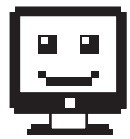
SCRATCH 3

ИЗУЧАЙТЕ ЯЗЫК ПРОГРАММИРОВАНИЯ, ДЕЛАЯ КРУТЫЕ ИГРЫ!

ЭЛ СВЕЙГАРТ



ПРОГРАММИРОВАНИЕ
ДЛЯ ДЕТЕЙ

AL SWEIGART

SCRATCH 3 PROGRAMMING PLAYGROUND

LEARN TO PROGRAM BY MAKING COOL GAMES

ЭЛ СВЕЙГАРТ

SCRATCH 3

ИЗУЧАЙТЕ ЯЗЫК ПРОГРАММИРОВАНИЯ, ДЕЛАЯ КРУТЫЕ ИГРЫ!

БОМБОРА
ИЗДАТЕЛЬСТВО
Москва 2023

УДК 004.43-053.2
ББК 32.973.26-018.1
С24

SCRATCH 3 PROGRAMMING PLAYGROUND
Al Sweigart

Свейгарт, Эл.

С24 Scratch 3. Изучайте язык программирования, делая крутые игры! / Эл Свейгарт ; [перевод с английского М. А. Райтман]. — Москва : Эксмо, 2023. — 224 с. — (Программирование для детей).

ISBN 978-5-04-122009-9

Scratch 3 — лучшая среда программирования для новичков. Она отличается удобным и интуитивно понятным интерфейсом, простотой в использовании и возможностью создавать адаптивные игры для различных устройств. Научиться программировать в ней несложно, а сам процесс точно не будет скучным и принесет море удовольствия. Благодаря этой книге дети научатся создавать несколько простых игр, которые станут отличным началом их пути в программировании.

УДК 004.43-053.2
ББК 32.973.26-018.1

ISBN 978-5-04-122009-9

ОТЗЫВЫ О КНИГЕ

«Впечатляющее руководство по мастерскому программированию в Scratch и созданию действительно увлекательных игр».

— Kirkus Reviews

«Мой сын научился сам создавать игры. И попутно он приобрел отличные навыки программирования. Еще мне понравилось, что книга привила ему терпение и настойчивость, не говоря уже о постановке целей. Это отличное руководство по программированию в Scratch!»

— Old Schoolhouse Magazine

«Книга понятная, полна юмора, каламбуров и прекрасных объяснений, как все устроено».

— I Programmer

«Я впечатлен, как много инструментов программирования Scratch осваивает читатель по мере чтения книги, и я думаю, что преподаватели и родители оценят этот справочник на 5 баллов».

— Джим Келли, GeekDad

«Если вы ищете что-нибудь новенькое для своих детей, фанатов Minecraft, и еще не пробовали Scratch, эта книга — отличное руководство, способное научить детей программировать и создавать классные игры».

— Tech Savvy Mama

ОГЛАВЛЕНИЕ

ОБ АВТОРЕ

Эл Свейгарт — разработчик программного обеспечения, автор технических книг и человек, у которого всегда при себе полотенце[1]. 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**.

О ТЕХНИЧЕСКОМ РЕЦЕНЗЕНТЕ

Оливия Родригес — преподаватель математики, информатики и естествознания, а также руководитель отдела вычислительной техники в частной начальной школе в Великобритании. Оливия получила сертификаты British Computer Society, Google Certified Educator, Microsoft Innovative Educator Expert и IDEA Award Teacher Ambassador. У нее врожденная страсть к обучению детей программировать. Оливия замужем за Гэри, и вместе они воспитывают годовалую дочь Женевьеву.

---

[1] 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). — *Прим. ред.*

БЛАГОДАРНОСТИ

Неправильно, если на обложке будет только мое имя. Эта книга не увидела бы свет без труда многих людей. Я хотел бы поблагодарить издателя Билла Поллока; научного редактора Фрэнсиса Со; выпускающего редактора Рэйчел Монаган; технического рецензента Оливию Родригес и корректора Ким Уимпсетт. Еще я хотел бы поблагодарить всех, кто работал над первым изданием этой книги: научных редакторов Лорел Чун и Тайлера Ортмана; технического рецензента Мартина Тан; корректора Энн Мари Уокер и всех сотрудников издательства No Starch Press.

Спасибо за разработку Scratch организации Lifelong Kindergarten MIT Media Lab, в частности влиятельным гениям Митчелу Резнику, Сеймуру Пейперту, Марвину Мински и Жану Пиаже. Обучая молодое поколение, не забывайте и о себе.

Особая благодарность Музею искусства и цифровых развлечений в Окленде, Калифорния. Участвовать в работе музея видеоигр оказалось крайне весело и полезно, как и обучать созданию Scratch-игр на субботних мастер-классах MADE. Если бы Алекс Хэнди, Майк Павоне и Уильям Морган не задумали создать мастер-классы по Scratch, мне бы никогда не пришла в голову идея этой книги.

ВВЕДЕНИЕ

Конечно, весело играть в видеоигры, 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 — умение 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 выходные 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/** и **inventwithscratch.com**. Также я рекомендую книгу Макса Уэйнрайта «25 Scratch 3 Games» (No Starch Press, 2019).

И помните: программирование — это практический навык, как карате или умение играть на гитаре. Конечно, вы 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; 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. Продвинутый арканоид.** Глава описывает методы создания игры, в которой простой кирпичик превращается

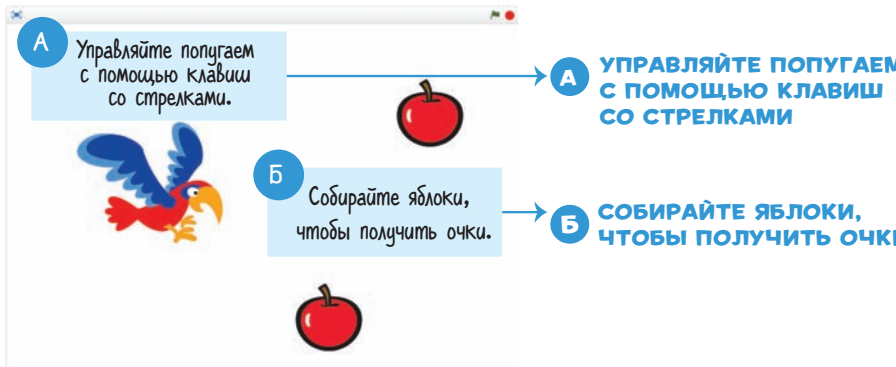
в гладкий после попадания по нему. Игру можно сделать красивее при помощи анимации, звуковых эффектов и многого другого.

- **Глава 6. Уничтожитель астероидов… в космосе!** Это клон классического космического шутера Asteroids. Для управления космолетом вы будете использовать мышь и клавиатуру.
- **Глава 7. 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.

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 «А», «Б», «В» в книге соответствуют этим функциям в эскизе.

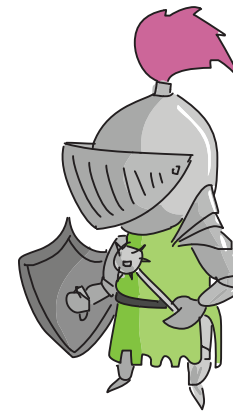


Дробление большой задачи на несколько мелких поможет вам думать организованно и сделает большую задачу не такой пугающей, какой она может показаться на первый взгляд. После того как будет создана и запущена простая версия игры, мы добавим новые функции, чит-коды и т. д. Для создания собственной игры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

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 на свой компьютер**.

ВЕБ-РЕСУРСЫ



Хотя среда Scratch и включает множество собственных ресурсов, вам потребуется и несколько дополнительных файлов, чтобы реализовать описанные в этой книге проекты. Эти файлы находятся в архиве, который можно загрузить по ссылке **http://addons.eksmo.ru/it/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 проектов 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 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

На сегодняшний день Scratch — одна 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 *скретчерами*, создают программы, совмещая в редакторе Scratch блоки кода. Хотя Scratch разрабатывался для детей от 8 до 16 лет, он давно перешагнул свои границы: им пользуются люди всех возрастов, в том числе и маленькие дети 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 ней 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 программирования, 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.

Вы найдете ответы на часто задаваемые вопросы по адресу: **https://scratch.mit.edu/info/faq/**.

ЗАПУСК SCRATCH

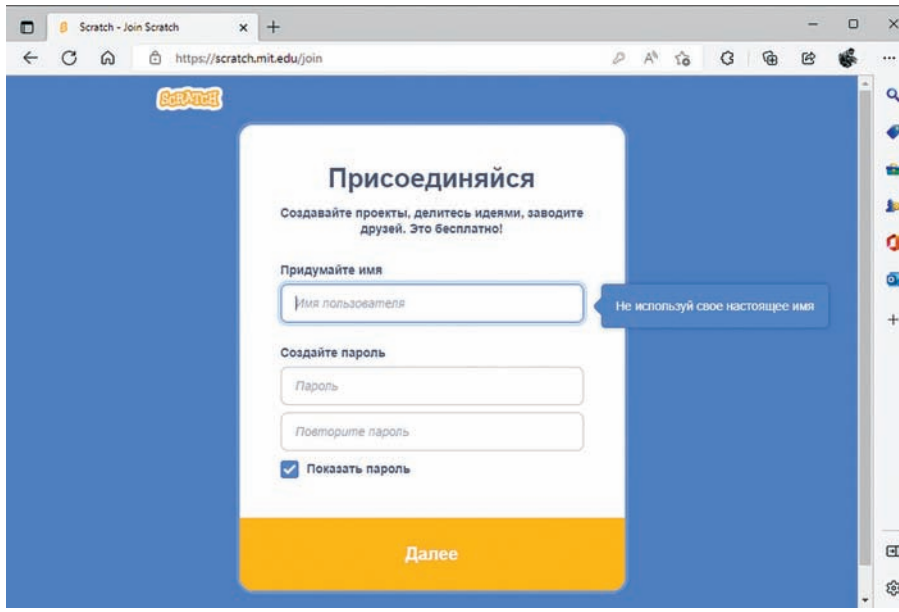
Чтобы начать знакомство со Scratch, откройте браузер и перейдите по адресу: **scratch.mit.edu**. Не имеет значения, какой операционной системой вы пользуетесь — Windows, macOS, Linux, Android или iOS; вы сможете запустить Scratch на своем планшете, смартфоне, ноутбуке или компьютере. И даже на Raspberry 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/**. Храните свои персональные данные в безопасности; не используйте свое настоящее имя и никому постороннему не сообщайте свой пароль, только родителям или учителям. Не сообщайте свой пароль никому, даже если человек выдает себя за сотрудника Scratch или MIT (настоящие сотрудники никогда его не спрашивают). Не используйте один и тот же пароль для защиты нескольких своих учетных записей в Интернете, потому что, если

пароль одной учетной записи будет взломан, хакер сможет получить доступ и к другим вашим аккаунтам.



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

АВТОНОМНЫЙ РЕДАКТОР

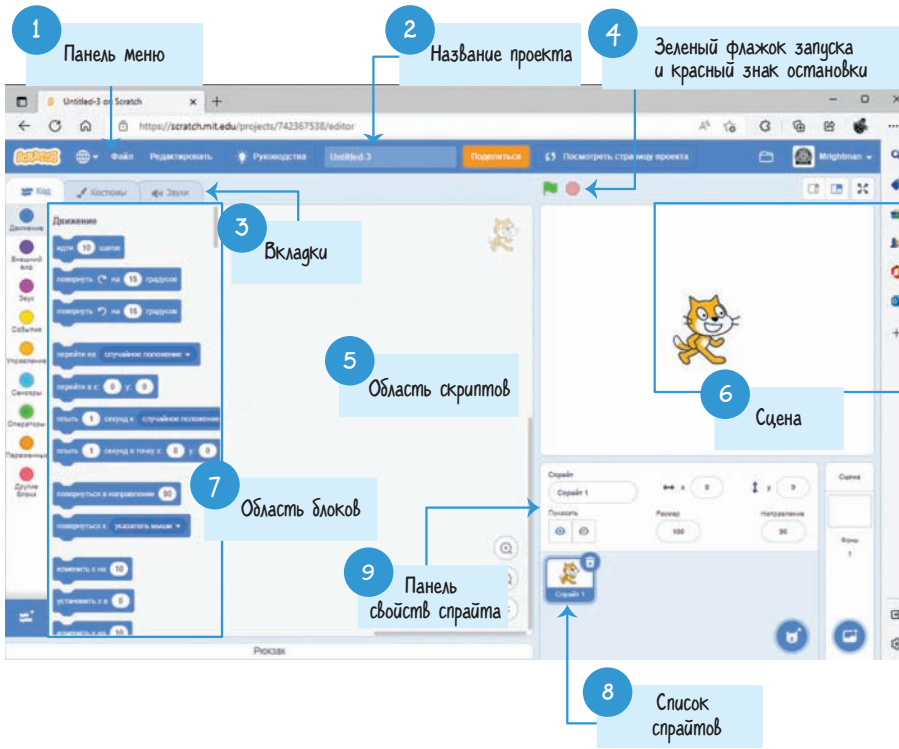
Автономный редактор позволяет программировать без подключения к Интернету. Если у вас нет доступа к Сети или если соединение нестабильно, вы можете установить автономный редактор на свой компьютер вместо того, чтобы использовать сайт Scratch. Единственное отличие состоит в том, что программы будут сохранены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

Автономный редактор Scratch доступен по адресу:
https://scratch.mit.edu/download/.

Примечание. Вам может попасться ранняя версия редактора Scratch 1.4 или 2.0. Не используйте их; они устарели и не имеют новых функций, которые есть в версии Scratch 3. Если вы используете веб-версию Scratch в браузере, это версия 3. Если вы скачиваете автономный редактор Scratch, убедитесь, что загружаете именно версию Scratch 3.

РЕДАКТОР SCRATCH И СПРАЙТЫ

В редакторе Scratch вы совмещаете блоки кода, чтобы создать игру, анимацию или иллюстрацию. Редактор открывается 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 ❻, а их поведением управляют соответствующие

блоки кода. Для всех новых проектов редактор автоматически запускается со спрайтом **Кот**, но вы можете удалить его и добавить собственные спрайты ❾. Программирование спрайта осуществляется путем добавления блоков кода в область скриптов ❺ в правой части экрана. Несколько объединенных блоков кода в Scratch называются *скриптом*.

Текстовое поле в верхней части редактора содержит название проекта ❷. Проекту стоит дать имя, описывающее его суть. Помните, что время от времени нужно сохранять проект. Для этого выберите команду **Файл ⇨ Сохранить сейчас** на панели меню ❶. Так вы не потеряете ваш проект, если браузер завершит работу с ошибкой.

В центре находится область блоков ❼, из которой вы будете брать блоки кода. В верхней части области блоков вы увидите 9 категорий блоков кода: **Движение**, **Внешний вид**, **Звук**, **События**, **Управление**, **Сенсоры**, **Операторы, Переменные** и **Другие блоки**. Каждый блок кода относится к одной из групп и оформлен цветом этой группы. Например, блок **Сказать** относится к фиолетовой категории **Внешний вид**. Запас доступных блоков кода безграничен; просто выбирайте их в области блоков и перетаскивайте в область скриптов.

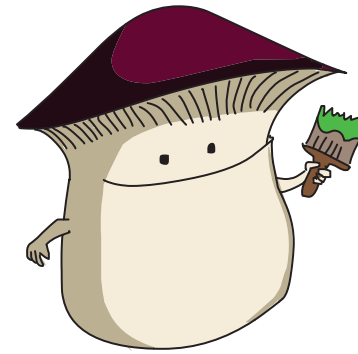
Каждый объект-спрайт имеет свои собственные скрипты. При выборе спрайта в списке спрайтов ❽ на панели свойств спрайта ❾ будут отображаться характеристики выбранного спрайта. Выберите вкладку **Скрипты** ❸ для отображения области скриптов. При выборе вкладки **Костюмы** вместо области скриптов появляется 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 ❹.

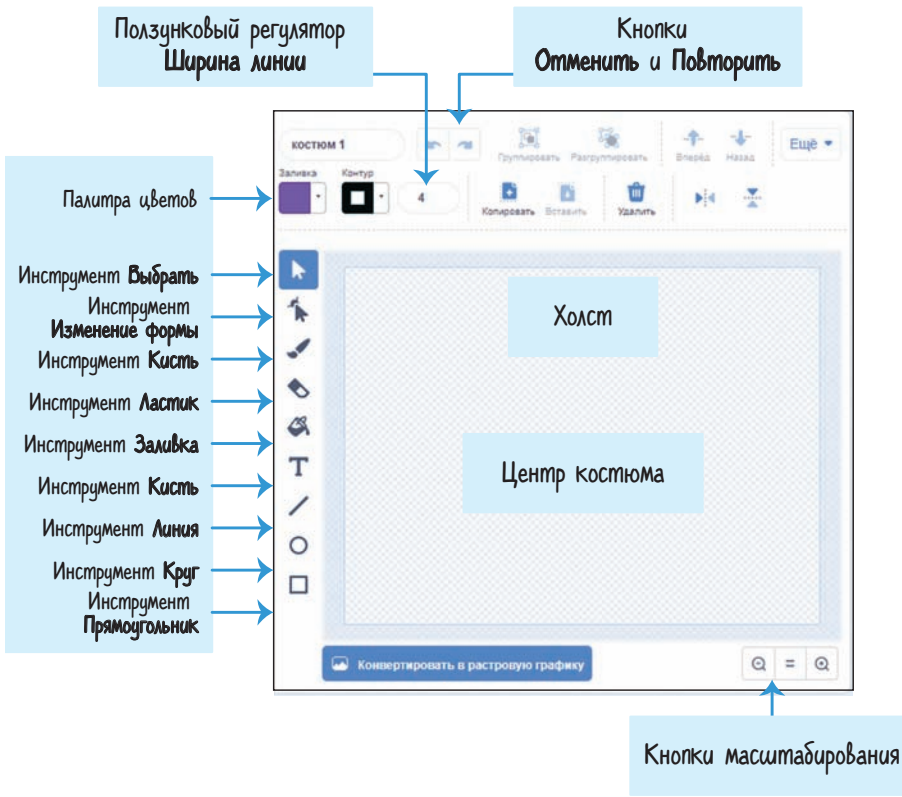
ГРАФИЧЕСКИЙ РЕДАКТОР

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, выберите пункт **Нарисовать**, который прячется в кнопке **Выбрать спрайт** в списке спрайтов. Вы можете изменить внешний вид спрайтов, переключаясь между множеством разных костюмов. Чтобы создать новый костюм для спрайта, выберите вкладку **Костюмы**, а затем выберите пункт **Рисовать**, который прячется в кнопке **Выбрать костюм**.



Графический редактор выглядит следующим образом:



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**, которые помогут исправить ошибки.

Экспериментируйте с графическим редактором, нажимая кнопки инструментов рисования и перемещая мышь по холсту, чтобы увидеть, как они работают. Изменяйте цвет и ширину инструментов рисования с помощью палитры цветов и ползунка, регулирующего ширину линии. С помощью инструмента **Кисть** можно рисовать прямо на костюме. 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 в списке слева и выберите пункт **Экспорт** в контекстном меню.

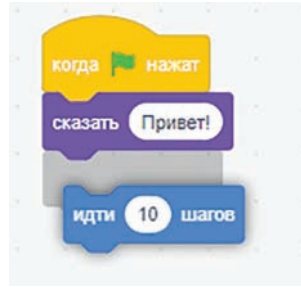
РАБОТА С БЛОКАМИ КОДА

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.

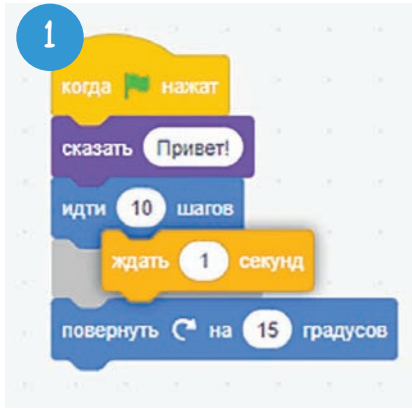
Добавление блоков

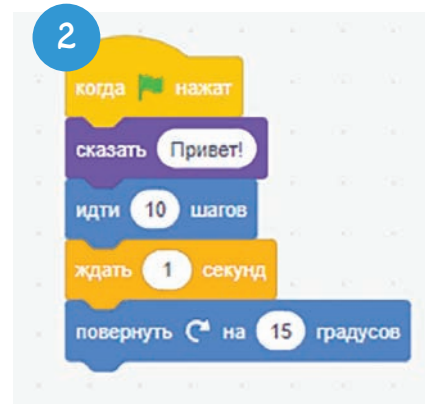
Чтобы создать новый блок кода, перетащите его из области блоков, расположенной по центру, в область скриптов. Блоки кода,

имеющие выемку на верхней стороне и выступ снизу, называются *блоками стека*. Чтобы соединить один блок стека с другим, перетащите первый блок ближе к верхней 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 серый фон: здесь будет помещен блок стека. Рисунки ниже показывают перемещение блока **Ждать 1 секунду** в середину скрипта.

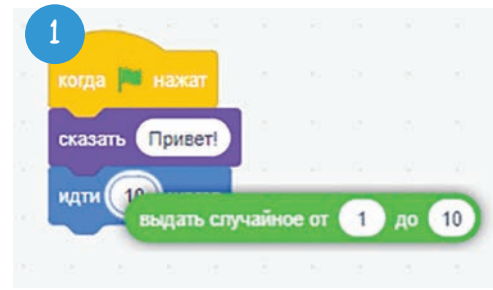


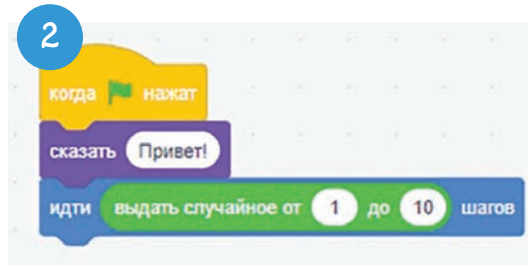


Вы можете изменить значение белого поля внутри блока, щелкнув мышью по этому полю и введя новое значение.

Скругленные блоки называются *блоками ссылок*. Они вставляются внутрь белых полей. Например, на следующих рисунках зеленый блок ссылки **Выдать случайное от 1 до 10** поместился внутри белого поля. Когда *левый край* блока ссылки оказывается

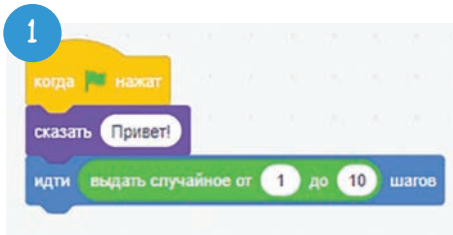
на белом поле, вокруг этого поля появляется белый контур. Если левый край блока ссылки находится не над белым полем, белый контур не будет появляться, и блок ссылки нельзя будет поместить внутрь.

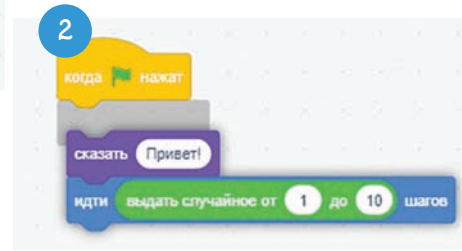




Удаление блоков

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.

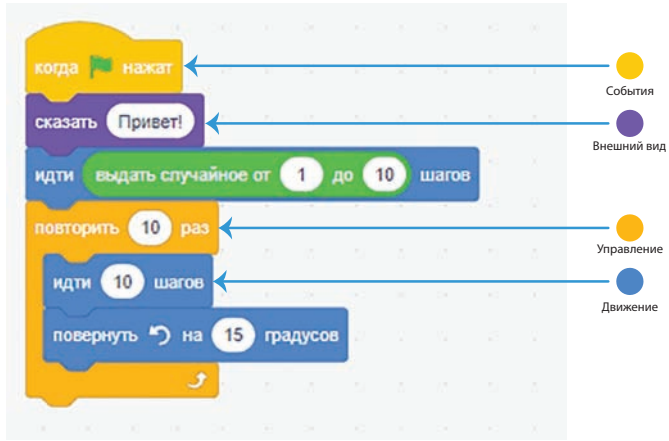


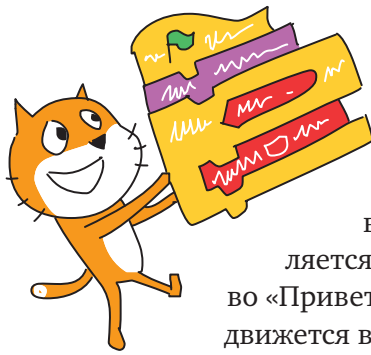


Существует и другой способ удаления блоков: щелкните правой кнопкой мыши по блоку и выберите команду **Удалить блок** в контекстном меню. Если вы случайно удалили нужные блоки, их можно восстановить, выбрав команду **Редактировать ⇨ Восстановить** в строке меню.

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 примере над спрайтом появляется всплывающее окно и отображается слово «Привет!». В цикле **Повторять всегда** спрайт движется вперед на 10 шагов, а затем поворачивается против часовой стрелки на 15 градусов. Когда программа доходит до выполнения последнего блока, цикл возвращает назад, к верхнему блоку в цикле. Все блоки, находящиеся в блоке **Повторять всегда**, будут работать зацикленно. Выполнение программы прекращается только после нажатия кнопки, которая выглядит как красный знак остановки.

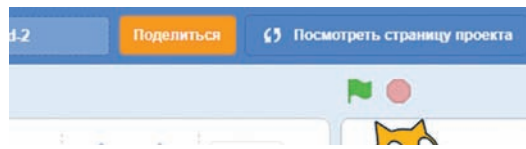
Кроме того, можно запустить скрипт или блок, дважды щелкнув по нему мышью. Но все же лучше запускать программы кнопкой в виде зеленого флажка.

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 смогут 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 **https://inventwithscratch.com/studio/**. В разделе **Добавить проекты** вставьте URL-адрес в текстовое поле и нажмите кнопку **Добавить по 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,

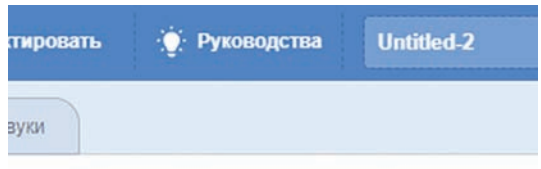
как вы. Десятки миллионов человек поделились своими программами на сайте Scratch, так что не расстраивайтесь, если у вашей программы мало просмотров: ее может быть сложно найти с таким количеством доступных программ.

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.

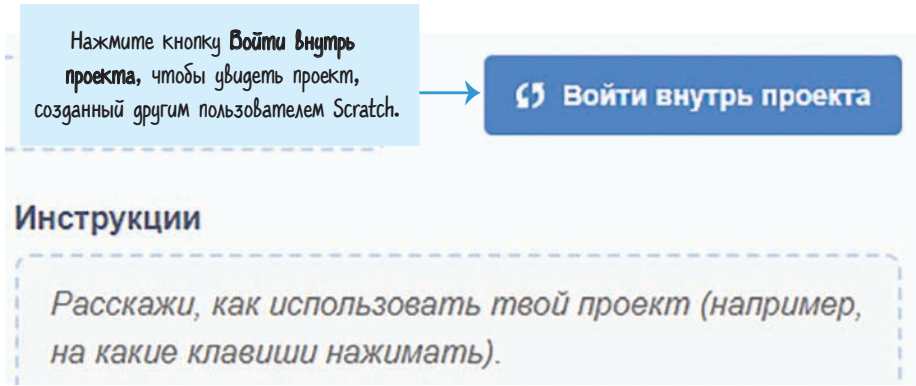
Руководства

В верхней части редактора Scratch находится кнопка меню **Руководства**. Нажмите эту кнопку, чтобы открыть окно **Выбрать руководство**. Там есть ссылки на несколько видеоуроков по Scratch.



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 **Войти внутрь проекта**, как показано здесь.



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 собственные версии.

Вам все еще нужна помощь и вы хотите поговорить с другим пользователем? Щелкните мышью по ссылке **Форумы** в нижней части сайта **https://scratch.mit.edu/**, чтобы посетить форумы для обсуждения.

ЗАКЛЮЧЕНИЕ

Редактор Scratch — это инструмент для творчества с большими возможностями. На сайте Scratch вы увидите самые разные проекты: от игр, мультфильмов и 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 руководств в редакторе и форумов на сайте Scratch.

Давайте создадим свою первую программу!

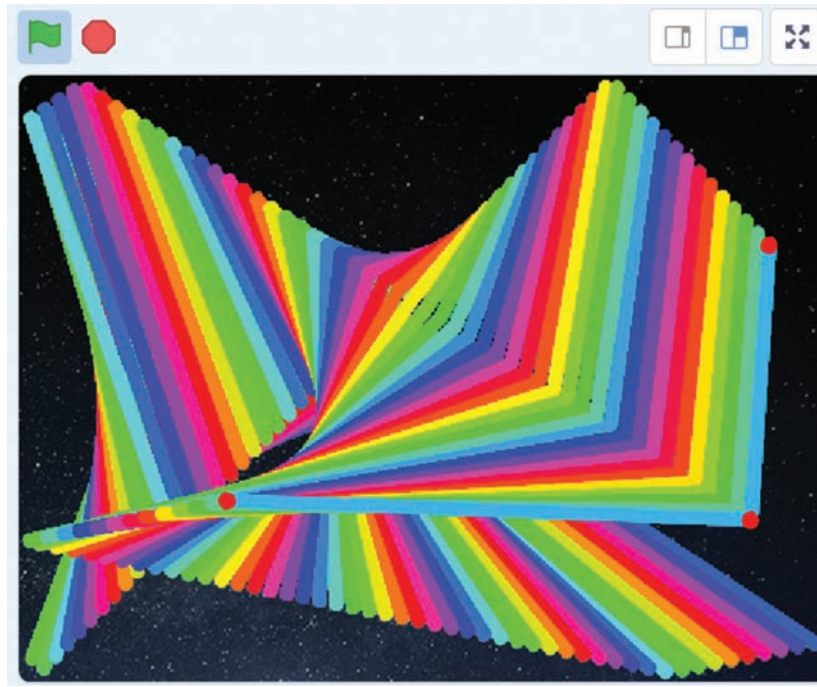
2

РАДУЖНЫЕ ЛИНИИ В КОСМОСЕ

В этой главе вы создадите классную анимацию — летящую V-образную радугу, оставляющую за собой яркий след. Эта программа была вдохновлена *демосценой* — 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 **https://scratch.mit.edu/projects/741002743**, чтобы посмотреть готовую анимацию.



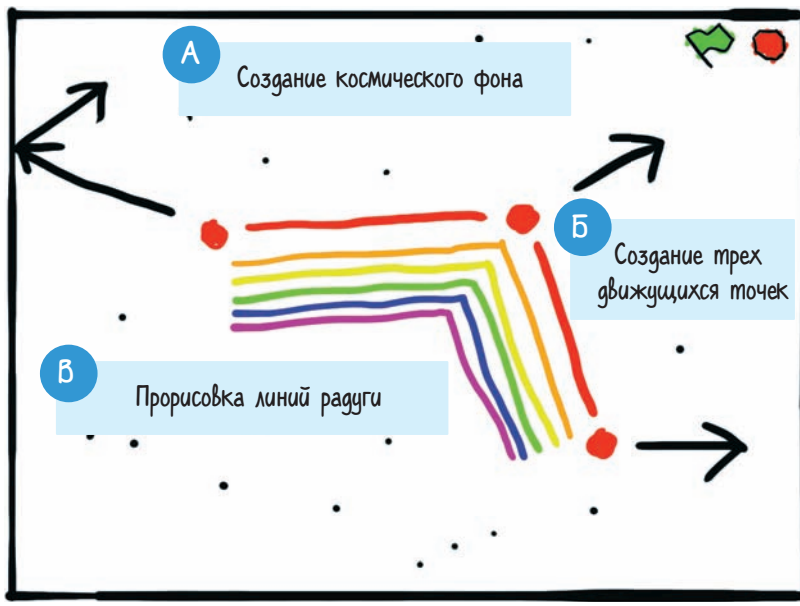
Так же как участники демосцены, вы можете создавать красивые программы. Давайте создадим наше собственное графическое демо в Scratch!

ЭСКИЗ ПРОЕКТА

Первый шаг превращения идеи в программу — представление о том, как она должна выглядеть. Проектирование программы поможет выяснить, какие спрайты вам нужны и как они будут себя вести. Я рекомендую рисовать 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. 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 сделать эскиз красивым. 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.



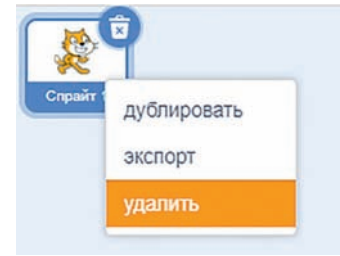
После того как вы закончили эскиз своей программы, можно начинать программировать! Перейдите на сайт Scratch по ссылке **https://scratch.mit.edu/**, зарегистрируйтесь на нем для получения учетной записи и авторизуйтесь в системе (наличие аккаунта позволяет сохранять созданные вами программы). После того 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 в космосе**. Давайте начнем с создания части А этой программы.

А. СОЗДАНИЕ КОСМИЧЕСКОГО ФОНА

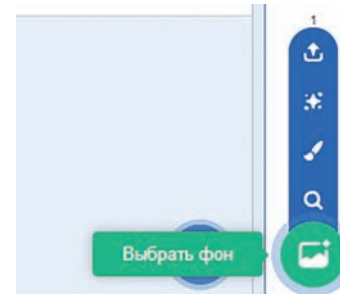
Во-первых, давайте уберем спрайты, которые мы не будем использовать, и создадим фон.

1. Очистка и настройка сцены

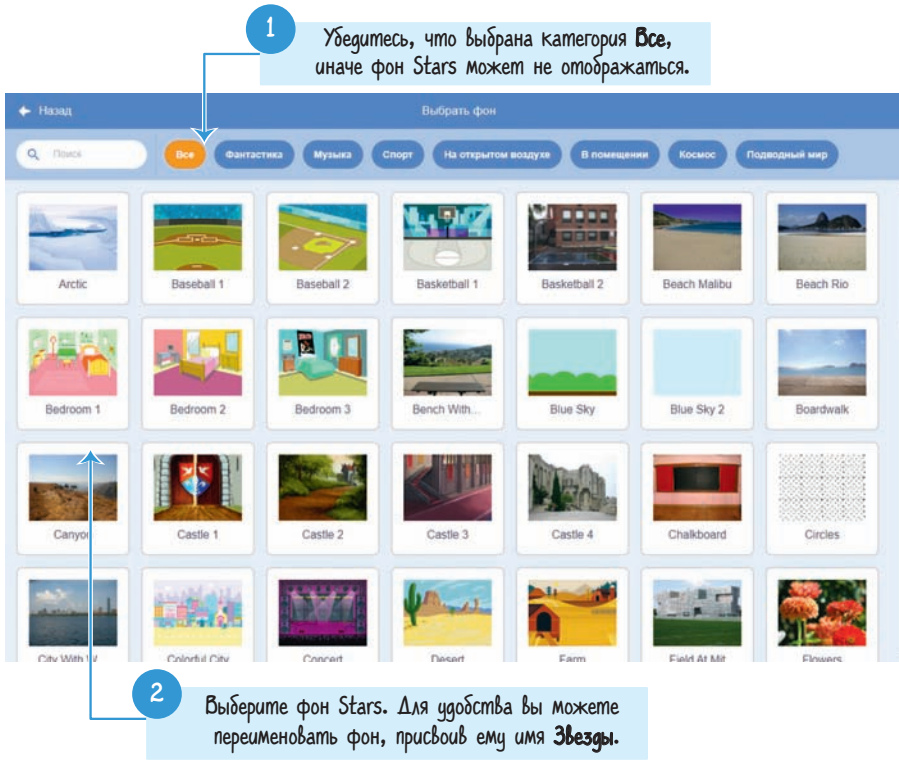
Каждый раз, когда вы создаете новый Scratch-проект, вы видите спрайт рыжего кота на пустой белой сцене. Для задуманной нами программы спрайт кота не понадобится, поэтому щелкните правой кнопкой мыши по элементу **Спрайт 1** в области спрайтов и выберите команду **Удалить** в контекстном меню (или щелкните мышью по значку корзины на миниатюре спрайта), чтобы убрать кота со сцены и из области спрайтов.



