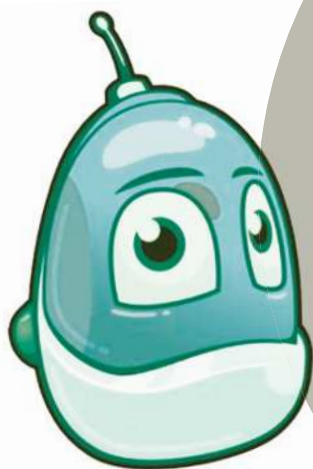




К. И. Астахова

Создаем игры с **KODU** Game Lab



К. И. Астахова

Создаем игры с KODU Game Lab

Под редакцией В. В. Тарапаты

Электронное издание



Москва
Лаборатория знаний
2019

Серия основана в 2018 г.

Астахова К. И.

А91 Создаем игры с Kodu Game Lab [Электронный ресурс] / К. И. Астахова ; под ред. В. В. Тарапаты. — Эл. изд. — Электрон. текстовые дан. (1 файл pdf : 125 с.). — М. : Лаборатория знаний, 2019. — (Школа юного программиста). — Систем. требования: Adobe Reader XI ; экран 10".

ISBN 978-5-00101-628-1

В данном учебном пособии рассказывается, как с помощью конструктора игр Kodu Game Lab создавать 3D-миры и компьютерные игры самых разных жанров — от простых «бродилок» до «стрелялок» и головоломок.

Книга разбита на темы, освоив которые на примерах небольших демо-игр, вы сможете создавать собственные полноценные игровые проекты.

Методика пошагового обучения, применяемая в пособии, поможет вам развить у себя алгоритмическое мышление, а также навыки программирования и разработки.

Книга предназначена для дополнительного образования в школе и дома. Она будет полезна учащимся 4–5 классов, а также учителям информатики, руководителям кружков и родителям маленьких программистов.

**УДК 004.9
ББК 32.97**

Деривативное электронное издание на основе печатного аналога:

Создаем игры с Kodu Game Lab / К. И. Астахова ; под ред. В. В. Тарапаты. — М. : Лаборатория знаний, 2019. — 122 с. : ил. — (Школа юного программиста). — ISBN 978-5-00101-189-7.



В соответствии со ст. 1299 и 1301 ГК РФ при устранении ограничений, установленных техническими средствами защиты авторских прав, правообладатель вправе требовать от нарушителя возмещения убытков или выплаты компенсации

ISBN 978-5-00101-628-1

© Лаборатория знаний, 2019

От автора

Дорогие друзья!

У вас в руках — книга, которая откроет перед вами совершенно новые горизонты. Вас ждет волшебный мир создания самых настоящих компьютерных игр и программирования!

Основным инструментом в нашем путешествии станет конструктор игр Kodu Game Lab. В нем можно создавать 3D-миры и настоящие компьютерные игры самых разных жанров: от простых «бродилок» до «стрелялок» и головоломок!

Книга разбита на темы, освоив которые, вы сможете создавать свои собственные игровые проекты. Для изучения приемов создания игр вы будете создавать небольшие **демо-игры**, которые помогут развить навыки программирования и разработки. На основе полученных знаний мы вместе создадим полноценные **игры**.

Эта книга станет важной ступенькой для будущих программистов — вы сможете развить у себя алгоритмическое мышление, которое понадобится не только в будущей профессии, но и при решении любой жизненной проблемы.

Надеемся, что наш интересный учебник станет для вас путеводной звездой в мире программирования и разработки игр, позволит реализовать ваши самые смелые творческие идеи и добиться успехов в учебной и будущей профессиональной жизни!

Удачи вам, друзья! Дерзайте!

Благодарности

Автор выражает искреннюю благодарность всем, кто принял участие в создании, редактировании и совершенствовании этой книги, а также тем, кто так или иначе повлиял на ее выход в свет.

Автор благодарит ООО «Компьютерная Академия ШАГ» (г. Москва) за содействие в проведении занятий по курсу «Создание игр в Kodu Game Lab», всех слушателей этого курса (поток 2016/2017 г.) за неотъемлемый вклад в идейную «копилку», создание собственных и совместных проектов, которые стали частью данной книги.

Не оценимы помощь и поддержка научного редактора книги Виктора Викторовича Тарапаты.

Профессор математического факультета МПГУ Надежда Николаевна Самылкина своими мудрыми советами помогла в написании методической и научной составляющей пособия.

Введение

Kodu Game Lab (лаборатория игр) — это среда разработки (конструктор), позволяющая создавать трехмерные игры без специальных знаний языков программирования. Проект Kodu разработан компанией Microsoft — лидером на рынке программного обеспечения, создавшим знаменитую операционную систему Windows.

Основной задачей в Kodu Game Lab является создание игровых миров, в которых будут находиться различные персонажи и объекты, взаимодействующие по установленным правилам.

Данная среда разработки игр содержит более двухсот готовых миров, на основе которых, путем их модификации, можно создавать свои, получая при этом первоначальный опыт работы с алгоритмами и их структурами. Любая программа в Kodu — это набор правил, которые определяют действия объекта (например, игрового персонажа).

Для написания таких правил в Kodu используются два оператора:

КОГДА + <условие>

ДЕЛАТЬ + <действие>

Особенность Kodu состоит в том, что разработчику, продумывая **сюжет** игры, **логику**, которой будут подчиняться действия персонажей, само **устройство мира**, в котором будут происходить действия, не нужно обращать особое внимание на способы составления программ. Причем идеи игр и игровых жанров практически лишены воображаемых границ — все зависит от вашей фантазии.

Пора начинать!

Загрузка и установка Kodu Game Lab

Зайдем на официальный сайт Kodu Game Lab:
<http://www.kodugamelab.com>



На главной странице нажмем¹ кнопку **Get Kodu**. Откроется загрузочная страница на сайте компании Microsoft. Здесь можно ознакомиться с системными требованиями (System requirements), а затем нажать кнопку **Download**.



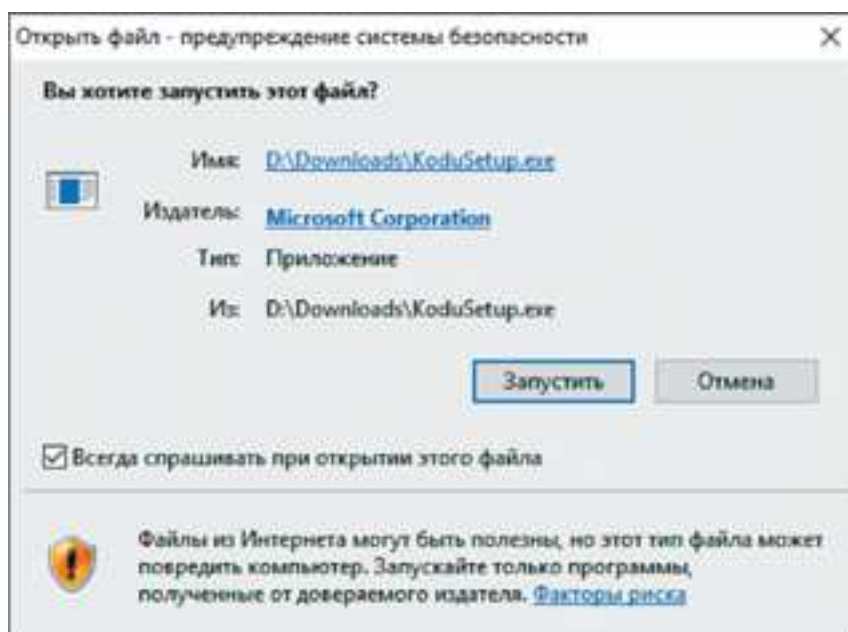
Как видим, здесь используется английский язык (English), но при установке Kodu его можно будет поменять.

Далее выбираем удобный формат установочного файла (обычно это *.exe) и начинаем загрузку нажатием кнопки **Next** (она расположена справа).

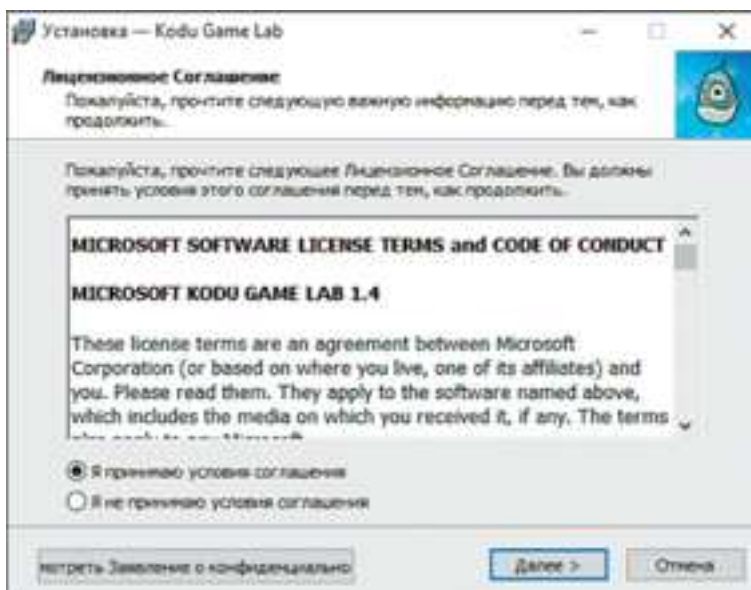
¹ Нажатие на кнопку, команду и др. подразумевает щелчок по ней *левой* кнопкой мыши.



Дожидаемся окончания загрузки и запускаем установочный файл:



В процессе установки выбираем русский язык (или любой другой из предложенных языков) и принимаем лицензионное соглашение:



После этого нажимаем **Далее** и ожидаем полного завершения установки. Теперь программа **Kodu Game Lab** установлена и готова к запуску.



Тема 1. Запуск Kodu Game Lab.

Создание мира

1.1. Главное меню

Запустим Kodu Game Lab. При первом запуске программа продемонстрирует видеоролик, после чего появится окно входа в учетную запись:



В строке **Creator** укажем имя, которое будет автоматически использоваться при сохранении игр. По желанию можно указать пароль в строке **Pin**, состоящий из четырех цифр.

Также можно поставить галочку в пункте **Keep me signed in when Kodu exists** для автоматического входа в учетную запись при запуске Kodu.

После нажатия кнопки **ОК** появится **Основное меню**:



Давайте разберемся с пунктами **Основного меню**:

- **ВОЗОБНОВИТЬ** — открытие последнего редактированного мира;
- **NEW WORLD** — создание нового мира;
- **ЗАГРУЗИТЬ МИР** — база миров, сохраненных на компьютере;
- **ДРУЗЬЯ** — база готовых игр в онлайн-сообществе;
- **ПАРАМЕТРЫ** — настройки программы;
- **ПОМОЩЬ** — краткий справочник по программе, список ее авторов;
- **ВЫЙТИ** — выход из программы.

Как видим, интерфейс в **Kodu Game Lab** русифицирован не до конца, поэтому некоторые команды, параметры, пункты меню и так далее придется рассматривать на языке оригинала. Некоторым инструментам мы дадим свои, более понятные, названия.


Пора приступать к созданию нашего первого мира!

Выберем пункт **NEW WORLD**, тем самым загрузив новый пустой мир.

1.2. Инструменты

Изучим основные инструменты, с помощью которых строится игровой мир:




1. Инструмент **РУКА** . По умолчанию в новом мире всегда выбран инструмент **РУКА**, который отвечает за движение камеры. Удерживая *левую* кнопку мыши и перемещая курсор в рабочей области экрана, мы двигаем камеру.

Колесико мыши позволяет масштабировать камеру (приближаться к объекту или отдаляться от него).

Внимание!

Будьте осторожны — камера может «улететь» в так называемый «Космос». Вернуть ее в свой мир можно, двигая левой кнопкой или колесиком мыши в сторону «Земли».

Если удерживать *правую* кнопку мыши и перемещать курсор в рабочей области экрана, то камера останется на месте, но вращается вокруг своей оси.

2. Следующий инструмент **ОБЪЕКТ** . Этот инструмент отвечает за создание персонажей, объектов и декораций. Нажмем *левую* кнопку мыши в любой области игрового мира — появится меню выбора объектов и персонажей в форме лепестков. Если навести курсор на какого-нибудь персонажа, отобразится краткое описание его возможностей.



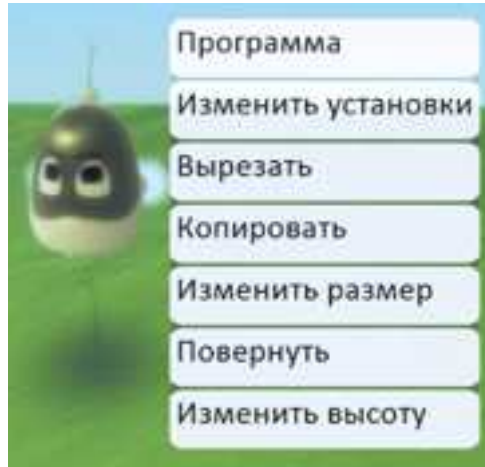
Создадим главного персонажа **Kodu** (далее будем называть его **Коду**), кликнув *левой* кнопкой мыши на верхнем лепестке. Рассмотрим **Коду** поближе. В этом нам помогут инструмент **РУКА** (выберем его снова), колесико и *правая* кнопка мыши.



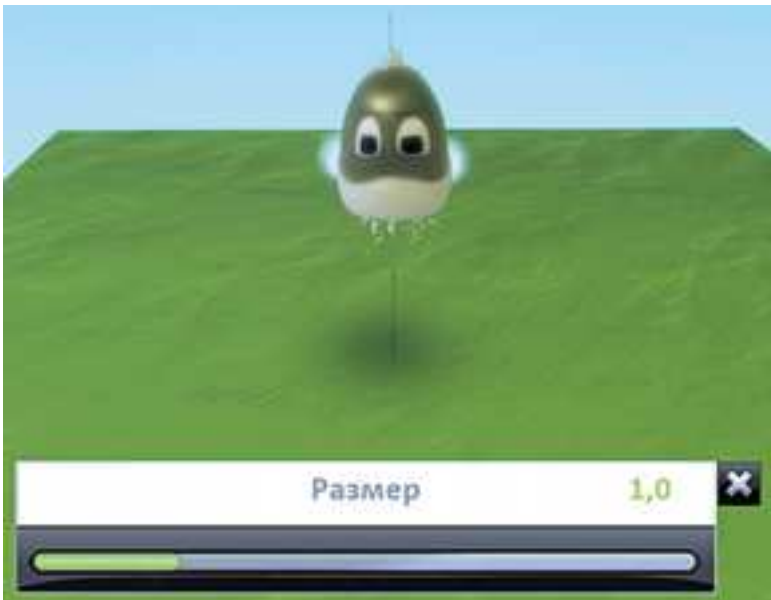
Вернемся к инструменту **ОБЪЕКТ** и переместим **Коду**, удерживая *левую* кнопку мыши, в центр земли. Как видим, при наведении курсора мыши на персонажа над ним появляется палитра цветов. Цвет персонажа можно изменить с помощью клавиш-стрелок **Влево** и **Вправо** на клавиатуре.



Теперь нажмем на **Коду** правой кнопкой мыши — появится меню настроек персонажа:



Здесь можно изменить внешний вид **Коду**. За это отвечают команды **Изменить размер**, **Повернуть** и **Изменить высоту**. При выборе одной из них появляется шкала для изменения значений. Регулировать значения можно *левой* кнопкой мыши или, для точности, стрелками на клавиатуре.





