструмент, 100

Все права защищены. Книга или любая ее часть не может быть скопирована, воспроизведена в электронной или механической форме, в виде фотокопии, записи в память ЭВМ, репродукции или каким-либо иным способом, а также использована в любой информационной системе без получения разрешения от издателя. Копирование, воспроизведение и иное использование книги или ее части без согласия издателя является незаконным и влечет уголовную, административную и гражданскую ответственность.

Пособие для развивающего обучения

ПРОГРАММИРОВАНИЕ ДЛЯ ДЕТЕЙ

Свейгарт Эл

**ПРОГРАММИРОВАНИЕ ДЛЯ ДЕТЕЙ**  
**Делай игры и учи язык Scratch!**

Директор редакции *Е. Капьев*  
Ответственный редактор *Е. Истомина*  
Художественный редактор *А. Гусев*

ООО «Издательство «Эксмо»  
123308, Москва, ул. Зорге, д. 1. Тел. 8 (495) 411-68-86.  
Home page: [www.eksmo.ru](http://www.eksmo.ru) E-mail: [info@eksmo.ru](mailto:info@eksmo.ru)

Өндіруші: «ЭКМО» АҚБ Баспасы, 123308, Мәскеу, Ресей, Зорге көшесі, 1 үй.  
Тел. 8 (495) 411-68-86.  
Home page: [www.eksmo.ru](http://www.eksmo.ru) E-mail: [info@eksmo.ru](mailto:info@eksmo.ru)  
Тауар белгісі: «Эксмо»

Қазақстан Республикасында дистрибьютор және өнім бойынша  
арыз-талаптарды қабылдаушының  
өкілі «РДЦ-Алматы» ЖШС, Алматы қ., Домбровский көш., 3«а», литер Б, офис 1.  
Тел.: 8(727) 2 51 59 89,90,91,92, факс: 8 (727) 251 58 12 вн. 107; E-mail: [RDC-Almaty@eksmo.kz](mailto:RDC-Almaty@eksmo.kz)  
Өнімнің жарамдылық мерзімі шектелмеген.  
Сертификация туралы ақпарат сайтта: [www.eksmo.ru/certification](http://www.eksmo.ru/certification)

Сведения о подтверждении соответствия издания согласно законодательству РФ  
о техническом регулировании можно получить по адресу: <http://eksmo.ru/certification/>

Өндірген мемлекет: Ресей  
Сертификация қарастырылмаған

Подписано в печать 17.08.2017. Формат 70х100<sup>1</sup>/<sub>16</sub>.  
Печать офсетная. Усл. печ. л. 24,63.  
Тираж экз. Заказ

ISBN 978-5-699-98943-0



9 785699 989430 >



В электронном виде книги издательства вы можете  
купить на [www.litres.ru](http://www.litres.ru)

ЛитРес:

один клик до книг



# КОГДА ВЫ ДАРИТЕ КНИГУ, ВЫ ДАРИТЕ ЦЕЛЫЙ МИР

## ХОТИТЕ ЗНАТЬ БОЛЬШЕ?

**Заходите на сайт:**

<https://eksmo.ru/b2b/>

**Звоните по телефону:**

+7 495 411-68-59, доб. 2261



ВАШ ЛОГОТИП  
НА ОБЛОЖКЕ

ВАШ ЛОГОТИП НА КОРЕШКЕ

ОБРАЩЕНИЕ  
К КЛИЕНТАМ  
НА ОБЛОЖКЕ



# СОЗДАВАТЬ ИГРЫ ТАК ЖЕ ВЕСЕЛО, КАК ИГРАТЬ В НИХ!

Scratch – самая популярная в мире платформа, разъясняющая миллионам новичков принципы программирования. Автор этой книги предлагает программировать на Scratch, создавая собственными руками классические игры, такие как «Змейка», «Фруктовый ниндзя», «Арканоид».

## Благодаря этой книге вы сможете:

- Сделать «Змейку». Собирайте яблоки и остерегайтесь собственного хвоста!
- Резать фрукты. Клон популярной игры «Фруктовый ниндзя»
- Прыгать по платформам, как в игре «Супер Марио»
- Использовать читы и рисовать одежду для персонажей!

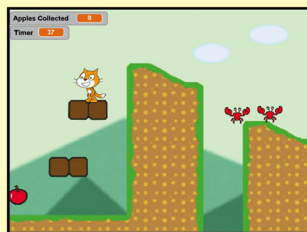
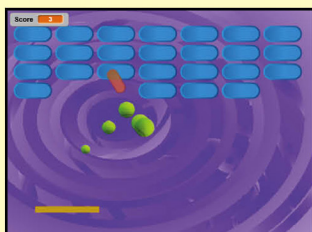
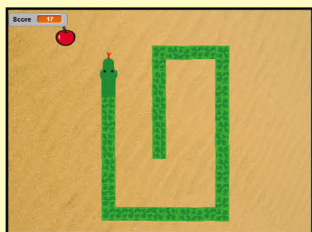
**ПРОГРАММИРОВАНИЕ  
ЕЩЕ НИКОГДА  
НЕ БЫЛО ТАКИМ ВЕСЕЛЫМ!**

## Об авторе

Эл Свейгарт – разработчик и преподаватель программирования для детей и взрослых, а также автор нескольких бестселлеров, посвященных изучению языка Python для начинающих.

В школе мы рекомендуем эту книгу даже взрослым. Цветные пазлы Scratch делают такие концепции, как переменные, циклы и массивы, простыми для понимания. Заигравшиеся в Scratch рискуют стать программистами, даже если просто хотели скрасить пару вечеров, помогая ребенку.

Вадим Резвый,  
сооснователь Moscow Coding School



www.nostarch.com

ISBN 978-5-699-98943-0



9 785699 989430 >