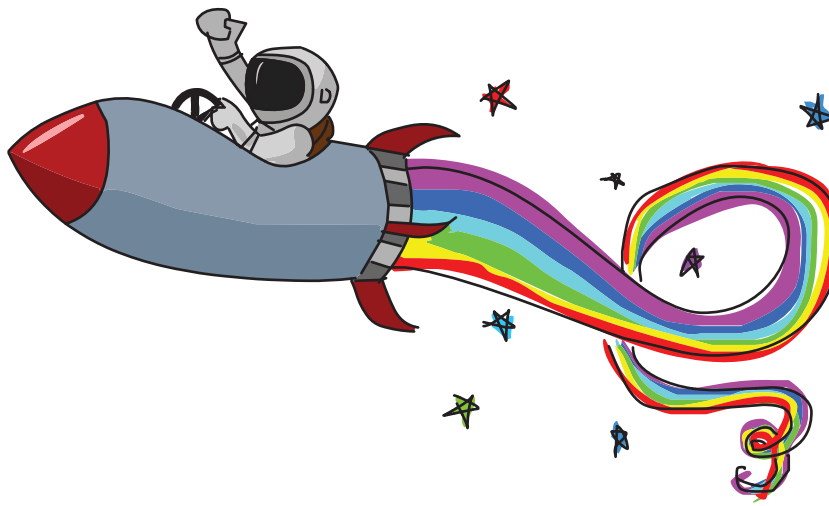
Нажмите кнопку **Выбрать фон** (она выглядит как изображение ландшафта).



Откроется окно **Выбрать фон**, в котором будут отображены все фоны, отсортированные в алфавитном порядке. Выберите фон **Stars**.

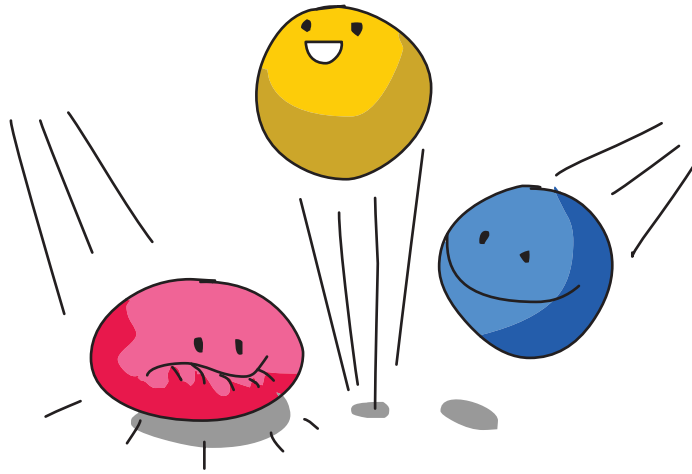


Теперь сцена выглядит как космическое пространство!



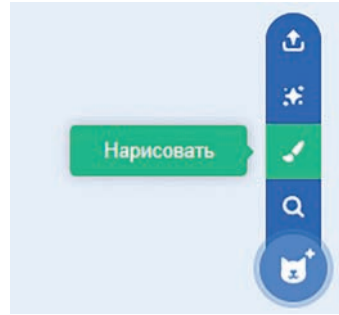
Б. СОЗДАНИЕ ТРЕХ ДВИЖУЩИХСЯ ТОЧЕК

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



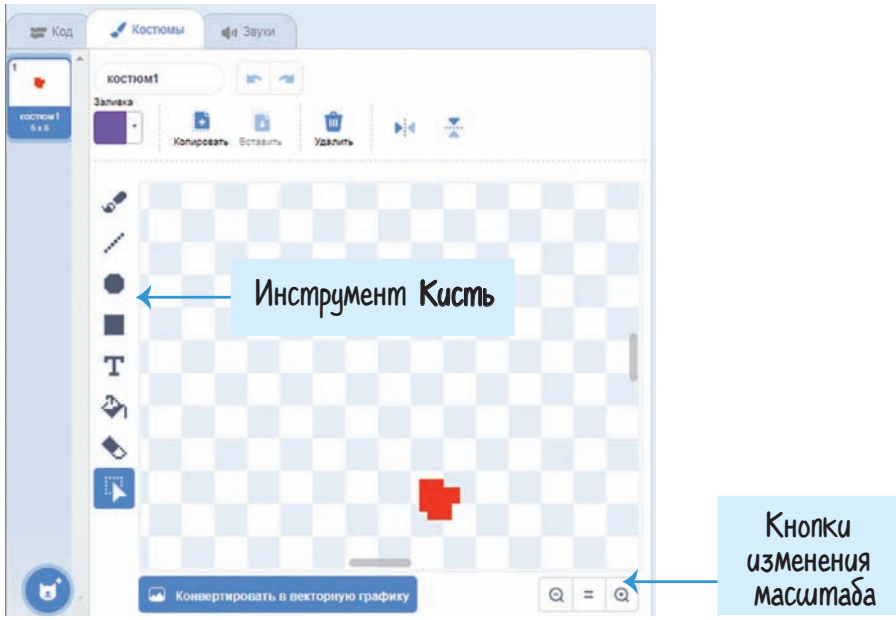
2. Рисование точки

Нажмите кнопку **Нарисовать,** которая прячется в кнопке **Выбрать спрайт**, которая выглядит как кисточка.

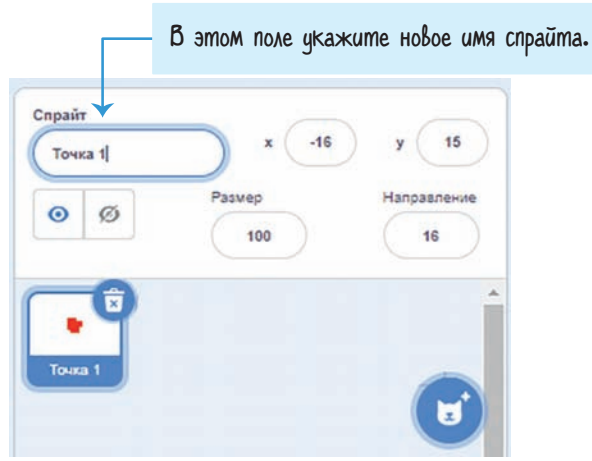


Новый спрайт с именем **Спрайт 1** появится в области спрайтов. При нажатии этой кнопки программа переключается на вкладку **Костюмы**, в которой вы увидите графический редактор. Используйте инструмент **Кисть**, чтобы нарисовать маленькую красную точку возле крестика в графическом редакторе. Для

большего удобства можно увеличить изображение, нажав кнопку масштабирования (она выглядит как увеличительное стекло).

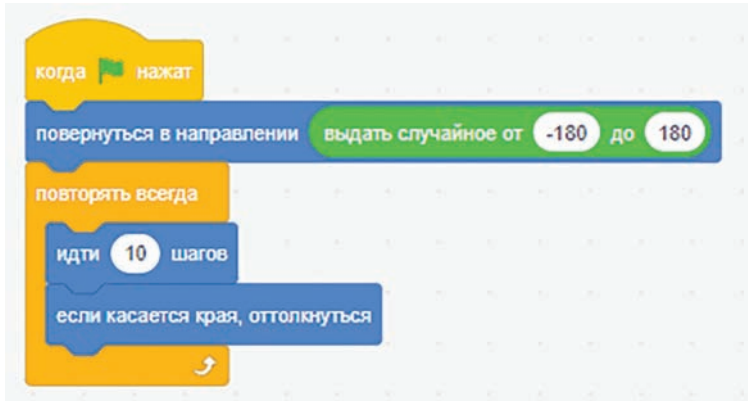


На панели свойств измените имя спрайта **Спрайт 1** на **Точка 1**.



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** (желтый), **Движение** (темно-синий), **Операторы** (зеленый) и **Управление** (светло-оранжевый). Если вам не совсем понятно, как перетащить эти блоки, посмотрите анимацию в разделе **Руководства.**

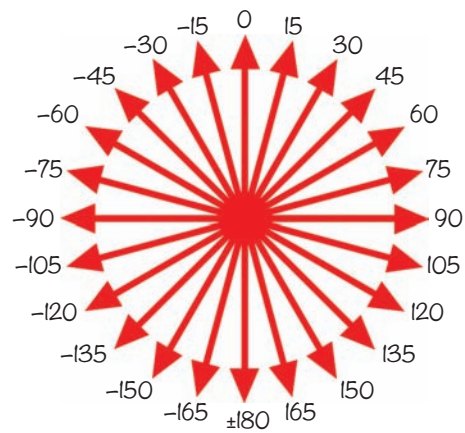


При нажатии кнопки в виде зеленого флага спрайт **Точка 1** появляется в случайной позиции между –180 и 180 градусами. Затем цикл **Повторять всегда** перемещает спрайт вперед на 10 шагов, и при достижении точкой края сцены она от него отскакивает. Таким образом, спрайт будет все время перемещаться.

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

ЭТО ИНТЕРЕСНО: НАПРАВЛЕНИЕ И ГРАДУСЫ

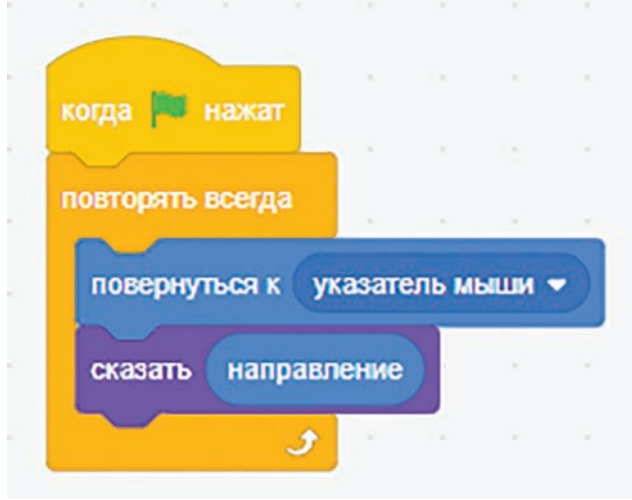
Такие слова, как «вверх» или «вправо», хорошо понятны людям. Но компьютеру нужно числовое значение, чтобы указать точное направление. Все спрайты в Scratch имеют свое собственное значение направления. Значения направления находятся в диапазоне от –180 до 180 градусов. Направление точно вверх задается значением 0 градусов. Указав значение в 90 градусов, вы задаете направление вправо. На следующем рисунке показаны 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.



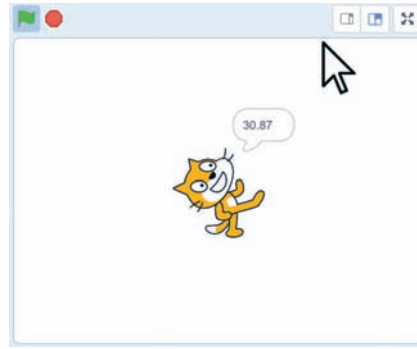
Блок **Выдать случайное от –180 до 180** выбирает случайное число для использования в качестве направления. Затем блок **Повернуться в направлении** указывает спрайту это направление. Это означает, что спрайт может быть направлен куда угодно.

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**, чтобы открыть новое окно редактора Scratch. Вы можете одновременно редактировать несколько программ Scratch.

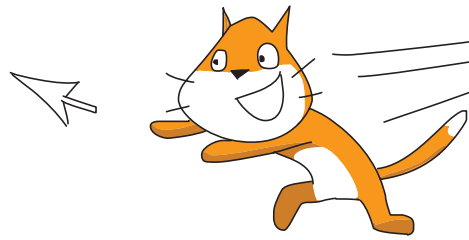
В области скриптов для спрайта кота с именем **Спрайт 1** добавьте код, показанный на следующем рисунке, используя блоки категорий **События** (желтый), **Управление** (светло-оранжевый), **Движение** (темно-синий) и **Внешний вид** (фиолетовый). Имейте в виду, что мы пишем совершенно отдельную от игры «Радужные линии в космосе» 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

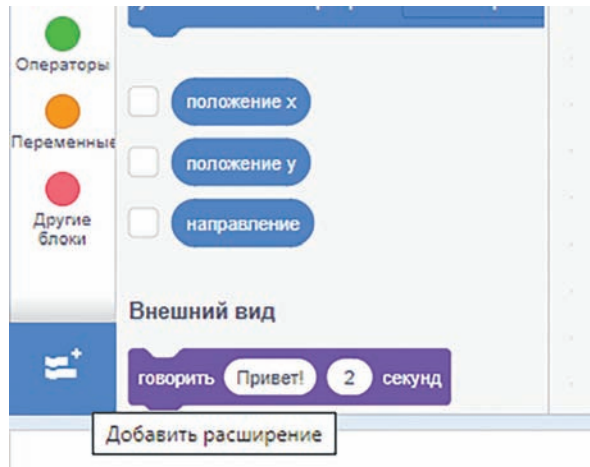
Нажмите кнопку в виде зеленого флага, чтобы проверить уже готовый код. Убедитесь, что все три точки движутся и отскакивают от края сцены. При дублировании спрайта **Точка 1** также продублировался и код спрайта. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

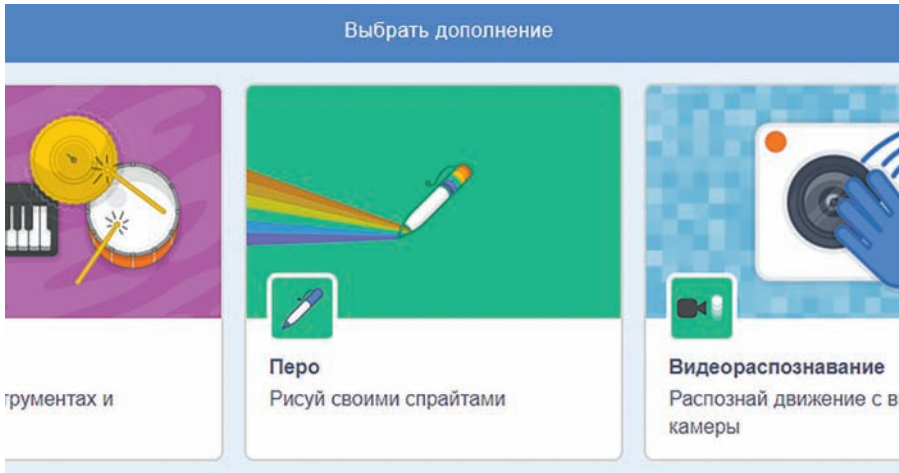
B. ПРОРИСОВКА ЛИНИЙ РАДУГИ

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

В этом проекте используются блоки категории **Перо**, которые по умолчанию не отображаются в списке слева. Необходимо нажать кнопку **Добавить расширение** в левом нижнем углу редактора и в появившемся окне выбрать расширение **Перо**.





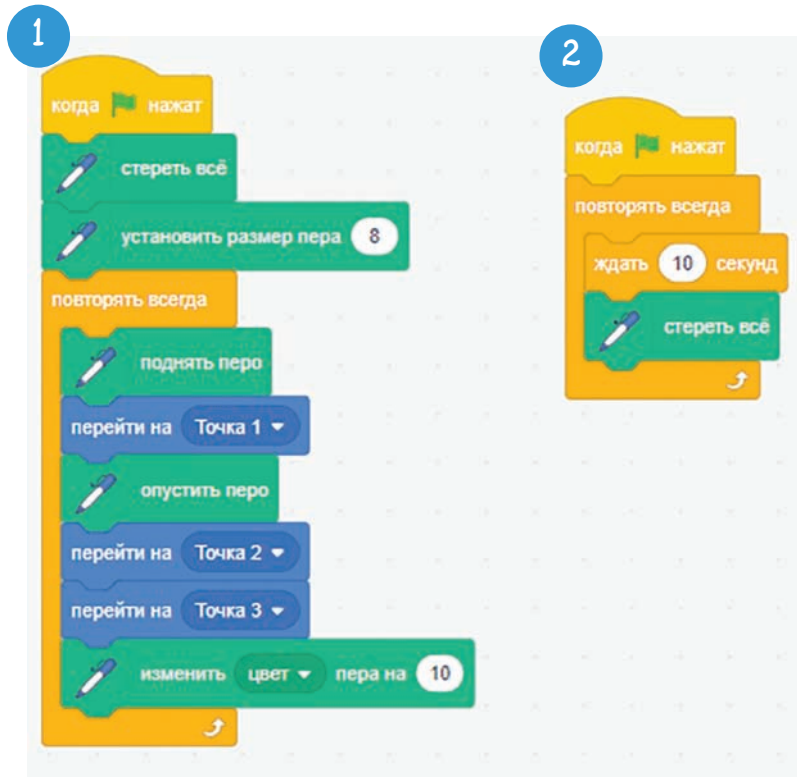
Щелкните правой кнопкой мыши по спрайту одной из отталкивающихся точек и выберите команду **Дублировать** в контекстном меню. Поскольку это дубликат, у него уже есть определенный код, который нам нужно будет удалить. В области скриптов нового спрайта удалите все блоки кода, нажав и удерживая кнопку мыши на блоке **Когда** ⚑ **нажат,** а затем перетащите его и все блоки ниже в левую часть окна.

На панели свойств переименуйте новый спрайт, присвоив ему имя **Рисующая точка**.

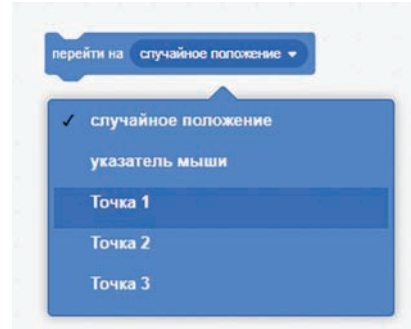
Добавьте два скрипта, показанных на следующем рисунке, в спрайт **Рисующая точка**. Вы можете найти эти блоки

в категориях **События** (желтый), **Перо** (темно-зеленый), **Управление** (светло-оранжевый) и **Движение** (темно-синий).

Напомню, что блоки кода в категории **Перо** появляются только в том случае, если вы добавили расширение **Перо**.



Убедитесь, что в скрипте ❶ вы используете блок **Перейти на**, а не блок **Перейти в x: y**. Кроме того, убедитесь, что вы поменяли в блоке **Перейти на** установленное по умолчанию значение **Указатель мыши.** Чтобы это сделать, выберите в раскрывающемся списке этого блока имя соответствующего спрайта.

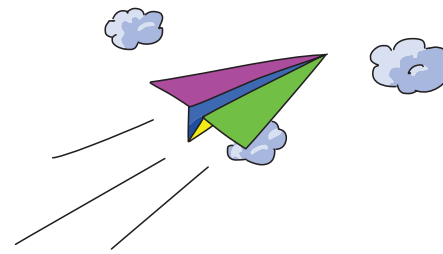


Перед тем как запустить код, давайте рассмотрим, как это работает. При нажатии кнопки в виде

зеленого флага в скрипте ❶ спрайт **Рисующая точка** запускает блок **стереть все**, 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 гигантский маркер на большом листе бумаги: 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.

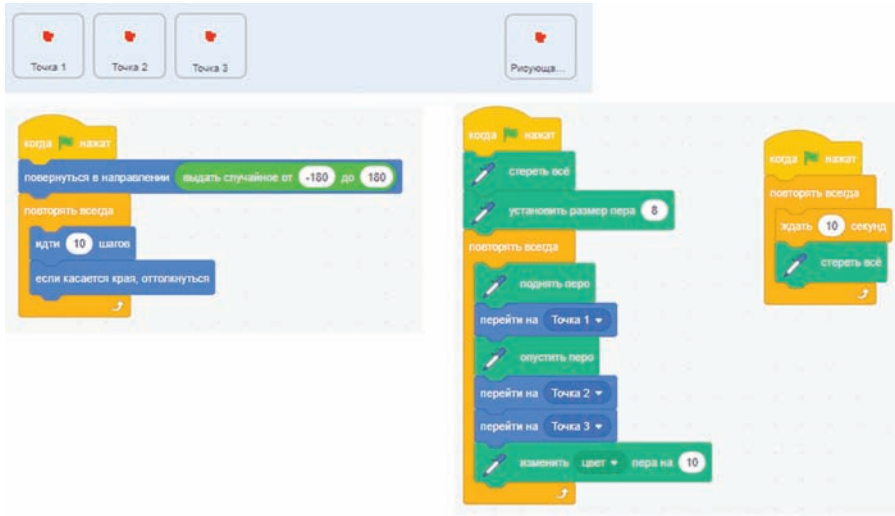


КОНТРОЛЬНАЯ ТОЧКА

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!

ГОТОВАЯ 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.

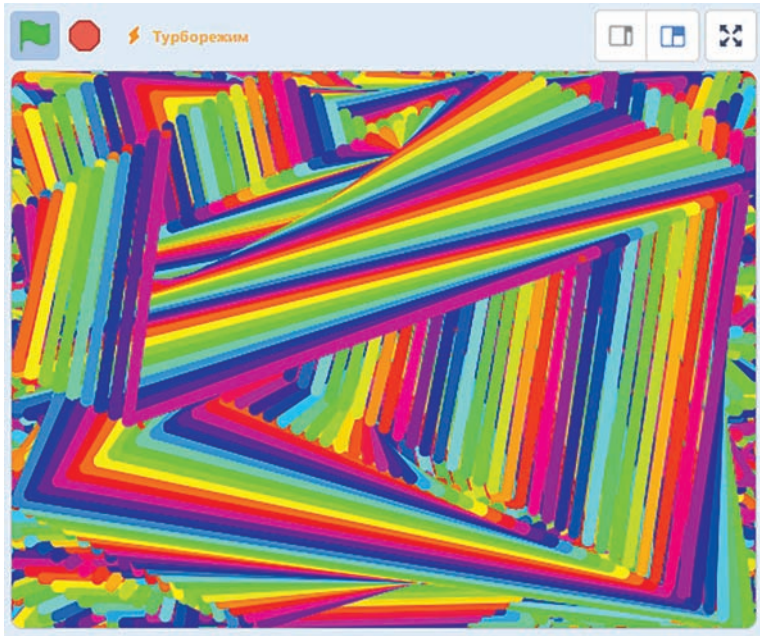


ТУРБОРЕЖИМ

Если удерживать клавишу **Shift** при нажатии кнопки в виде зеленого флага, можно запустить программу в турборежиме. 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 космосе** в турборежиме. Почти мгновенно экран будет

заполнен! Для завершения турборежима снова нажмите кнопку в виде зеленого флага, удерживая клавишу **Shift**.



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 — 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

Ответьте 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.)

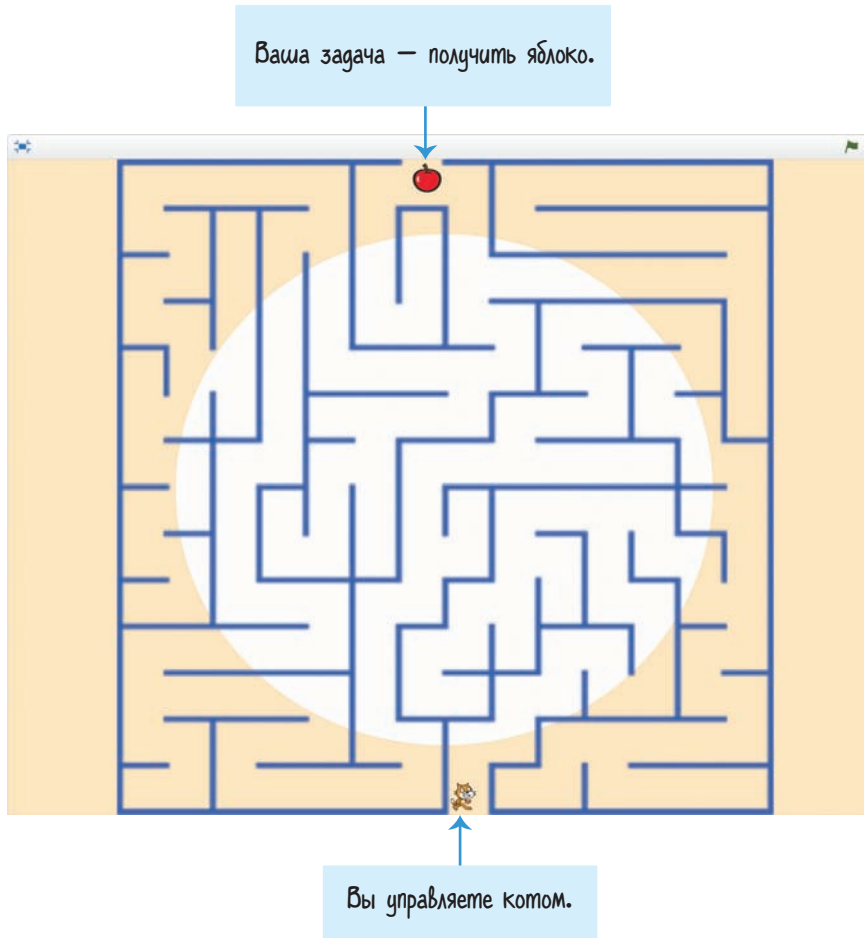
1. Что происходит в процессе движения спрайта после запуска блока **Опустить перо**?
2. Во время перемещения спрайта за ним не рисуется линия. Почему возникла такая ошибка?
3. Какой блок в программе **Радужные линии в космосе** отвечает за то, что линии выглядят радужными?
4. Какой блок кода нужно использовать, чтобы сделать радужные линии толще?
5. Как включается турборежим? 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 **Спрайт 1**. Как его переименовать?

3
БЕГУЩИЙ В ЛАБИРИНТЕ

Вероятно, вы уже играли в какую-нибудь игру-лабиринт. Но пробовали ли вы сделать такую игру сами? Лабиринты может быть сложно пройти, но их легко программировать. В этой главе вы создадите игру, в которой игрок должен провести кота через лабиринт. В конце его ждет приз — вкусное яблоко!

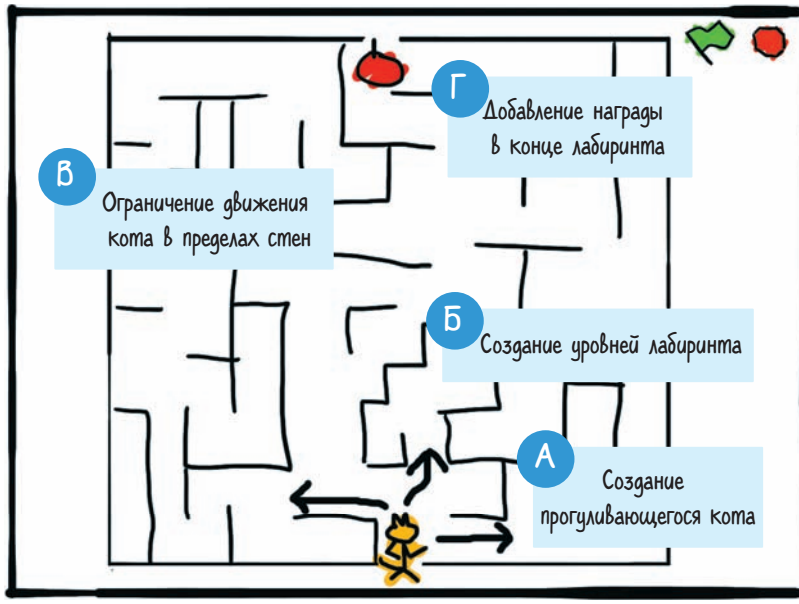
Вы узнаете, как управлять котом при помощи клавиатуры и как препятствовать движению, используя стены.

Перед тем как начать создавать код игры, взгляните на финальную программу. Перейдите по ссылке **https://scratch.mit.edu/projects/741003262** 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 показан на рисунке ниже.



Если вы хотите сэкономить время, можете начать с файла скелета проекта, который называется *Глава 03/ Проект.sb3* и находится в запакованном файле с примерами. Перейдите по ссылке **http://addons.eksmo.ru/it/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 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/ Проект.sb3*.

Даже если вы не используете готовый проект, 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**.

A. СОЗДАНИЕ ПРОГУЛИВАЮЩЕГОСЯ КОТА

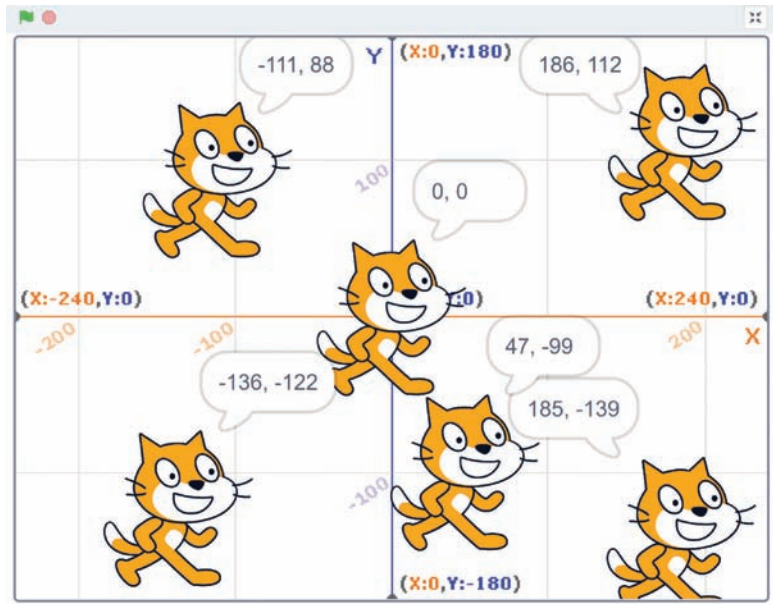
В игре «Бегущий в лабиринте» пользователь будет управлять спрайтом кота. В части A вы создадите код для управления котом клавишами со стрелками на клавиатуре.

ЭТО ИНТЕРЕСНО: КООРДИНАТЫ X И Y

Чтобы кот двигался по сцене, вам нужно использовать координаты. *Координатами* называются числа, которые указывают точное местоположение. Координата *x* — это 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, *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** и выберите фон.) Я добавил несколько спрайтов котов, сообщающих свои координаты.



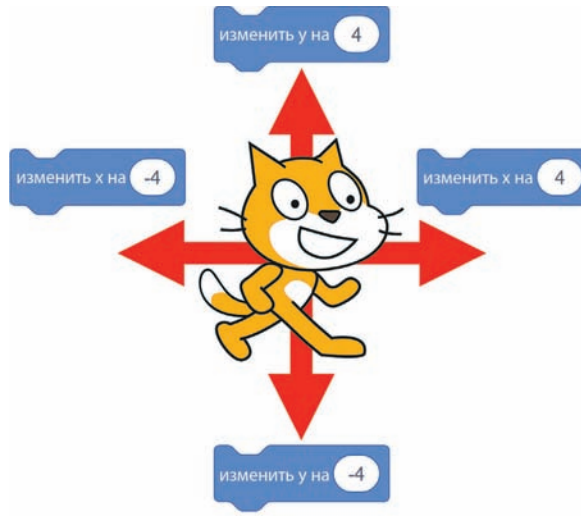
Крайняя правая точка на боковой стороне сцены имеет координату *x*, равную 240. Левее этой точки координаты *x* уменьшаются. В центре сцены координата *x* равна 0. Слева от центра координаты *x* становятся отрицательными числами. Крайняя левая точка на боковой стороне сцены имеет координату *x*, равную –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.

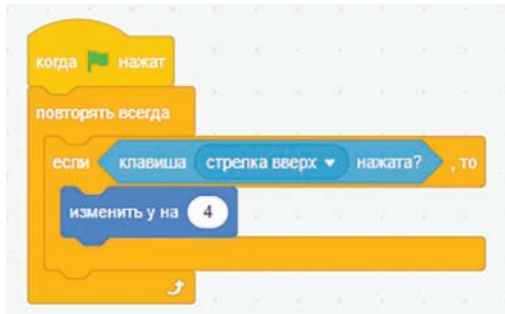
Чтобы спрайт переместился...	**измените его...**	**на...**
вправо	координату x	положительное число
влево	координату x	отрицательное число
вверх	координату y	положительное число
вниз	координату y	отрицательное число



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 изменить координату на отрицательное число — это то же самое, что поставить знак минус перед положительным числом.

1. Добавление кода движения для спрайта игрока

Первый фрагмент кода, который вы добавите, будет отвечать за перемещение спрайта кота (который называется **Спрайт 1**) с помощью клавиш со стрелками. Сначала переименуйте спрайт кота 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. Цикличный блок **Повторять 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 **Повторять 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, чтобы игра не заканчивалась. Если вам кажется, что ваша программа что-то не делает, убедитесь, что вы не забыли добавить блок **Повторять всегда**.

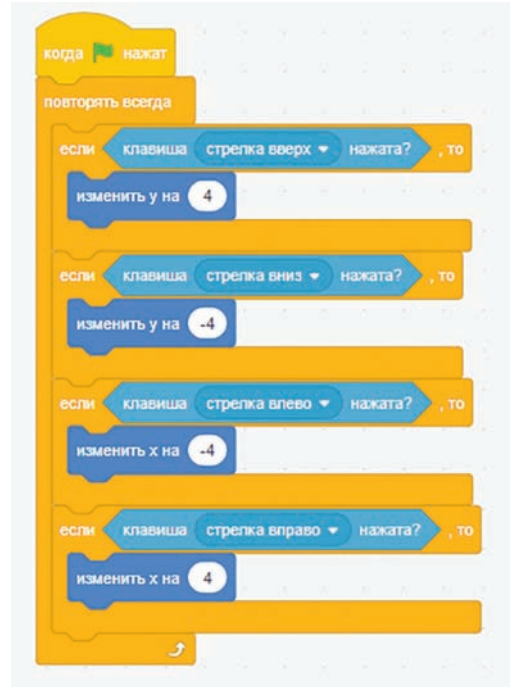
Когда вы пишете код самостоятельно, убедитесь, что вы используете блок **Изменить *y* на**, а не блоки **Изменить *x* на** или **Установить *y* на**. Если ваша программа не работает должным образом, проверьте, совпадает ли ваш код с кодом в этой книге.

КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага и попробуйте переместить кота, нажав клавишу ↑. 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: ←, ↓ и →. Он похож на код для перемещения вверх спрайта **Рыжий кот**. Чтобы сэкономить время, вы можете щелкнуть правой кнопкой мыши или нажать и удерживать оранжевый блок **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 все клавиши со стрелками одну за другой. После проверки клавиши → Scratch возвращается в верхнюю часть цикла и снова проверяет клавишу ↑. Он делает это 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.sb3*, который находится в архиве с примерами. В редакторе Scratch выберите команду меню **Файл ⇨ Загрузить с компьютера**, чтобы загрузить файл *лабиринт-часть-а.sb3*, а затем переходите к части Б.