3. Инструмент для редактирования мира **ПУТЬ**. К этому инструменту мы вернемся позже, так как он требует умения создавать программы для персонажей.

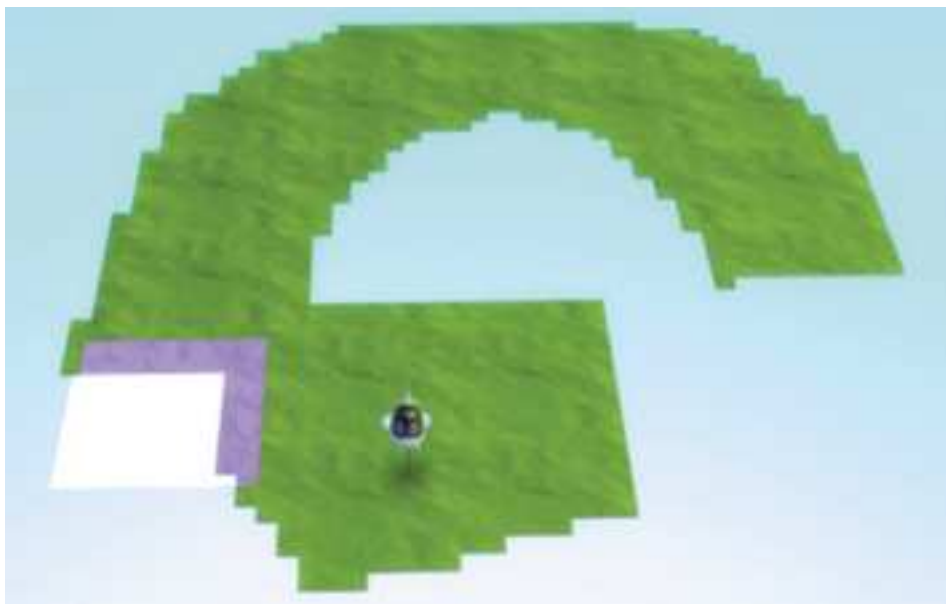
4. Перейдем на соседний инструмент **КИСТЬ ЗЕМ-**



ЛИ. Он позволяет создавать участки земли. При выборе этого инструмента курсор примет вид полупрозрачной кисти. Нажмем *левой* кнопкой мыши на свободной области мира — появится новый участок.



Чтобы удалить созданный ландшафт, нужно нажать на него *правой* кнопкой мыши:




Размер кисти можно увеличить или уменьшить с помощью клавиш-стрелок на клавиатуре:



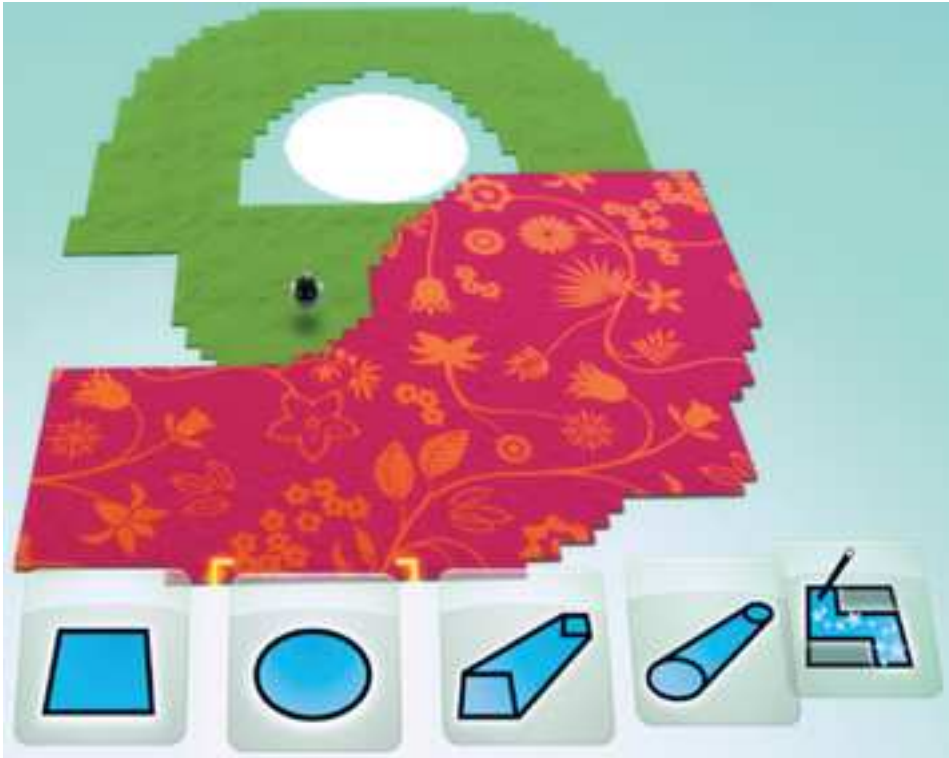
Создание земли напоминает рисование кистью в графических редакторах, отсюда и название инструмента.

Кстати!

Для каждого инструмента всегда доступны подсказки в левой верхней части экрана.

Над инструментом **КИСТЬ ЗЕМЛИ** изображены два параметра: . Левый отвечает за выбор цвета земли. Воспользуемся им и создадим ландшафт нового цвета:







Правый параметр отвечает за выбор формы кисти. Попробуем нарисовать участок земли в форме круга. Для этого выберем круговую форму кисти, увеличим ее до желаемого размера нового участка земли и кликнем мышью — получится круглый остров:



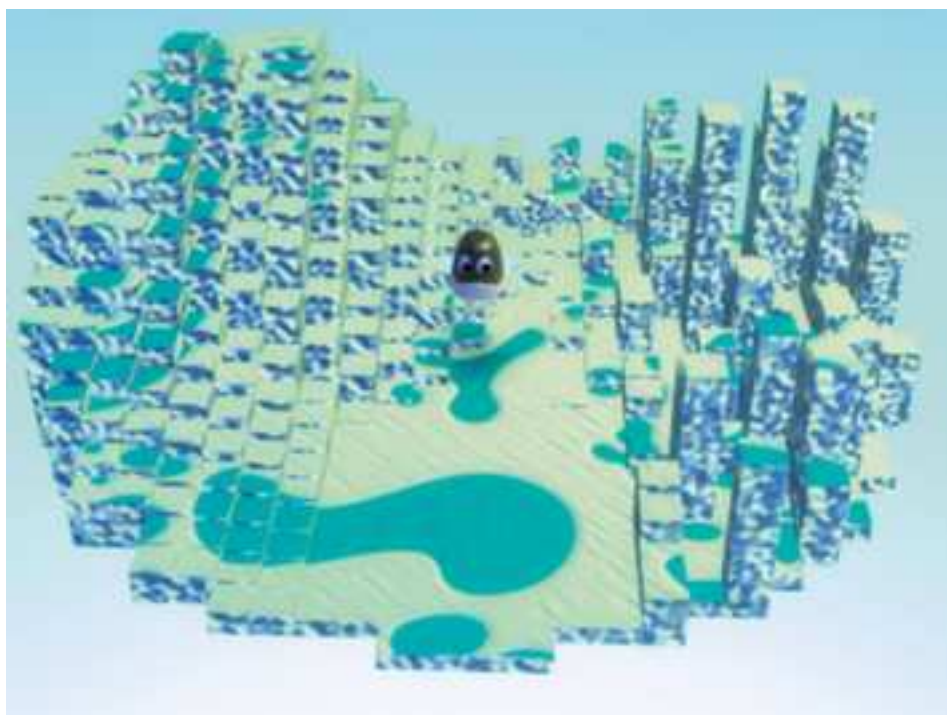
Следующие три инструмента (пункты 5–7) отвечают за создание рельефа земли.


5. Инструмент **ХОЛМЫ**  предназначен для создания холмистой местности.

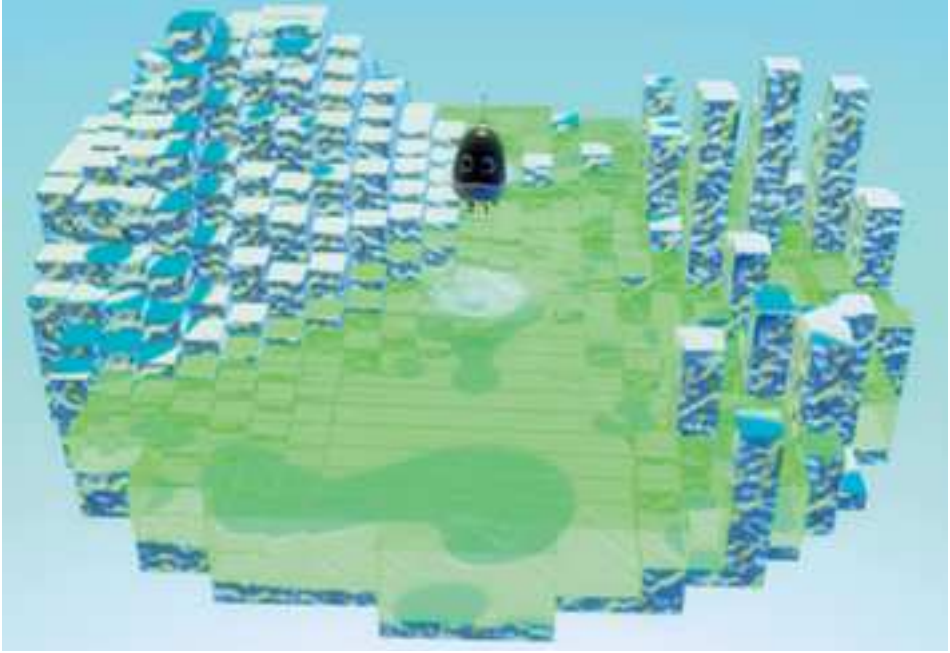
6. Инструмент **СГЛАЖИВАНИЕ**  выравнивает, делает гладкими неровные участки земли.

7. Инструмент **СКАЛЫ**  создает достаточно резкие неровности, на которые не смогут взобраться персонажи.

Заметим, что у этих трех инструментов также можно изменять форму кисти. В качестве тренировки создадим новый рельеф земли с их помощью:




8. Перейдем на следующий немаловажный инструмент **ВОДА** . Он позволяет создавать в мире воду, причем цвет ее можно выбирать. Заполним водой наш рельеф, удерживая для этого *левую* кнопку мыши. Мы создали неровный рельеф, поэтому вода сначала будет заполнять самые низкие участки земли, постепенно поднимаясь выше:

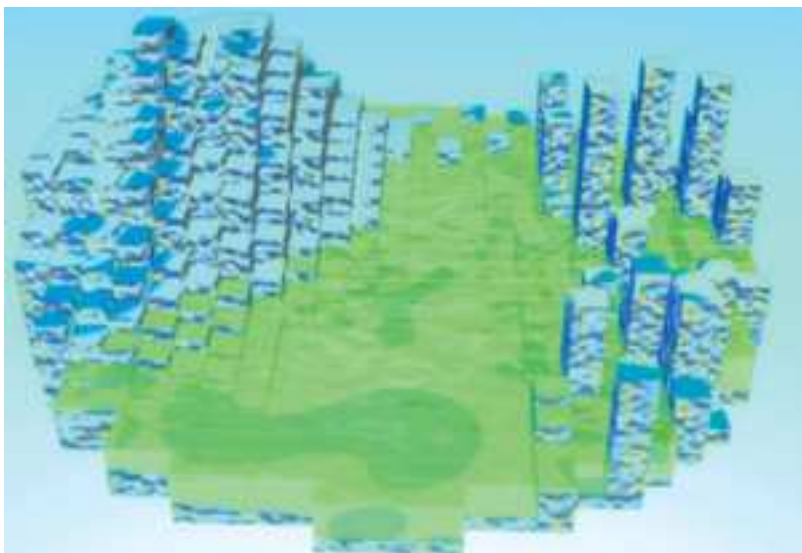


Таким образом, для создания водоемов сначала необходимо позаботиться о рельефе земли и только потом заполнять созданные участки водой.

Кстати!

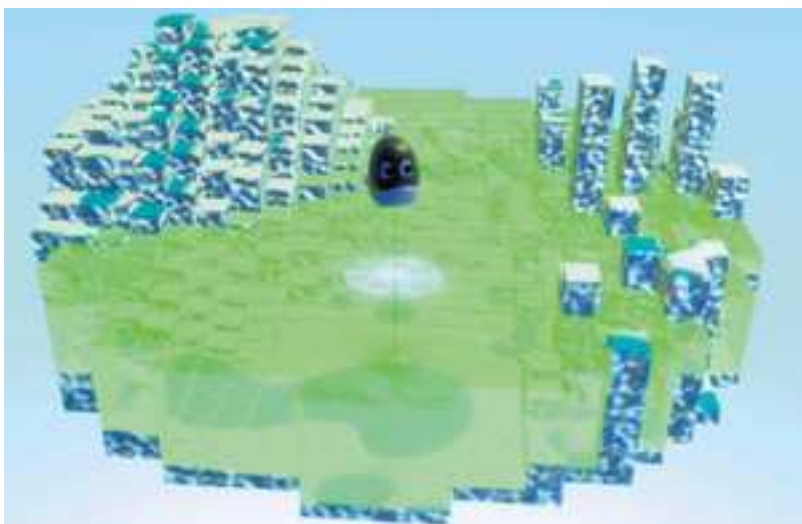
Изначально рельеф земли создается плоским (из одного слоя) и «углубить» его в таком виде не получится. Сначала придется поднять общий уровень рельефа, а уже затем сделать углубления для реки, озера и других водоемов.

9. Последний инструмент, позволяющий редактировать мир, — **УДАЛЕНИЕ ОБЪЕКТОВ** . Этот инструмент позволяет удалять персонажей и объекты, созданные с помощью инструмента **ОБЪЕКТ**. При выборе курсор примет вид кисти, форму и размер которой можно изменять так же, как и в инструменте **КИСТЬ ЗЕМЛИ**. Попробуем удалить Коду, кликнув по нему выбранным инструментом:




Не стоит волноваться, если допущена ошибка и удалены не те объекты: в **Kodu Game Lab** предусмотрена отмена последнего действия. Чтобы вернуться на предыдущее выполненное действие, нужно кликнуть на команде **ОТКАТИТЬ** в левой части экрана, где в скобках указано количество шагов для возврата.

Нажмем команду **ОТКАТИТЬ** один раз, и удаленный Коду вернется в мир:



1.3. Параметры мира

Завершает линейку инструментов инструмент **ПАРА-**

МЕТРЫ МИРА . Это десятый инструмент программы. Он открывает настройки, которые применяются к целому миру сразу, а не к отдельным его элементам.