Б. СОЗДАНИЕ УРОВНЕЙ ЛАБИРИНТА

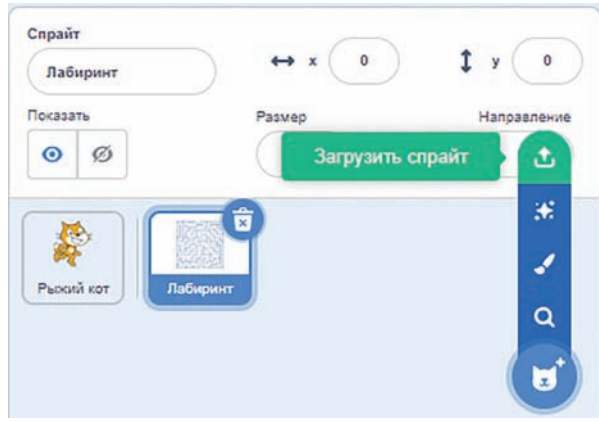
Далее мы создадим спрайт лабиринта и установим фон. Если все время проходить один и тот же лабиринт, играть быстро надоест, поэтому мы сделаем в игре несколько уровней.

3. Загрузка изображений лабиринта

Вы можете сами нарисовать спрайт лабиринта или использовать изображения из архива с примерами. Одно из изображений лабиринта — файл *Лабиринт.sprite3*.

В редакторе Scratch выберите пункт **Загрузить спрайт**, который скрывается в кнопке **Выбрать спрайт** (с изображением

кошачьей мордочки), и выберите файл *Лабиринт.sprite3,* 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**; она часто используется для анимации спрайта. Ваша область спрайтов должна выглядеть так.

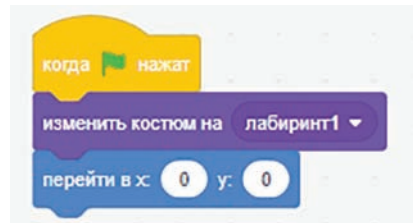


4. Изменение фона

Давайте добавим немного лоска в игру, поместив в качестве фона изображение. Вы можете использовать любой фон, который вам нравится. Изменить его можно, нажав кнопку **Выбрать фон**, чтобы открыть одноименное окно. 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**, **Внешний вид** 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 **Лабиринт 1** в блоке **Изменить 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.sb3*, который находится в архивном файле. В редакторе Scratch выберите команду меню **Файл ⇨ Загрузить с компьютера**, чтобы загрузить файл *лабиринт-часть-б.sb3*, а затем переходите к части В.

В. Ограничение движения кота в пределах стен

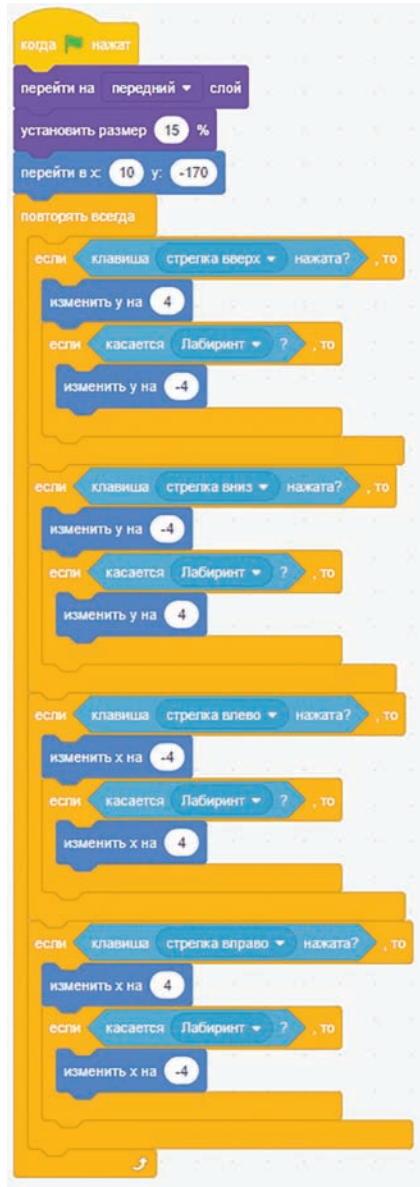
Теперь при нажатии кнопки с изображением зеленого флага вы сможете перемещать кота по лабиринту. Но вы также будете иметь возможность перемещать кота сквозь стены лабиринта, потому что ничто в программе не мешает этому.

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

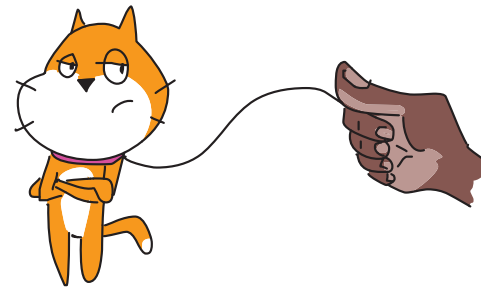
Давайте добавим код, который будет проверять, касается ли кот синих стен. Если да, то кот должен будет отходить назад. Поэтому, если кот движется вправо и касается стены, он должен автоматически передвинуться влево. Это устранит лишние движения игрока и сделает так, чтобы кот не перемещался сквозь стены. 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** большой, как Годзилла, и выглядит нереалистично. Добавьте блок **Установить размер %** из категории **Внешний вид**, чтобы уменьшить спрайт **Рыжий кот**.

Поскольку для прохождения лабиринта нужно, чтобы спрайт **Рыжий кот** всегда отображался поверх лабиринта, вам нужно добавить блок **Перейти на передний 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.sb3*, который находится в архиве с примерами. В редакторе Scratch выберите команду меню **Файл ⇨ Загрузить с компьютера**, чтобы загрузить файл *лабиринт-часть-в.sb3*, а затем переходите к части Г.

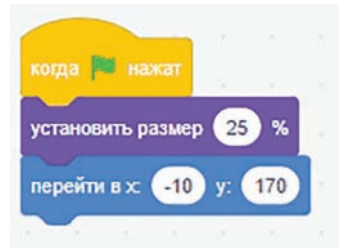
Г. ДОБАВЛЕНИЕ НАГРАДЫ В КОНЦЕ ЛАБИРИНТА

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

7. Создание спрайта яблока

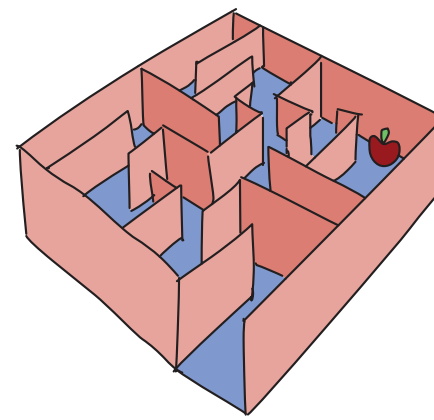
Нажмите кнопку **Выбрать спрайт**, чтобы открыть одноименное окно, и выберите спрайт **Apple**, чтобы добавить его в область спрайтов. Для удобства вы можете переименовать его в **Яблоко**.

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**.



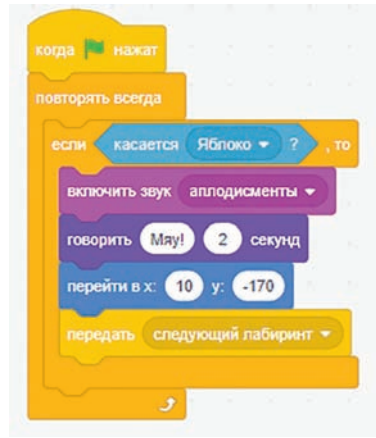
8. Выявление момента, когда игрок находит яблоко

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 спрайт **Рыжий кот**. Перейдите на вкладку **Звуки** в верхней части области блоков, а затем нажмите кнопку **Выбрать звук**. Эта кнопка выглядит как динамик.

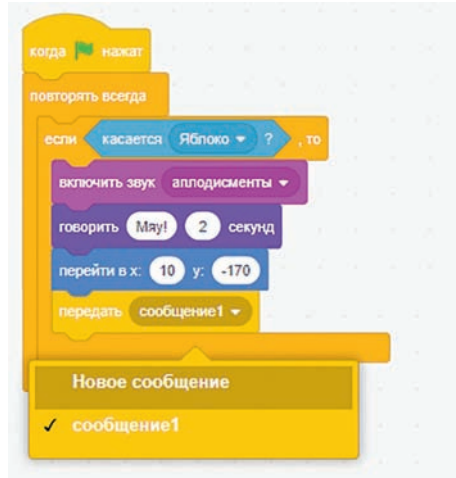


В появившемся окне **Выбрать звук** выберите звук **Cheer**, чтобы загрузить звук аплодисментов. Для удобства вы можете переименовать его в **Аплодисменты**. Теперь перейдите на вкладку **Скрипты**. Блок **Передать** будет передавать сообщение, активирующее сценарий блока **Когда я получу**.

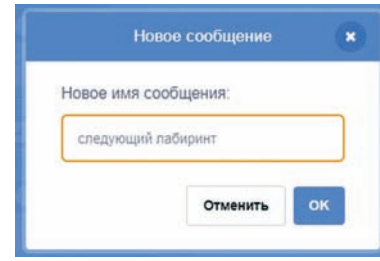
Добавьте код, показанный на следующем рисунке, в спрайт **Рыжий кот**.



Чтобы передавать сообщение **Следующий лабиринт**, выберите пункт **Новое сообщение** в раскрывающемся списке блока **Передать**.

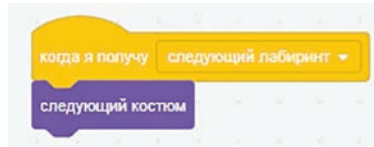


В появившемся окне введите текст **Следующий лабиринт** в качестве имени сообщения и нажмите кнопку **ОК**.



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**.

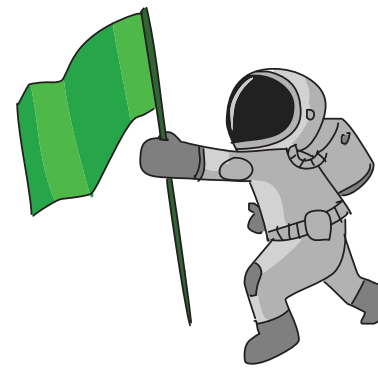
КОНТРОЛЬНАЯ ТОЧКА

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.

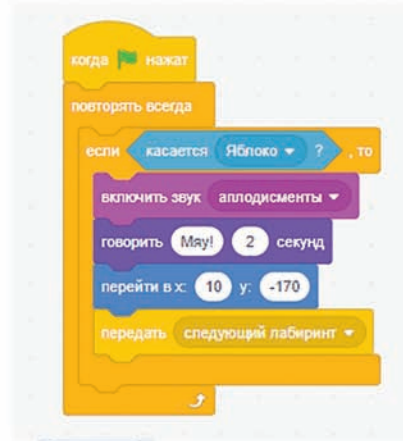
ГОТОВАЯ ПРОГРАММА

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.sb3*.

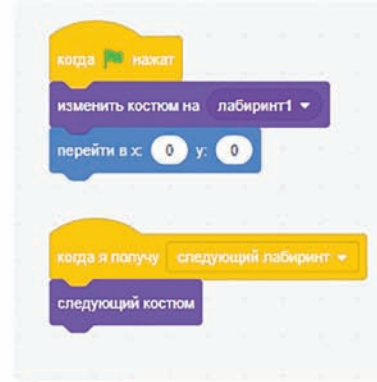
Я надеюсь, код этой программы-лабиринта не кажется вам слишком запутанным.



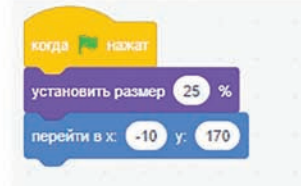

когда нажат
перейти на передний слой
установить размер 15 %
перейти в x: 10 y: -170
повторять всегда
если клавиша стрелка вверх нажата? , то
изменить y на 4
если касается Лабиринт ? , то
изменить y на -4
если клавиша стрелка вниз нажата? , то
изменить y на -4
если касается Лабиринт ? , то
изменить y на 4
если клавиша стрелка влево нажата? , то
изменить x на -4
если касается Лабиринт ? , то
изменить x на 4
если клавиша стрелка вправо нажата? , то
изменить x на 4
если касается Лабиринт ? , то
изменить x на -4
Рыжий кот


когда нажат
повторять всегда
если касается Яблоко ? , то
включить звук аплодисменты
говорить Мяу! 2 секунд
перейти в x: 10 y: -170
передать следующий лабиринт


Лабиринт


когда нажат
изменить костюм на лабиринт1
перейти в x: 0 y: 0
когда я получу следующий лабиринт
следующий костюм

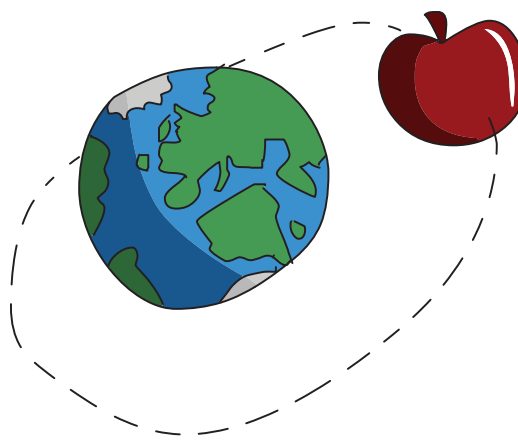

Яблоко


когда нажат
установить размер 25 %
перейти в x: -10 y: 170

ВЕРСИЯ 2.0: РЕЖИМ ДЛЯ ДВУХ ИГРОКОВ

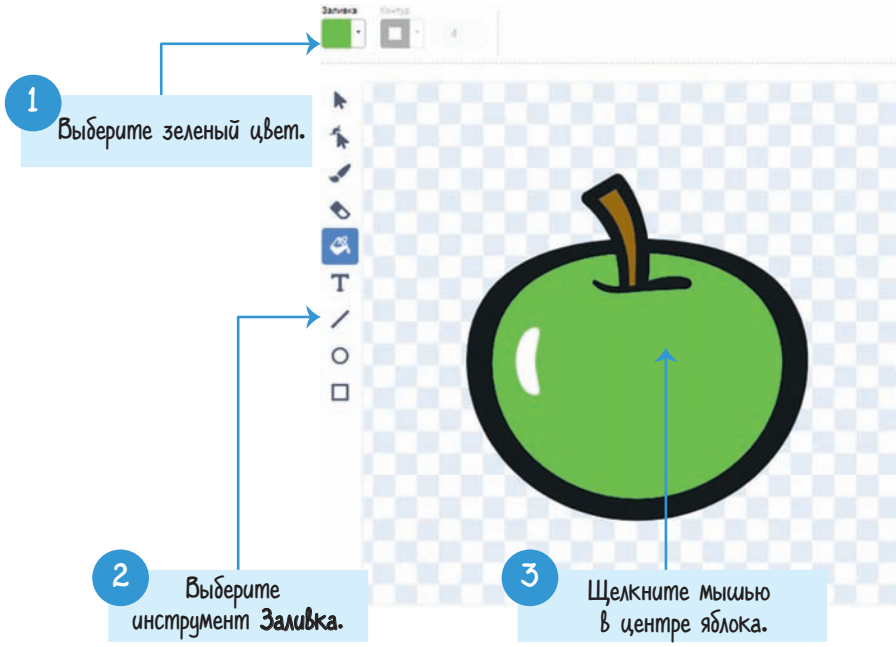
Теперь, когда базовая версия игры-лабиринта готова, вы можете выполнить некоторую доработку и добавить к ней небольшие улучшения, по одному за раз. Это называется итеративная разработка. Она помогает нам дополнять игру постепенно, чтобы не делать ее такой большой, что вы не сможете завершить начатое.

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.



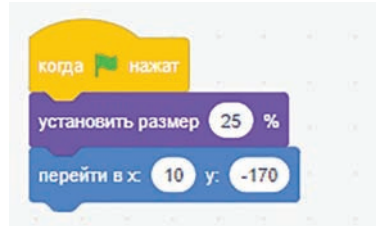
Дублирование спрайта Яблоко

Второму игроку тоже нужна цель. Щелкните правой кнопкой мыши (или нажмите и удерживайте) по спрайту **Яблоко** и выберите команду **Дублировать** в контекстном меню, чтобы создать копию яблока и его кода. Новому спрайту автоматически присваивается имя **Яблоко 2**. Выберите спрайт **Яблоко 2** и перейдите на вкладку **Костюмы**. Выберите зеленый цвет заливки, 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, **Яблоко 2** будет выглядеть так, как показано на следующем рисунке.



Изменение кода спрайта Яблоко 2

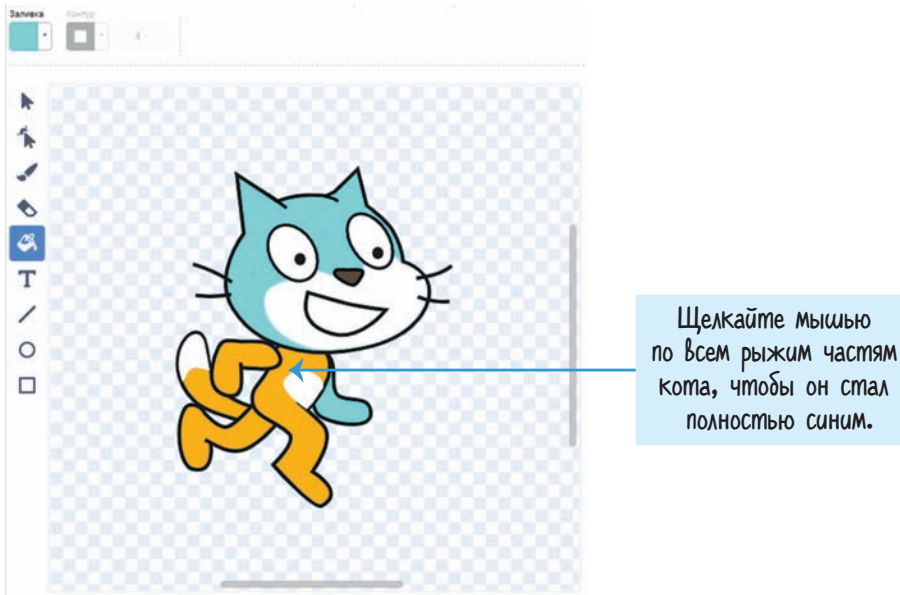
А теперь измените код спрайта **Яблоко 2**, чтобы он выглядел так, как показано на следующем рисунке, и зеленое яблоко появлялось в нижней части лабиринта, а не сверху.



Дублирование спрайта Рыжий кот

Теперь давайте добавим второй спрайт кота. Щелкните правой кнопкой мыши (или нажмите и удерживайте) по спрайту **Рыжий кот** и выберите из меню **Дублировать**, чтобы сделать копию этого спрайта и его кода. Новый спрайт автоматически получает имя **Рыжий кот 2**. Спрайты **Рыжий кот** и **Рыжий кот 2** должны различаться, чтобы игроки их не перепутали. Как и в случае

со спрайтом **Яблоко 2**, перейдите на вкладку **Костюмы** и измените цвет спрайта **Рыжий кот 2** с рыжего на синий.

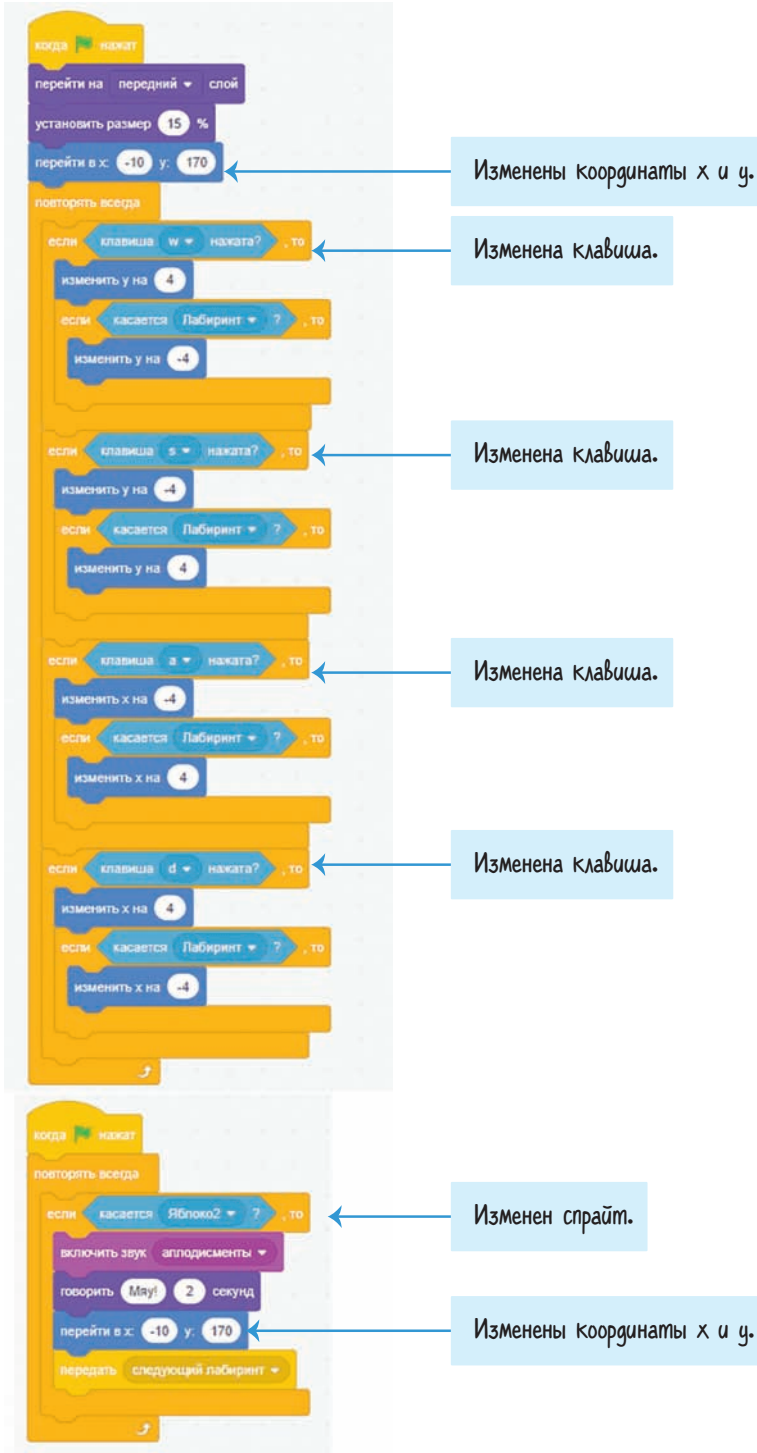


На панели свойств переименуйте спрайт **Рыжий кот 2** в **Синий кот**.

Изменение кода спрайта Синий кот

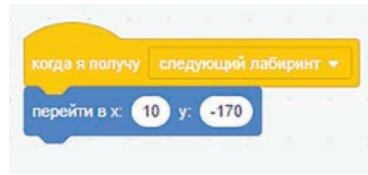
На данный момент спрайт **Синий кот** имеет тот же код, что и спрайт **Рыжий кот**. Вам нужно изменить код; в противном случае клавиши со стрелками будут управлять обоими спрайтами сразу. Второй игрок будет управлять спрайтом **Синий кот** с помощью клавиш **W**, **A**, **S** и **D**. Эти клавиши часто используются в качестве альтернативы клавишам ↑, ←, ↓ и → 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 Яблоко 2**.

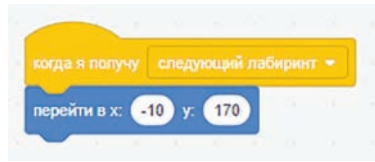

когда нажат
перейти на передний слой
установить размер 15 %
перейти в x: -10 y: 170
повторять всегда
если клавиша w нажата? , то
изменить y на 4
если касается Лабиринт ? , то
изменить y на -4
если клавиша s нажата? , то
изменить y на -4
если касается Лабиринт ? , то
изменить y на 4
если клавиша a нажата? , то
изменить x на -4
если касается Лабиринт ? , то
изменить x на 4
если клавиша d нажата? , то
изменить x на 4
если касается Лабиринт ? , то
изменить x на -4
когда нажат
повторять всегда
если касается Яблоко2 ? , то
включить звук аплодисменты
говорить Мяу! 2 секунд
перейти в x: -10 y: 170
передать следующий лабиринт
Изменены координаты x и y.
Изменена клавиша.
Изменена клавиша.
Изменена клавиша.
Изменена клавиша.
Изменен спрайт.
Изменены координаты x и y.

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.

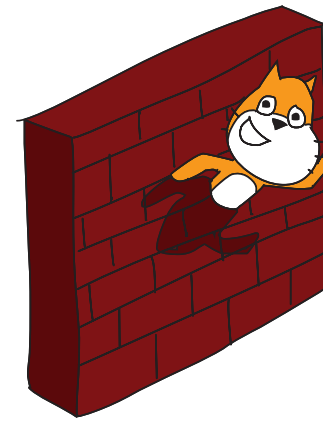
КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый код. Попробуйте передвинуть обоих игроков с помощью клавиш ↑, ←, ↓, → 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 изменили игру «Бегущий в лабиринте», создав режим с двумя игроками. Найдите друга для этой гонки. Игрок 1 использует клавиши ↑, ←, ↓ и →, а игрок 2 — клавиши **W**, **A**, **S** и **D**.

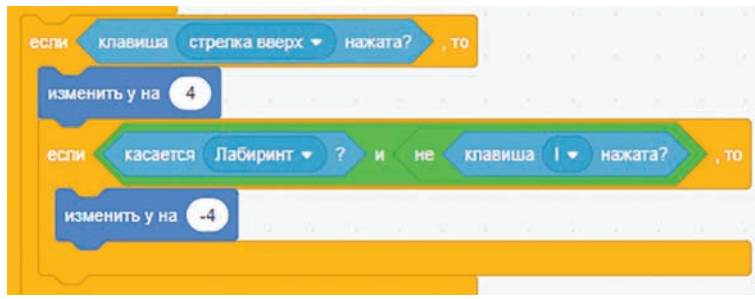
ЧИТ-РЕЖИМ: УМЕНИЕ ПРОХОДИТЬ СКВОЗЬ СТЕНЫ

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 специальная клавиша удерживается в нажатом положении.



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 l нажата?**. На рисунке ниже показан код для клавиши ↑. 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!

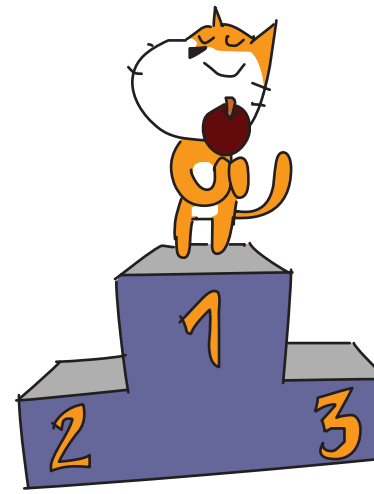
ЗАКЛЮЧЕНИЕ

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;
- содержится чит-режим, благодаря которому коты могут проходить сквозь стены.

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

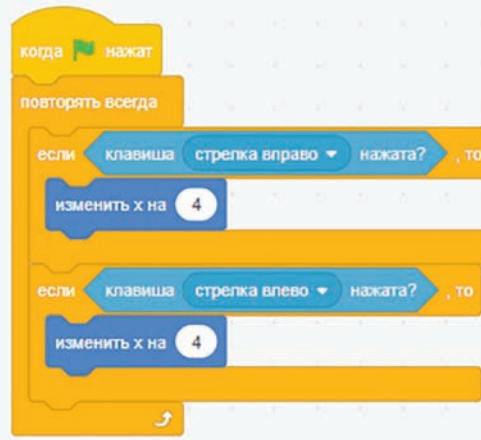
В главе 4 вы поиграете в баскетбол. В этой игре вид будет сбоку, в отличие от предыдущей игры, в которой используется вид сверху. Но это означает, что вы сможете добавлять прыжки и учитывать силу тяжести. Эти возможности часто используются во многих Scratch-играх.



ОБЗОРНЫЕ ВОПРОСЫ

Ответьте 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 *y* 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?

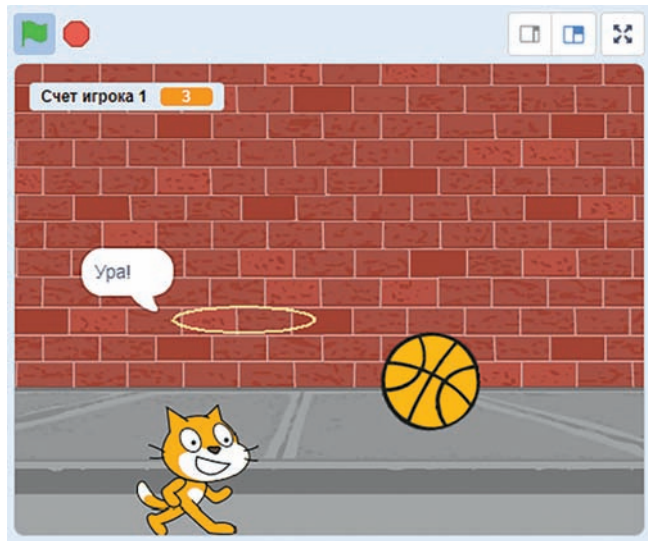


4

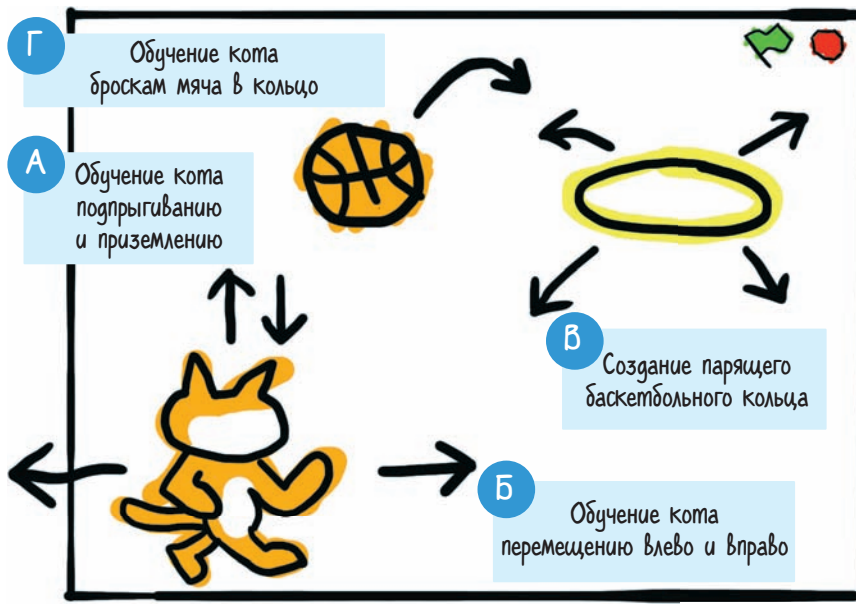
БАСКЕТБОЛ С УЧЕТОМ СИЛЫ ТЯЖЕСТИ

Во многих играх-платформерах, таких как «Супер-Марио» и «Соник Суперъежик», 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, пройдя по ссылке **https://scratch.mit.edu/projects/741003231**.



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 проекта под названием *Глава 04/Проект.sb3,* находящегося в запакованном файле с примерами, который вы скачали ранее по ссылке **https://eksmo.ru/files/Scratch_Sweigart.zip**. В файл проекта уже загружены все спрайты; вам нужно всего лишь перетащить блоки кода в каждый спрайт.

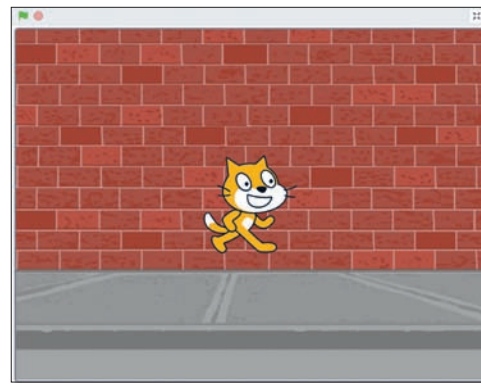
А. Обучение кота подпрыгиванию и приземлению

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

1. Добавление кода силы тяжести к спрайту кота

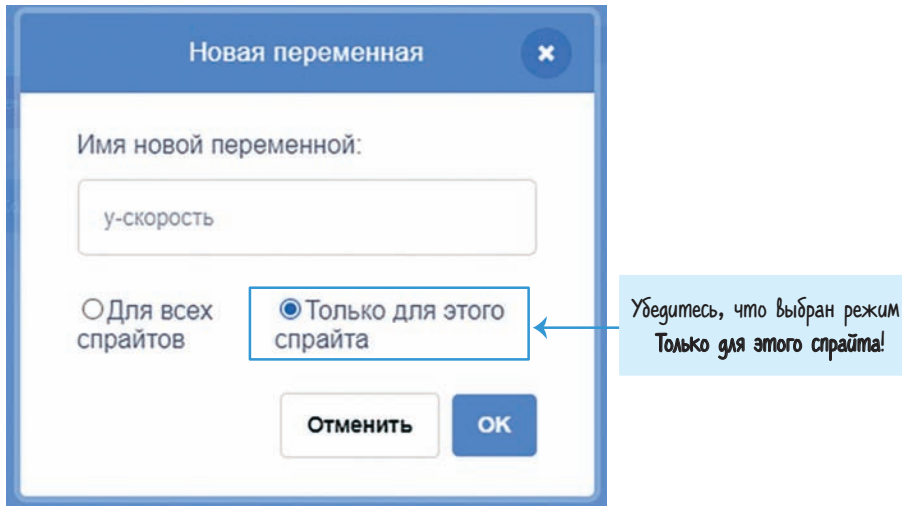
На панели свойств измените имя спрайта **Спрайт 1** на **Кот**. Далее в текстовом поле в верхней части редактора Scratch переименуйте программу *Untitled* в **Баскетбол**.

Нажмите кнопку **Выбрать фон**, чтобы открыть окно библиотеки фонов. Выберите вариант **Wall 1**, чтобы изменить фон. Для удобства вы можете переименовать фон, присвоив ему имя **Кирпичная стена**. Сцена будет выглядеть следующим образом.

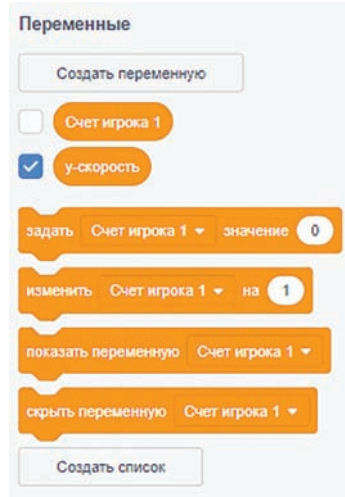


Для программирования силы тяжести требуются переменные. Переменную можно представить как коробочку, в которой хранятся числа или текст. Ее можно открыть при необходимости и использовать ее содержимое в своей программе. Мы создадим числовую переменную, которая задает скорость падения кота.

Первым делом удостоверьтесь, что в области спрайтов выбран спрайт **Кот**. Затем перейдите на вкладку **Скрипты**. В категории **Переменные** оранжевого цвета нажмите кнопку **Создать переменную**, в результате чего появится окно **Новая переменная**. Введите текст ***y*-скорость** в качестве имени переменной. (*Скорость* означает, как быстро какой-либо объект движется и в каком направлении.) Когда значение переменной ***y*-скорость** представляет собой положительное число, кот движется вверх. Когда значение переменной ***y*-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**.

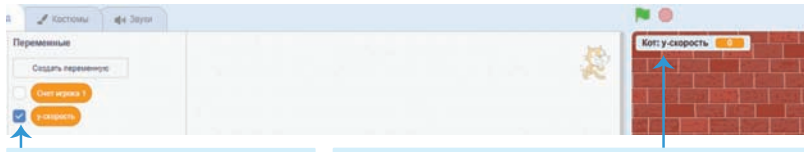


В категории **Переменные** появится несколько новых блоков. Одним из них будет скругленный блок переменной ***y*-скорость**, который можно увидеть на этом рисунке.