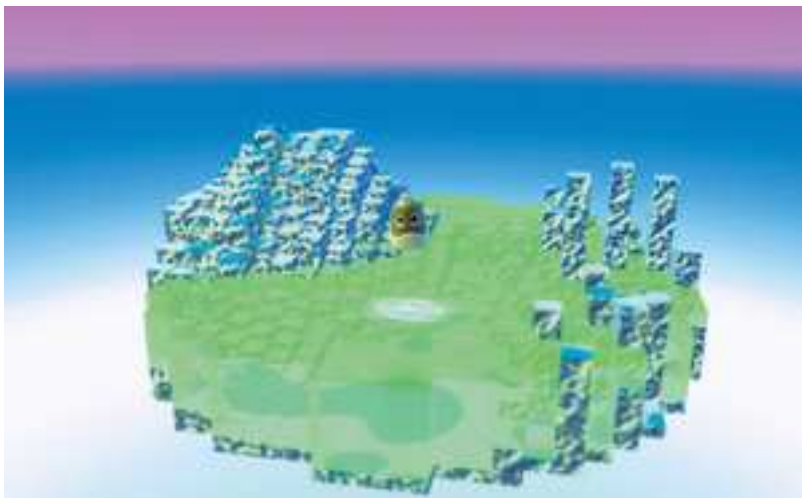
Выберем этот инструмент. Откроется список настроек. Большинство из них мы изучим позже, однако некоторые можно применить уже сейчас.

С помощью колесика мыши найдем параметр **НЕБО**. Здесь можно выбрать цвет неба:



Следующий параметр **ОСВЕЩЕНИЕ** настроит необходимый свет. С помощью *левой* кнопки мыши или стрелок на клавиатуре выберем новый цвет неба и освещение.

Вернуться в свой мир можно нажатием клавиши ESC:



1.4. Сохранение мира

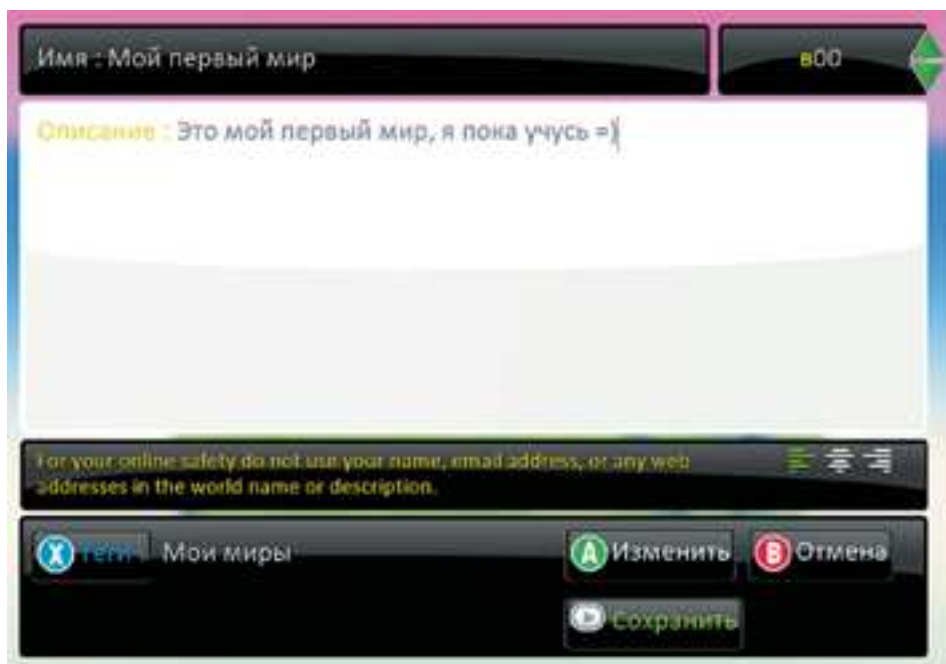
Чтобы сохранить созданный мир, нажмем на **ДОМИК**



. Откроется **Главное меню**. Для сохранения игры выберем пункт **Сохранить мой мир**:



Откроется окно, где в поле **Имя** введем название созданного мира, например **Мой первый мир**:



По желанию можно заполнить описание и добавить теги (ключевые слова для поиска мира).

Когда поля заполнены, нажмем **Сохранить**. Мир будет сохранен «внутри» программы **Kodu Game Lab**.

1.5. Сохранение мира на диске компьютера (экспорт)

Вам наверняка захочется продемонстрировать созданную игру другу или обменяться файлом для совместной разработки.

Для того чтобы выгрузить созданную игру на диск компьютера или флешку, зайдём в **Главное меню** или **Основное меню** (начальный экран), а затем нажмем **Загрузить мир**:



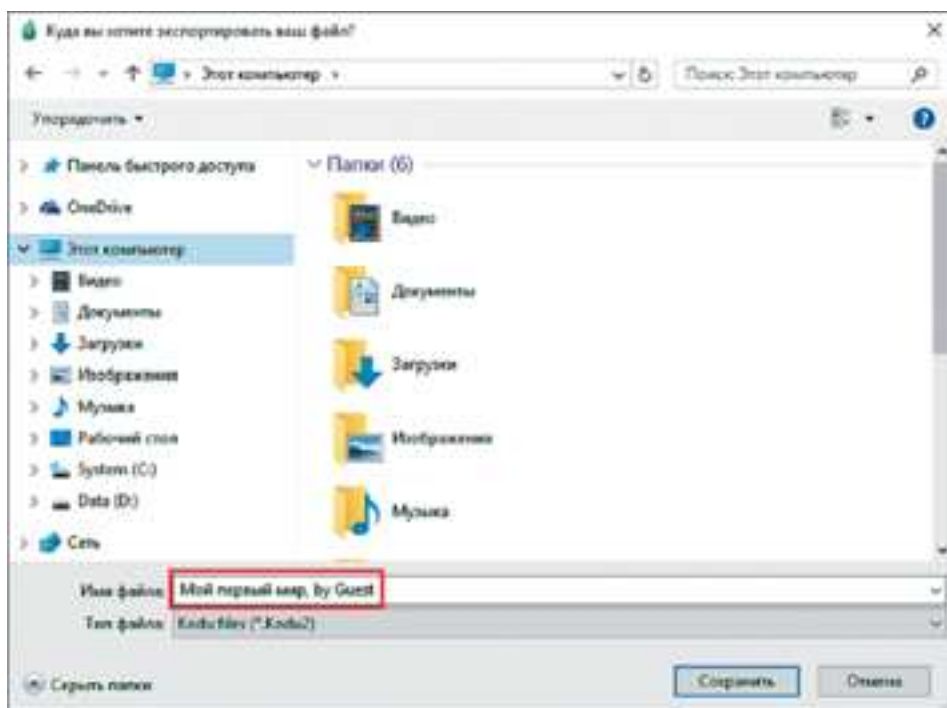
Мы попадем в базу сохраненных миров (на диске компьютера). Здесь хранятся все сохраненные игры, а также примеры игр/миров других разработчиков. Последняя сохраненная игра будет первой в списке:



Нажмем *левой* кнопкой мыши на свой созданный мир и выберем пункт **Экспорт**:



Появится окно, где следует выбрать папку, в которой мы планируем сохранить игру:



Программа автоматически укажет имя файла, которое было написано при сохранении. После этого нажмем кнопку **Сохранить**.

Теперь файл игры можно перемещать и загружать на другие компьютеры, где установлена программа **Kodu Game Lab**.

1.6. Задания для самостоятельной работы

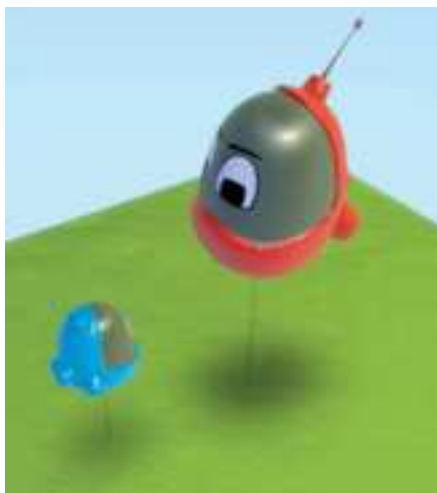
1. Измените параметры персонажа **Коду** следующим образом:

- **Размер:** 2,5;
- **Высота:** 1,75;
- **Поворот:** 250;
- **Цвет:** зеленый.

Для этого откройте параметры персонажа, нажав на него *правой* кнопкой мыши.

Скопируйте **Коду**, выбрав команду **Копировать**. Кликните на свободной области *правой* кнопкой и выберите команду **Вставить (Kodu)**.

Измените параметры второго **Коду**. Пусть он будет другого цвета, меньше размером, приподнят над землей и повернут лицом к первому **Коду**:



2. Создайте реалистичный мир, например поляну с расположенными на ней лесом, озером, тропинками или подводный мир с растительностью, рыбами и другими персонажами.

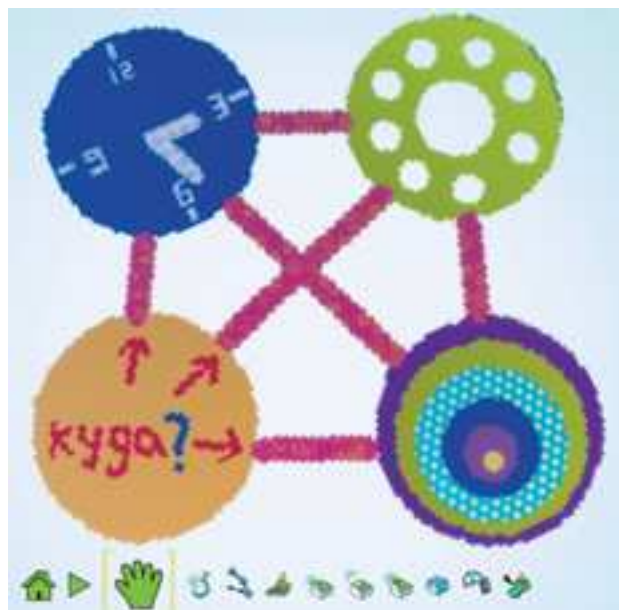
Подсказка. Первым делом продумайте рельеф земли, затем заполните мир водой, после этого можете добавлять объекты.

Пример готового мира:



3. Помните историю про «Алису в Стране чудес»? С помощью инструмента **КИСТЬ ЗЕМЛИ** создайте территорию нестандартной формы и цвета; возможно, земля будет напоминать какой-то предмет или явление. Напишите для Алисы подсказки на земле.

Пример «Страны чудес»:



Тема 2. Начинаем программировать. Простые условия

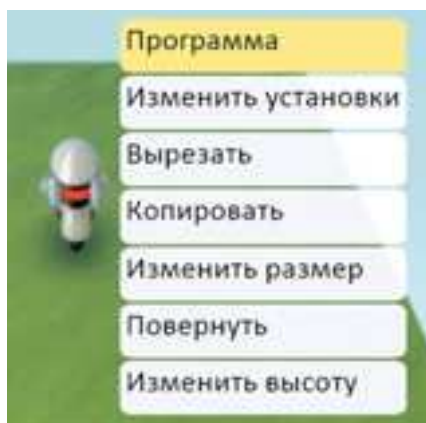
2.1. Первая программа. Движение

Запустим **Kodu Game Lab**. В **Основном меню** программы выберем **NEW WORLD**. Откроется новый пустой мир.

Сначала изучим четыре способа движения персонажа. Для этого создадим **демо-игру**: в пустом мире добавим какого-нибудь персонажа, например **Байкера**, и немного увеличим исходный размер земли:



1. Движение с помощью клавиши. С помощью инструмента **ОБЪЕКТ** нажмем на **Байкера** *правой* кнопкой мыши и в открывшемся меню выберем пункт **Программа**:



Мы попадем на **Страницу 1**, где будет создаваться программа для выбранного объекта.

Вспомним, что программа в **Kodu Game Lab** состоит из условий и действий. Они записываются в строках с помощью карточек и имеют следующую конструкцию: **КОГДА** + <условие>, **ДЕЛАТЬ** + <действие>.

**Кстати!**

Программисты используют угловые скобки <...> для обозначения тех условий и действий, которые должны стоять на этом месте.

Заполним условие, которое позволит персонажу двигаться. Стандартно управление выглядит следующим образом: персонаж двигается вперед, назад, влево, вправо при нажатии соответствующих клавиш WASD или клавиш-стрелок на клавиатуре.

Начнем заполнять условие, нажав на знак «+» возле условия **КОГДА**. Появится выбор карточек условий в форме лепестков. Выберем карточку **КЛАВИШИ**, затем еще раз кликнем на знаке «+» и выберем карточку **WASD** либо карточку **СТРЕЛКИ**.

Далее нажмем на знак «+» у действия **ДЕЛАТЬ**. В выборе действий укажем карточку **ДВИГАТЬСЯ**. Получится следующая строка программы:



Эту строку можно записать словами:

КОГДА + Клавиши + WASD
ДЕЛАТЬ + Двигаться

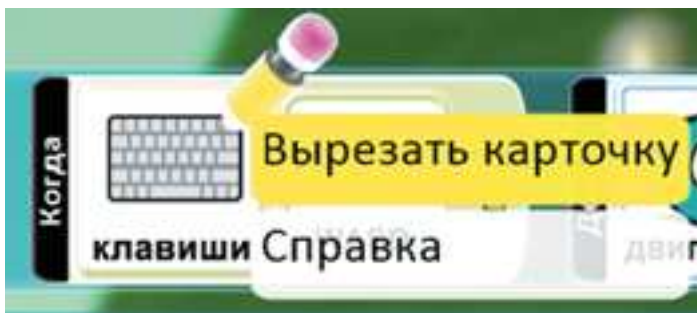
Выйдем из **Программы**, нажав клавишу **ESC** на клавиатуре. Запустим игру повторным нажатием **ESC** или кликнем на значке **ИГРАТЬ** ►.

Мы видим, что камера переместилась за «спину» **Байкера** и теперь при нажатии указанных клавиш он двигается.



2. Свободное движение. Этот способ позволит **Байкеру** двигаться самостоятельно. Для этого вернемся в **Программу**, выбрав инструмент **ОБЪЕКТ**, а затем нажав на персонажа *правой* кнопкой мыши и выбрав команду **Программа**.

При самостоятельном движении персонажа уже не нужно указывать клавиши для управления движением. Поэтому нажмем *правой* кнопкой мыши на карточки **КЛАВИШИ** и **WASD** (или **СТРЕЛКИ**) и удалим их, выбрав команду **Вырезать карточку**:



Чтобы **Байкер** двигался самостоятельно, напомним следующую строку программы:

КОГДА + Всегда

ДЕЛАТЬ + Двигаться + Свободно



Запустим игру и посмотрим, что получилось. Действительно, **Байкер** теперь перемещается по миру самостоятельно.



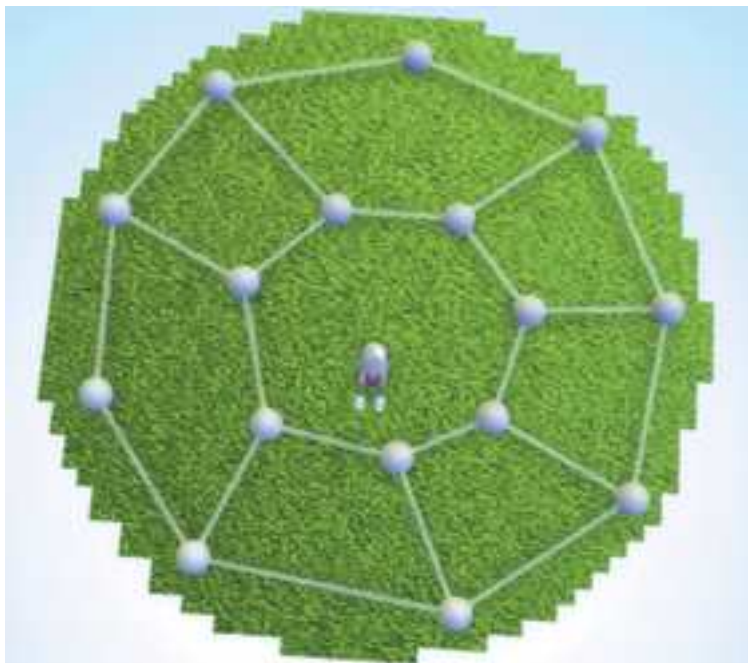
3. Движение по путям. Этот способ движения также позволит **Байкеру** двигаться самостоятельно, но на этот раз мы зададим ему конкретный путь перемещения.