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

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



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

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

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.

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 **Переменные** 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, указанные в качестве значения переменной. Когда вы используете переменные, их значения (число или текст) меняются *во время работы программы*.

Чтобы установить значение переменной, используйте 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**, что 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.)

С помощью переменной блок **Сказать** может отображать разный текст во время работы программы. 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 — причина 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; затем скрипт запускает цикл **Повторять 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

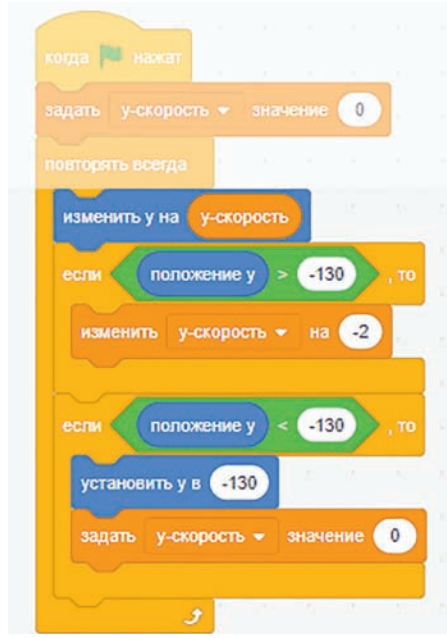
программного цикла положение по оси *y* будет меняться все быстрее и быстрее, что заставляет кота так же быстрее и быстрее падать.

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 ***y*-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 перебро-шен на уровень земли –130 и ***y*-скорость** вернется обратно к 0, чтобы остановить падение спрайта.

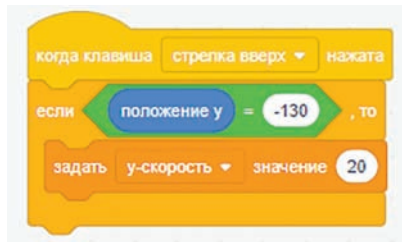
КОНТРОЛЬНАЯ ТОЧКА

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 ***y*-скорость**, сбросив флажок рядом с ее именем в блоке в оранжевой категории **Переменные**.

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 ↑, переменной ***y*-скорость** присваивается положительное значение 20, что заставляет спрайт **Кот** подпрыгивать вверх. Но значение переменной ***y*-скорость** все равно будет меняться на –2 при каждом прохождении цикла. Поэтому, хотя сначала кот и подпрыгивает на 20, после прохождения цикла он будет на 18, затем 16 и т. д. 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, а не в воздухе!

Когда переменная ***y*-скорость** имеет значение 0, спрайт **Кот** находится в верхней точке прыжка. После этого за каждое прохождение цикла значение переменной ***y*-скорость** изменяется на –2, и спрайт продолжает падать, пока не упадет на землю. Попробуйте поэкспериментировать с различными числами для блоков **Задать *y*-скорость значение** и **Изменить *y*-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 ↑ и убедитесь, что кот подпрыгивает вверх и падает вниз. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

Б. ОБУЧЕНИЕ КОТА ПЕРЕМЕЩЕНИЮ ВЛЕВО И ВПРАВО

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

4. Добавление кода ходьбы к спрайту Кот

Добавьте код, показанный на следующем рисунке, в нижнюю часть скрипта спрайта **Кот**.

Внутри цикла **повторять 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** создает впечатление, что кот движется.

КОНТРОЛЬНАЯ ТОЧКА

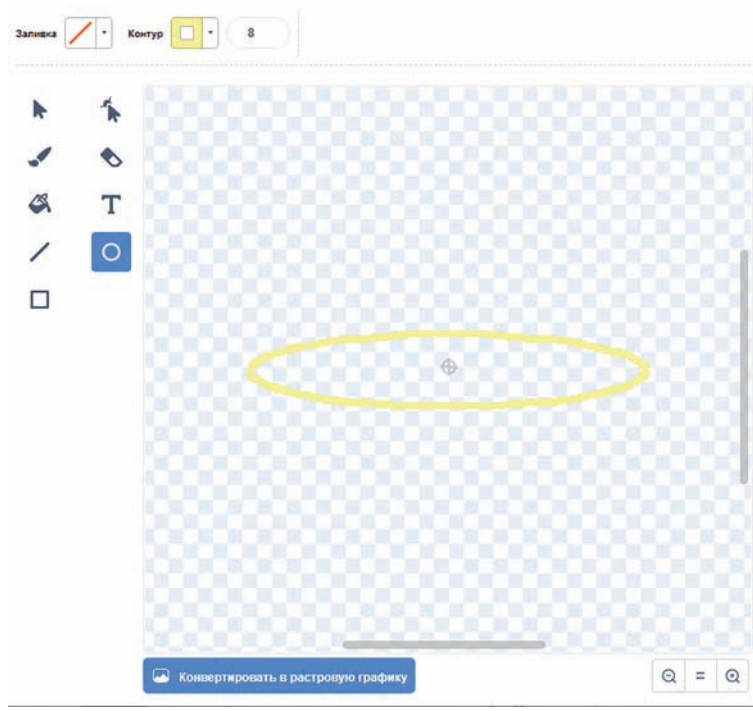
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Нажимайте клавиши ← и → и убедитесь, что кот движется в правильном направлении. Если при нажатии кнопки ← кот идет назад, это именно то, что нужно. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

B. СОЗДАНИЕ ПАРЯЩЕГО БАСКЕТБОЛЬНОГО КОЛЬЦА

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

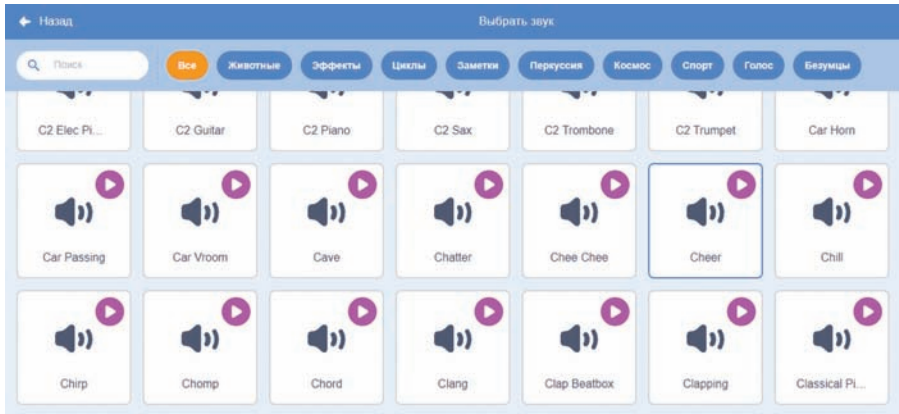
5. Создание спрайта кольца

Нажмите кнопку **Нарисовать**, которая прячется в кнопке **Выбрать спрайт** в списке спрайтов. Кнопки инструментов для рисования отобразятся в правой части графического редактора. Выберите желтый цвет и, используя инструмент **Круг**, нарисуйте кольцо. Вы также можете указать нужное значение в поле справа от раскрывающегося списка **Контур**, чтобы сделать линию кольца толще. Убедитесь, что крестик графического редактора находится в центре кольца.

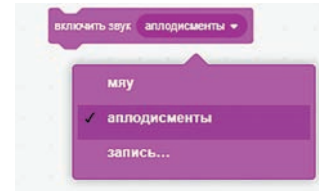


Присвойте созданному спрайту имя **Кольцо** на панели свойств. Давайте сделаем так, чтобы спрайт **Кольцо** проигрывал звук аплодисментов, когда игрок попадает в кольцо. Для этого загрузите

звук **Cheer**. Перейдите на вкладку **Звуки** в верхней части области блоков, а затем нажмите кнопку **Выбрать звук**. В открывшемся окне выберите звук **Cheer**. Для удобства вы можете переименовать звук, присвоив ему имя **Аплодисменты**.



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



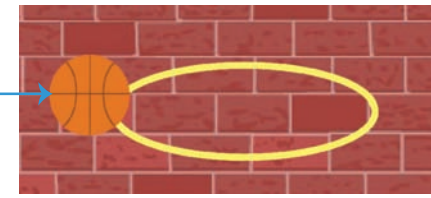
Добавьте код, показанный на следующем рисунке, в спрайт **Кольцо**, чтобы сделать его летающим в верхней половине сцены. Вам нужно создать сообщение, выбрав в раскрывающемся списке блока **Когда я получу** пункт **Новое сообщение**. Назовите новое сообщение **Свисток**.

Скрипт ❶ программирует кольцо на перемещение в новое положение каждую секунду. Играть в игру с движущимся кольцом намного сложнее! Скрипт ❷ 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.

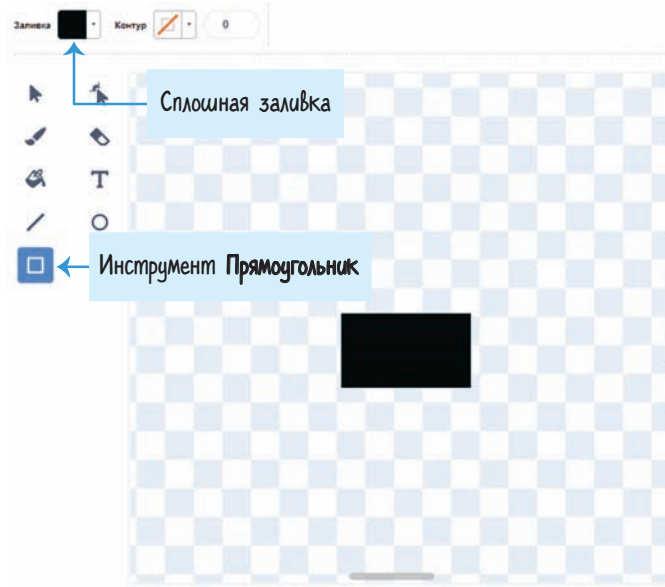
6. Создание хитбокса

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

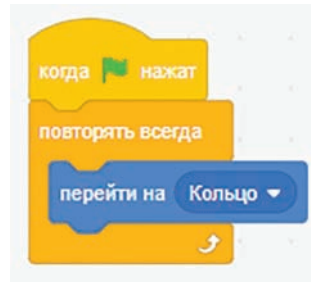
Если программа будет проверять, коснулся ли мяч кольца, бросок будет засчитан. Но в баскетболе не такие правила.



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, выбрав пункт **Нарисовать**, которые прячется в кнопке **Выбрать спрайт** в списке спрайтов. Нарисуйте маленький черный квадрат в середине крестика, используя инструмент **Прямоугольник** и выбрав сплошной вариант заливки. Назовите этот спрайт **Хитбокс**. Он будет выглядеть так.

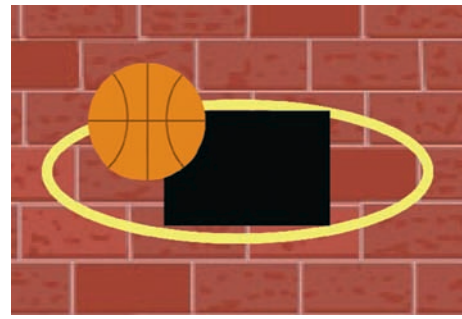


Добавьте код, показанный на следующем рисунке, к спрайту **Хитбокс**.

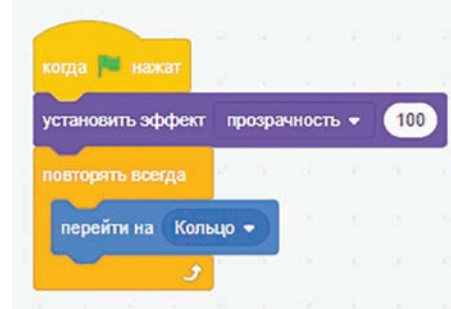


Спрайт **Хитбокс** теперь будет следовать за спрайтом **Кольцо** независимо от того, куда он переместится.

На шаге 9 мы напишем программу, которая в качестве попадания будет учитывать только касание баскетбольного мяча спрайта **Хитбокс**, а не спрайта **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. Добавьте блок **Установить эффект: прозрачность** (изначально этот блок отображается на панели блоков как **Установить эффект: цвет**, но вы можете изменить значение **цвет** на **прозрачность**) к спрайту **Хитбокс** и установите значение 100.



Существует различие между блоками **Спрятаться** и **Установить эффект: прозрачность 100**. Если вы используете блок **Спрятаться**, чтобы сделать спрайт **Хитбокс** невидимым, соприкасающиеся блоки не обнаружат, что мяч коснулся спрайта **Хитбокс** и игрок никогда не забросит мяч в кольцо. Блок **Установить эффект: прозрачность 100** делает спрайт невидимым, но 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. ОБУЧЕНИЕ КОТА БРОСКАМ МЯЧА В КОЛЬЦО

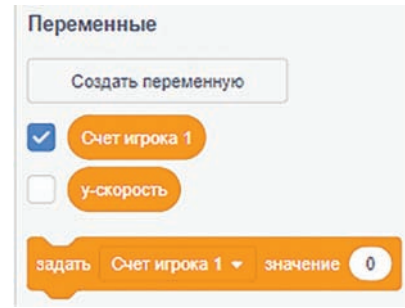
В этой части вы добавите в игру баскетбольный мяч, который кот будет бросать. Как и кот, мяч будет подчиняться закону тяготения и падать на землю.

7. Создание спрайта баскетбольного мяча

Нажмите кнопку **Выбрать спрайт**, чтобы открыть одноименное окно. Выберите спрайт **Basketball**. Для удобства вы можете переименовать спрайт в **Баскетбол**.

Затем перейдите на вкладку **Звуки** в верхней части области блоков и нажмите кнопку **Выбрать звук**, чтобы открыть одноименное окно. Выберите звук **Pop**. Для удобства его можно переименовать в **Хлопок**. Перейдите на вкладку **Скрипты** в верхней части области блоков, чтобы снова отобразить область скриптов.

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

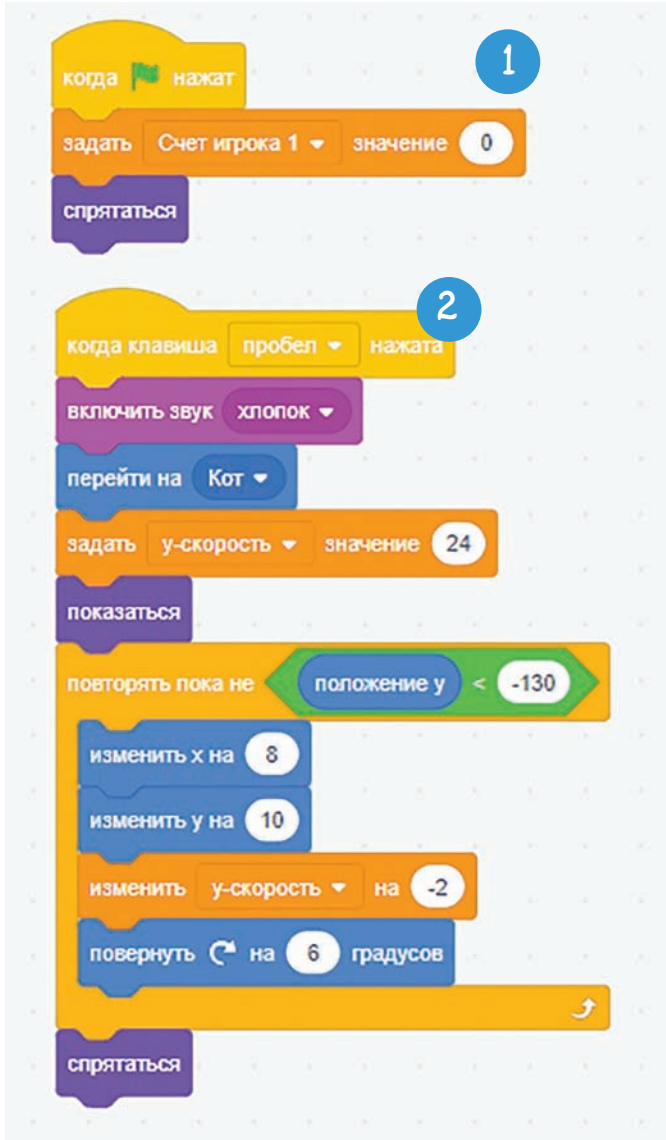


Нажмите кнопку **Создать переменную** еще раз, чтобы создать еще одну переменную с именем **Счет игрока 1**, но на этот раз установите переключатель в положение **Для всех спрайтов**. (Имя переменной **Счет игрока 1** начинается с заглавной буквы, потому что ее будет видно на сцене. Сбросьте

флажок напротив имени переменной **y-скорость**, чтобы ее не было видно на сцене.) В оранжевой категории **Переменные** появятся новые блоки переменных.

8. Добавление кода для спрайта Баскетбол

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



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

Скрипт ❷ использует код, аналогичный коду спрайта **Кот**. Когда игрок нажимает клавишу **Пробел**, баскетбольный мяч появляется перед котом и начинает двигаться вперед. Код присваивает переменной ***y*-скорость** спрайта **Баскетбол** значение — положительное число. Как вы помните, переменной ***y*-скорость** спрайта **Кот** тоже 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 игрок в следующий раз 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.

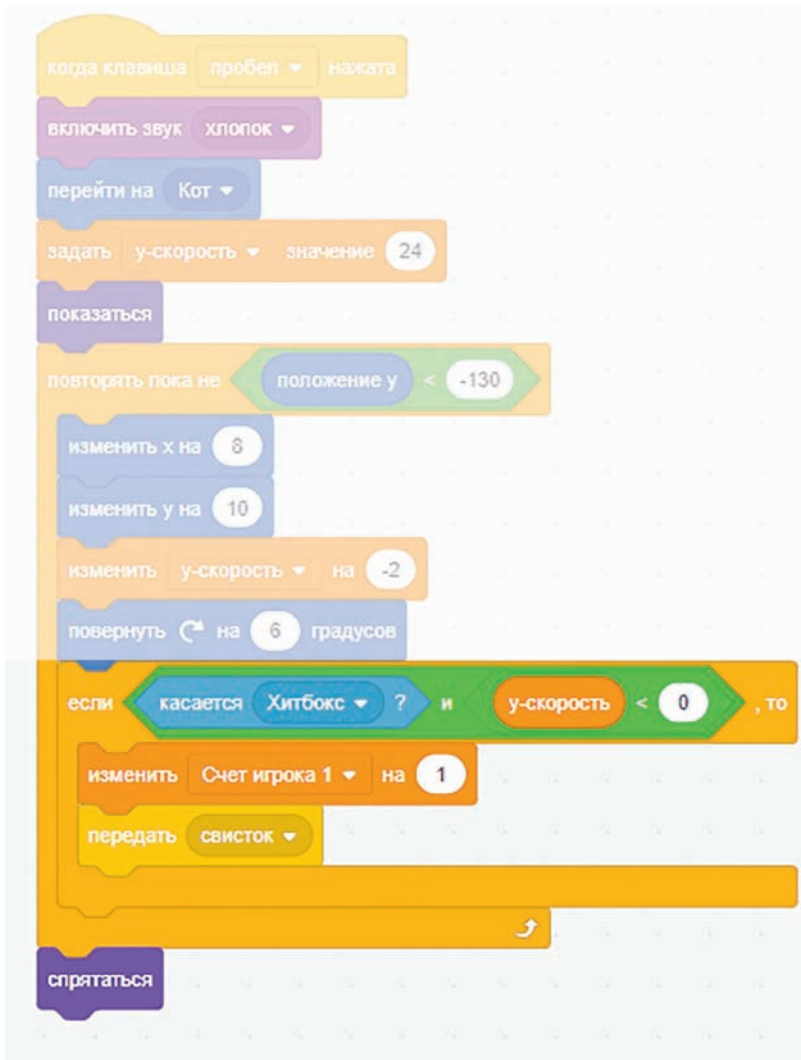
9. Учет успешных бросков

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. Но есть одна загвоздка: бросок не должен засчитываться, если баскетбольный мяч проходит через кольцо *снизу вверх*.

Имейте в виду: если значение переменной ***y*-скорость** положительное, то блок **Изменить *y* на *y*-скорость** будет направлять движение спрайта **Баскетбол** вверх. Если значение переменной ***y*-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 движется 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 ***y*-скорость** было меньше 0. Чтобы бросок был засчитан, оба эти условия — и **Касается Хитбокс?**, и ***y*-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.

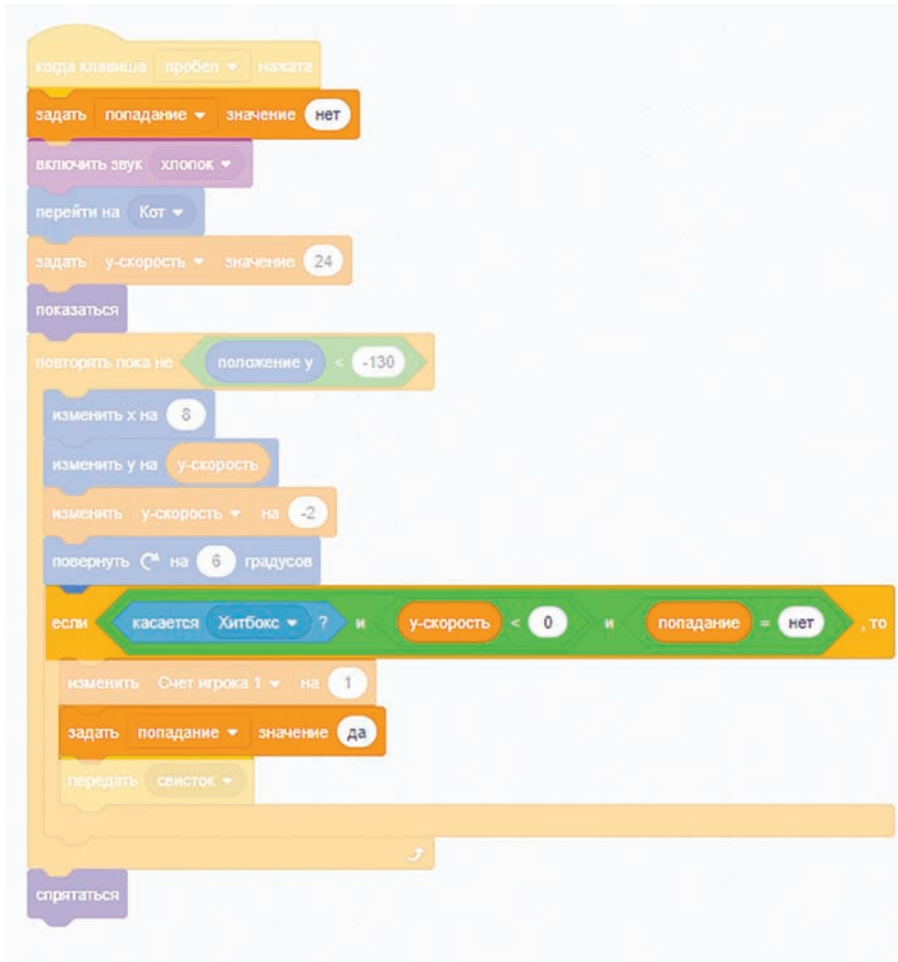
10. Исправление ошибки в счете

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

Цикличный 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 нам надо, 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.

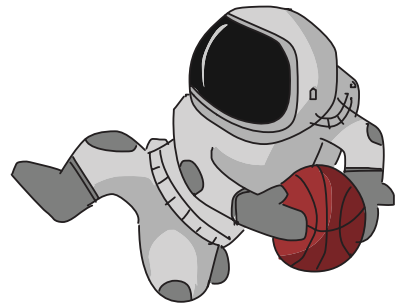
Перейдите в оранжевую категорию **Переменные** в верхней части области блоков, а затем нажмите кнопку **Создать переменную**. Присвойте переменной имя **Попадание** и установите переключатель в положение **Только для этого спрайта**. Затем измените код спрайта **Баскетбол** следующим образом.



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 ***y*-скорость** отрицательно (баскетбольный мяч падает).
3. Переменной **Попадание** присвоено значение **Нет**.



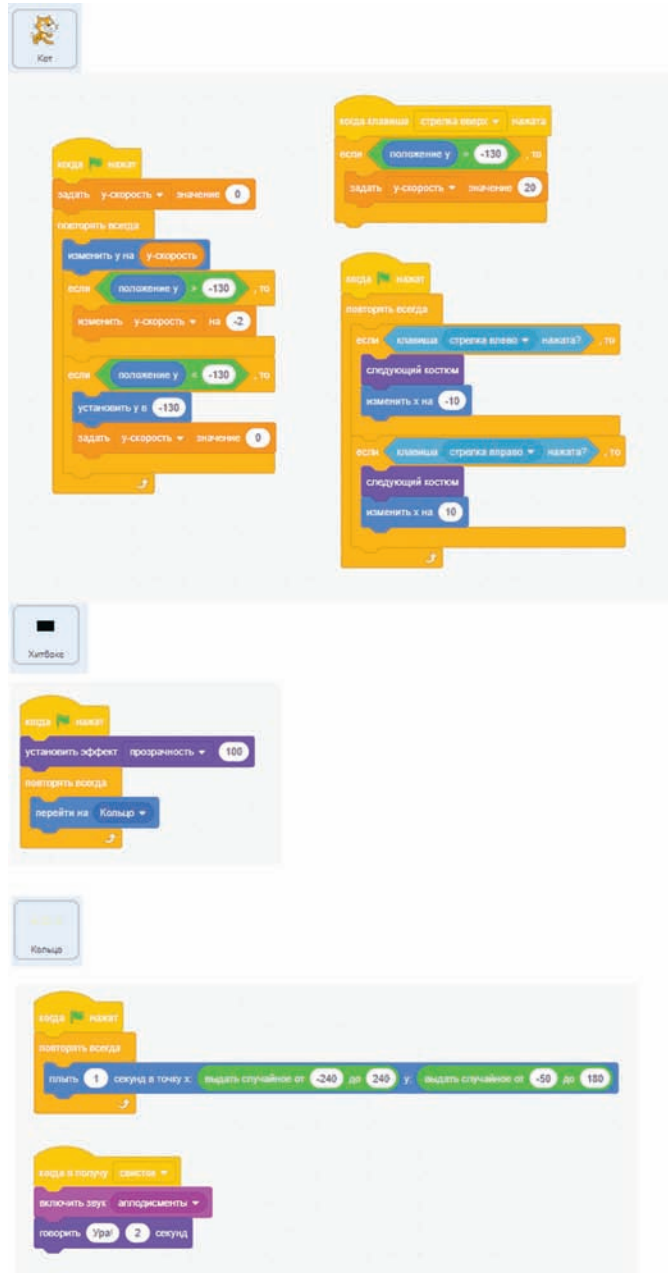
Когда спрайт **Баскетбол** впервые обнаруживает удачный бросок, он увеличивает значение переменной **Счет игрока 1** на 1 и присваивает переменной **Попадание** значение **Да**. При последующих проверках этого броска значение переменной **Попадание** не будет равно **Нет**, поэтому бросок больше не будет 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.

ГОТОВАЯ ПРОГРАММА

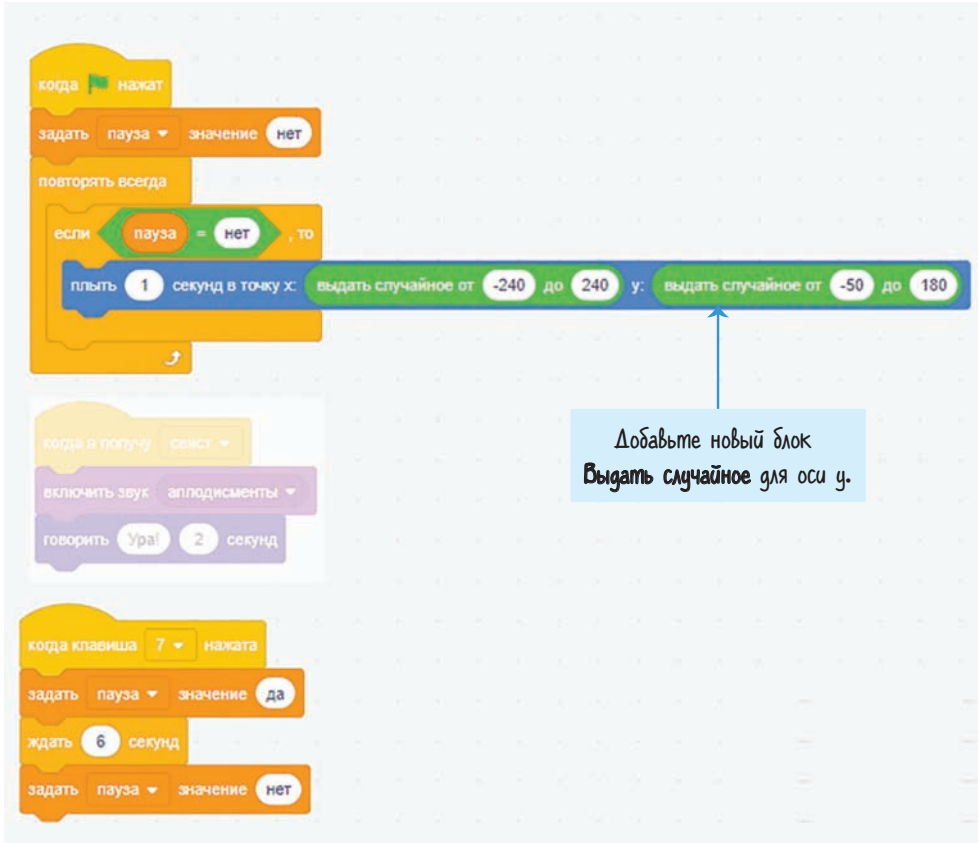
Ниже показана финальная версия кода. Если ваша программа не работает правильно, сверьте ее код с приведенным ниже.



ЧИТ-РЕЖИМ: ОСТАНОВКА КОЛЬЦА

Движущееся баскетбольное кольцо — довольно сложная 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.



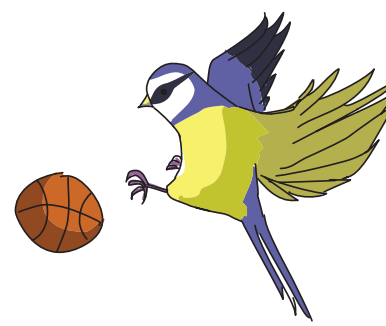
КОНТРОЛЬНАЯ ТОЧКА

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.

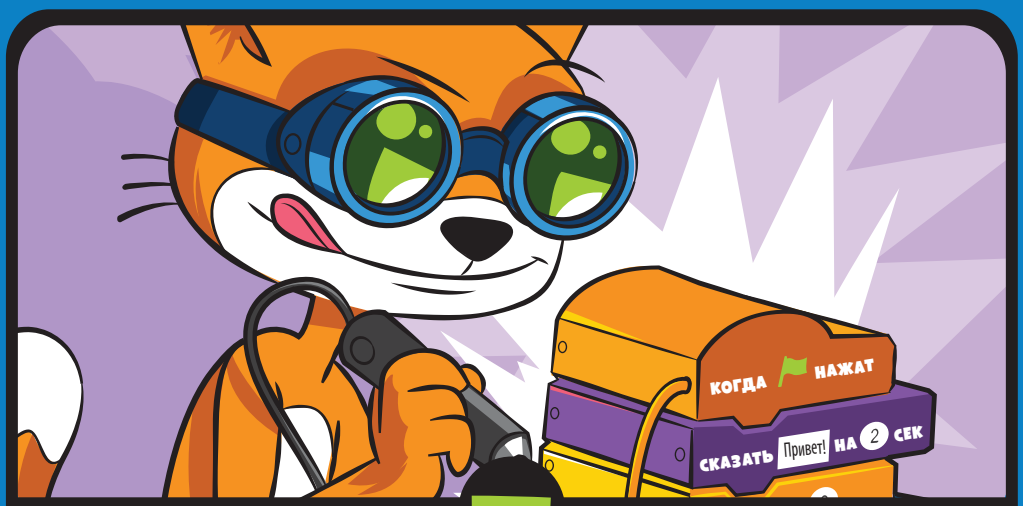


Использовать силу тяжести в этой программе было довольно просто. К тому времени, когда вы доберетесь до главы 7, вы сможете создать продвинутую игру-платформер с более сложными прыжками и падениями. Но для начала потребуется еще попрактиковаться со Scratch. В главе 5 вы создадите игру с боковым режимом просмотра и используете функцию клонирования, чтобы дублировать спрайты десятки раз.

ОБЗОРНЫЕ ВОПРОСЫ

Ответьте 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. Чем игра с боковым режимом просмотра (например, «Баскетбол») отличается от игры с видом сверху (например, игры «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?

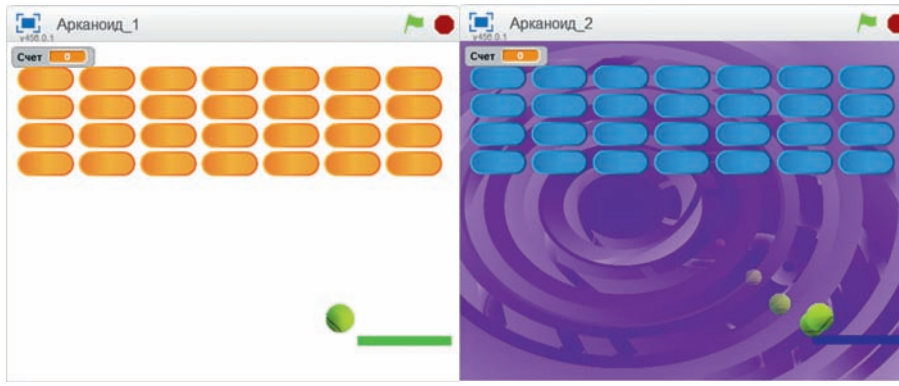


5 ПРОДВИНУТЫЙ АРКАНОИД

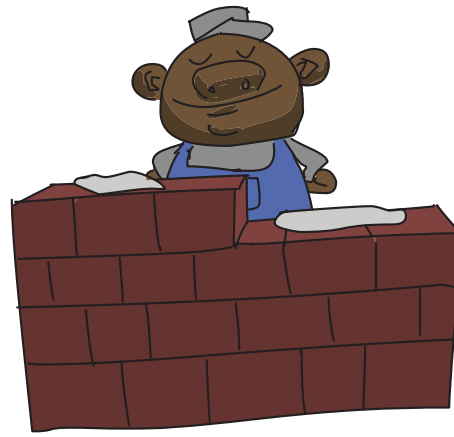
Вы когда-нибудь видели классическую игру «Арканоид»? Игрок управляет платформой-ракеткой в нижней части экрана, отбивая шарик, который разбивает кирпичики в верхней части экрана. Игрок проигрывает, когда шарик пролетает мимо ракетки. Хотя эта игра достаточно проста для программирования, она быстро может 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 **https://scratch.mit.edu/projects/741003198**.



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 проекта, который называется *Глава 05/Проект.sb3* и находится в запакованном файле с примерами, который вы скачали ранее по ссылке **http://addons.eksmo.ru/it/Scratch_Sweigart.zip**. В файл проекта уже загружены все спрайты, так что вам нужно всего лишь перетащить блоки кода в каждый спрайт.