Для реализации этого способа необходимо создать путь. Выйдем из **Программы** и выберем инструмент

ПУТЬ  .

Кликнем *левой* кнопкой мыши там, где будет начинаться путь (маршрут). При каждом нажатии будут образовываться «шарики», так называемые **УЗЛЫ** пути. Их можно передвигать, а также добавлять новые к уже имеющимся.

Изобразим путь, по которому **Байкер** будет перемещаться:



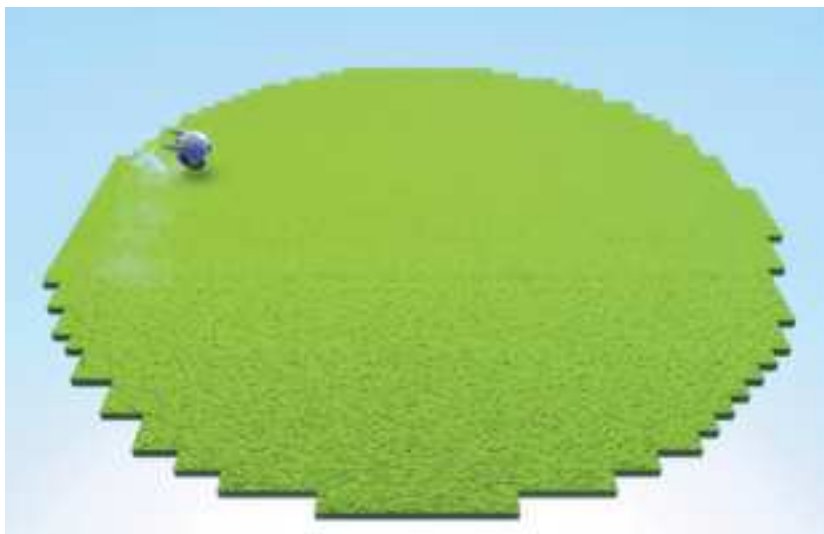
Чтобы закончить создание пути, кликнем *правой* кнопкой мыши в любой точке мира, и инструмент станет неактивным.

Теперь нужно научить **Байкера** ездить по этим путям. Для этого вернемся в **Программу** и отредактируем строку действий:

ДЕЛАТЬ + Двигаться + По путям



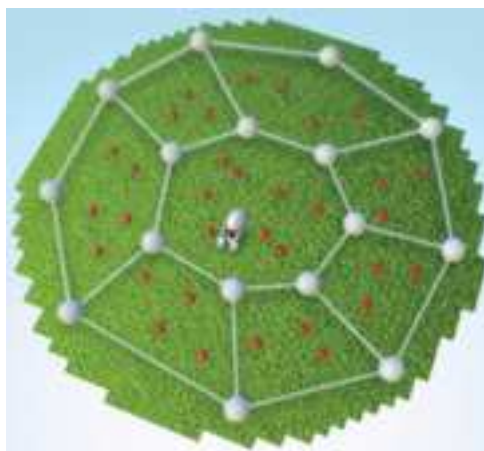
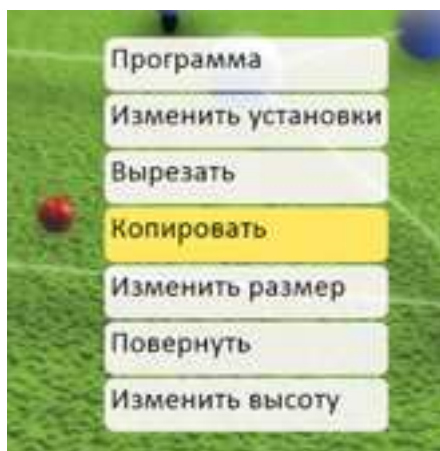
Запустим игру и посмотрим результат:



Как мы видим, созданный путь отображается в мире только в режиме редактирования, а в самой игре он не виден.

4. **Движение к цели.** Для этого способа нам понадобятся объекты для «поиска». Сделаем так, чтобы **Байкер** подъезжал, например, к яблокам и ел их.

С помощью инструмента **ОБЪЕКТ** создадим некоторое количество яблок и расположим их на земле в произвольном порядке.



Знаете ли вы, что...

существует способ «множественного копирования» объектов. Для этого достаточно создать, например, одно яблоко, нажать *правой* кнопкой мыши на него, выбрать команду **КОПИРОВАТЬ**, затем зажать клавишу **ALT** на клавиатуре и кликать *левой* кнопкой мыши там, где нужно добавить (вставить) скопированные яблоки.

Суть движения к цели состоит в том, что **Байкер** будет двигаться к объекту, который видит. Составим программу:

КОГДА + Вижу + Яблоко
ДЕЛАТЬ + Двигаться + К нему

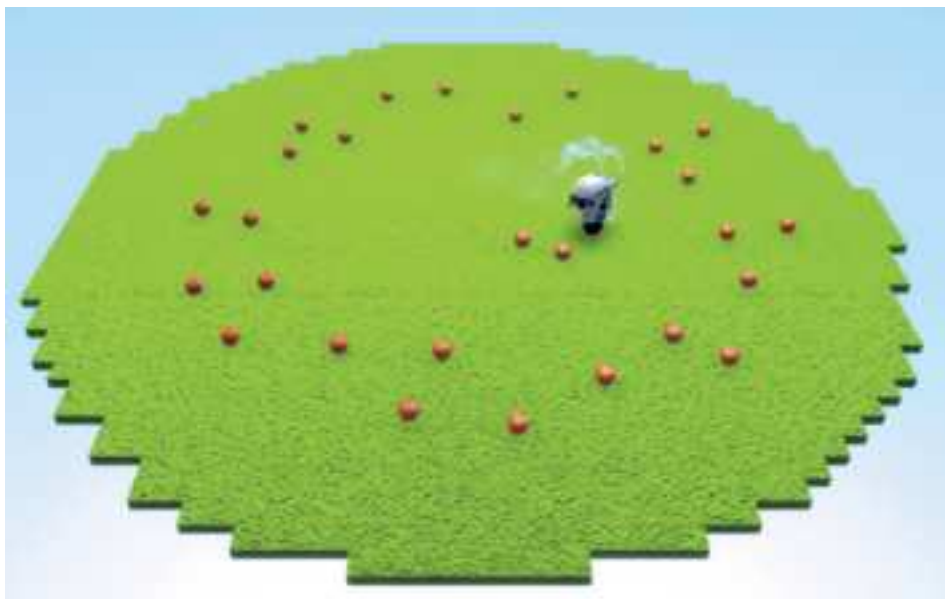


Теперь **Байкер** двигается к первому яблоку, которое он заметил, но к остальным не едет. Почему? Потому что яблоко должно исчезнуть, чтобы **Байкер** поехал к другому. Дополним программу второй строкой:

КОГДА + Касание + Яблоко
ДЕЛАТЬ + Съесть + Это



Теперь яблоки исчезают, и **Байкер** двигается до тех пор, пока не съест их все:



Сохраним игру под названием **Движение** (см. разд. 1.4).



2.2. Задания для самостоятельной работы

1. Создайте новый красочный мир с холмами и озером (или рекой).

2. На основе созданного мира разработайте игру, в которой на трех деревьях растут разные по цвету яблоки: красные, зеленые и черные. Главный персонаж **Коду** управляется *с помощью клавиш*. Если **Коду** съедает красное яблоко, то становится красным; съедает зеленое яблоко — становится зеленым; съедает черное яблоко — становится черным и говорит: «Фу! Это яблоко гнилое!»

2.3. Игра «Гонки»

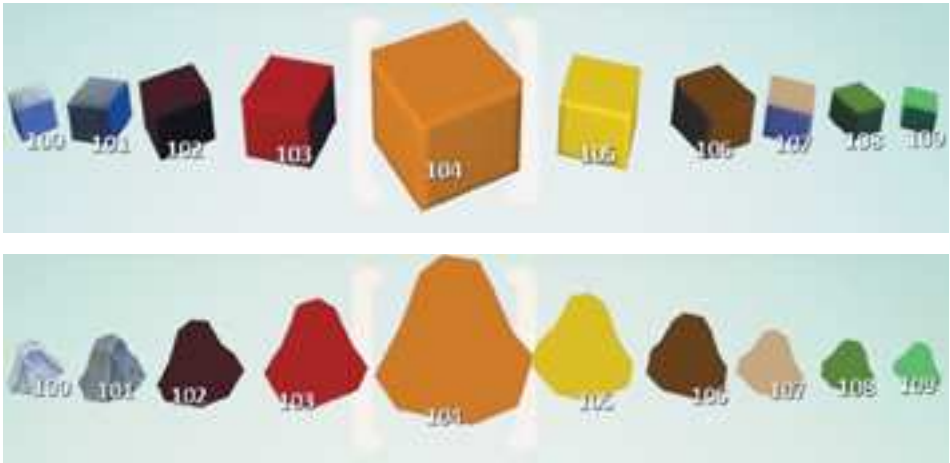
Создадим игру «Гонки», чтобы подробнее рассмотреть способы движения.

Сюжет игры: проводятся гонки на мотоциклах, мы играем за одного из **Байкеров**, а четверо других **Байкеров** проходят маршрут самостоятельно (компьютерные соперники). Гонки проходят на сложной трассе. Посмотрим, кто придет к финишу первым!

1. Для начала создадим территорию и маршрут для гонщиков. Стоит проявить фантазию: на местности должны встречаться холмы и даже препятствия. Гоночную трассу обозначим серым цветом земли, а места старта и финиша — черно-белыми полосами:



Кстати, любую территорию можно сделать гладкой в процессе рисования. Для этого выберем инструмент **КИСТЬ ЗЕМЛИ**, найдем подходящий цвет и нажмем на клавиатуре стрелку «вниз». Блок цвета сменится на небольшой холмик:



Такой кистью можно создавать гладкий ландшафт.

2. Теперь запрограммируем **Байкера**, за которого будем играть. Чтобы он отличался от других, выделим его особым цветом.

Откроем программу **Байкера**. Для начала пропишем, что он будет двигаться под управлением клавиш (*движение по клавишам*).



Далее добавим условие выигрыша и проигрыша. Условие можно сформулировать так: если первым пересек линию финиша **Байкер**, то он выиграл, а если соперник, то **Байкер** проиграл. Отметим цветом линию **финиша**, используя новый цвет земли, например красный:



Тогда в программе **Байкера** получится следующее условие:

КОГДА + Земля + Тип (47 — красная земля)
ДЕЛАТЬ + Победа



3. Настроим персонажей-соперников. Создадим одного из них (всего их будет четыре) и построим для него маршрут с помощью инструмента **ПУТЬ**:



Внимание!

Каждый компьютерный соперник будет иметь индивидуальный путь, поэтому при создании пути необходимо выбрать его цвет с помощью клавиш-стрелок **Влево** и **Вправо**. Мы создали **синий** путь для синего соперника.

Напишем соответствующую программу:

КОГДА + Всегда

ДЕЛАТЬ + Двигаться + По путям + Синий

КОГДА + Земля + Тип (47 — красная земля)

ДЕЛАТЬ + Конец

Кстати!

Подумайте, почему в конце второй строки мы поставили карточку **Конец**. Что получится, если пропустить карточку **Конец**?



Аналогично создадим программы еще для трех персонажей-соперников:



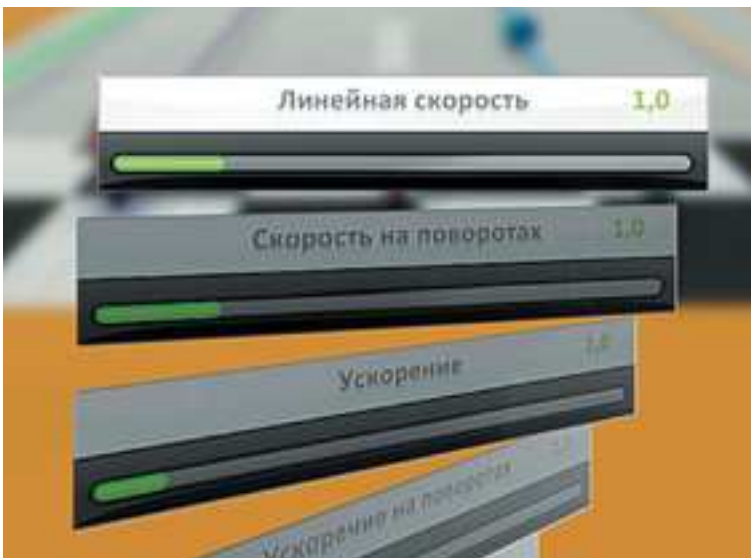
Подсказка. Обращаем внимание на шкалу справа на экране: она постепенно заполняется. С чем это связано? С тем, что игры в **Kodu Game Lab** используют оперативную память компьютера, и если мы расходует слишком много памяти, шкала становится красной. В этом случае игра будет либо «подвисать», либо вовсе не запустится. Эта шкала называется «Шкалой ресурсов». Нужно следить, чтобы она была заполнена как можно меньше.

Запустим игру и проверим ее работоспособность.



При желании можно установить персонажам более высокую скорость передвижения. Для этого нужно кликнуть *правой* кнопкой на персонаже (любого из **Байкеров**) и выбрать команду **Изменить установки**.

В списке первые параметры отвечают за скорость движения по прямой, скорость на поворотах и так далее. Эти параметры можно настроить по желанию.



Советуем не бояться и ставить скорость соперников выше, чем скорость игрока; в результате игра станет сложнее для прохождения.

Народная мудрость. Интересна та игра, которая одновременно и сложна в прохождении, и проходима.

2.4. Задания для самостоятельной работы

1. Создайте извилистую и холмистую трассу для гонок. Предусмотрите препятствия на дороге, а также невозможность «схитрить», то есть «срезать» трассу и добраться до финиша коротким путем. Игра не должна быть легкой.

2. Попробуйте реализовать состязание с другими участниками, а не только с **Байкерами**. Что из этого получится? Будет ли такая игра справедливой?

3. Создайте игру «**Перейди дорогу**», где действуют главный персонаж **Коду**, которым игрок управляет с помощью клавиш, и несколько **Байкеров**, которые двигаются по путям.

Цель игры: перейти дорогу, не коснувшись **Байкеров**. Если **Байкер** совершил наезд на **Коду**, то засчитывается проигрыш.

Пример игры:



Тема 3. Игры в жанре «Сражение»

3.1. Коду против Зámка

Народная мудрость. Лучший бой тот, который не состоялся.

Игры в жанре «Сражение» следует выделить в отдельную тему, так как в них необходимо учитывать следующие факторы:

- есть главный персонаж(и), у которого(ых) есть враг;
- у персонажа(ей) имеется определенное количество очков жизни;
- в процессе игры наносится определенное количество очков урона персонажам;
- есть возможность победить или проиграть.

Пример. Создадим демо-игру, в которой пользователь играет за **Коду** против объекта **Зáмок**.

Запустим программу **Kodu Game Lab**, создадим новый пустой мир и добавим в него **Коду** и **Зáмок**. Увеличим исходный размер земли так, чтобы **Коду** был на некотором расстоянии от **Замка**:

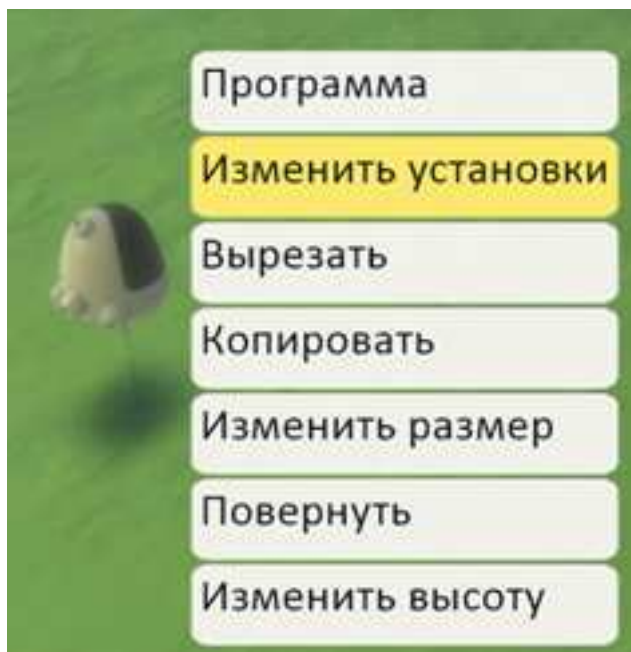


1. В этой игре мы будем управлять **Коду**, поэтому в **Программе** укажем ему *движение с помощью клавиш*:

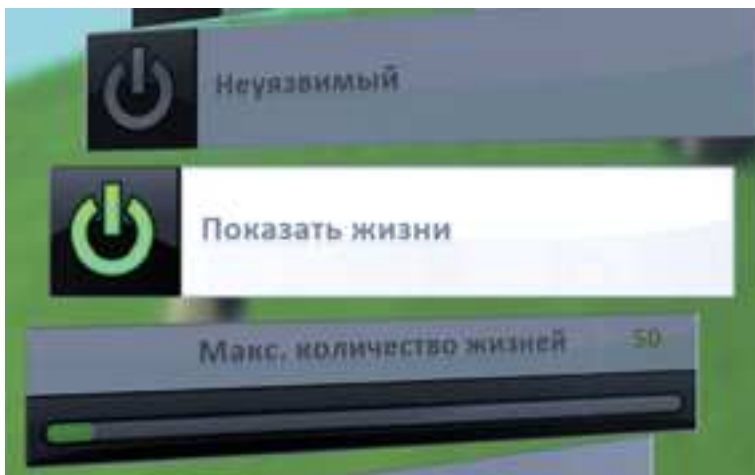


Замок будет стоять на месте, поэтому программу движения для него прописывать не нужно.

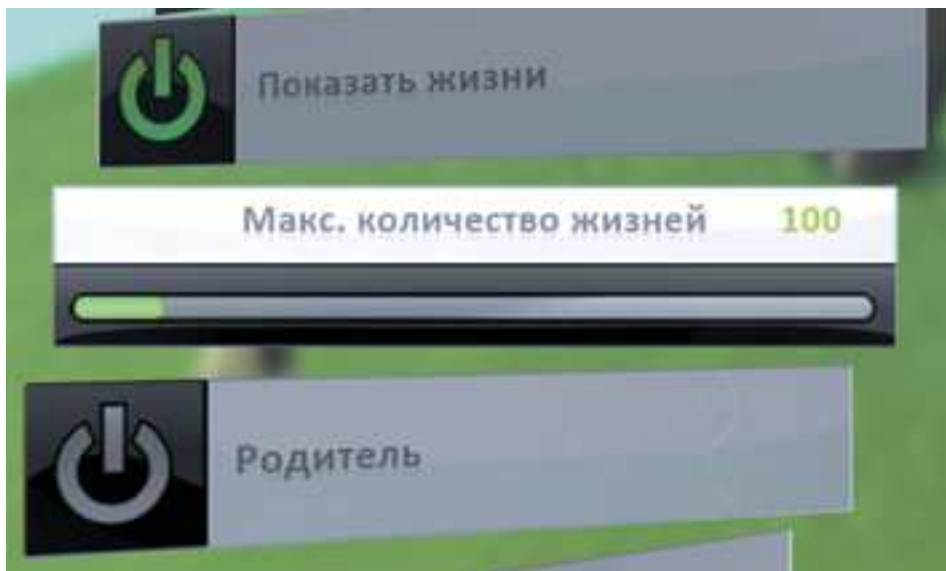
2. Теперь укажем очки жизни обоим персонажам, включив **полосу жизни** над их головами. Кликнем *правой* кнопкой мыши по **Коду** и выберем пункт **Изменить установки**.



Откроется меню настроек персонажа. Найдем в нем пункт **Показать жизни** и нажмем на значок «включения» этого параметра. Он загорится зеленым цветом:



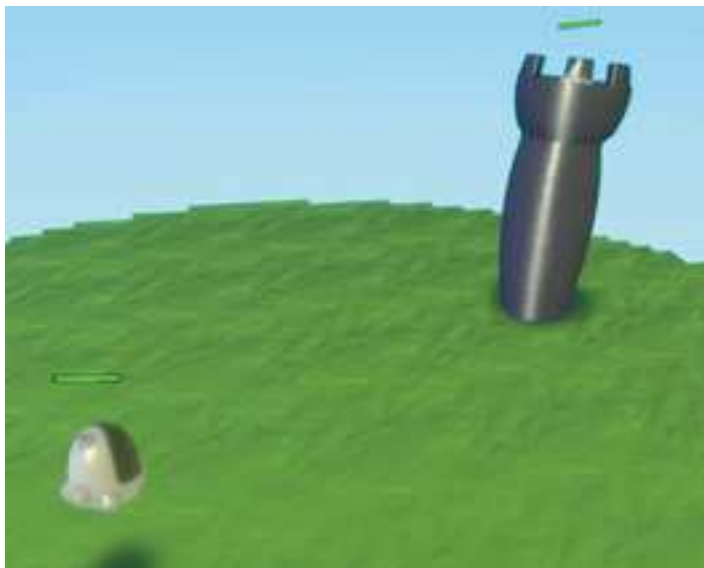
Далее нужно указать количество очков жизни. За это отвечает пункт **Макс. количество жизней**. С помощью стрелок на клавиатуре или левой кнопки мыши установим значение **100**:



После этого можно выйти из меню настроек, нажав клавишу **Esc**.

Проделаем то же самое с объектом **Зáмок**, но для него количество очков жизни установим равным **300**.

Вот как выглядит наше поле сражения:



3. Теперь реализуем сражение между персонажами. Пусть **Коду** будет стрелять **Пульками**, а **Замок** — **Ракетами**.

Добавим программу **Коду**:

КОГДА + Мышь + Левая
ДЕЛАТЬ + Стрелять + Пульки



Похожую программу напомним и для **Замка**, который стреляет в обнаруженного **Коду**:

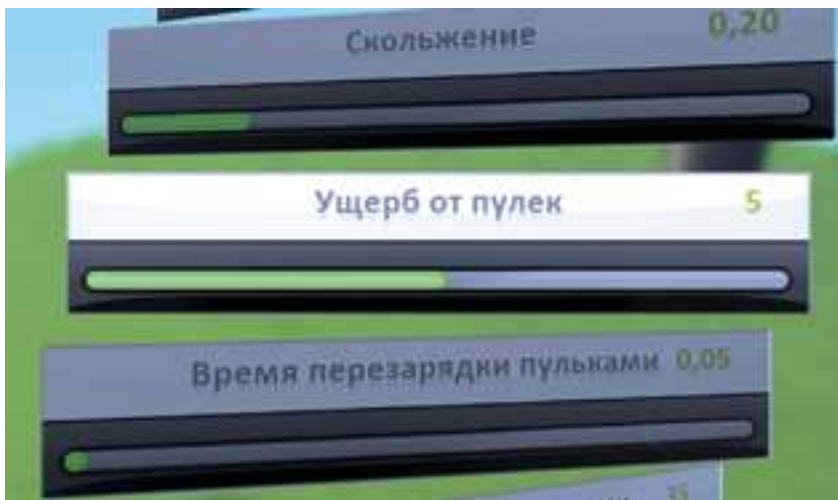
КОГДА + Вижу + Коду
ДЕЛАТЬ + Стрелять + Ракеты + Это



Запустим игру и посмотрим, что получилось. Обратим внимание на то, после скольких попаданий исчезает **Коду**? А **Зáмок**?

Теперь настроим выстрелы каждого персонажа, для чего снова зайдём в установки персонажей. Сначала настроим **Коду**, нажав на него правой кнопкой мыши и выбрав пункт **Изменить установки**.

Так как **Коду** стреляет пулями, то нам нужна группа настроек, отвечающая за выстрелы пулями. Прокручиваем список вниз и находим нужную настройку **Ущерб от пуль**:

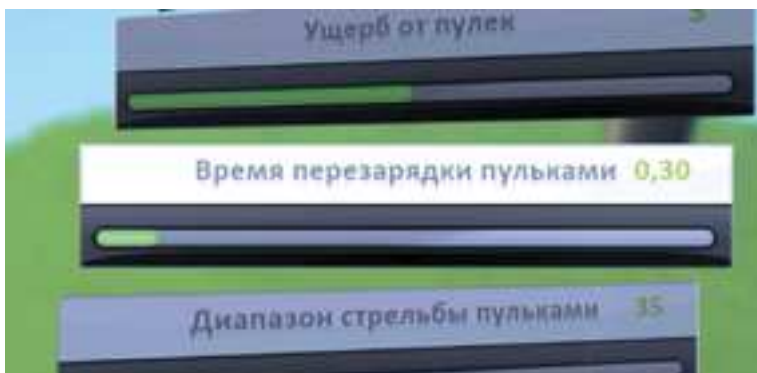


Вспомним, что для **Зáмка** мы установили **300 очков** жизни, а одна пуля даёт всего **5 очков** урона. Увеличим значение до **15 очков**.

Внимание!

Не путайте со значением –15. Отрицательные значения добавляют очки жизни противнику, «исцелят» его.

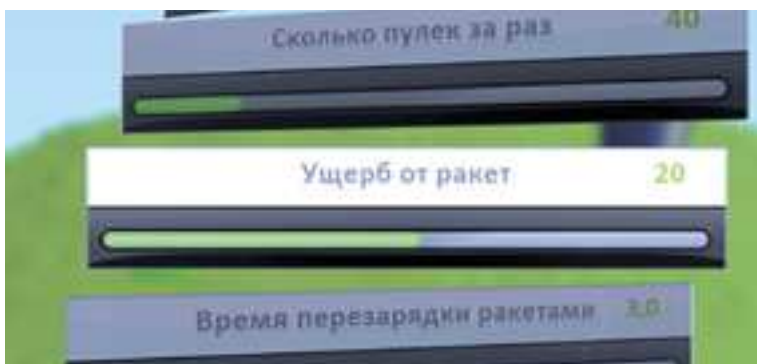
Как видим, пульки вылетают одна за другой слишком быстро, это тоже можно изменить в пункте **Время перезарядки пулями**. Установим значение на **0,30**, то есть пульки вылетают через 0,3 секунды. Это примерно 3 пульки в секунду:



Остальные пункты можно настроить по своему усмотрению:

- **Диапазон стрельбы пулями** — определяет расстояние, на которое может улететь пулька;
- **Скорость пуль** — устанавливает, насколько быстро пулька летит к цели;
- **Сколько пуль за раз** — максимальное число пуль, выпускаемое одной очередью.

Перейдем в установки **Замка**. Так как **Замок** стреляет ракетами, то найдем группу настроек, отвечающих за выстрелы ракетами. Изменим пункт **Ущерб от ракет** на значение **20**:



Запустим игру и посмотрим, что изменилось.



4. Игра почти готова, осталось настроить условия выигрыша и проигрыша.

Представим, что персонаж **Коду** пал в бою. Сколько очков жизни у него осталось? Верно, у него **0 очков** жизни. Гибель **Коду** — это проигрыш для игрока. Добавим новую строку в программу **Коду**:

КОГДА + Жизнь + Равна (equals) + 0 очков
ДЕЛАТЬ + Конец



Подсказка. Карточка **Жизнь** находится в разделе **Еще**; карточка **0 очков** находится в разделе **Баллы**; карточка **Конец** находится в разделе **Игра**.

Перейдем в программу **Замка**. Добавим аналогичное условие, не забывая учесть, что если падет **Замок**, то для игрока это будет победа.



Запустим игру и протестируем итоговый результат.

При желании можно изменить настройки по своему усмотрению, создать дополнительных соперников и/или союзников, разнообразив и усложнив игру.

Важно! Не забывайте периодически сохранять вашу игру, особенно по окончании работы в Kodu.

3.2. Игра «Утром спасение»

Давайте создадим новую игру, которая поможет нам применить полученные знания, — сделаем симулятор игры **ЗОМБИ-АПОКАЛИПСИС!**

Сюжет игры: персонаж **Байкер** ночью спасается от **Коду-зомби**, которые пытаются его догнать и укусить. Ночью **Байкер** не может нанести урон **Коду** и может только ждать, когда наступит утро. Когда наступает утро, все зомби исчезают.

Приступим к реализации игры. Сначала создадим главного персонажа — **Байкера** и составим для него

уже знакомую программу управления и условие проигрыша:



Важно! Не забываем добавить **Байкеру** полосу жизни и установить ее значение на **100**.

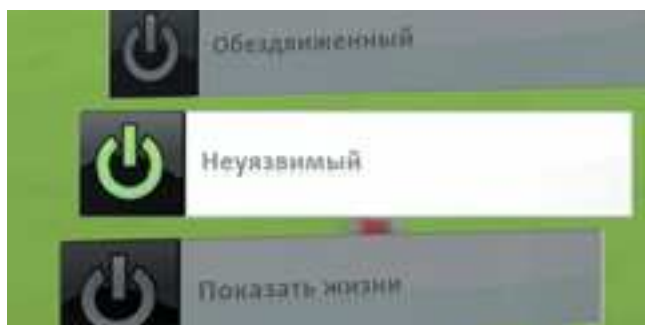
Теперь создадим одного персонажа **Коду** и напомним для него следующую программу:

- 1) ДЕЛАТЬ + Эмоции + Злой — так **Коду-зомби** будет постоянно злым, потому что отсутствие условия перед действием автоматически формирует условие «Всегда»;
- 2) КОГДА + Вижу + Байкер
ДЕЛАТЬ + Двигаться + К нему — погоня за **Байкером**;
- 3) КОГДА + Касание + Байкер
ДЕЛАТЬ + Ущерб + Это + 20 Очков — нанесение урона **Байкеру**:



Подсказка. Карточка **Ущерб** находится в пункте **Сражение**.

Наших зомби будет много, поэтому, чтобы они не наносили урона друг другу в толпе, в установках сделаем **Коду** неуязвимым:



Остается настроить наступление утра.