А. СОЗДАНИЕ ПЛАТФОРМЫ-РАКЕТКИ, ПЕРЕМЕЩАЕМОЙ ВЛЕВО/ВПРАВО

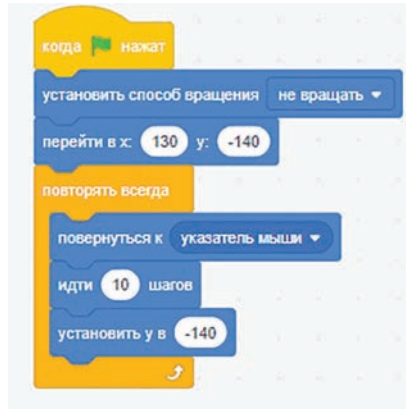
Игрок будет управлять платформой-ракеткой, двигая мышкой. Мяч отскакивает от ракетки к кирпичикам, но игрок проигрывает, если мяч летит мимо ракетки.

1. Создание спрайта платформы

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

Затем нажмите кнопку **Выбрать спрайт** и добавьте из открывшегося окна зеленый спрайт **Paddle**. На панели свойств 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.



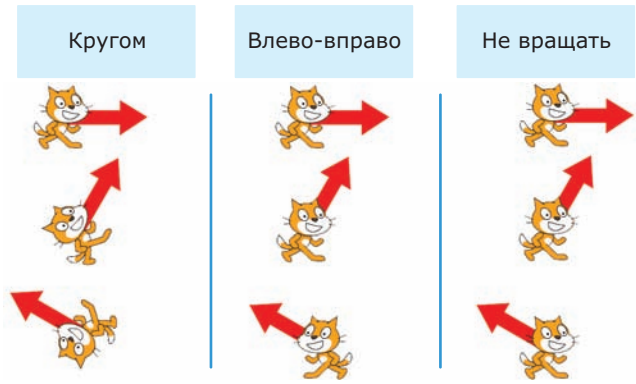
Спрайт **Платформа** постоянно движется на 10 шагов за мышью. При этом его положение по оси *y* остается равным –140.

Горизонтальное движение платформы является результатом двух действий: следования за мышью на 10 шагов и установки положения платформы по оси *y* 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: СПОСОБ ВРАЩЕНИЯ

Способ вращения определяет, как выглядит спрайт, когда меняет направление. Существуют три типа поворотов: **Кругом**, **Влево-вправо** и **Не вращать**. Вы можете настроить тип поворота с помощью блока **Установить способ вращения** из категории с синими блоками **Движение**.



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

Поскольку мы изменяем направление спрайта **Платформа**, нам необходимо установить способ вращения спрайта с помощью блока **Установить способ вращения**. Спрайт **Платформа** запрограммирован поворачиваться и следовать за мышью, но для нашей игры нужно, чтобы спрайт всегда выглядел плоским и горизонтальным. Поэтому мы выбираем способ 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.

Б. НАСТРОЙКА ОТСКАКИВАНИЯ МЯЧА ОТ СТЕН

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

2. Создание спрайта мячика

Нажмите кнопку **Выбрать спрайт**, чтобы открыть одноименное окно, и выберите спрайт **Tennis Ball**. 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

Повторять всегда спрайт **Мячик** начинает двигаться. Когда спрайт **Мячик** будет касаться края сцены, он будет отскакивать в новом направлении.

КОНТРОЛЬНАЯ ТОЧКА

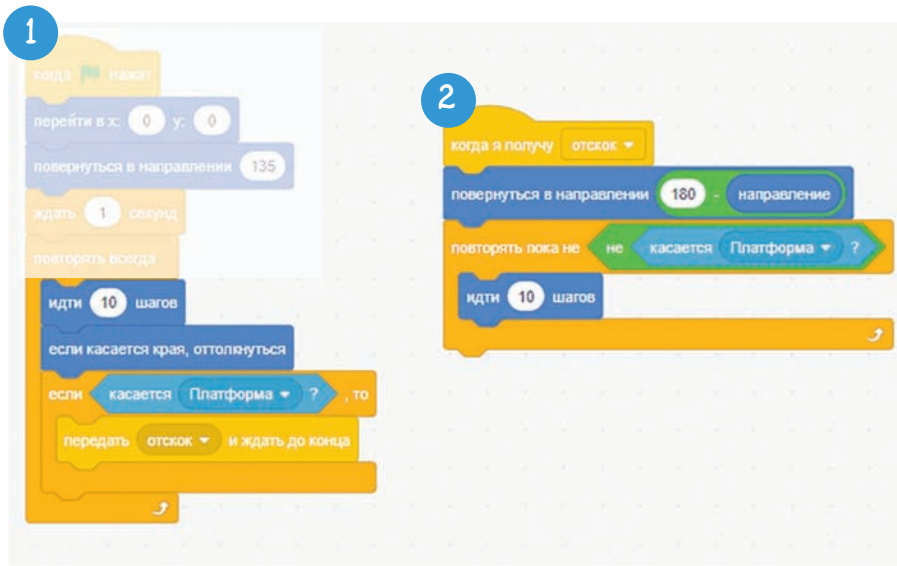
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что спрайт **Мячик** движется по сцене и отскакивает от краев. Он не будет отскакивать от ракетки, поскольку этот код вы еще не добавили. Нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

B. НАСТРОЙКА ОТСКАКИВАНИЯ МЯЧА ОТ РАКЕТКИ

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

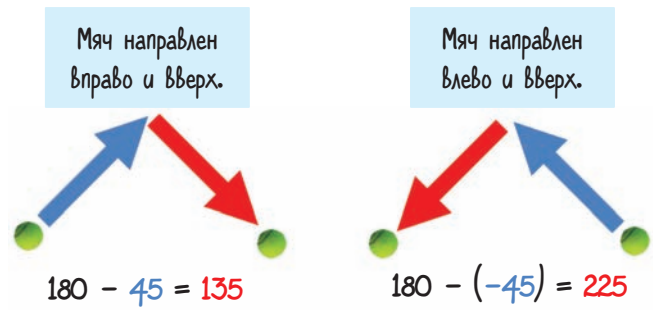
3. Добавление кода отскакивания к спрайту теннисного мяча

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 ❶, чтобы управлять тем, что происходит, когда мяч касается ракетки в скрипте ❷.

Код **Повернуться в направлении 180 — направление** в скрипте ❷ может показаться немного сложным. Но это уравнение лишь вычисляет направление, в котором мяч будет отскакивать, основываясь на текущем направлении движения. 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.

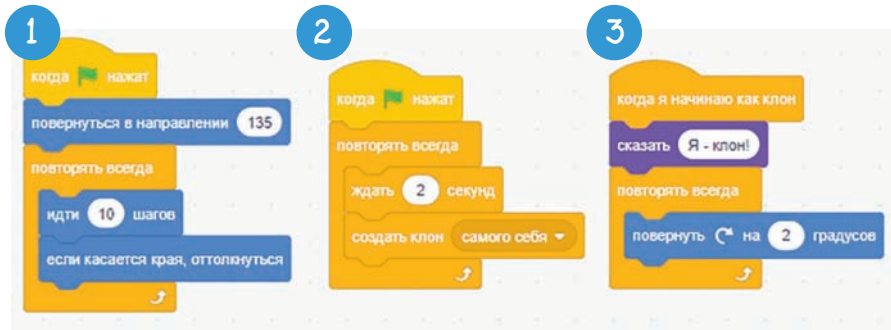
КОНТРОЛЬНАЯ ТОЧКА

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, множество враждебных персонажей, которые выглядят одинаково, груду 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 ❶ программирует спрайт **Кот** отскакивать от границ сцены точно так же, как это делает спрайт **Мячик** в игре «Арканоид». В скрипте ❷ мы создаем клон снова и снова каждые две секунды. В скрипте ❸ используется блок **Когда я начинаю как**

клон, чтобы управлять поведением клонированных спрайтов. Как вы думаете, что произойдет, когда вы запустите этот код? Запустите его сейчас, чтобы узнать, правильны ли ваши предположения.

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

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 **Выбрать спрайт**, чтобы открыть одноименное окно, и выберите спрайт **Button 2**. На панели свойств спрайта присвойте этому спрайту имя **Кирпичик**.

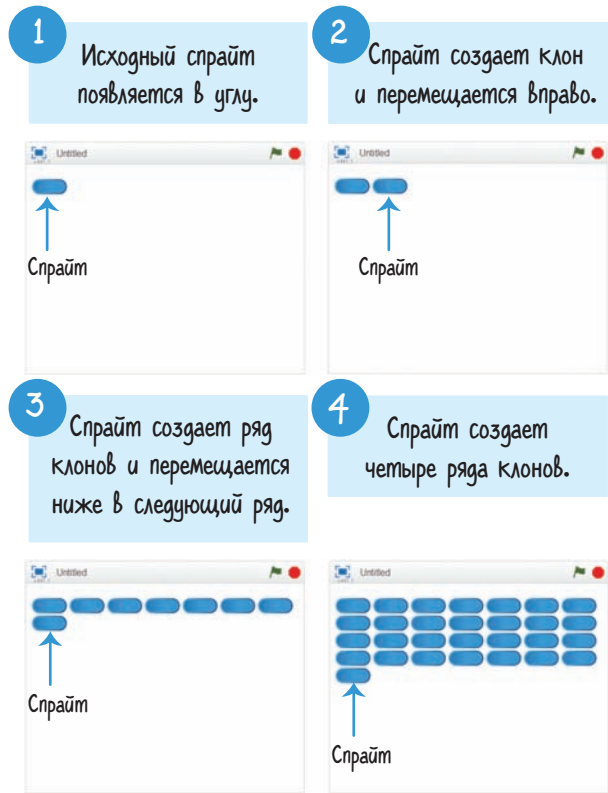
Далее вам нужно создать новую переменную. Выберите оранжевую категорию **Переменные** и нажмите кнопку **Создать переменную**. Присвойте этой переменной имя **Счет** и установите переключатель в положение **Для всех спрайтов**. Затем добавьте код, показанный на следующем рисунке, к спрайту **Кирпичик**.

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**.

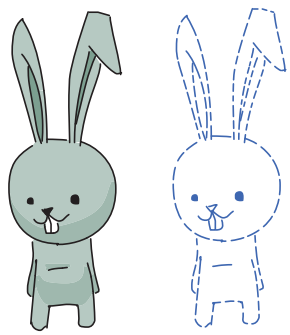
5. Клонирование спрайта Кирпичик

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



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 О. Блок **Повторить 4 раза** повторяет код, который создает ряд из семи кирпичиков, программируя создание четырех рядов клонов 6. Семь клонов кирпичиков, умноженных на четыре ряда, составляют 28 клонов. 29-й кирпичик на предыдущем рисунке — это исходный спрайт, не клон, и позже мы его спрячем.



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.

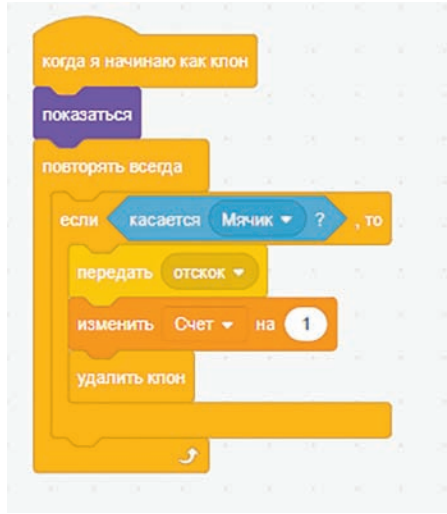
Представьте себе, что вам пришлось бы дублировать спрайты, вместо того чтобы клонировать их. Тогда вам пришлось бы изменить все 28 спрайтов **Кирпичик**, чтобы изменить код. Клонирование экономит много времени!

Д. 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 когда он попадает на ракетку. Программа добавляет 1 очко к счету игрока, а затем клон удаляет сам себя.

КОНТРОЛЬНАЯ ТОЧКА

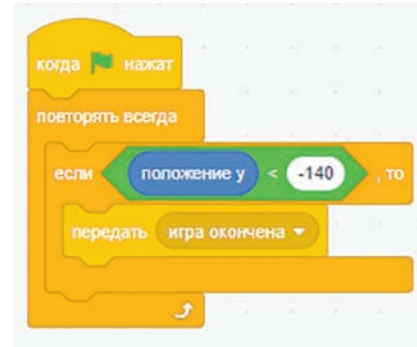
Нажмите кнопку в виде зеленого флага, чтобы проверить уже имеющийся фрагмент кода. Убедитесь, что верхняя часть сцены заполняется клонами кирпичиков и клоны исчезают, когда спрайт **Мячик** ударяется о них и отскакивает. Затем нажмите кнопку в виде красного знака остановки и сохраните вашу программу.

E. 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

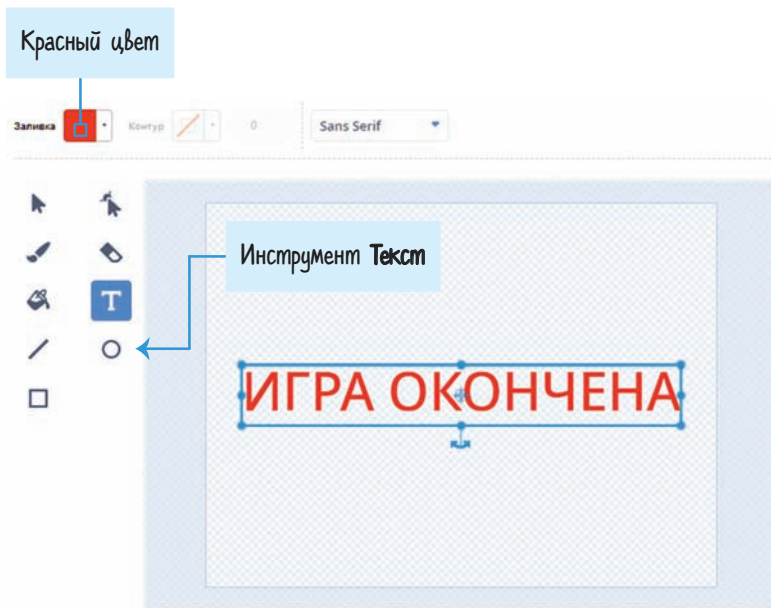
Когда спрайт **Мячик** пролетает мимо спрайта **Платформа** — другими словами, когда значение ***y*-позиции** спрайта **Мячик** составляет менее чем –140, — игра окончена. После этого спрайт **Мячик** должен передать сообщение **Игра окончена**. Добавьте показанный ниже код к спрайту **Мячик**.



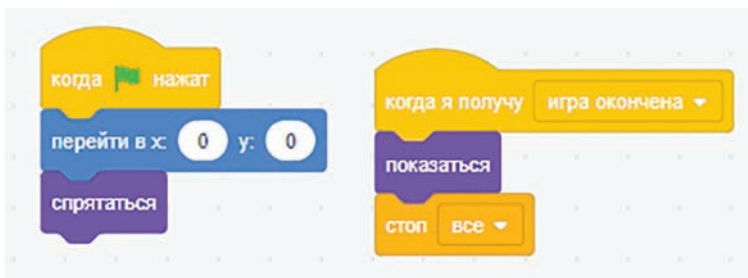
Передача сообщения **Игра окончена** означает, что нужно отобразить спрайт **Игра окончена**. Давайте приступим к следующему спрайту.

8. Создание спрайта Игра окончена

Нажмите кнопку **Нарисовать,** которая прячется в кнопке **Выбрать спрайт** в списке спрайтов. Когда появится графический редактор, воспользуйтесь инструментом **Текст**, чтобы написать текст «Игра окончена!» красным шрифтом.



На панели свойств присвойте этому спрайту имя **Игра окончена**. Затем добавьте к нему код, показанный на следующем рисунке.



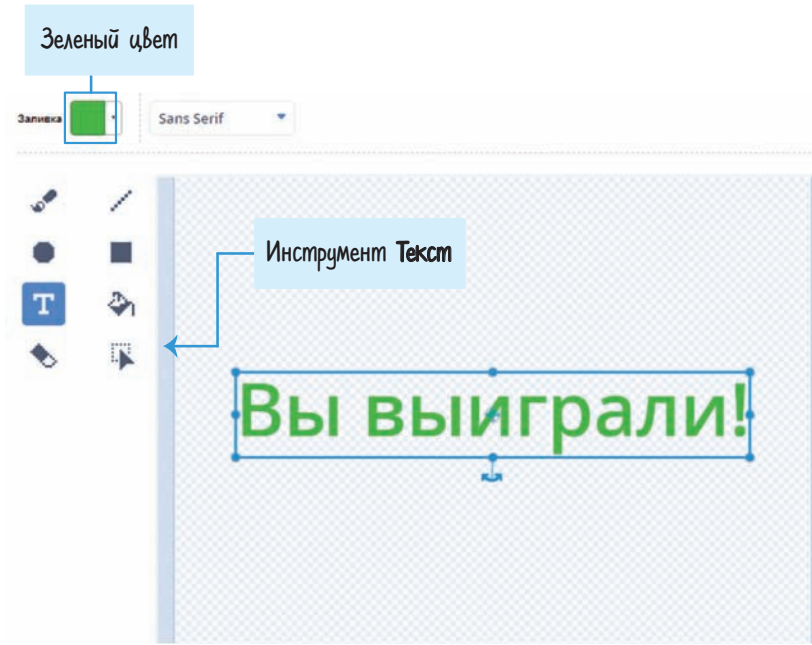
Спрайт остается скрытым до тех пор, пока не получит сообщение **Игра окончена**. После этого блок **Стоп все** останавливает все спрайты.

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. Сохранитесь.

9. Создание спрайта Вы выиграли

Нажмите кнопку **Нарисовать,** которая прячется в кнопке **Выбрать спрайт** в списке спрайтов. 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).



На панели свойств присвойте этому спрайту имя **Вы выиграли**. Добавьте к нему код, показанный на следующем рисунке.

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**.

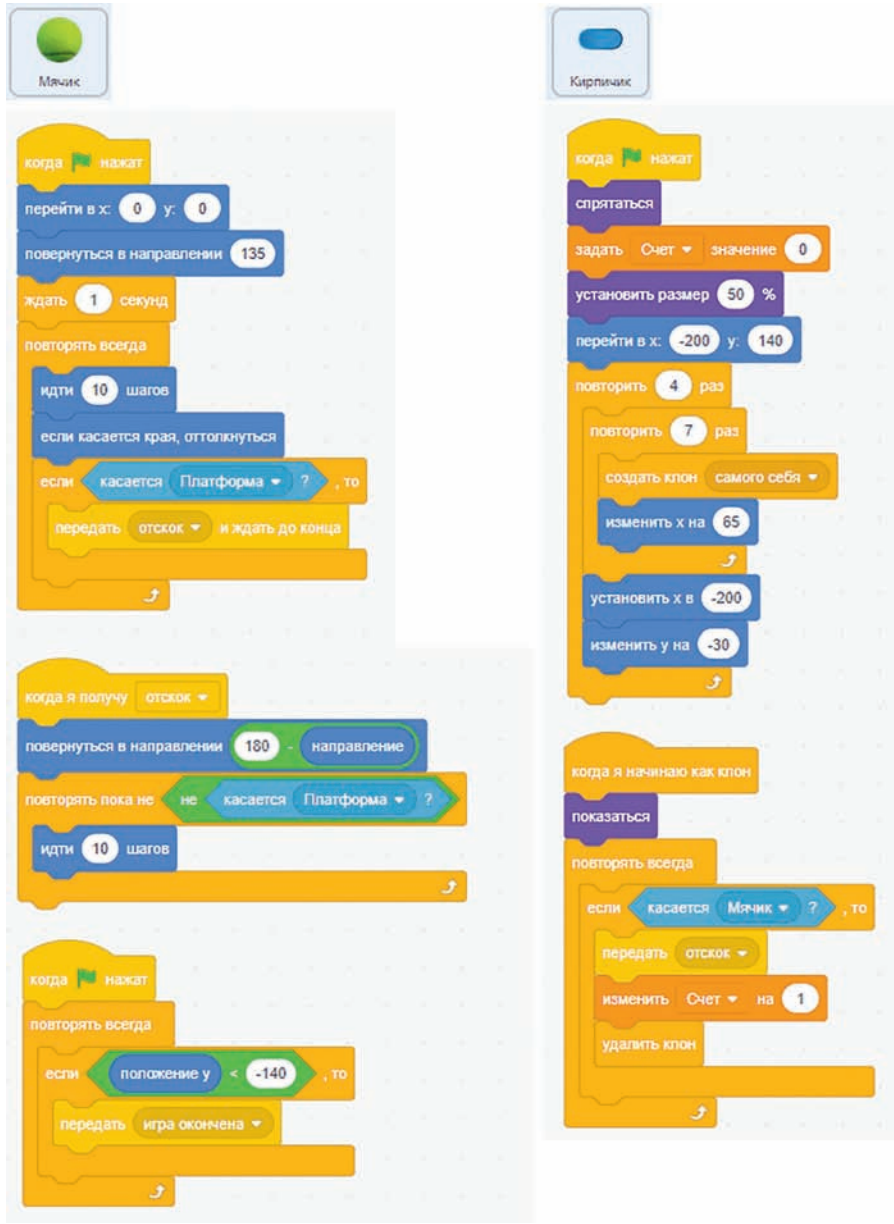
После того как в игре отобразится спрайт **Вы выиграли**, программа прекращает все остальные спрайты с помощью блока **Стоп все**.

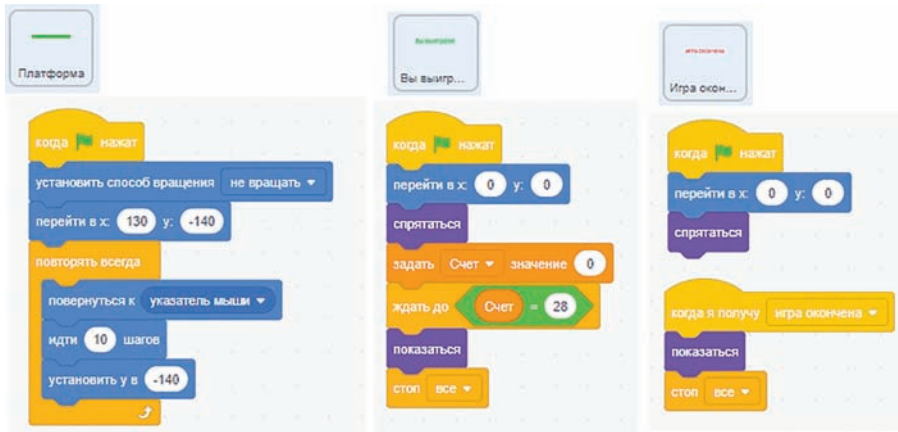
КОНТРОЛЬНАЯ ТОЧКА

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.

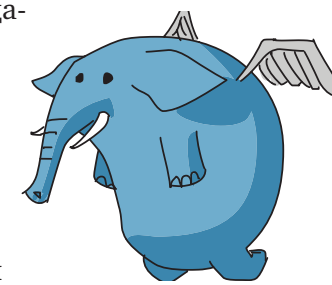
ГОТОВАЯ ПРОГРАММА

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

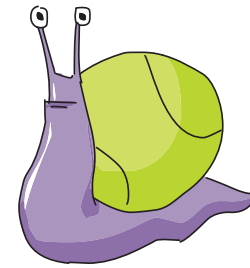




ВЕРСИЯ 2.0: ПРИДАЕМ ИГРЕ ЛОСК



Ваша игра уже отлично работает. Теперь давайте улучшим ее. Множество идей для улучшения игры «Арканоид» пришли из текста «Придай лоск или проиграешь!» Мартина Йонассона и Петри Пурхо. 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 в захватывающую и красочную. Улучшенная игра выглядит профессиональнее, чем простая[2].



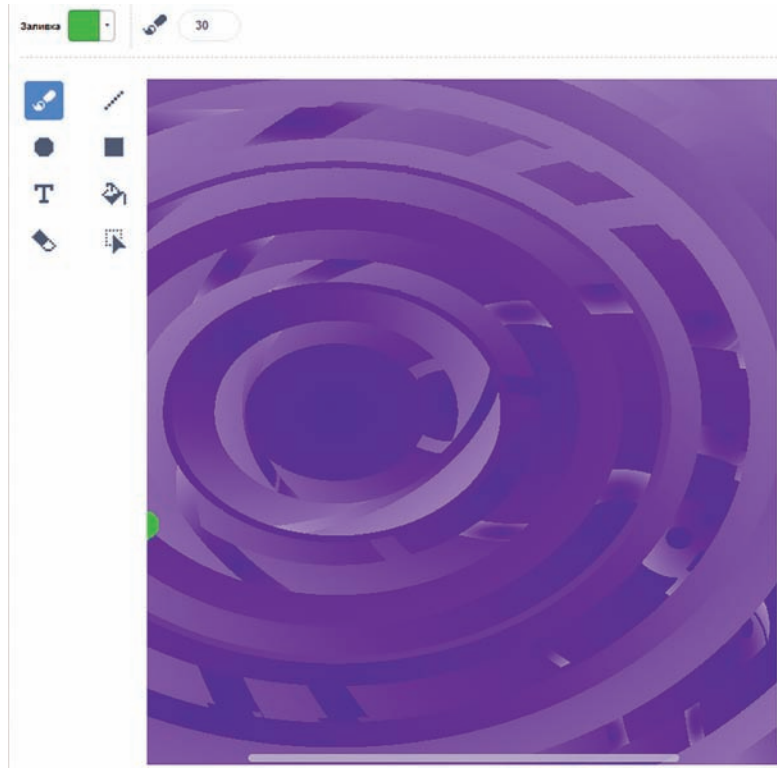
Вы можете добавить много красивых деталей к «Арканоиду», чтобы игра выглядела интересно и привлекательно. А лучше всего то, что вы можете использовать эти приемы улучшения в любой из ваших игр. Перед тем как заняться кодом, взгляните на полную версию игры на сайте **https://scratch.mit.edu/projects/741419764/**.

---

[2] Вы можете посмотреть упомянутую презентацию об улучшении игры «Арканоид», пройдя по ссылке **https://www.gdcvault.com/play/1016487/Juice-It-or-Lose**. — *Прим. ред.*

Создание классного фона

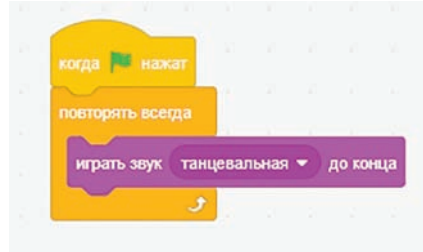
Чтобы игра выглядела круче, можно воспользоваться простым способом — нарисовать интересный фон. Нажмите кнопку **Выбрать фон** и выберите фон **Neon Tunnel**. В графическом редакторе выберите понравившийся цвет и с помощью инструмента **Заливка** залейте некоторые плитки на стенах туннеля. Создайте необычный, но интересный фон. Для удобства переименуйте его в **Неоновый туннель**.



Добавление музыки

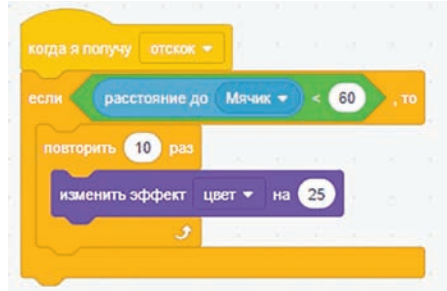
Звуковое сопровождение создает настроение и оживляет игру. Выберите в области спрайтов пункт **Сцена** и перейдите на вкладку **Звуки** в верхней части области блоков. Нажмите кнопку **Выбрать звук** (в виде динамика), чтобы открыть одноименное окно. Выберите звук **Dance celebrate**. Для удобства вы можете переименовать звуковое сопровождение, присвоив имя **Танцевальная**.

Затем перейдите на вкладку **Скрипты** и добавьте код, показанный на следующем рисунке, в область 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**.



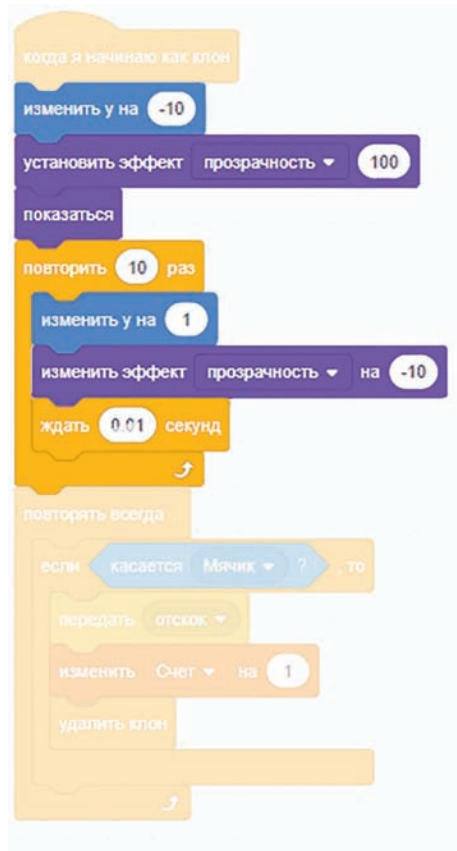
Сообщение **Отскок** передается, когда спрайт **Мячик** отскакивает от клона спрайта **Кирпичик** или от спрайта **Платформа**. Блок **Если расстояние до Мячика < 60, то** отвечает за изменение цвета платформы, когда спрайт **Мячик** отскакивает от спрайта **Платформа** (и означает, что мяч будет менее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

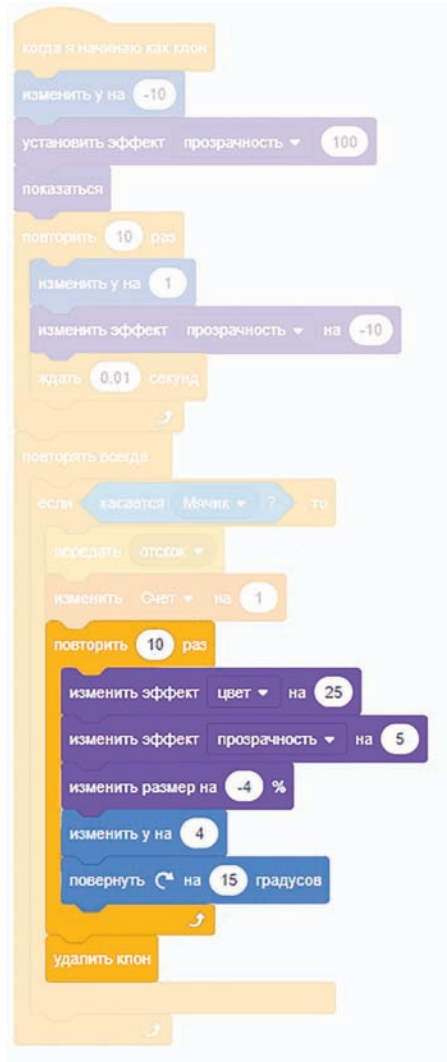
Клоны кирпичиков в игре появляются довольно скучно. Они просто становятся видны, как только их клонировали. Для того чтобы анимировать их появление, измените код спрайта **Кирпичик**, чтобы он соответствовал рисунку ниже.

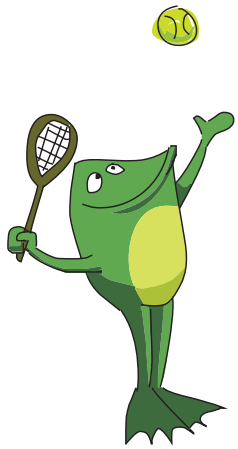


Присвойте блоку **Установить эффект: прозрачность 100**, а затем постепенно снижайте его. Благодаря этому эффекту станет возможным, чтобы клоны кирпичиков медленно исчезали, а затем мгновенно появлялись. Также этот код устанавливает появление клонов кирпичиков на 10 шагов ниже их конечного положения и медленно поднимает их, изменяя их положение по оси *y* 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, а не мгновенным, как раньше.

Теперь исчезновение клонов кирпичиков стало более захватывающим. Блоки **Изменить эффект на** внутри цикла **Повторить** сделают так, чтобы клоны кирпичиков мигали разными цветами и 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 *y* на 4** поднимает их вверх, а блок **Повернуть ↻ на 15 градусов** поворачивает их. Такое анимированное исчезновение выглядит очень привлекательно.

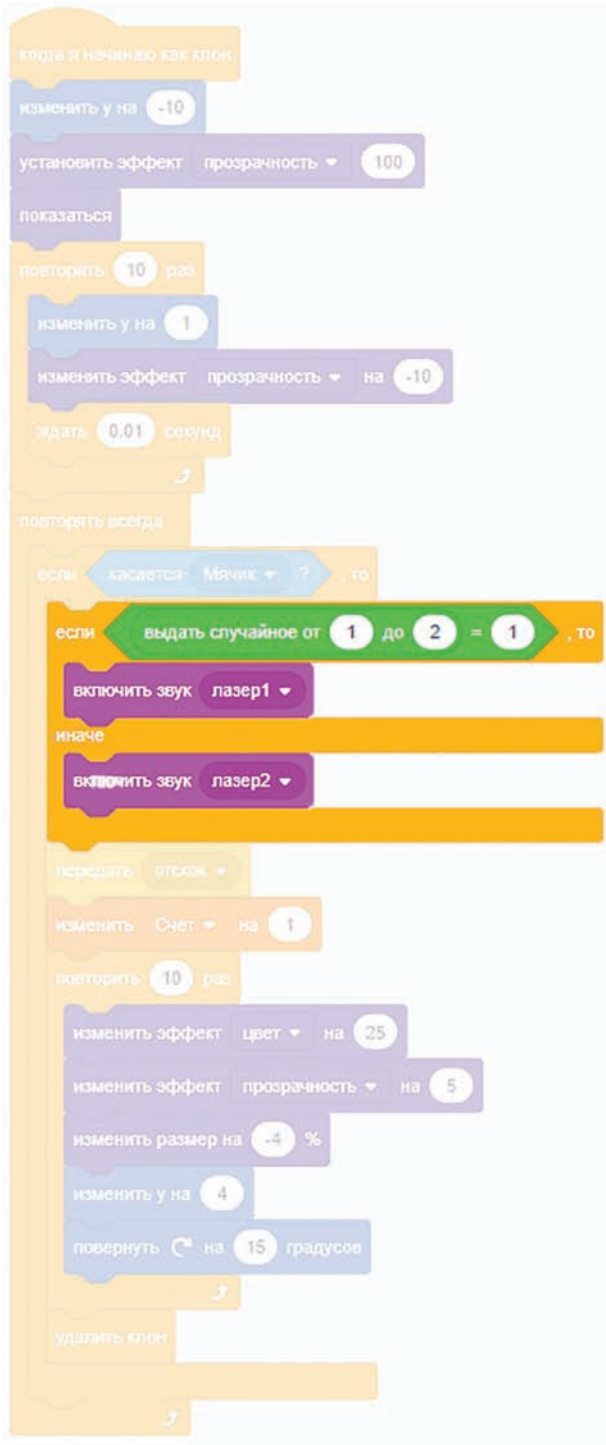
КОНТРОЛЬНАЯ ТОЧКА

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 исчезают из поля зрения медленно, а не мгновенно. 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, когда они исчезают. Выберите спрайт **Кирпичик** в области спрайтов, а затем перейдите на вкладку **Звуки** над областью блоков. Нажмите кнопку **Выбрать звук**, чтобы открыть одноименное окно, и выберите звук **Laser1**. Повторите этот шаг, чтобы добавить звук **Laser2**. Для удобства вы можете переименовать их в **Лазер1** и **Лазер2.**

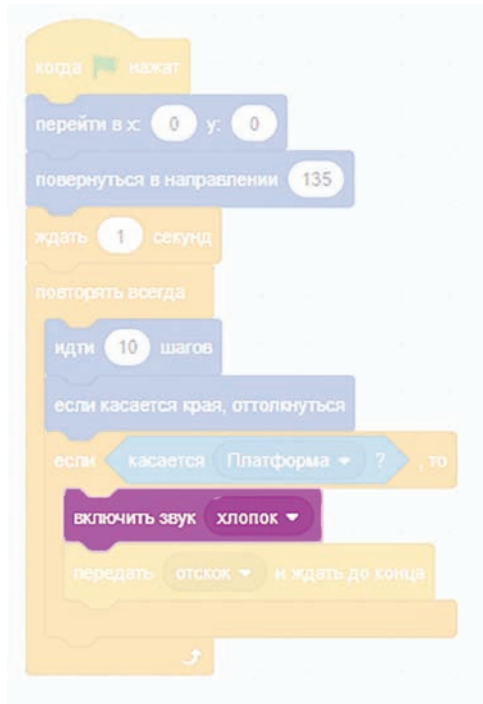
Измените код спрайта **Кирпичик** в соответствии с приведенным ниже рисунком.