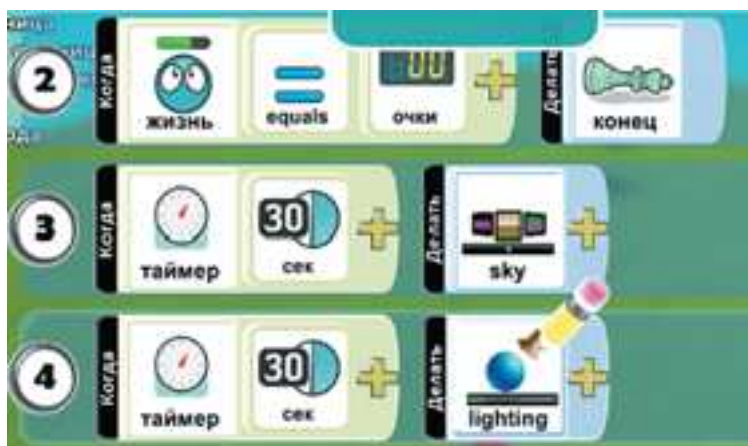
Условием смены дня и ночи будет время, поэтому в программу **Байкера** (кстати говоря, здесь выбор персонажа для записи программы не важен) добавим следующие строки:

3) КОГДА + Таймер + 30 сек

ДЕЛАТЬ + Небо (sky) — здесь нужно выбрать любое дневное небо;

4) КОГДА + Таймер + 30 сек

ДЕЛАТЬ + Освещение (lighting) — здесь нужно выбрать любое дневное освещение.



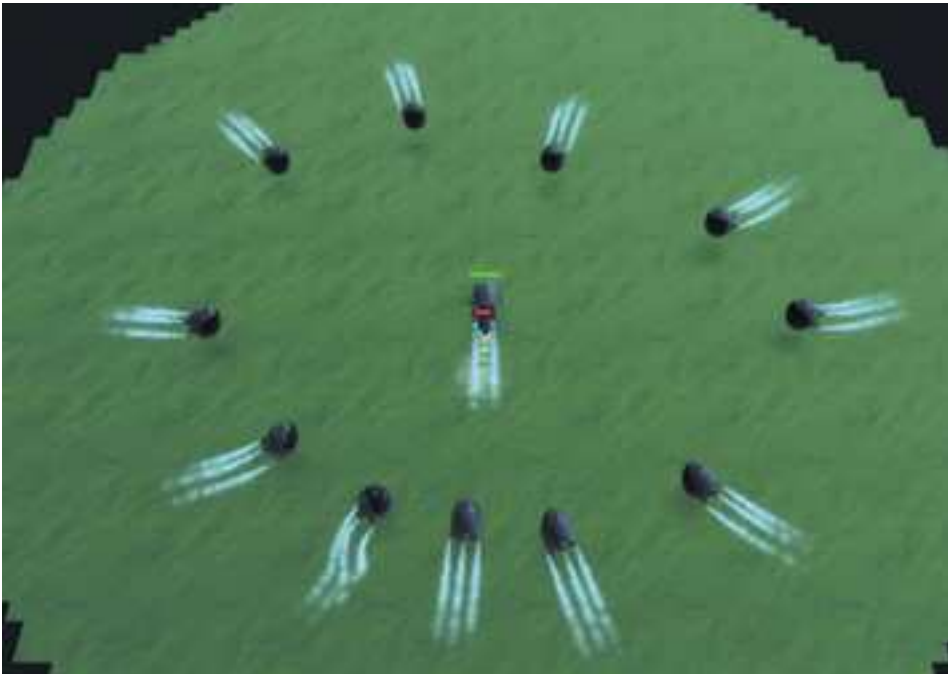
Важно! Не забудьте установить в начале игры ночное небо и освещение.

Теперь настроим программу **Коду-зомби** так, чтобы он исчезал при наступлении утра. Это условие тоже будет зависеть от времени:

4) КОГДА + Таймер + 30 сек
ДЕЛАТЬ + Нокаут + Я



С помощью множественного копирования создадим еще десяток таких же **Коду-зомби** и запустим игру:



3.3. Задания для самостоятельной работы

1. Самостоятельно настройте выигрыш в игре «Утром спасение» и сделайте апгрейд игры, например, чтобы **Байкер** мог отстреливаться от зомби. Украсьте территорию деревьями и другим декором, добавьте объекты, которые будут лечить персонажа — проявите фантазию!

2. Создайте игру «Дуэль», в которой будут сражаться два персонажа с равными возможностями. Например, в новом мире создайте персонажей **Коду** и **Байкера** на небольшом расстоянии друг от друга:



Настройте персонажей следующим образом:

Коду: управление с помощью клавиш, стреляет пулями с уроном в **5 очков**, каждые **5 секунд** создает камень, о который спотыкается **Байкер**.

Байкер: управление «Если вижу, то двигаюсь к...», стреляет пулями в **Коду** с уроном в **5 очков**.

Если погиб **Коду**, то в игре наступает проигрыш, если погиб **Байкер**, то — победа. Количество жизней у игроков должно быть одинаково.

3. Создайте игру «Коду против байкеров».

Игрок управляет **Коду**, у которого **200 очков** жизни и который стреляет пулями по **10 очков** урона.

На поле есть пять свободно передвигающихся **Байкеров**, у каждого из которых по **150 очков** жизни и который стреляет пулями по **20 очков** урона.

Еще есть **Босс-Байкер**, стоящий на месте до тех пор, пока не «увидит» **Коду**. **Босс-Байкер** в 2 раза больше обычного **Байкера**, у него **500 очков** жизни, и стреляет он ракетами по **30 очков** урона.

Помимо этого, в игре есть **Дерево**, на котором 1 раз в **30 секунд** вырастает целебное яблоко. Оно добавляет **50 очков** жизни тому, кто его съел.

С помощью рельефа земли устройте различные укрытия для **Коду**, отдельное место для **Босса-Байкера**, грамотно распределите по уровню простых **Байкеров**.

В качестве бонуса добавьте **невидимый** летающий объект **Тарелка**, который движется по заданному пути и случайным образом создает **Сердечки**, которые могут полностью излечить любого персонажа.

Подсказка. Объекты могут лечить других с помощью карточки **Лечить** в пункте **Сражение**.

Тема 4. Счетчики

4.1. Часы, прямой отсчет времени

Многие игры требуют применения различных счетчиков, в том числе для учета времени. В следующей **демо-игре** мы познакомимся с тем, как создавать счетчики. Они могут пригодиться нам в разных ситуациях, например, если мы захотим сделать игру на время или засчитать выигрыш игроку при наборе определенного количества очков.

Приступим!

Создадим новый мир и реализуем в нем **прямой и обратный отсчет** времени. Для этого добавим любой объект, который можно запрограммировать (например, того же **Коду**), и напишем программу для **прямого отсчета** времени:

- 1) КОГДА + Таймер + 1 сек
ДЕЛАТЬ + Очки + Синий + 01 Очко — здесь мы создали объект, который ведет отсчет начиная с нуля и прибавляет к счетчику по единице 1 раз в секунду;
- 2) КОГДА + Счет + Синий + Равен (equals) + 10 Очков
ДЕЛАТЬ + Победа — таким образом, игра будет завершена по прошествии **10 секунд** с момента ее начала.



Запустим игру и посмотрим, что получилось:



Логика программы состоит в следующем: каждую секунду добавляется очко в счетчик, и как только счет достигает 10, в игре наступает **Победа**.

4.2. Часы, обратный отсчет времени

Теперь посмотрим, как работает обратный отсчет. В отличие от прямого отсчета, в обратном необходимо совершить три действия:

- 1) установить значение, от которого мы будем считать числа в обратном порядке;
- 2) вычитать очки из счетчика каждую секунду;
- 3) когда счет достигнет нуля, в игре засчитать **Проигрыш**.

Для решения задачи сначала удалим созданную программу прямого отсчета времени. Кликнем на номере строки в программе персонажа *левой* кнопкой мыши, при этом на выбранной строке отобразятся стрелки перемещения. Далее нажмем на номер строки *правой* кнопкой мыши и выберем пункт **Вырезать строку**:



Таким же образом удалим вторую строку.

Теперь добавим новые строки в программу и проверим результат:

- 1) ДЕЛАТЬ + Установить счет (set score) + Красный + 10 Очков + Один раз — установка начального значения для счетчика;
- 2) КОГДА + Таймер + 1 сек
ДЕЛАТЬ + Вычесть + Красный + 01 Очко — вычитание единицы из счетчика 1 раз в секунду.
- 3) КОГДА + Счет + Красный + Ниже + 0 Очков
ДЕЛАТЬ + Конец — конец игры по достижении счетчиком нуля:



Почему же мы не поставили знак равенства в третьей строке, как в предыдущей программе? Получилась бы фраза: когда счетчик **равен 0**, то наступает **Проигрыш**. Все дело в том, что, если поставить знак «=» вместо сравнения **Ниже**, программа будет моментально выдавать **Проигрыш**. Это происходит из-за того, что с запуском игры сама среда **Kodu Game Lab** не успевает переключить счетчик с нуля на другое число. Поэтому в начале игры всегда, на очень малую долю секунды, мы имеем значение **0**. Карточка **Конец** реагирует на него и поэтому срабатывает еще в начале игры. Чтобы этого не происходило, мы используем

сравнение **меньше 0** для наступления **Проигрыша**. Отсюда и строка программы: *когда счет ниже нуля, то наступает Проигрыш*.

Кстати!

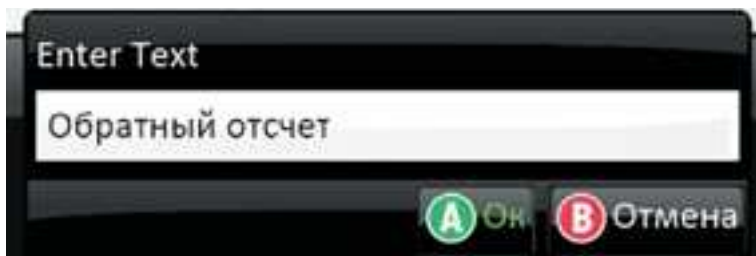
В Kodu Game Lab можно добавлять имя к создаваемому счетчику. Чтобы это сделать, нужно зайти в **ПАРАМЕТРЫ МИРА** и найти функцию **Вывод счета: [Цвет счета]**.



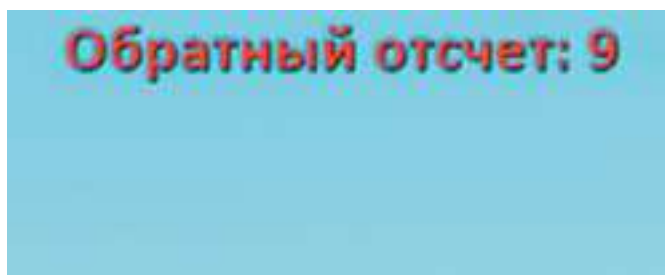
В зависимости от цвета созданного счетчика найдем наш цвет, например красный, и кликаем на пункте **Loud Labeled** (видимый подписанный счетчик):



Появится строка ввода, где мы можем написать название счетчика, например **Обратный отсчет**:



Теперь при запуске игры счетчик будет подписан:



Попробуйте сами в новом мире создать с помощью счетчика секундомер, который, как настоящие часы, будет считать секунды и минуты.

4.3. Задания для самостоятельной работы

1. Реализуйте игру «Поймай за время». Игрок управляет **Байкером**. Вокруг **Байкера** стоят **10 деревьев**. На каждом **дереве** в случайный момент времени вырастает яблоко. С начала игры идет обратный отсчет времени. *Цель игры:* за 30 секунд съесть как можно больше яблок (каждое съеденное яблоко добавляется в счетчик «Яблоки»). Когда **Байкер** съедает черное яблоко, его движения замедляются (карточка **Глухой**); когда съедает золотое яблоко, добавляется **2 очка** в счетчик «Яблоки».

2. Создайте игру «Гонки-2» (за основу можно взять созданную игру «Гонки»), в которой **Байкеры** соревнуются между собой за то, чтобы прийти первым к финишу. В этой игре трасса должна представлять собой замкнутую линию, где линия старта является одновременно и линией финиша. Задача игрока прийти первым на третьем круге. Также реализуйте прямой подсчет времени в игре.

3. Создайте игру «PaintBall».

Правила игры таковы: команда из пяти синих игроков играет против команды из пяти красных игроков. Цель каждой команды — попасть в противника выстрелом шарика краски. Вы играете, например, за команду красных (или любых других, цвет можно выбрать по своему усмотрению) и управляете только одним персонажем. Остальные члены вашей команды играют самостоятельно. Если вы попали во всех противников, то победа за вами; если от вашей команды никого не осталось, то проигрыш. Не забудьте сделать так, чтобы нельзя было «убивать» игроков из своей команды. Используйте знания счетчика в реализации этой игры.



Совет. Чтобы ускорить процесс создания игры, сначала запрограммируйте одного персонажа и лишь потом скопируйте его, получив таким образом команду.

Тема 5. Дороги и стены

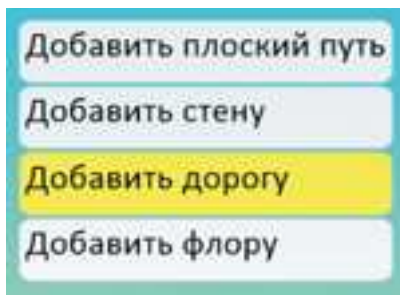
5.1. Подробнее о путях

Мы уже умеем настраивать движение персонажа с помощью инструмента **ПУТЬ**. Но этот инструмент пригодится и для создания дорог, стен и даже прорисовки флоры.

Кстати!

Под созданием флоры в **Kodu Game Lab** понимается создание цветочного поля.