когда я начинаю как клон
изменить y на -10
установить эффект прозрачность 100
показаться
повторить 10 раз
изменить y на 1
изменить эффект прозрачность на -10
ждать 0.01 секунд
повторять всегда
если касается Мячик ? , то
если выдать случайное от 1 до 2 = 1 , то
включить звук лазер1
иначе
включить звук лазер2
передать отскок
изменить Счет на 1
повторить 10 раз
изменить эффект цвет на 25
изменить эффект прозрачность на 5
изменить размер на -4 %
изменить y на 4
повернуть на 15 градусов
удалить клон

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 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 этому рисунку.



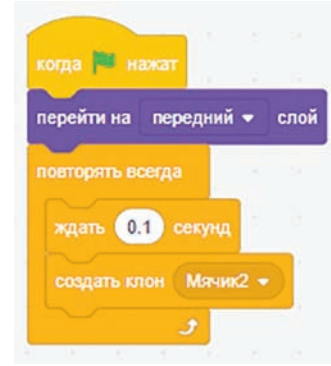
Добавление хвоста к мячику

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**: он будет

реагировать на сообщение **Отскок** всякий раз, когда исходный спрайт **Мячик** совершает отскок. Вместо этого нажмите кнопку **Выбрать спрайт**. В открывшемся окне выберите спрайт **Tennis Ball** для создания другого спрайта под названием **Мячик 2**. Клонируйте этот второй мяч, чтобы создать след. В отличие от клона спрайта **Мячик**, клон спрайта **Мячик 2** не содержит блок **Когда я получаю отскок**. Добавьте код, показанный на следующем рисунке, к спрайту **Мячик 2**.

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.

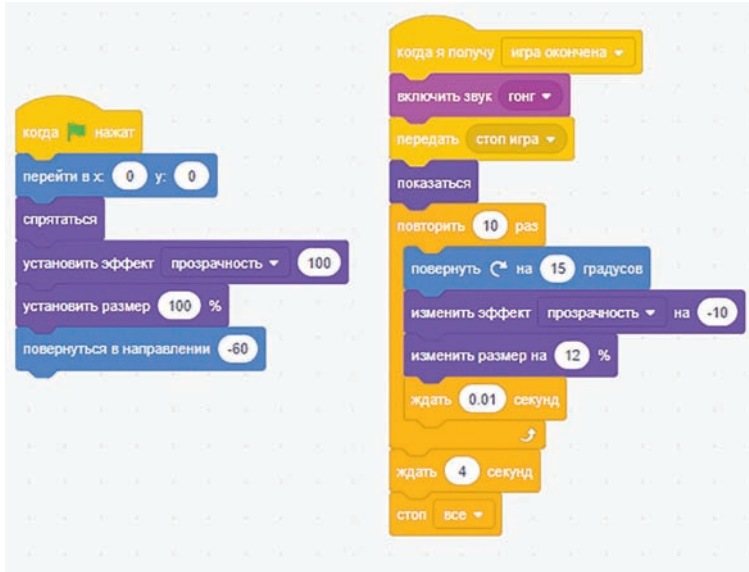
Этот скрипт создает новый клон спрайта **Мячик 2** через 0,1 секунды ожидания, что создает след из мячиков.

КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить код, который готов к этому моменту. Убедитесь, что след из уменьшающихся клонов спрайта **Мячик 2** 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!» выглядело бы интереснее, как и анимированное появление клонов кирпичиков. Измените код спрайта **Игра окончена**, чтобы он соответствовал коду на рисунке ниже. Сначала загрузите звуковой эффект **Gong**, нажав кнопку **Выбрать звук** 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 прозрачности 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 прозрачности до –10. За это отвечает код анимации, находящийся внутри блока **Повторить 10 раз**. Блоки **Повернуть ↻ на 15 градусов** и **Изменить размер на 12%** поворачивают и увеличивают текст. После паузы в четыре 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.

Блок **Стоп другие скрипты спрайта** вместо блока **Стоп все** нужно использовать потому, 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 продолжат 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 с анимацией, 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 приведенным на следующем рисунке.

Вам нужно будет загрузить звуковой эффект под названием **Gong**, нажав кнопку **Выбрать звук** на вкладке **Звуки**. Для удобства вы можете переименовать звук в **Гонг**.

В приведенном коде имеются два набора анимации. Один находится в блоке **Повторить 10 раз**, а другой в блоке **Повторить 2 раза**. Код в блоке **Повторить 10 раз** отвечает за появление на экране спрайта **Вы выиграли**, увеличивает его и перемещает вверх. После этой короткой анимации блок кода **Повторить 2 раза** увеличивает яркость спрайта до 50, ждет 0,1 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!», созданные 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 интересными. Но это лучше делать после того, как вы убедитесь, что 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 игр. 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

Ответьте 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?**

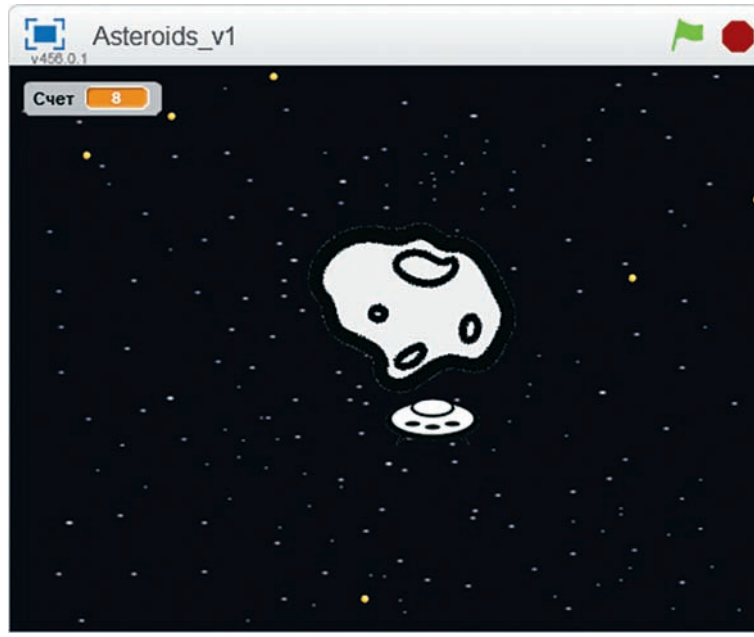
6

УНИЧТОЖИТЕЛЬ АСТЕРОИДОВ... В КОСМОСЕ!

Asteroids — классическая игра, разработанная в 1979 году компанией Atari. С тех пор многие программисты переделывали эту игру, ставшую 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!» в названии игры делает ее привлекательнее.)

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.

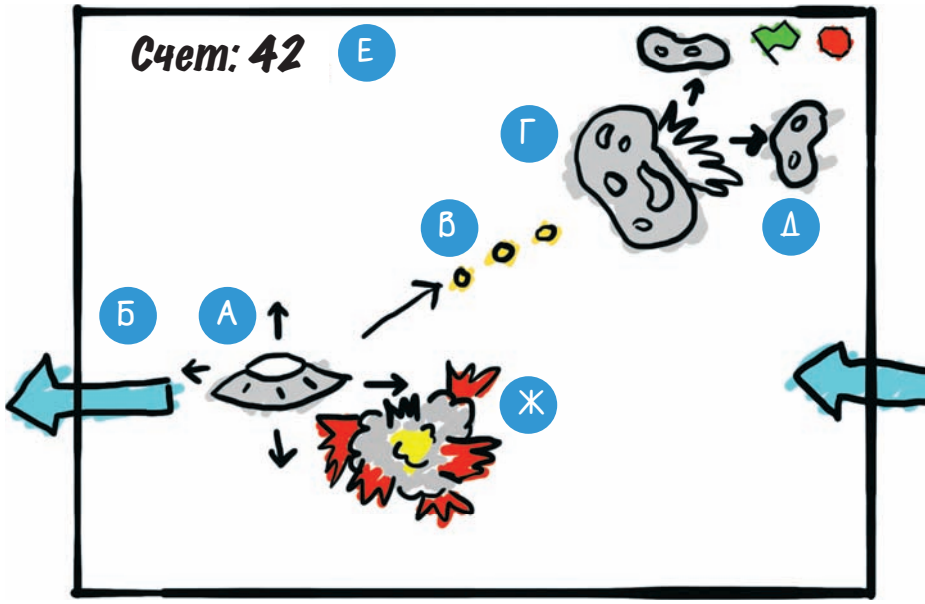
Прежде чем приступить к написанию кода, посмотрите окончательную версию игры на сайте **https://scratch.mit.edu/projects/741003023**.



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 летающих 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 проекта с именем *Глава 08/Проект.sb3*, находящегося в запакованном файле с примерами, который вы скачали ранее по ссылке **http://addons.eksmo.ru/it/Scratch_Sweigart.zip**. В файл проекта уже загружены все спрайты; вам нужно всего лишь перетащить 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 назовите его **Астероиды**. Нажмите кнопку **Выбрать фон**, выберите фон **Stars** и нажмите кнопку **ОК**. Для удобства вы можете переименовать фон в **Звезды**.

Мы не будем использовать спрайт кота, который помещен изначально в каждый проект Scratch. Прежде чем продолжить, щелкните правой кнопкой мыши по этому спрайту и выберите в контекстном меню команду **Удалить** (либо щелкните мышью по значку корзины на спрайте).

1. Создание спрайта Космолет

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

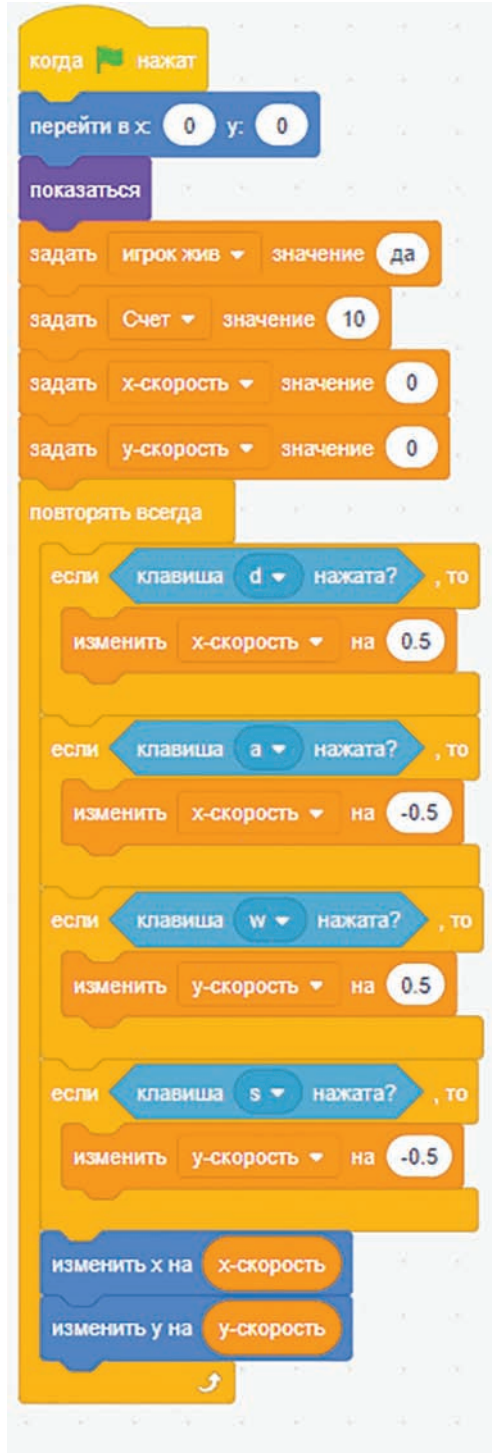
В редакторе Scratch выберите пункт **Загрузить спрайт**, который скрывается в кнопке **Выбрать спрайт**. Затем выберите изображение *космолет.png* в папке с примерами к книге.

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

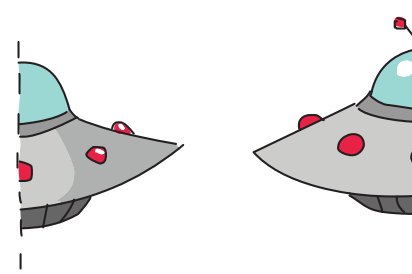

когда нажат
перейти в x: 0 y: 0
показаться
задать игрок жив значение да
задать Счет значение 10
задать x-скорость значение 0
задать y-скорость значение 0
повторять всегда
если клавиша d нажата? , то
изменить x-скорость на 0.5
если клавиша a нажата? , то
изменить x-скорость на -0.5
если клавиша w нажата? , то
изменить y-скорость на 0.5
если клавиша s нажата? , то
изменить y-скорость на -0.5
изменить x на x-скорость
изменить y на y-скорость

Вы можете спросить, 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 ***x*-скорость** и ***y*-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 перемещают космолет в правильном направлении. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

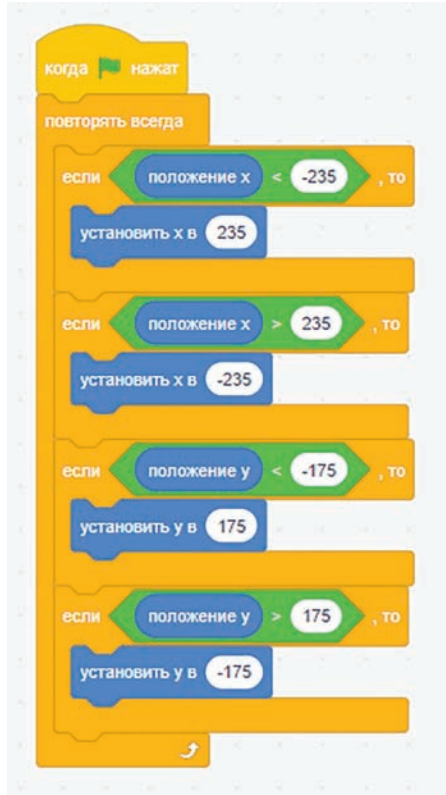
Б. ВЫХОД КОСМОЛЕТА ЗА КРАЯ СЦЕНЫ



Проверяя код, заметили ли вы, что спрайт **Космолет** мгновенно останавливается, когда врезается 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 пяти 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 или на сцену.

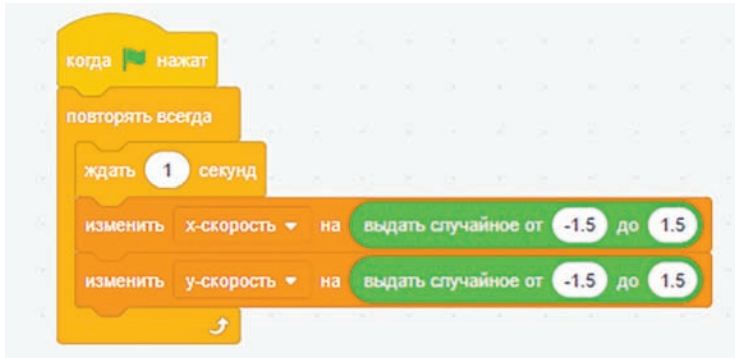
КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить 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 **Повторять всегда** значения переменных ***x*-скорость** и ***y*-скорость** случайным образом немного изменяются

после паузы в одну секунду. Это значит, 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.

B. 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

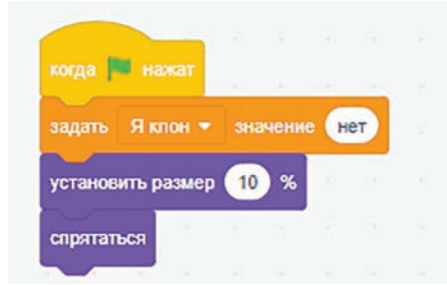
В библиотеке Scratch есть спрайт, который мы можем использовать для создания в своей программе мощных шаров бластера. Нажмите кнопку **Выбрать спрайт**. Выберите спрайт **Ball** и нажмите **ОК**. На панели свойств присвойте спрайту имя **Шар бластера**.

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**, чтобы открыть одноименное окно. Выберите звук **Laser 1**. Для удобства вы можете переименовать звуковой фрагмент в **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 **Переменные** нажмите кнопку **Создать переменную**. Присвойте переменной 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**.

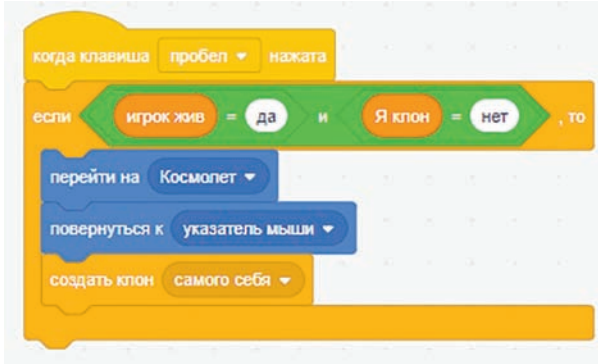


В начале игры исходный спрайт скрывается и остается скрытым всю игру. На сцене появляются только его клоны. Кроме того, спрайт, который мы используем, слишком велик для игры. Поэтому установите его размер равным 10 процентам от исходного, чтобы сделать его меньше.

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**, работает 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 сам себя и исчезает со сцены.

КОНТРОЛЬНАЯ ТОЧКА

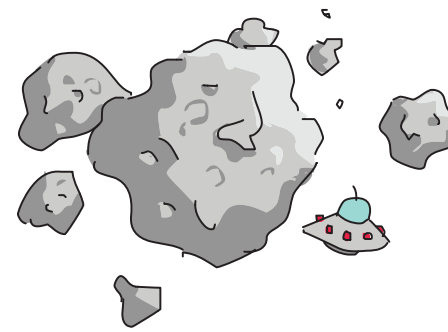
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Прицельтесь с помощью мыши и нажмите клавишу **Пробел**. Убедитесь, что клоны спрайта **Шар бластера** появляются у спрайта **Космолет** и двигаются по направлению к мыши. Клоны должны выходить за пределы сцены и в конечном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

Г. СОЗДАНИЕ ЛЕТАЮЩИХ 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

В редакторе Scratch выберите пункт **Загрузить спрайт**, который скрывается в кнопке **Выбрать спрайт**, и выберите файл *Астероид.png*. Этот файл находится в папке с примерами.

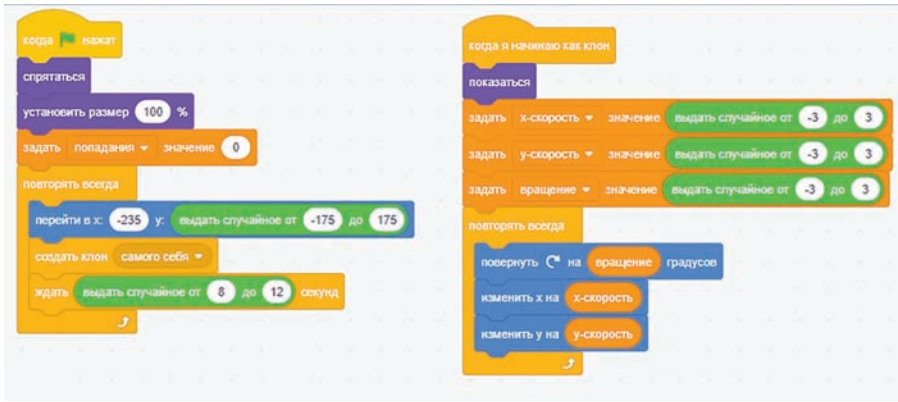


Выберите оранжевую категорию **Переменные**, а затем нажмите кнопку **Создать**

переменную. Установите переключатель в положение **Только для этого спрайта** и присвойте переменной имя **Попадания**. Повторите эти шаги, чтобы создать переменные с именами ***x*-скорость**, ***y*-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.

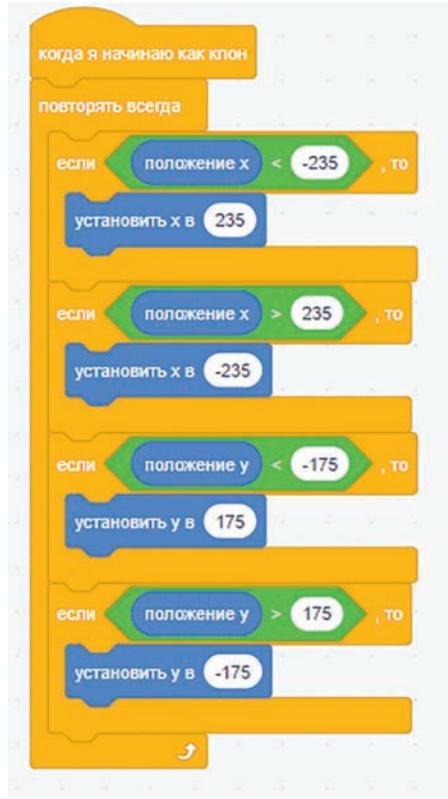
Давайте напишем код, с помощью которого спрайт **Астероид** будет создавать новые клоны. Каждый клон будет иметь случайную скорость и вращение. Результатом будет непредсказуемый рой астероидов… в космосе!

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 отображается, получает случайную скорость и вращение и начинает движение.

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

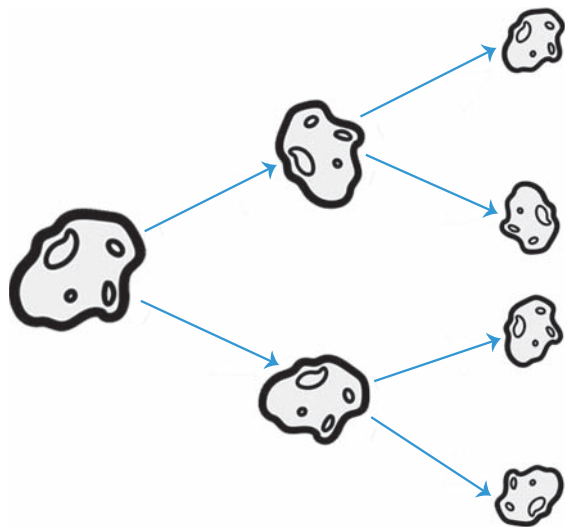


КОНТРОЛЬНАЯ ТОЧКА

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.

Д. Создание астероидов, раскалывающихся надвое при попадании

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



6. Добавление кода раскалывания астероида

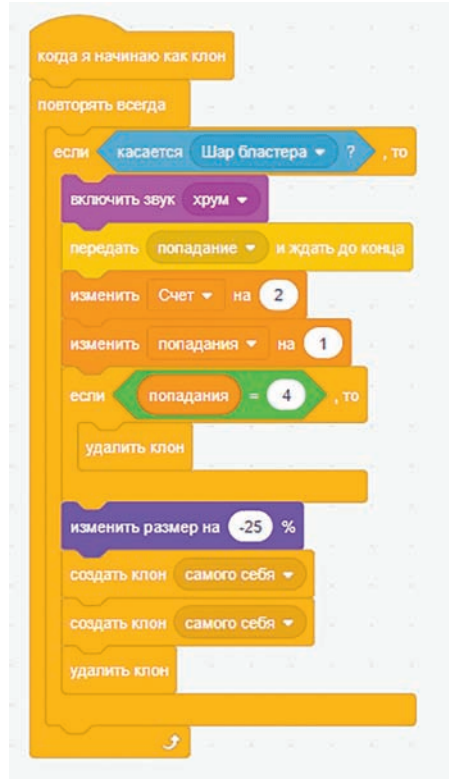
На вкладке **Звуки** нажмите кнопку **Выбрать звук**, чтобы открыть одноименное окно. Выберите звук **Chomp**. Для удобства вы можете переименовать звуковой файл в **Хрум**. Звук **Хрум** будет воспроизводиться всякий раз, когда клон спрайта **Шар бластера** попадает в астероид.

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

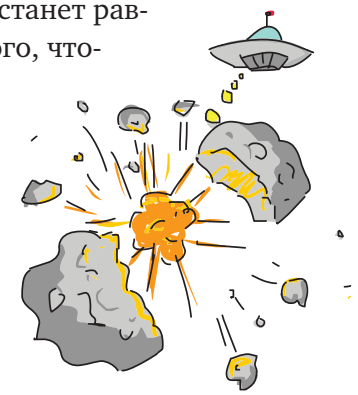
Когда клон спрайта **Шар бластера** попадет в клон спрайта **Астероид**, последний воспроизведет звуковой эффект **Хрум** и передаст сообщение **Попадание**.

Затем **Астероид** прибавит 2 к значению переменной **Счет** и 1 к значению переменной **Попадания**. «Подстреленный» астероид немного уменьшится, изменив размер на –25; когда он («родитель»)

дважды клонирует себя, его «дочерние» клоны будут меньшего размера. Наконец, клон спрайта **Астероид** удалит сам себя.



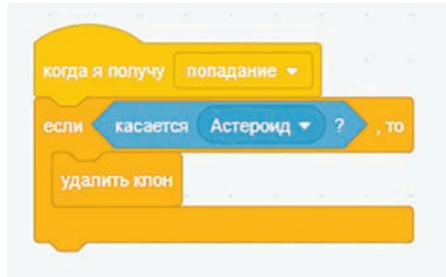
Значения переменной **Попадания** двух маленьких «дочерних» клонов будут на единицу больше, чем значение переменной «родителя». Когда астероид получит четвертое попадание, а значение его переменной **Попадания** станет равно 4, код уничтожит этот клон вместо того, чтобы создать два новых. (Код после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

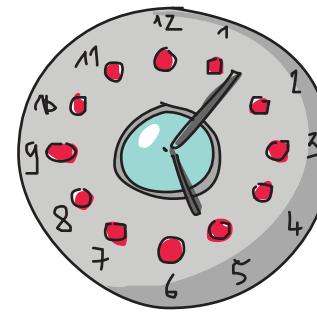


Все клоны спрайта **Шар бластера** получат сообщение **Попадание**, но только те, которые в настоящее время касаются спрайта **Астероид**, будут удалены. (Один достигнет астероида, потому что сообщение передается только с помощью спрайта **Астероид**, когда он касается спрайта **Шар бластера**.) На экране 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.

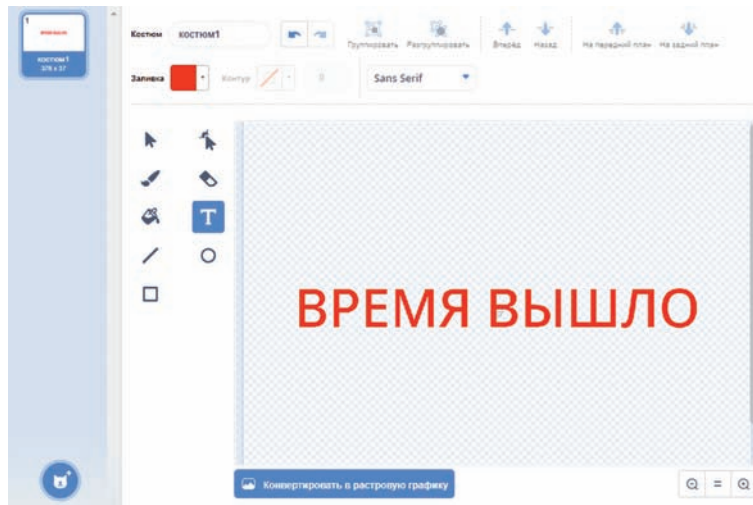
Е. ВЕДЕНИЕ СЧЕТА И СОЗДАНИЕ ТАЙМЕРА



Игра «Уничтожитель астероидов… в космосе!» быстро усложняется, когда астероиды превращаются в тучу крошечных осколков, летающих по всей сцене. Успешная стратегия игры — 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 так, 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,** которая прячется в кнопке **Выбрать спрайт** в списке спрайтов. В графическом редакторе выберите инструмент **Текст** и напишите фразу «ВРЕМЯ ВЫШЛО» красными буквами.



На панели свойств переименуйте спрайт, присвоив ему имя **Время вышло**. Добавьте код, показанный на следующем рисунке, в спрайт **Время вышло**.

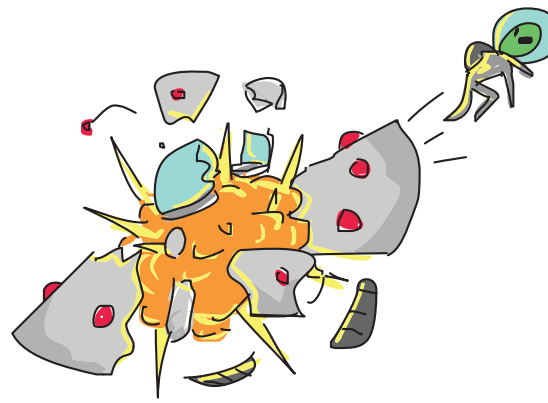
Этот код в начале игры скрывает спрайт **Время вышло** и уменьшает значение переменной **Счет** на единицу после паузы в 1 секунду. Когда значение переменной **Счет** достигает 0, код отображает спрайт **Время вышло**.

КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что значение переменной **Счет** каждую секунду уменьшается на единицу. 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 доступны восемь изображений. Эти костюмы находятся в файле *Взрыв.sprite3* в архивном файле.

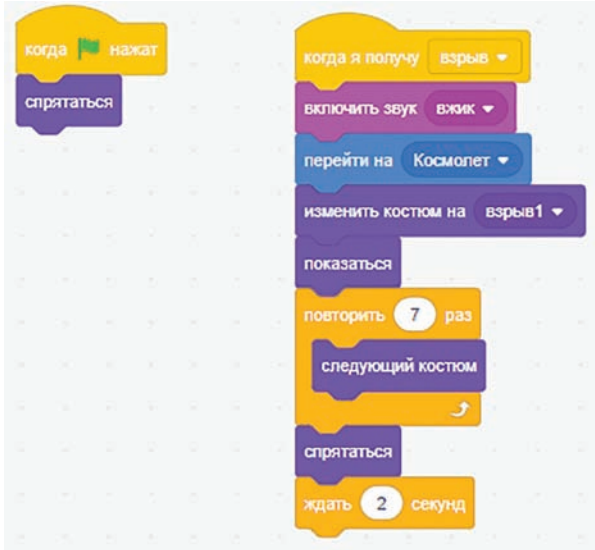
В редакторе Scratch выберите пункт **Загрузить спрайт**, который скрывается в кнопке **Выбрать спрайт**. Выберите файл *Взрыв.sprite3* и нажмите кнопку **OK**. Восемь костюмов для анимации взрыва появятся на вкладке **Костюмы** спрайта **Взрыв**.

10. Добавление кода спрайта Взрыв

Для спрайта **Взрыв** вы создадите новое сообщение — **Взрыв**. Когда спрайт **Взрыв** получит это сообщение, он последовательно покажет свои костюмы и таким образом создаст 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**, чтобы открыть одноименное окно. Выберите звук **Alien creak 2**. Для удобства можно переименовать его во **Вжик**.

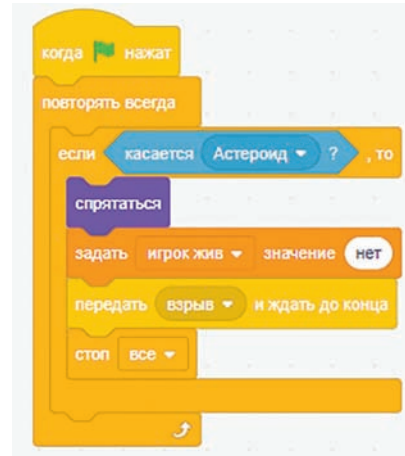
Добавьте код, показанный на следующем рисунке, в спрайт **Взрыв**.



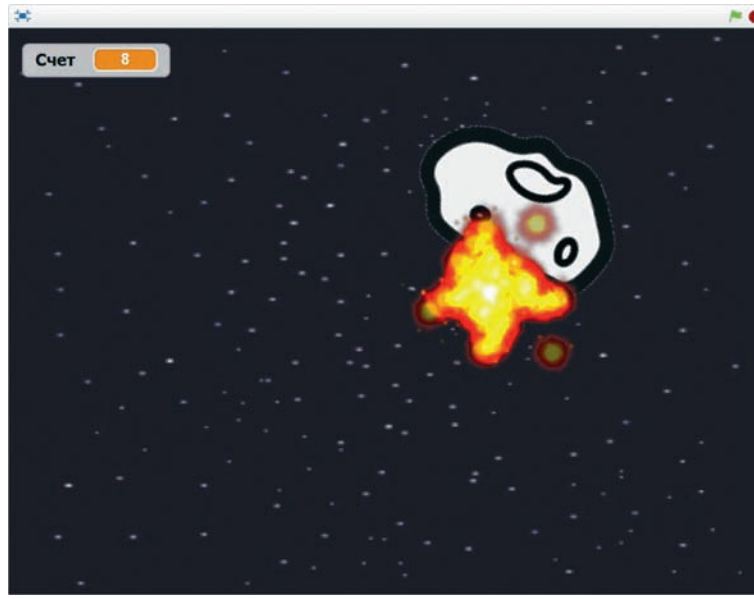
Спрайт **Взрыв** остается скрытым до тех пор, пока он не получит сообщение **Взрыв**. Затем он запускает воспроизведение звука, переходит в положение на сцене, где находится космолет, и переключает костюмы семь раз, чтобы создать зрелищную анимацию взрыва.

11. 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.

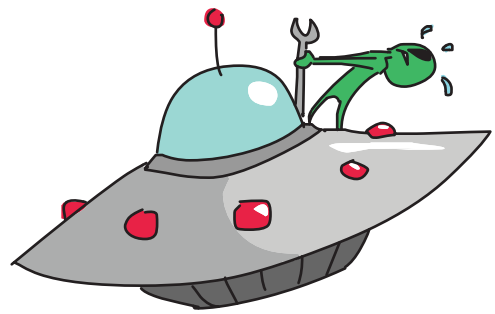


КОНТРОЛЬНАЯ ТОЧКА

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.sb3*.

ВЕРСИЯ 2.0: ОГРАНИЧЕНИЕ БОЕЗАПАСА

Как только вы освоите игру, она может стать слишком легкой для вас. Вот что упрощает игру: 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 **Переменные** и нажмите кнопку **Создать переменную**. Создайте переменную с именем **Энергия** и установите переключатель в положение **Для всех спрайтов**. В области блоков установите флажок **Энергия**, чтобы этот блок появился на сцене.

Значение переменной **Энергия** будет составлять 10 в начале игры. Затем оно 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 ❶ — совершенно новый; он присваивает значение 10 переменной **Энергия**, прежде чем перейти к циклу **Повторять 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 ❷ 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**.