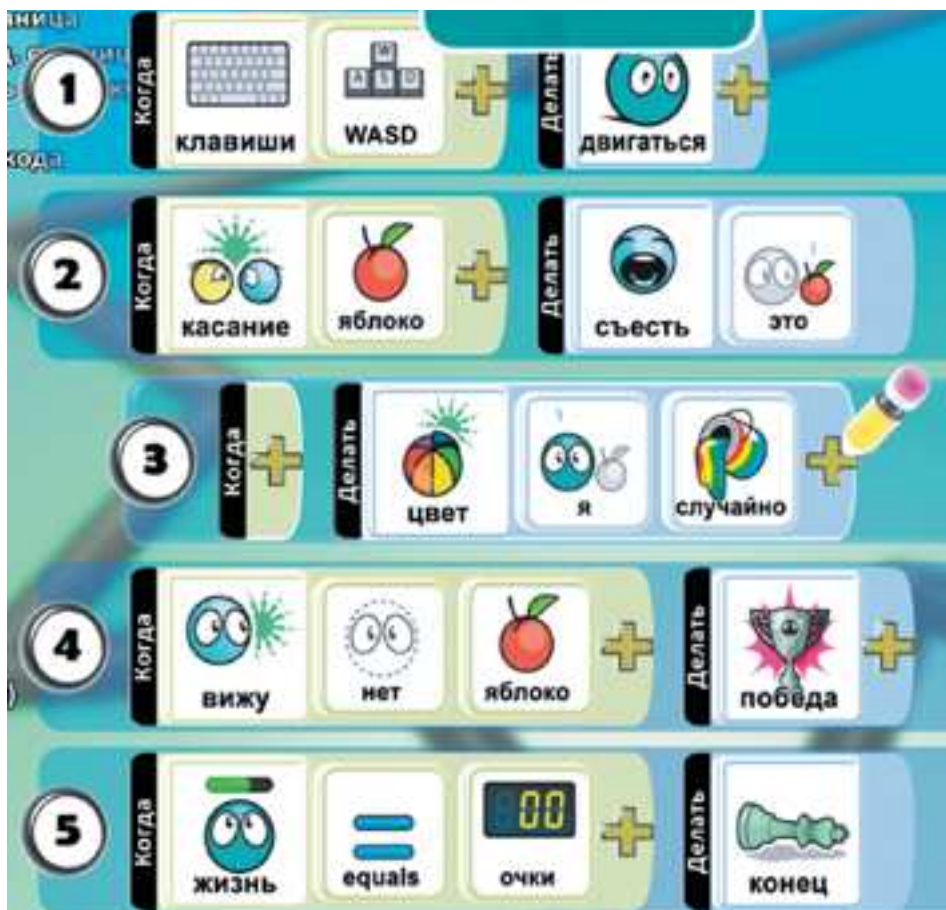
Рассмотрим возможности инструмента **ПУТЬ** на примере. Создадим новый мир и удалим участок земли, который дан по умолчанию, — он нам не пригодится. Выберем инструмент **ПУТЬ**, кликнем *правой* кнопкой мыши в любом месте мира и выберем из меню пункт **Добавить дорогу**:



Используя правила создания путей, нарисуем дорогу произвольной формы, например в виде звезды:



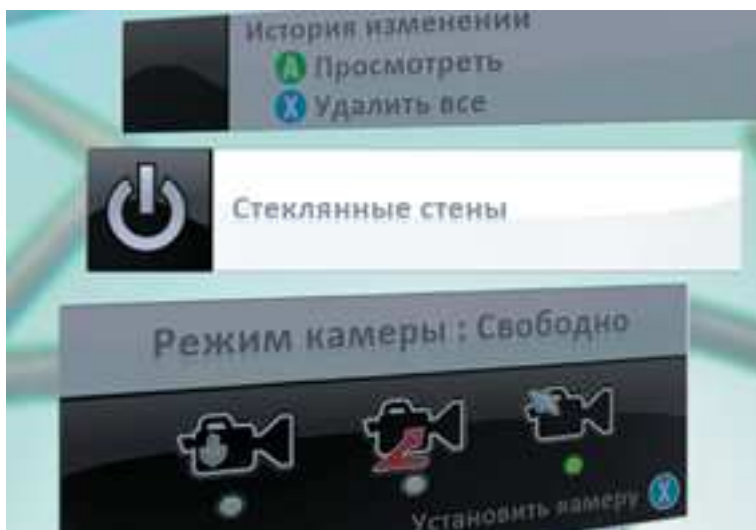
Воспользуемся этой дорогой для какой-нибудь не-сложной демо-игры. Например, создадим **Байкера**, который будет собирать по дороге яблоки и каждый раз менять свой цвет на новый:



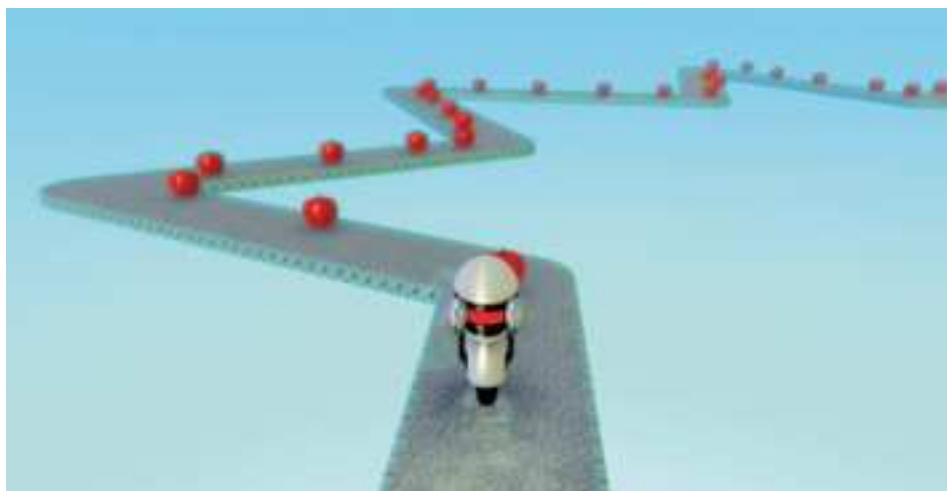
Внимание!

Чтобы сдвинуть строку вправо, нужно кликнуть и за-жать *левую* кнопку мыши на номере строки, а затем «перетащить» всю строку вправо. Зачем так делать, вы узнаете немного позже.

Чтобы игра стала интереснее, добавим **Байкеру** воз-можность упасть. Для этого в **ПАРАМЕТРАХ МИРА** отключим пункт **Стеклянные стены**:



Добавим несколько яблок на дорогу, а затем запустим и протестируем игру.



Теперь **Байкер** движется по дороге, но может упасть, если съедет с пути. Каждое съеденное яблоко меняет его цвет.

Что нового мы заметили в программе этой демо-игры? Верно, строка № 3 сдвинута вправо, а строка № 4 имеет необычную карточку **Нет**.

Разберемся с этим подробнее!

5.2. Наследование. Родительские и дочерние действия

Дело в том, что строка № 3 является дочерней по отношению к строке № 2. Сдвинутая вправо строка становится зависимой от родительской (предыдущей) строки и повторяет условие **КОГДА**. Ниже на первой картинке мы видим обычный способ составления программы, на второй — с использованием **Наследования**.



Обе программы имеют одинаковый результат, однако **Наследование** позволяет уменьшить затраты памяти в компьютере (так как сокращается количество команд для исполнения) и в целом ускоряет процесс создания программы и игры.

Если представить условия-наследования в виде предложения, то получится:

«КОГДА + <условие>

ДЕЛАТЬ + <действие1> **И** <действие2>»

Точно такие же действия можно выполнять и с условиями. Например, мы создаем игру, в которой для выигрыша нужно собрать **10** яблок и затем прыгнуть в озеро. Тогда для программирования выигрыша нужно написать следующую программу с использованием **Наследования**:



Если представить эту программу в виде предложения, то получится:

«КОГДА + <условие1> **И** <условие2>
ДЕЛАТЬ + <действие>»

В этом случае победа будет наступать только тогда, когда выполнены *оба условия одновременно*.

5.3. Отрицание

Посмотрим на два примера программ и поймем, в чем различие двух условий.



Первое условие:

КОГДА + <вижу свет>

ДЕЛАТЬ + <двигаться к нему>,

второе условие:

КОГДА + <НЕ вижу свет>

ДЕЛАТЬ + <грустить>

Добавление карточки **Нет** позволяет отрицать условие, то есть *инвертировать* его (менять условие на противоположное).

Так мы можем придумывать и реализовывать новые игры.

5.4. Задания для самостоятельной работы

1. Создайте лабиринт с помощью стен. Запрограммируйте любого управляемого персонажа так, чтобы игрок успел пройти лабиринт за заданное количество секунд. В качестве условия выигрыша используйте землю, раскрашенную в красный цвет.

2. Усовершенствуйте игру — спрячьте в лабиринте несколько артефактов, которые игрок должен найти. Воспользуйтесь в своей игре функцией **Наследование**.

Пример внешнего вида игры:



Подсказка. Чтобы усложнить прохождение игры, добавьте строку «камера от первого лица»:



3. Создайте модель города с сетью дорог, зданиями, парком, пляжем и другими объектами окружающей среды. Подумайте, как расположить жителей этого города и транспортные средства. Игрок здесь — турист, который осматривает город. Проявите фантазию и максимально продумайте детали. Какие задания мог бы выполнить игрок в вашем городе?

Тема 6. Страницы программ

6.1. Меняем поведение персонажей

Раньше мы не рассматривали в программах такое понятие, как **страницы**. Все программы мы описывали на так называемом листе или странице № 1:



Зачем же нужны страницы № 2, 3 и так далее?

Каждая новая страница в программе — это принципиально новое поведение персонажа. Так, если на первой странице персонаж собирал яблоки, то при переходе на вторую страницу он забудет про яблоки и будет делать что-то новое.

Для примера создадим демо-игру, в которой есть персонаж **Коду** и он живет в доме. Когда в мире день, он выходит гулять на улицу, а когда наступает ночь — возвращается домой.

Реализуем нашу задумку. Добавим **Коду** и запрограммируем его следующим образом:



Далее перейдем к странице № 2 и пропишем поведение **Коду** в ночное время суток:



В хижину добавим условия смены дня и ночи и небольшой бонус:

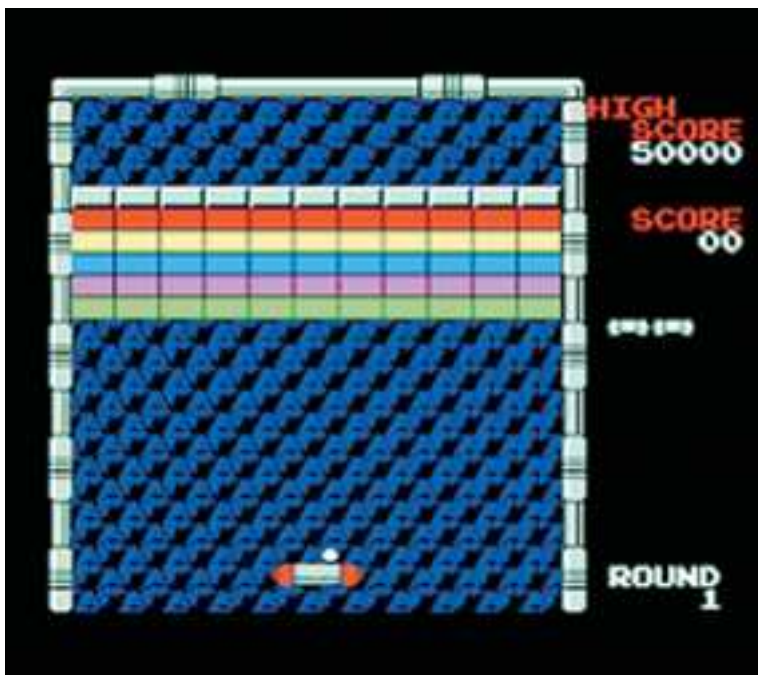


Запустим и протестируем игру.



6.2. Игра «Арканойд»

Рассмотрим еще одну игру, идея которой возникла в 1986 году. Называется она «Арканойд»:



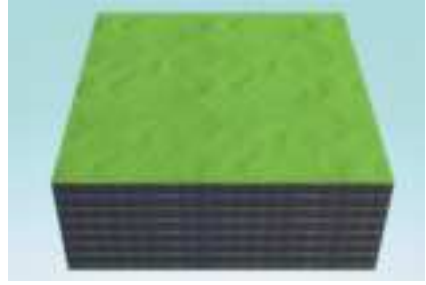
Главный персонаж — платформа, которая перемещается влево и вправо с помощью стрелок. Есть шарик, который двигается сам и сбивает объекты.

Цель игры: управлять платформой так, чтобы шарик сбил все объекты, не пролетев мимо платформы.

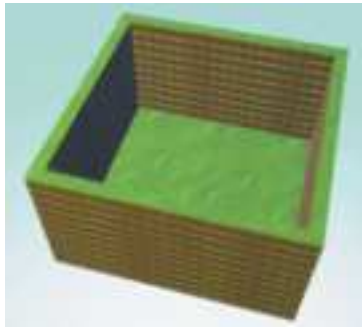
Приступим к разработке!

На основе игры «Арканойд» создадим свою, в которой будут действовать персонажи **Тумба**, **Шайба** и **Камни**. Игрок управляет тумбой, которая не дает шайбе упасть мимо. Шайба в свою очередь отскакивает от тумбы и сбивает камни.

В новом мире создадим небольшой участок земли квадратной формы. Используя инструмент **ХОЛМЫ** и форму кисти **Волшебная кисть**, немного приподнимем уровень земли.



Теперь сформируем стены. Это можно сделать уже известным образом с помощью дополнительной функции инструмента **ПУТЬ**. Или можно взять инструмент **ХОЛМЫ** и, глядя на участок строго сверху вниз (перпендикулярно), убрать внутреннюю область с помощью уменьшенной кисти квадратной формы. В результате должно получиться примерно так:



Пришло время добавить персонажей в игру.

Добавим **Тумбу** и расположим ее вдоль южной стены. Ее размер должен быть равен **1,8**. Полосу земли под тумбой окрасим в красный цвет.



Как найти южную стену? Все просто. В нижнем правом углу есть компас, указывающий направление и позволяющий ориентироваться в пространстве.

Стрелка компаса всегда указывает на север (N — Nord). В этой игре нам важно расположить камеру так, чтобы север оказался сверху.



Запрограммируем **Тумбу**. Она двигается вдоль южной стены, и когда все камни будут сбиты, игрок побеждает:

1) КОГДА + Клавиши + WASD

ДЕЛАТЬ + Двигаться + Вост/зап — таким образом, мы разрешаем тумбе движение только по горизонтали (влево и вправо);

2) КОГДА + Вижу + Камень + Нет

ДЕЛАТЬ + Победа — когда камней в игре не остается, засчитываем победу;

3) КОГДА + Счет + Красный + Ниже + 0 Очков

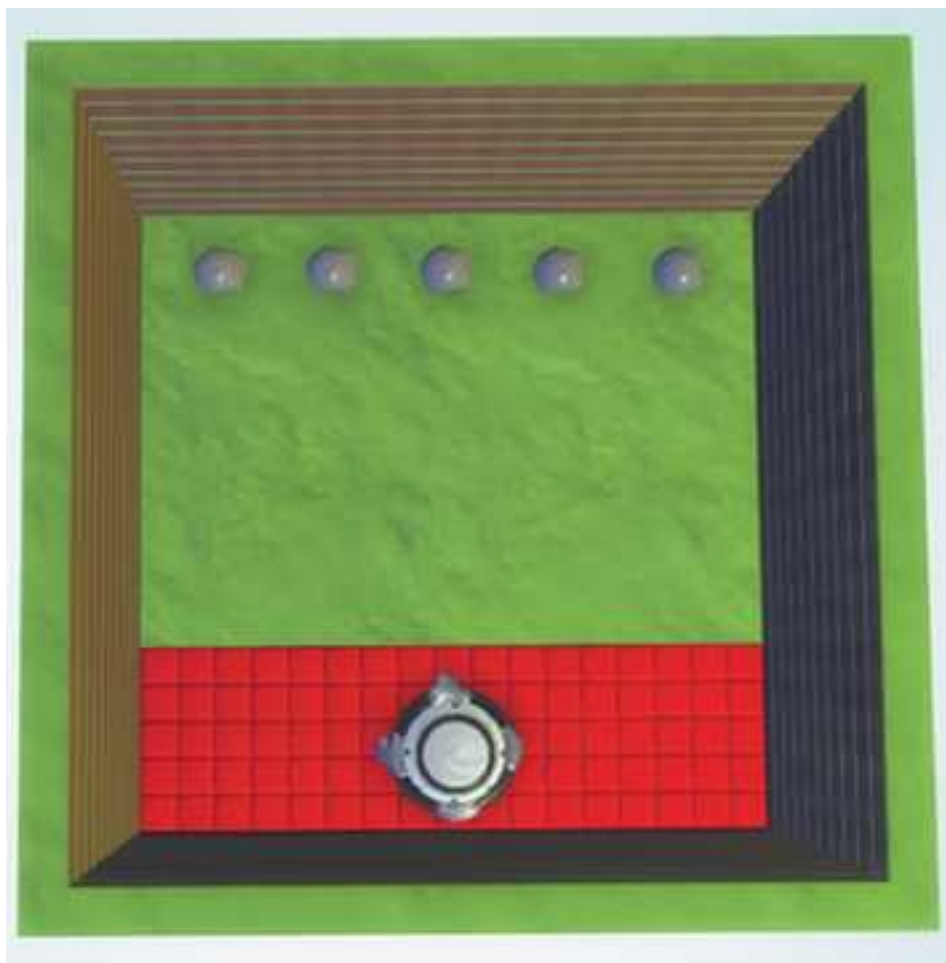
ДЕЛАТЬ + Конец



У северной стены поставим камень и запрограммируем его так, чтобы каждые **10 секунд** он создавал свою копию:



Теперь скопируем запрограммированный камень и размножим его до пяти штук.