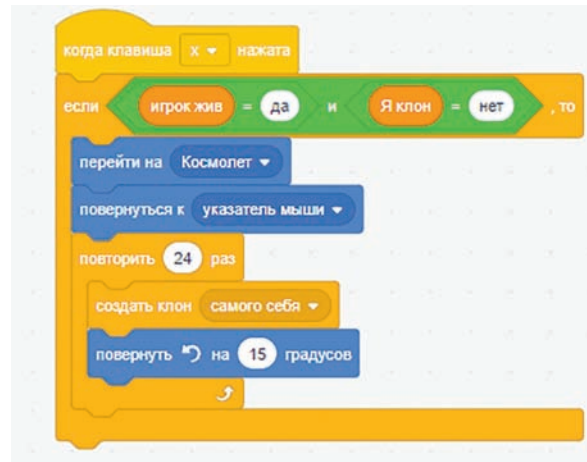
КОНТРОЛЬНАЯ ТОЧКА

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; ее взрыв 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 ***x*-скорость** и ***y*-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. Но 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 **Энергия**, 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 «Супер-Марио» или «Мясной пацан»[3]. В ней будут не только прыжки и гравитация, как в игре «Баскетбол», но и возможность разрабатывать для нее пользовательские уровни, не меняя код.

[3] Super Meat Boy — компьютерная инди-игра в жанре платформера. — *Прим. ред.*

ОБЗОРНЫЕ ВОПРОСЫ

Ответьте 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. 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?

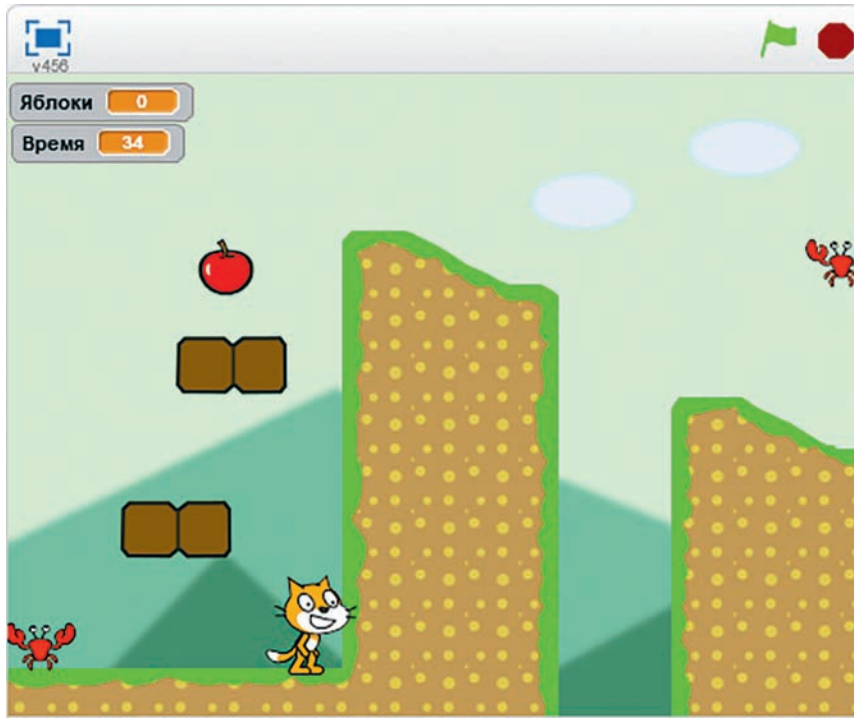
7

ПРОДВИНУТЫЙ ПЛАТФОРМЕР

Первая видеоигра из серии «Супер-Марио» была представлена в 1985 году и стала лучшей игровой франшизой 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*.

В этой главе мы создадим такую игру с помощью Scratch. 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!

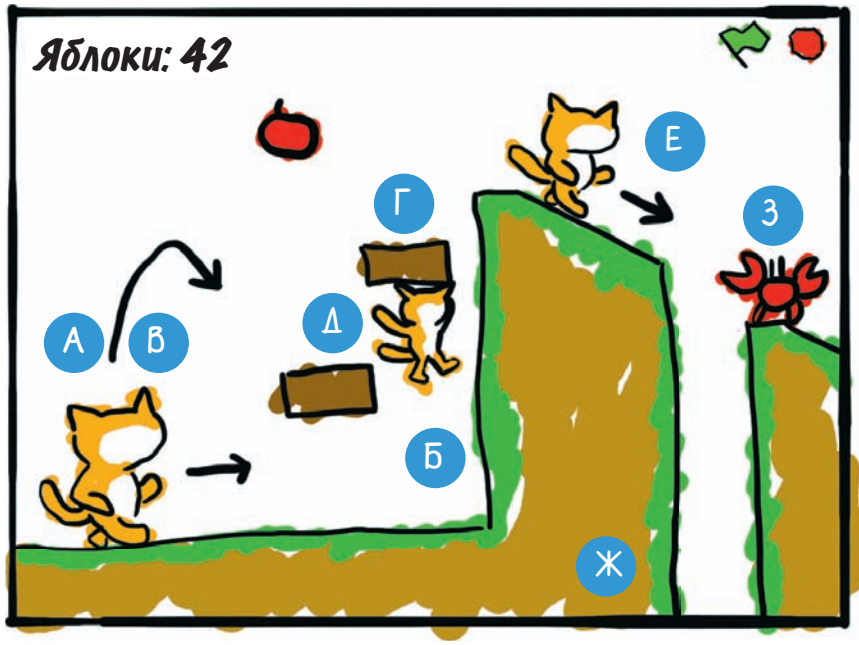
Прежде чем начать писать код игры, посмотрите готовую программу, пройдя по ссылке **https://scratch.mit.edu/projects/741003143**.



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 и собирает яблоки, которые появляются случайным образом. Крабы ходят и прыгают по платформам в случайном порядке.



Вот что мы будем делать в каждом разделе этой главы:

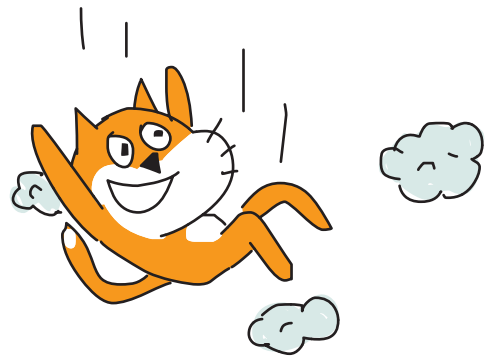
А. Имитация гравитации, падения и приземления.
Б. Использование крутых склонов и стен.
В. Обучение кота высоким и низким прыжкам.
Г. Добавление возможности обнаружения препятствий сверху.
Д. Использование хитбокса для спрайта **Кот**.
Е. Улучшение анимации ходьбы.
Ж. Создание уровня.
З. Добавление крабов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

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

Если вы хотите сэкономить время, вы можете начать с файла проекта с именем *Глава 09/Проект.sb3*, находящегося в запакованном файле с примерами, который вы скачали ранее по ссылке **http://addons.eksmo.ru/it/Scratch_Sweigart.zip**. В файл проекта уже загружены все спрайты; 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.



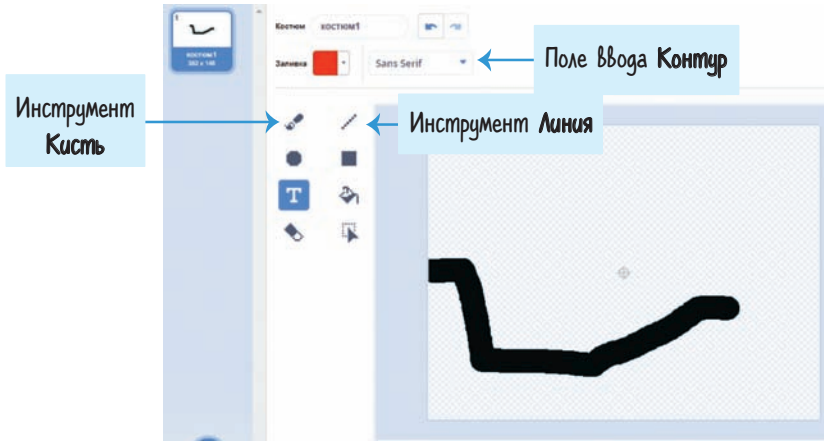
Для начала присвойте проекту имя **Платформер**, указав его в текстовом поле в левом верхнем углу редактора Scratch.

1. Создание спрайта Земля

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

Выберите пункт **Нарисовать,** который прячется в кнопке **Выбрать спрайт** в списке спрайтов, 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 поля ввода **Контур** в верхней части графического редактора. Обязательно нарисуйте пологий склон справа и крутой склон слева.

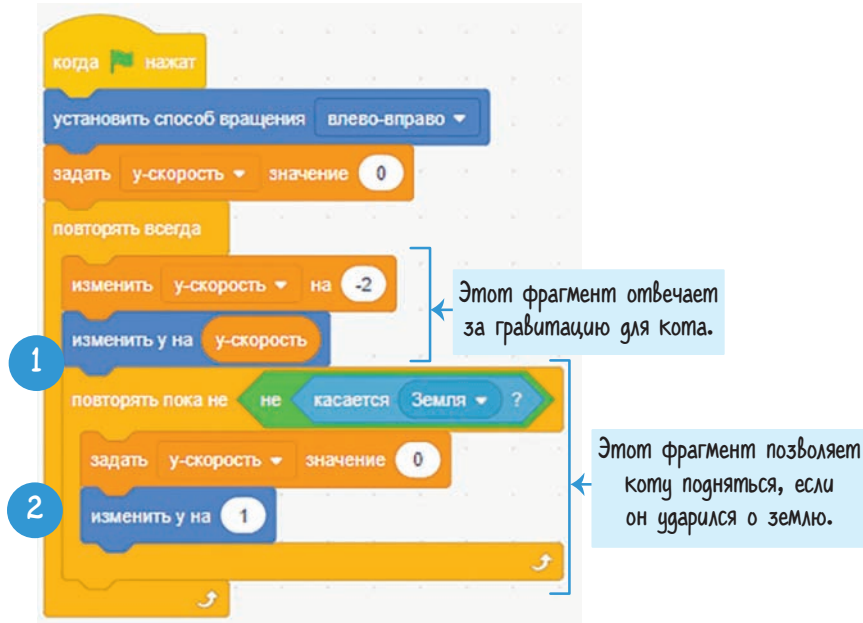
На панели свойств присвойте созданному спрайту имя **Земля**. Кроме того, переименуйте **Спрайт 1** в спрайт **Кот**.



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



Этот код в своем цикле **Повторять 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 ***y*-скорость**, а затем перемещает спрайт **Кот** по оси *y* в позицию со значением переменной ***y*-скорость**, заставляя кота падать все быстрее и быстрее. Если вы программировали игру «Баскетбол» в главе 4, код падения должен быть вам знаком.

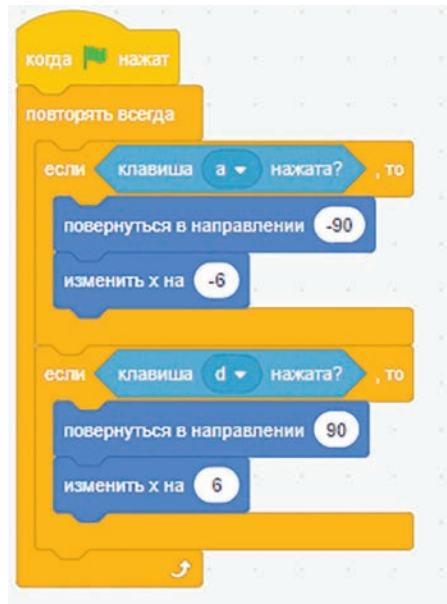
Но блок **Повторять, пока не** ❷ будет запускать цикл до тех пор, пока спрайт **Кот** не перестанет касаться спрайта **Земля**. (Если кот все еще находится в воздухе и падает, он не коснется земли, поэтому код в цикле будет пропущен.) Внутри этого цикла переменной ***y*-скорость** присваивается значение 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**. 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 6. Код для перемещения влево и вправо напишем позже.

КОНТРОЛЬНАЯ ТОЧКА

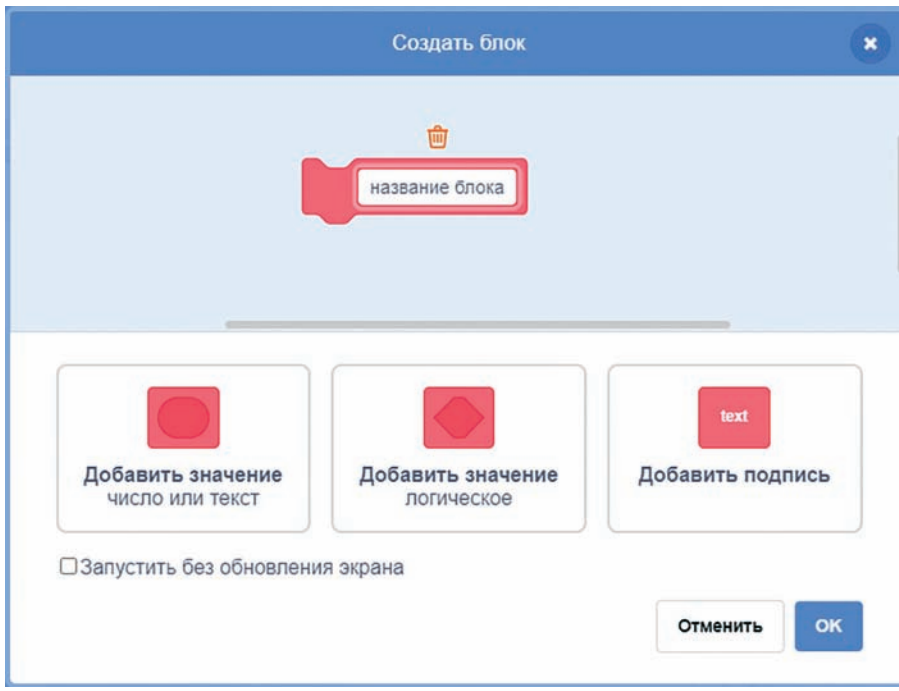
Нажмите кнопку в виде зеленого флага, чтобы проверить уже готовый фрагмент кода. Нажимайте клавиши **A** и **D**, чтобы перемещать кота по склонам. Если кот сорвется с края спрайта **Земля** и упадет за пределы нижней части сцены, то он должен появиться сверху. Затем нажмите кнопку в виде красного знака 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 *Платформер1.sb3* 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 розовые пользовательские блоки. Перейдите в категорию **Другие блоки** и нажмите кнопку **Создать блок**. Присвойте блоку имя **Обработка земли** и установите флажок **Запустить без обновления экрана**. Если этот флажок установлен, Scratch запустит код в розовом пользовательском блоке в турборежиме, даже если игрок не включил турборежим нажатием кнопки в виде зеленого флажка с клавишей **Shift**. Без турбо-режима данный код будет выполняться слишком медленно.



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 готовый код. Двигайте кота, используя клавиши, или используйте мышь, чтобы сбросить кота с верхней части сцены вниз. Теперь спрайт **Кот** 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.sb3* из архива с примерами и продолжайте чтение с этого момента.

Б. Использование крутых склонов и стен

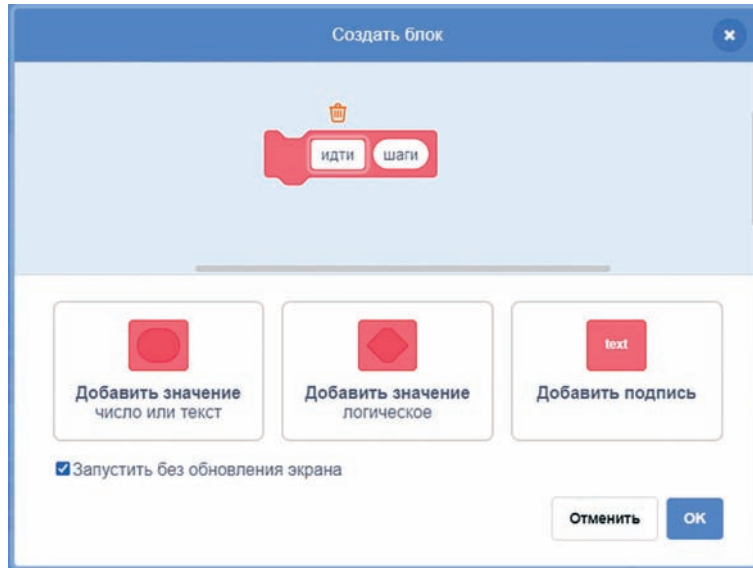
Спрайт **Земля** имеет возвышенности и склоны, по которым может ходить кот, и вы можете в графическом редакторе придать спрайту **Земля** практически любую форму. Это существенное улучшение по сравнению с хождением по ровной поверхности нижней части сцены, как в игре «Баскетбол». Но теперь проблема в 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 кнопкой мыши в области скриптов и выберите **Очистить блоки**, чтобы выстроить скрипты в аккуратные ряды.

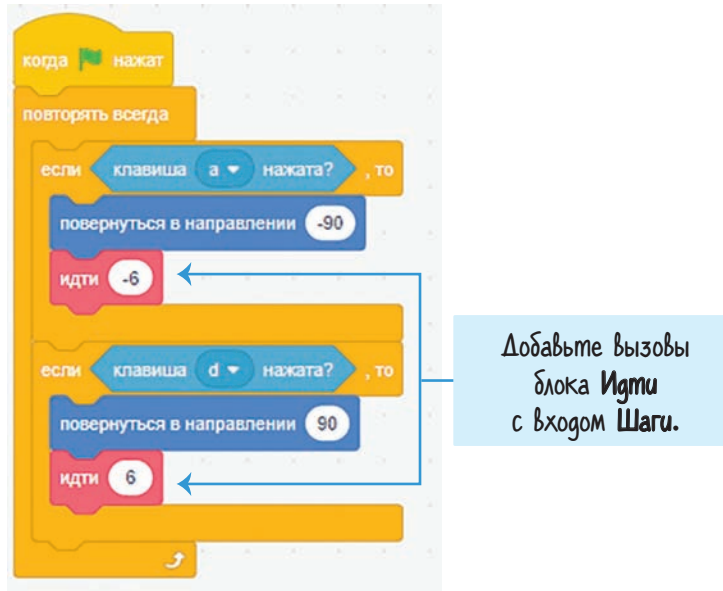
5. Добавление кода для крутого склона

Теперь нам нужно отредактировать код ходьбы для спрайта **Кот**, а также добавить новый код. Вместо простого присваивания определенного значения позиции *x* мы будем использовать еще один пользовательский блок. Давайте присвоим ему имя **Идти** 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 **Идти**, нажмите кнопку **Создать блок** в розовой категории **Другие блоки**. Нажмите кнопку **Добавить числовое поле**, чтобы сделать вход **Шаги**. Когда мы хотим вызвать новый блок **Идти**, 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 непростой, 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 **Идти**. В блоке **Определить Идти** блок входа **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 **Идти –6**) и вправо (с помощью блока **Идти 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 блокирует 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.

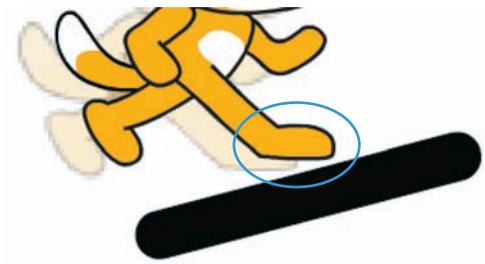
Если значение переменной **Склон** меньше 8, то наклон не очень крутой. Спрайт **Кот** может подняться по склону, поэтому скрипт **Определить Идти** ничего больше делать не будет.

Но если значение переменной **Склон = 8**, цикл **Повторять, пока не** перестанет запускаться. Этот код в переводе на человеческий

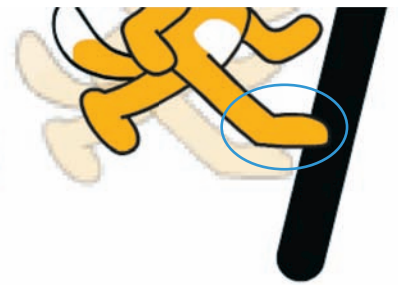
язык будет звучать так: «Спрайт был поднят на 8 единиц, но он все еще касается спрайта **Земля**, значит, это крутой склон».

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 **Идти** на –1 дает противоположное значение входа и переменной.

Пологий склон: Спрайт **Кот** поднимается на 8 единиц и больше не касается земли. Спрайт **Кот** может подняться по этому склону.



Крутой склон: Спрайт **Кот** поднимается на 8 единиц и все еще касается земли. По этому склону спрайт **Кот** не может подняться.



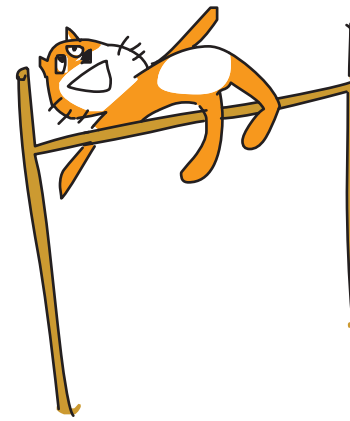
Этот код в точности похож на код в игре «Бегущий в лабиринте» в главе 3, который запрещает игроку проходить сквозь стены.

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 его остановить. Нажмите красную кнопку остановки и сохраните программу.

Если вы запутались, откройте файл *Платформер_3.sb3* из архива с примерами и продолжайте чтение с этого момента.

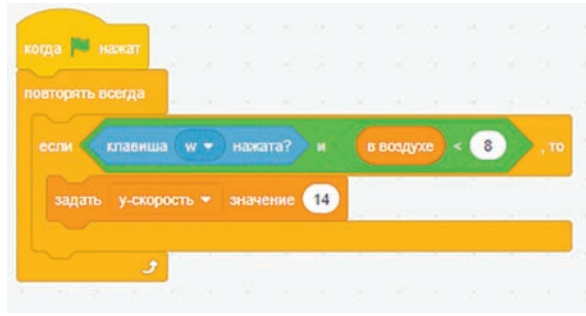
В. ОБУЧЕНИЕ КОТА ВЫСОКИМ И НИЗКИМ ПРЫЖКАМ



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 платформерах 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».

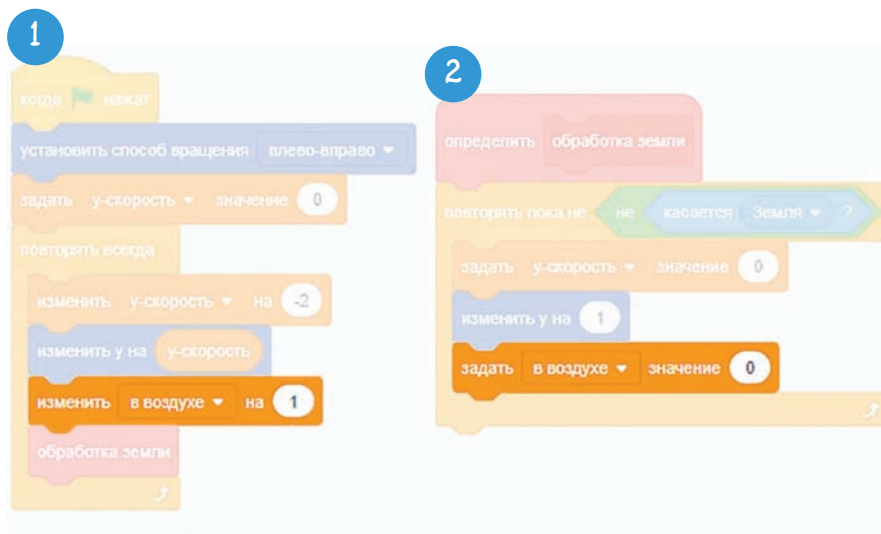
6. Добавление кода прыжка

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 кот будет находиться не на земле. Добавьте код, показанный на следующем рисунке, в спрайт **Кот**.



Цикл **Повторять 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 ***y*-скорость** устанавливается равным 14. За это отвечает код в цикле **Повторять всегда** ❶. Он изменяет положение по оси *y* спрайта **Кот** на положительное значение переменной ***y*-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 ***y*-скорость** продолжает устанавливаться равным 14, вместо того чтобы уменьшаться. Это происходит из-за блока **Изменить *y*-скорость на –2**. Поэтому спрайт совершает прыжок вверх дольше, чем если бы игрок удерживал клавишу **W** всего на одну итерацию в цикле. Но в конечном счете 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 ***y*-скорость** будет уменьшаться, как и ожидалось, и в конечном счете 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 стоит на земле, и не может делать двойные прыжки. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

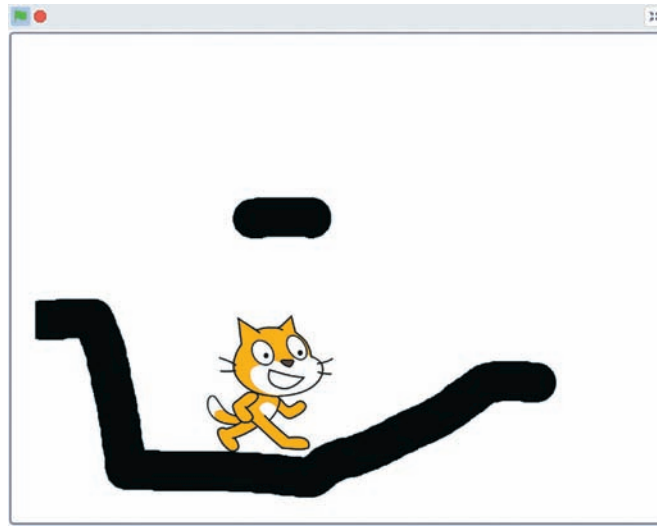
Если у вас возникли проблемы, откройте файл *Платформер_4.sb3* из архива с примерами и продолжайте чтение с этого момента.

Г. ДОБАВЛЕНИЕ ВОЗМОЖНОСТИ ОБНАРУЖЕНИЯ ПРЕПЯТСТВИЙ СВЕРХУ

Теперь кот может ходить по земле, и стены будут препятствовать ему двигаться сквозь них. Но если он подпрыгивает и ударяется головой 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 удается, перерисуйте платформу немного ниже.

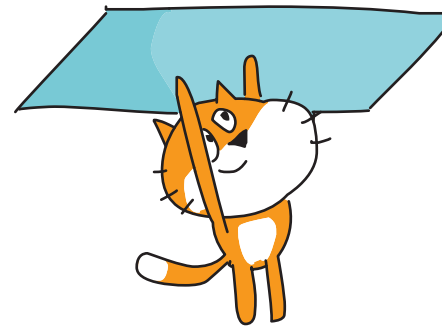
КОНТРОЛЬНАЯ ТОЧКА

Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Попрыгайте котом под низкой платформой. Обратите внимание: 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

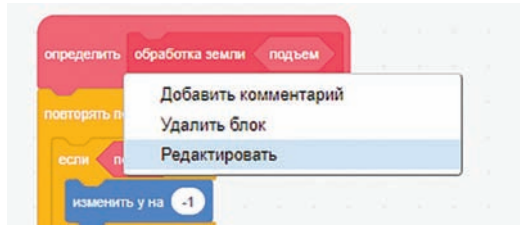
Проблема с кодом находится в пользовательском блоке **Обработка земли**. Этот код предполагает, что спрайт **Кот** всегда падает сверху и, если спрайт **Кот** касается спрайта, он должен быть

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 ***y*-скорость** выше 0. Итак, давайте отредактируем пользовательский блок **Обработка земли**, чтобы добавить новый логический вход с именем **Подъем**.

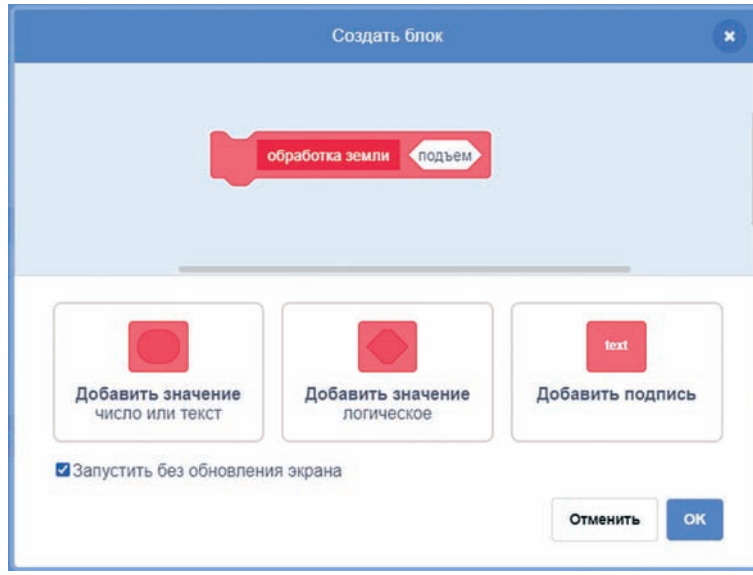


Логическое (булево) значение может быть только истинным или ложным. Мы используем логический вход, потому что нам нужно знать, больше ли нуля значение переменной ***y*-скорость**, когда происходит первое обращение к блоку **Обработка земли**. Это истинное или ложное значение хранится во входе **Подъем** точно так же, как хранилось бы в переменной. Если мы поместим код ***y*-скорость > 0** в блок **Если, то** вместо **Подъем**, то кот окажется сверху на потолке вместо того, чтобы наткнуться на него.

Щелкните правой кнопкой мыши (или нажмите и удерживайте) по блоку **Определить Обработка земли** и выберите в меню команду **Редактировать**.



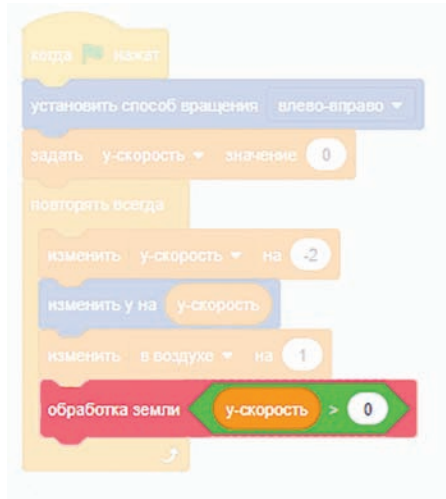
Нажмите кнопку **Добавить значение (логическое)**. Присвойте этому новому полю ввода имя **Подъем** и нажмите кнопку **ОК**.



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 Обработка земли** так, чтобы он соответствовал коду на рисунке ниже.

Если кот движется вверх, то из-за блока **Изменить *y* 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 ***y*-скорость > 0**.



Блок **Обработка земли *y*-скорость > 0** определяет вход **Подъем** как истинный, если значение переменной ***y*-скорость** больше 0 (то есть если спрайт прыгает и перемещается вверх). Если значение переменной ***y*-скорость** не превышает 0, то спрайт либо падает, либо стоит, и вход **Подъем** определяется как ложный.

Таким образом, блок **Определить Обработка земли** определяет, должен ли он запустить блок **Изменить *y* на –1** (чтобы спрайт **Кот** не мог подняться сквозь потолок) или запустить код **Изменить *y* 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 ***y*-скорость** должно быть присвоено значение 0, чтобы спрайт **Кот** перестал падать или прыгать.

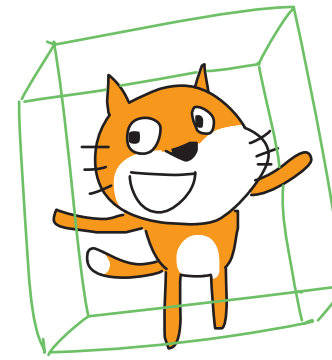
КОНТРОЛЬНАЯ ТОЧКА

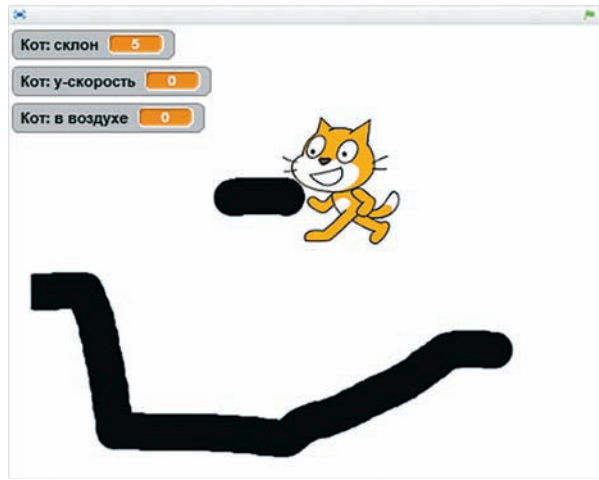
Нажмите кнопку в виде зеленого флага, чтобы проверить код, готовый к этому моменту. Пройдите котом под низкую платформу и подпрыгните. Кот должен врезаться головой в платформу, а не оказаться сверху на этой платформе. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Если у вас возникли проблемы, откройте файл *Платформер_5.sb3* из архива с примерами и продолжайте чтение с этого момента.

Д. 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.

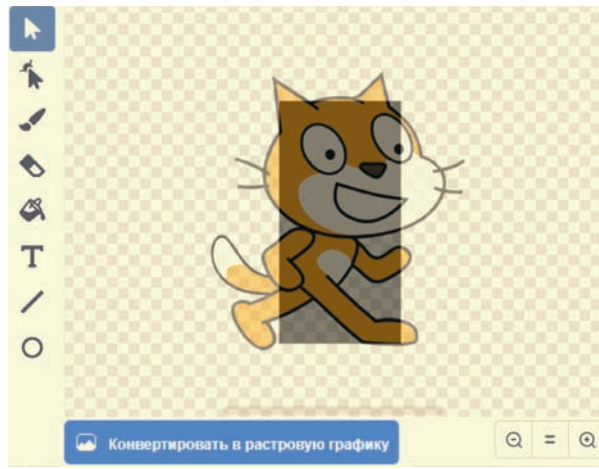




К счастью, мы можем исправить эту проблему, используя концепцию «хитбокс», которую применяли в игре «Баскетбол».

9. Добавление костюма Хитбокс к спрайту Кот

Перейдите на вкладку **Костюмы** спрайта **Кот**. Затем выберите пункт **Рисовать**, который скрывается в кнопке **Выбрать костюм**, и нарисуйте черный прямоугольник, который покрывает бóльшую часть (но не всю) площади двух других костюмов. На следующем рисунке показан полупрозрачный первый костюм **Кот**, чтобы вы могли видеть, сколько площади перекрывает черный прямоугольник.

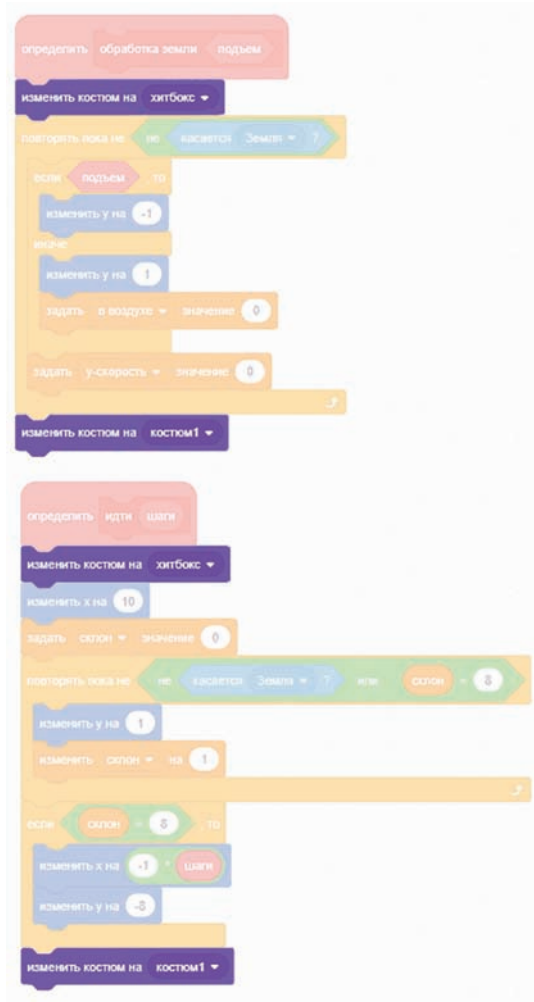


Присвойте этому костюму имя **Хитбокс**. Всякий раз, когда код платформера проверяет, касается ли спрайт **Кот** спрайта **Земля**, мы перед проверкой переключаем костюм на черный прямоугольник костюма **Хитбокс**, а после проверки возвращаемся к обычному костюму. Совершая эти действия, **Хитбокс** определяет, касается ли кот земли.

Эти переключения костюма будут закодированы розовыми пользовательскими блоками с параметром **Запускать без обновления экрана**, поэтому костюм **Хитбокс** никогда не отобразится на экране.

10. Добавление кода хитбокса

Мы добавим блоки **Изменить костюм на** в начале и конце обоих розовых пользовательских блоков. Измените код спрайта **Кот** так, чтобы он соответствовал коду на рисунке.



Эти блоки из фиолетовой категории **Внешний вид** будут изменять костюм на **Хитбокс**. Поскольку **Хитбокс** — это простой прямоугольник без выступающих частей, которые могут «цепляться» за платформы (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.sb3* из архива с примерами и продолжайте чтение с этого момента.

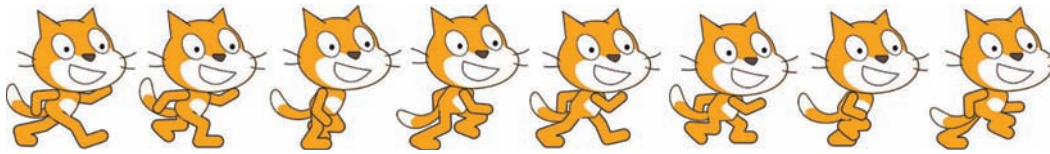
E. УЛУЧШЕНИЕ АНИМАЦИИ ХОДЬБЫ

Спрайт **Кот**, с которого начинается любой проект Scratch, изначально содержит два костюма с именами **Костюм 1** и **Костюм 2**.

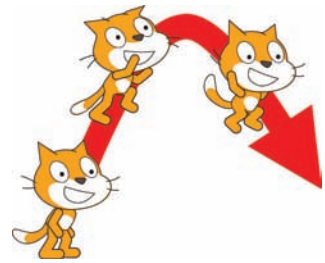
Костюм 1

Костюм 2

Конечно, вы можете сделать простую анимацию ходьбы, переключаясь только между этими двумя костюмами. Но пользователь Scratch с ником griffpatch создал целую серию костюмов ходьбы для кота — см. ссылку **scratch.mit.edu/users/griffpatch/**.



Он 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 нужное время будет переключаться между этими костюмами. Пользователь griffpatch создал несколько классных программ Scratch с использованием этих костюмов — см. **scratch.mit.edu/users/griffpatch/**.

11. 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, прыгающего и падающего кота 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**. Выберите пункт **Загрузить костюм**, который скрывается в кнопке **Выбрать костюм**, и выберите документ *Остановка.svg*, чтобы загрузить файл. Так будет создан новый костюм с именем *Остановка.svg*.

Удалите исходные костюмы **Костюм 1** и **Костюм 2**, но сохраните костюм **Хитбокс**. Поместите костюмы в следующем порядке

(важно, чтобы этот порядок вы соблюдали точно) и со следующими именами:

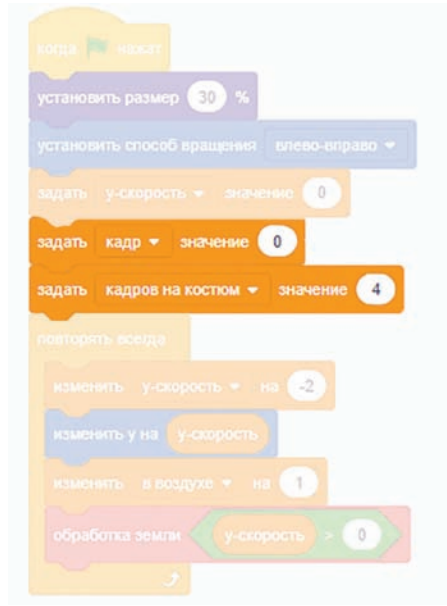
1. Остановка.
2. Прыжок.
3. Падение.
4. Прогулка1.
5. Прогулка2.
6. Прогулка3.
7. Прогулка4.
8. Прогулка5.
9. Прогулка6.
10. Прогулка7.
11. Прогулка8.
12. Хитбокс.

Каждый костюм имеет не только имя (например, **Прогулка1**, **Прыжок** или **Падение**), но и число. Номер костюма присваивается в соответствии с порядком расположения костюма на вкладке **Костюмы**. Например, верхний костюм называется **Остановка**, но также он является костюмом 1. Костюм под ним называется **Прыжок**, 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 сложновато 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**  **нажат** спрайта **Кот**.

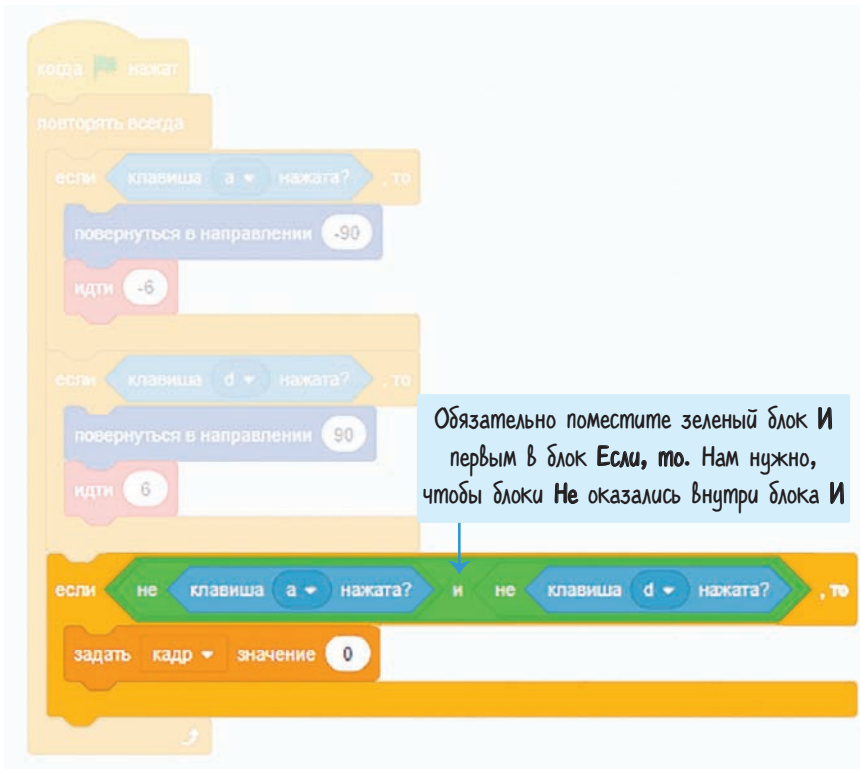


Теперь установка завершена.

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 Идти**, 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**  **нажат** 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 розовой категории **Другие блоки** нажмите кнопку **Создать блок** и присвойте блоку имя **Правильный костюм**. 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**, либо один из костюмов **Прогулка**. Помните, что скрипт **Когда** ⚑ **нажат** сохраняет значение переменной **Кадр** равным 0, если игрок не нажимает клавишу **A** или **D**. Поэтому, когда значение переменной **Кадр** равно 0, блок **Изменить костюм на Остановка** отображает костюм **Остановка**. 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 **Изменить костюм на целое меньшее 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 деления по модулю, результат которой — остаток от деления целого числа на другое целое число. Например, результат деления по модулю выражения 7/3 будет 1, так как 7/3 = 2 + 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 номера 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.

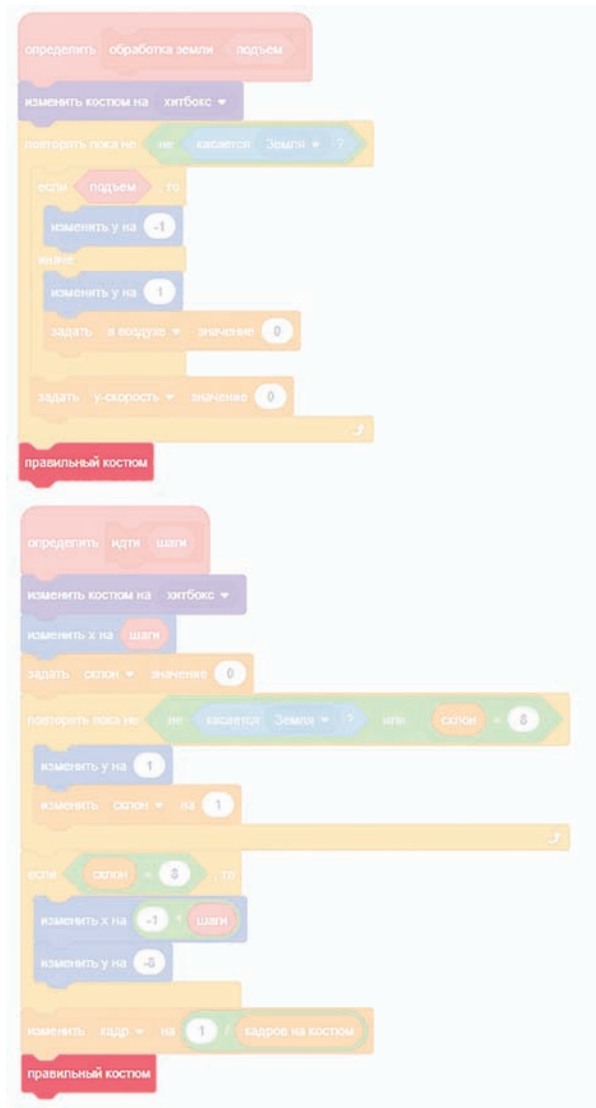
Блок **Остаток от деления на** поможет нам совершить это возвращение с номерами костюмов. Мы можем управлять, какой костюм отображается с помощью математического трюка: поскольку остаток от деления 8/8 будет равен 0, значение переменной **Кадр**, равное 8, отобразит первый костюм **Прогулка!** К этому 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 ***y*-скорость**, чтобы увидеть, падает ли кот (то есть если значение переменной ***y*-скорость** меньше или равно 0) или подпрыгивает (то есть

если значение переменной ***y*-скорость** больше 0), и переключиться на правильный костюм. На этом заканчивается код **Определить правильный костюм**.

Замените блок **Изменить костюм на костюм 1** в блоках **Определить Обработка земли** и **Определить Идти** 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**, **Прыжок** и **Падение** отображаются в нужное время. Затем нажмите кнопку в виде красного знака остановки и сохраните программу.

Если у вас возникли проблемы, откройте файл *Платформер_7.sb3* 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 на спрайте **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**. Выберите пункт **Загрузить костюм**, который скрывается в кнопке **Выбрать костюм**, и выберите файл *ФонПлатформера.png*, который находится в архиве с примерами. После того как этот костюм загружен, вы можете удалить предыдущий костюм.

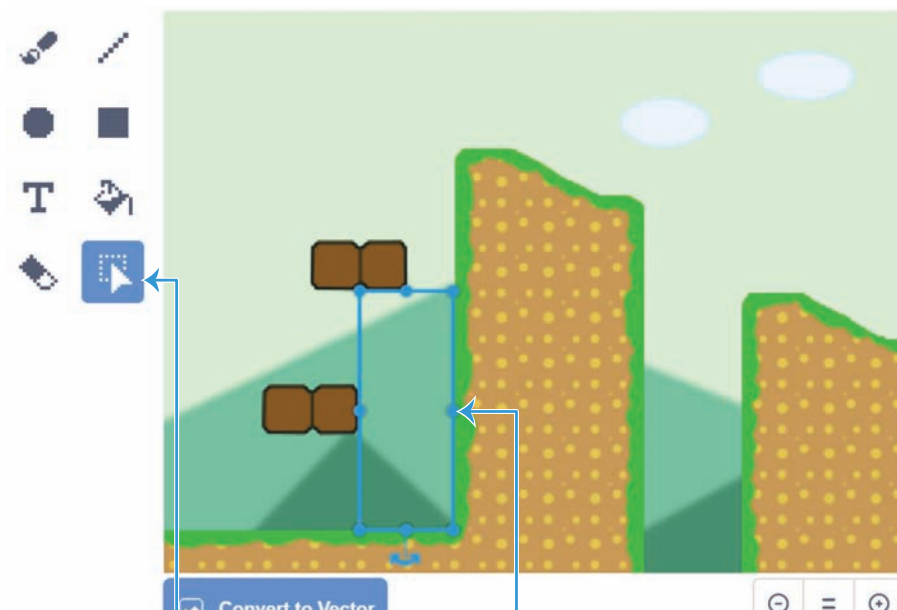
Недостаточно добавить файл *ФонПлатформера.png* в качестве костюма для спрайта **Земля**, вы также должны загрузить его в качестве фона сцены. Выберите пункт **Загрузить фон**, который скрывается в кнопке **Выбрать фон**, и выберите файл *ФонПлатформера.png*, чтобы загрузить его. Нам нужно загрузить 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

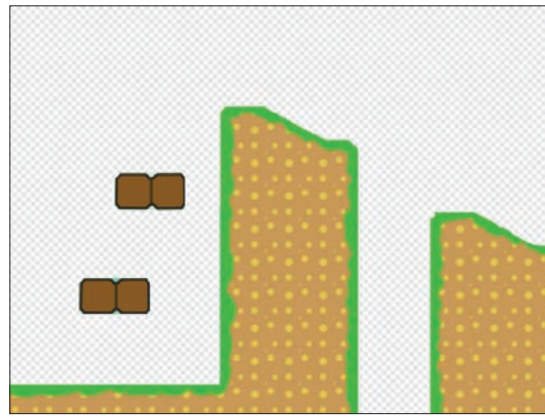
Код игры-платформера основан на том, что спрайт **Кот** коснулся спрайта **Земля**. Костюм спрайта **Земля** — это хитбокс, поэтому, если костюм спрайта **Земля** — это прямоугольник, который занимает всю сцену, он будет взаимодействовать со сценой так, как будто вся сцена — сплошная земля. Нам нужно 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.



Используйте инструмент **Ластик** для стирания непрямоугольных областей. Если вы допустили ошибку, нажмите кнопку **Отменить** в верхней части графического редактора для отмены удаления.

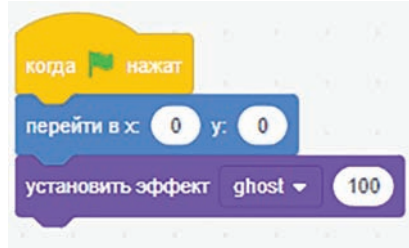
Продолжайте удалять фоновые области костюма, пока не останутся только изображения платформ.



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 вам просто нужно нажать кнопку **Загрузить костюм** 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 прозрачности 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.

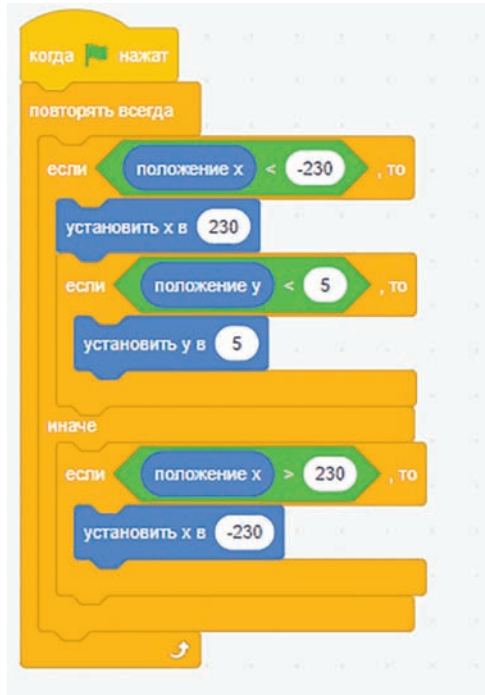
КОНТРОЛЬНАЯ ТОЧКА

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, его **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. 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.sb3* 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.

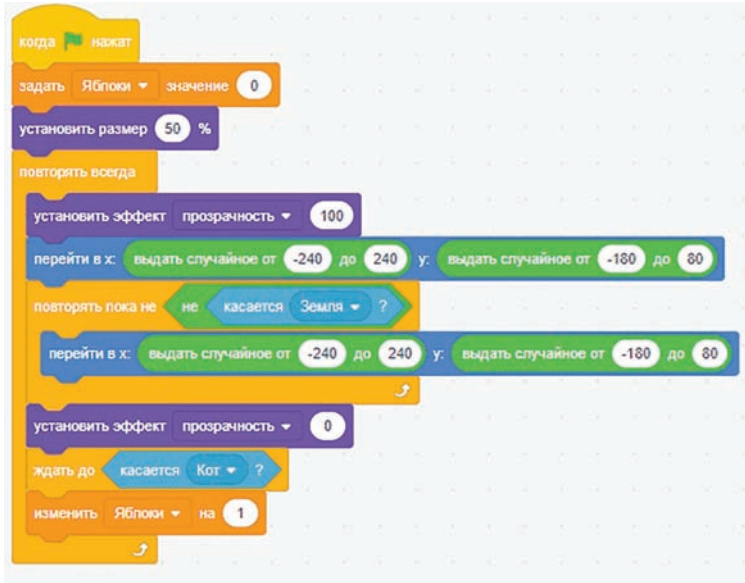
Теперь все, что нам нужно сделать, это собрать игру, используя те части, которые у нас есть. Мы добавим яблоко, которое будет появляться на сцене случайным образом, и нескольких врагов, которые будут пытаться коснуться спрайта **Кот** и украсть яблоки.

17. Добавление спрайта Яблоко и кода для него

Нажмите кнопку **Выбрать спрайт** и в появившемся окне выберите спрайт **Apple**. Для удобства вы можете переименовать спрайт, присвоив ему имя **Яблоко**.

Как и в предыдущих играх, мы будем использовать переменную для отслеживания счета игры. Выберите оранжевую категорию **Переменные**, 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%.

Во время игры спрайт **Яблоко** должен появляться на сцене случайным образом в разных местах. Мы делаем спрайт **Яблоко** невидимым, используя блок **Установить эффект: прозрачность 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 появления 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 прозрачности по-прежнему имеет значение 100. Когда спрайт **Яблоко** обнаруживает место, которое не касается спрайта **Земля**, он снова становится видимым с помощью блока **Установить эффект: прозрачность 0**.

Затем спрайт **Яблоко** ждет, пока спрайт **Кот** его коснется. Когда это происходит, значение переменной **Яблоки** увеличивается на один, а циклы спрайта **Яблоко** находят 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.

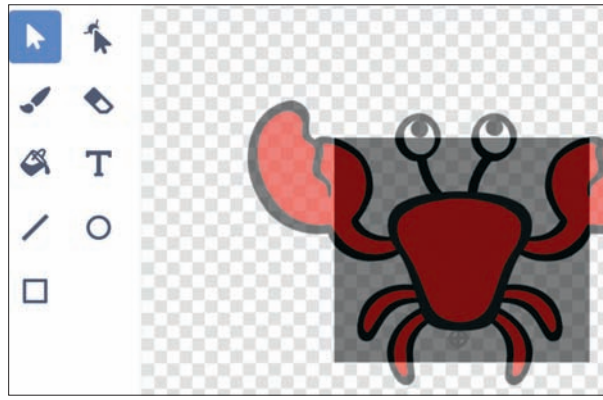
18. Создание спрайта Краб

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 (или нажмите и удерживайте) 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, отвечающим 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 в **Краб-а** и **Краб-б** соответственно.

Спрайт **Краб** по-прежнему обладает всеми костюмами спрайта **Кот** (**Остановка**, **Падение**, **Прогулка 1** 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).

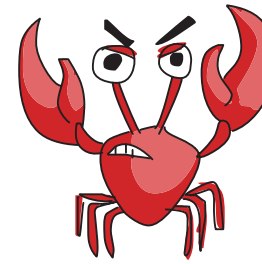


19. Разработка искусственного интеллекта врага

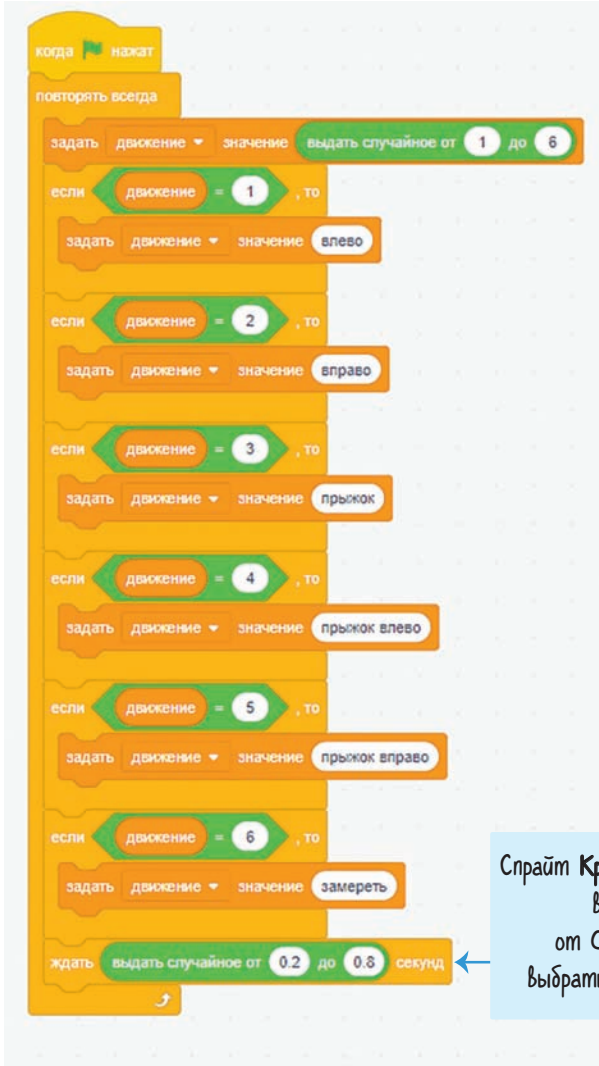
В играх понятие *искусственный интеллект (ИИ)* относится к коду, который управляет движениями врагов и их реакцией на игрока. В нашем платформере крабы не особенно блещут интеллектом: они будут просто двигаться случайным образом.

В оранжевой категории **Переменные** 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.
2. Идет вправо.
3. Прыжок вверх.
4. Прыжок влево.
5. Прыжок вправо.
6. Стоит на месте.



Движения спрайта **Краб** будут определяться произвольно и часто меняться. Добавьте код, показанный на следующем рисунке, в спрайт **Краб**.

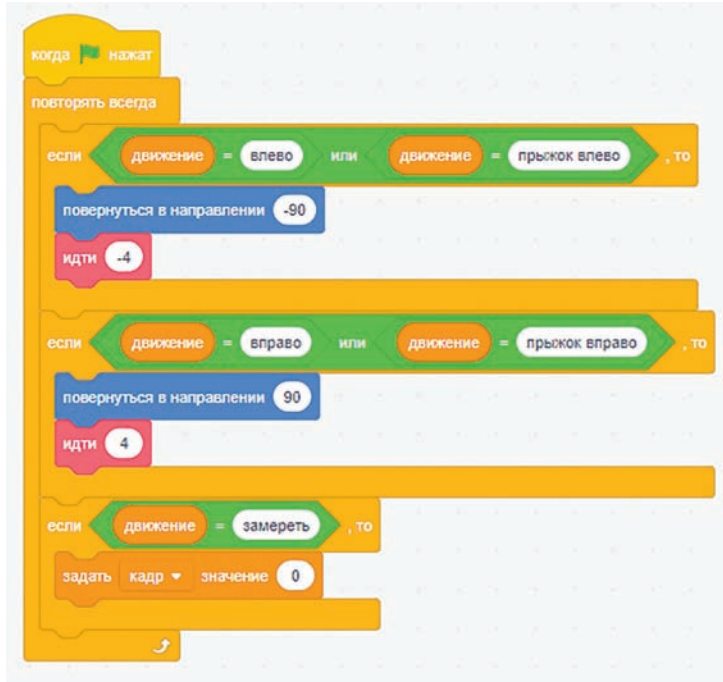


Спрайт **Краб** ждет случайное количество времени в промежутке от 0,2 до 0,8 с, прежде чем выбрать новое случайное движение.

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 на рисунке ниже.



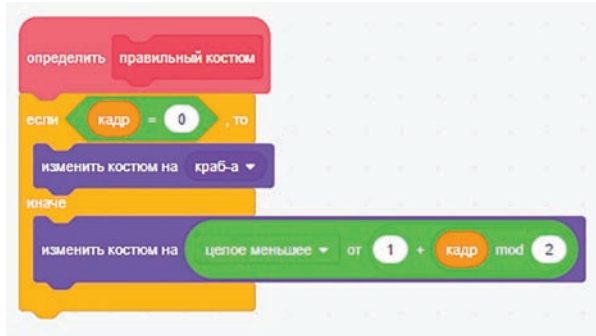
Как и в случае со спрайтом **Кот**, этот код позволяет спрайту **Краб** ходить влево и вправо. Измените значения в блоке **Идти** на –4 и 4, чтобы краб двигался медленнее, чем игрок.

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.

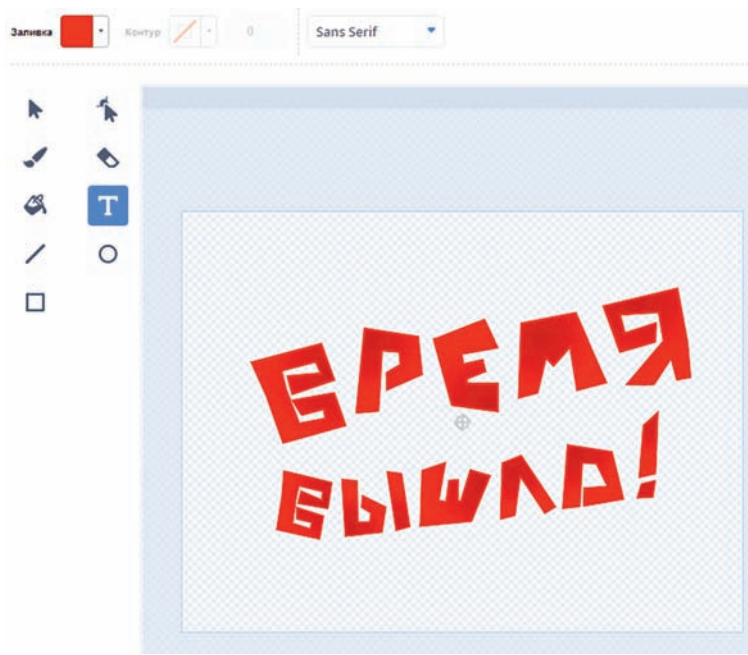
С двумя крабами играть будет интереснее, поэтому щелкните правой кнопкой мыши (или нажмите и удерживайте) по спрайту **Краб** в области спрайтов и выберите в контекстном меню пункт **Дублировать**.

КОНТРОЛЬНАЯ ТОЧКА

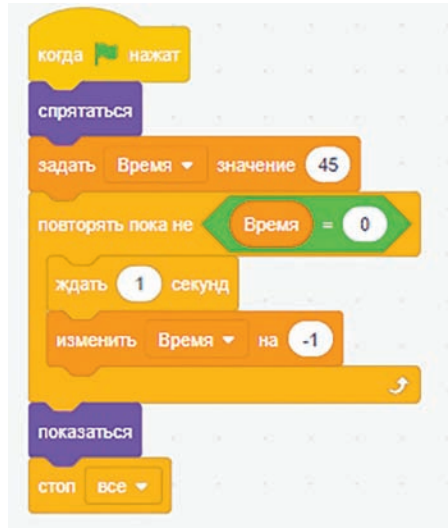
Нажмите кнопку в виде зеленого флага, чтобы проверить готовый фрагмент кода. Убедитесь, что на сцене движутся два краба. Когда один из них коснется кота, краб должен сказать: «Есть одно!» — и одно из ваших яблок должно пропасть. Если же у кота нет яблок, краб просто должен сказать: «Яблоки!» Проверив 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. Создайте новый спрайт, выбрав пункт **Нарисовать,** который прячется в кнопке **Выбрать спрайт** в списке спрайтов, 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 «Продвинутый 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.sb3*.

ЗАКЛЮЧЕНИЕ

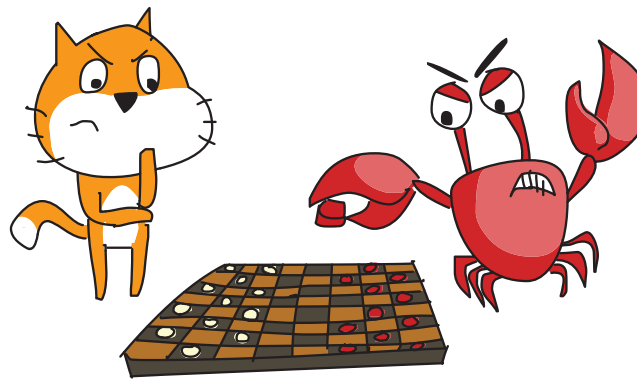
Вы сделали это! Вас можно назвать мастером Scratch! Игра «Продвинутый платформер» — самый сложный и объемный проект в этой книге. Вы объединили и использовали множество различных концепций, чтобы сделать эту игру, так что эту главу стоит прочитать еще несколько раз.

В этой главе вы создали игру, в которой:

- используется спрайт земли, на котором стоит игрок;
- используются розовые 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 — 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

Ответьте 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. Розовые пользовательские блоки помогают избежать дублирования кода. Почему это хорошо?
2. В чем сходство ввода розового пользовательского блока и переменной?
3. Где можно использовать вход розового 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/**) — 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/**) — на этой дискуссионной площадке скретчеры делятся идеями, а также задают вопросы и отвечают на них.

Книга Макса Уэйнрайта «25 Scratch 3 Games» (No Starch Press, 2019), **https://nostarch.com/25scratchgames/**. В этой книге рассматривается еще больше проектов в Scratch.

ScratchEd (**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**.
11. На панели свойств спрайта 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 ↑, ←, ↓ и →.
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** и выберите звук **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 ***y*-скорость**.
5. Когда кот находится в воздухе, числовое значение переменной ***y*-скорость** уменьшается. Когда значение переменной ***y*-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. 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 7

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

Пособие для развивающего обучения

ПРОГРАММИРОВАНИЕ ДЛЯ ДЕТЕЙ

Свейгарт Эл

SCRATCH 3

Изучайте язык программирования, делая крутые игры!

Главный редактор *Р. Фасхутдинов*
Руководитель направления *В. Обручев*
Ответственный редактор *Д. Калачева*
Младший редактор *Д. Данилова*
Художественный редактор *Е. Пуговкина*
Компьютерная верстка *Э. Брегис*
Корректоры *Л. Макарова*, *Ю. Никитенко*

Страна происхождения: Российская Федерация
Шығарылған елі: Ресей Федерациясы

ООО «Издательство «Эксмо»
123308, Россия, город Москва, улица Зорге, дом 1, строение 1, этаж 20, каб. 2013.
Тел.: 8 (495) 411-68-86.
Home page: www.eksmo.ru E-mail: info@eksmo.ru
Өндіруші: «ЭКСМО» АҚБ Баспасы,
123308, Ресей, қала Мәскеу, Зорге көшесі, 1 үй, 1 ғимарат, 20 қабат, офис 2013 ж.
Тел.: 8 (495) 411-68-86.
Home page: www.eksmo.ru E-mail: info@eksmo.ru.
Тауар белгісі: «Эксмо»
Интернет-магазин : www.book24.ru

Интернет-магазин : www.book24.kz
Интернет-дүкен : www.book24.kz
Импортёр в Республику Казахстан ТОО «РДЦ-Алматы».
Қазақстан Республикасындағы импорттаушы «РДЦ-Алматы» ЖШС.
Дистрибьютор и представитель по приему претензий на продукцию,
в Республике Казахстан: ТОО «РДЦ-Алматы»
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) 251-59-90/91/92; E-mail: RDC-Almaty@eksmo.kz
Өнімнің жарамдылық мерзімі шектелмеген.
Сертификация туралы ақпарат сайтта: www.eksmo.ru/certification

Сведения о подтверждении соответствия издания согласно законодательству РФ
о техническом регулировании можно получить на сайте Издательства «Эксмо»
www.eksmo.ru/certification
Өндірген мемлекет: Ресей. Сертификация қарастырылмаған

Дата изготовления / Подписано в печать 13.12.2022. Формат 70x100$^{1}/_{16}$.
Печать офсетная. Усл. печ. л. 18,15.
Тираж экз. Заказ

ISBN 978-5-04-122009-9

9 785041 220099 >

SCRATCH 3

ЭЛ СВЕЙГАРТ – РАЗРАБОТЧИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И АВТОР КНИГ ПО ПРОГРАММИРОВАНИЮ ДЛЯ ДЕТЕЙ. НАПИСАЛ БОЛЕЕ 15 ОБУЧАЮЩИХ КНИГ ПРО SCRATCH И PYTHON. НА ЯЗЫКЕ ОРИГИНАЛА ИХ МОЖНО БЕСПЛАТНО ПРОЧИТАТЬ НА САЙТЕ WWW.INVENTWITHPYTHON.COM.

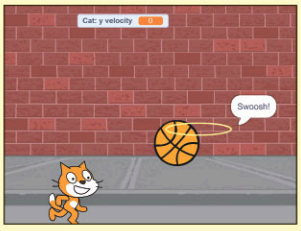
SCRATCH 3 — лучшая среда программирования для новичков. Она отличается удобным и интуитивно понятным интерфейсом, простотой в использовании и возможностью создавать адаптивные игры для различных устройств. Научиться программировать в ней несложно, а сам процесс точно не будет скучным и принесет море удовольствия.

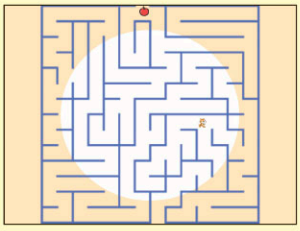
БЛАГОДАРЯ ЭТОЙ КНИГЕ ДЕТИ БЫСТРО НАУЧАТСЯ СОЗДАВАТЬ ТАКИЕ УВЛЕКАТЕЛЬНЫЕ ИГРЫ, КАК:

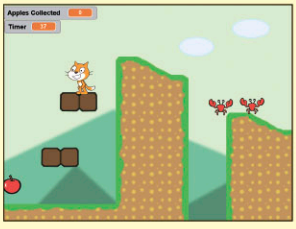
- **Классическая «змейка»**
- **Раннер «Бегущий в лабиринте», в котором главный герой — кот**
- **Аркада «Разрушитель астероидов» с управлением космолетом**
- **Аркада Fruit Slicer (клон знаменитого Fruit Ninja)**
- **Аркада Brick Breaker (ремейк классической игры Breakout)**
- **Платформер с анимацией ходьбы и прыжков, платформами и врагами с искусственным интеллектом**

Каждой из этих игр посвящена отдельная глава, которая включает в себя пошаговые инструкции, наглядные иллюстрации, контрольные вопросы для самопроверки и творческие задачи. Программирование еще никогда не было таким захватывающим!

КНИГА ОТЛИЧНО ПОДОЙДЕТ ДЛЯ ДЕТЕЙ СТАРШЕ 10 ЛЕТ.







ISBN 978-5-04-122009-9

БОМБОРА
ИЗДАТЕЛЬСТВО

БОМБОРА – лидер на рынке полезных и вдохновляющих книг. Мы любим книги и создаем их, чтобы вы могли творить, открывать мир, пробовать новое, расти. Быть счастливыми. Быть на волне.

bombora.ru bomborabooks bombora

no starch press