Осталось создать **Шайбу** по центру территории. Ее цель — самостоятельно передвигаться вверх и вниз, «отпрыгивая» от **Тумбы** (два разных поведения на двух разных страницах). Запишем программу для нее:

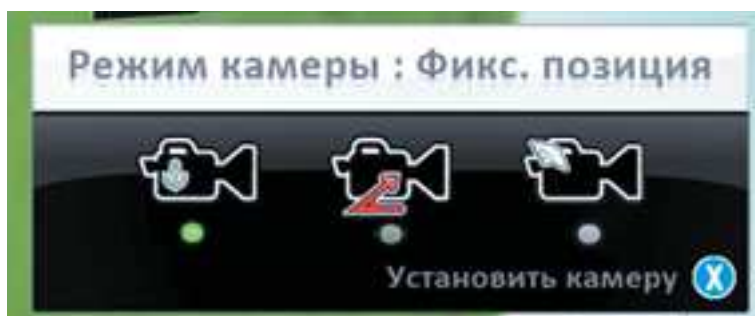
Поведение 1 (движение вперед): **Шайба** двигается вперед. Если она сбивает камень, засчитываем одно очко, при этом шайба двигается назад, то есть переходит на Поведение 2. Если **Шайба** не попала в камень и «задела» красную полосу, то вычитаем одно очко из счетчика.



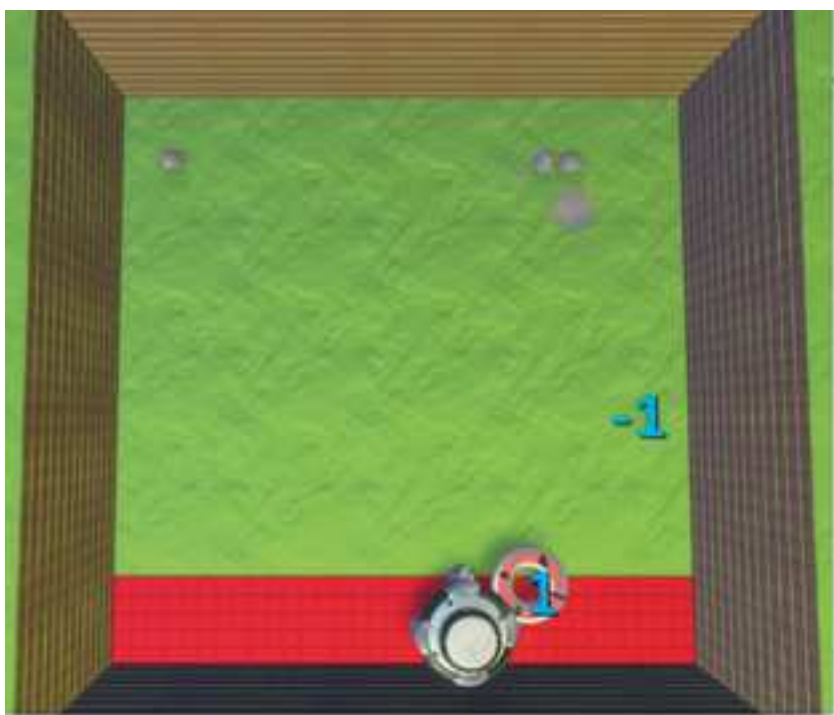
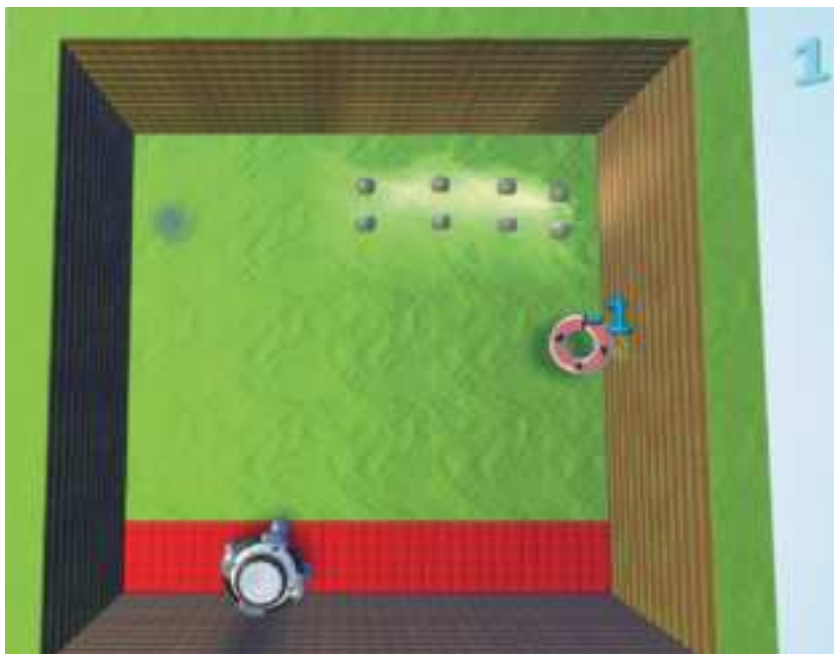
Поведение 2 (движение назад): **Шайба** двигается назад. Если она касается **Тумбы**, то снова начинает движение вперед, то есть возвращается на Поведение 1. Если **Шайба** касается красной полосы, мы также указываем на переход к Поведению 1, чтобы не прописывать вычет очков заново:



Не забудем также зафиксировать **Режим камеры** в **ПАРАМЕТРАХ МИРА**, чтобы следить за игрой с одного ракурса:



Запустим игру и протестируем ее.



6.3. Задания для самостоятельной работы

1. Добавьте в игру «Арканоид» дополнительные настройки по своему усмотрению. Например, можно сделать так, чтобы из камней выпадали различные бонусы, изменяющие ход игры.

2. Разработайте игру «Оборотень», в которой персонаж **Коду** имеет два поведения:

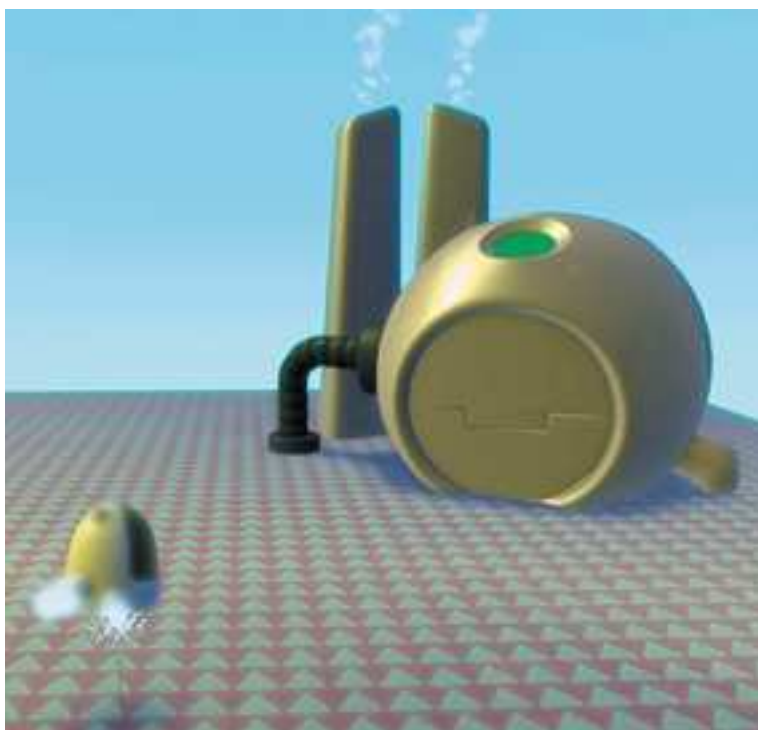
- днем он поддается управлению, ходит по миру и собирает яблоки;
- когда наступает ночь, **Коду** превращается в Оборотня (увеличивается в размере и переходит в эмоцию гнева) и больше не поддается управлению игроком. Через какое-то время снова наступает день, и все повторяется.

3. Усовершенствуйте игру «Оборотень». Пусть, например, на ночь управление передается другому персонажу, цель которого с помощью «волшебных» пуль превратить **Коду-оборотня** обратно в нормальное существо.

Тема 7. Возможности функции Родитель

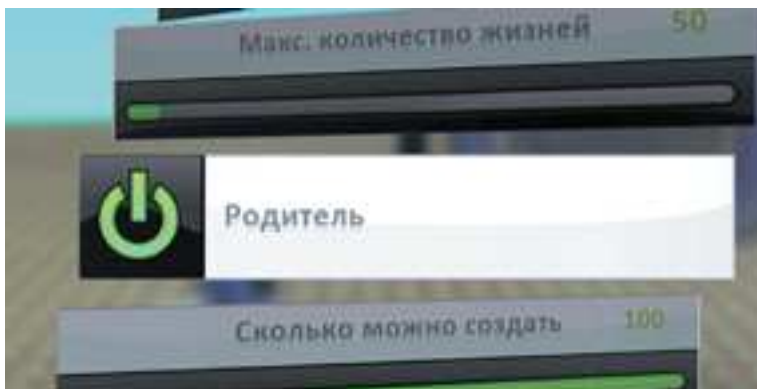
В **Kodu Game Lab** можно создавать персонажей не только с помощью инструмента **ОБЪЕКТ**. Любого персонажа можно запрограммировать так, чтобы он создавал себе подобных. За это отвечает функция **Родитель**.

Рассмотрим работу этой функции на примере демо-игры «Клон». Создадим небольшой мир и в нем поставим **Завод**. По сюжету игры он умеет клонировать персонажей по их «просьбе». Мы будем управлять **Коду**, цель которого создать себе одного клона.

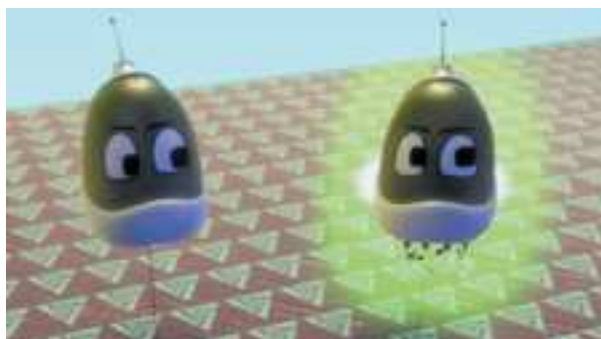


Создадим будущего клона: скопируем уже созданного **Коду** (в нем должна быть предварительно реализована программа движения по клавишам).

В скопированном **Коду** выберем команду **Изменить установки** и включим функцию **Родитель**.



Выйдя из меню установок, заметим, что около второго **Коду** появилось сияние, а если запустить игру, то он и вовсе исчезнет.



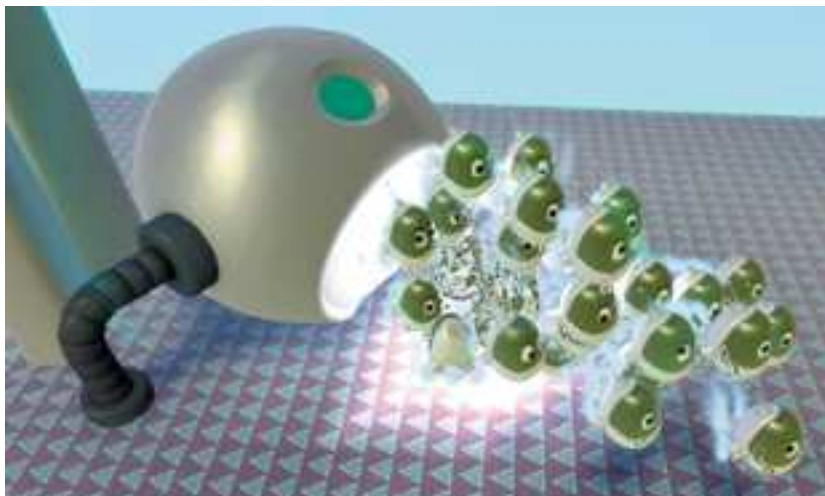
Это произошло потому, что **Коду** еще «не появился на свет».

Отсюда можно сделать вывод: функция **Родитель** включается для тех персонажей, которые *будут* созданы под влиянием других объектов.

В программе **Завода** при его касании с первым **Коду** (у которого нет функции **Родитель**) реализуем создание **Коду-клона**.



Запустим игру и посмотрим, что получилось:



Как видим, при контакте **Завода** с **Коду** появляется слишком большое количество клонов. Чтобы появлялся только один клон, в установках **Коду-клона** нужно задать значение параметра **Сколько можно создать** равным 1.

7.1. Задания для самостоятельной работы

1. Дополните игру: запрограммируйте **Коду-клона** на выполнение действий по вашему усмотрению. Например, пусть при появлении он скажет какую-нибудь фразу или подойдет к заводу и получит еще одного **Коду-клона**. Попробуйте придумать продолжение игры самостоятельно.

2. Создайте игру «Аквариум». Наполните мир водой и запустите в него рыбок. В том месте, где игрок кликает левой кнопкой мыши, появляются яблоки, камни или другие объекты, напоминающие корм. Рыбки плывут к появляющимся объектам и поедают их. Если рыбок не кормить, они со временем могут умереть.

Цель игры: накормить всех рыбок.

3. Рассмотрите функцию **Родитель** с точки зрения массового появления одинаково запрограммированных персонажей: создайте игру «Тир», где главный персонаж **Коду** стоит на месте, но может вращать головой. *Левая* кнопка мыши отвечает за стрельбу пулями. На расстоянии от **Коду** появляются персонажи трех видов из какого-либо объекта (например, рыбки, байкеры и летающие тарелки), которые двигаются по горизонтали.

Цель игры: попасть пулями во всех появившихся персонажей.

7.2. Игра «Рыбки»

Используем функцию **Родитель** и все наши знания на полную мощь — создадим серьезный проект «**Рыбки**»!

В данной игре мы будем управлять рыбой, которая может есть рыб своего размера и меньше, а от рыб большего размера должна уплывать, чтобы не быть съеденной. Съедая определенное количество рыбок, наша рыбка вырастает и становится сильнее.

Приступим!

Создадим новый мир и в нем стены. Заполним мир водой — получился аквариум.



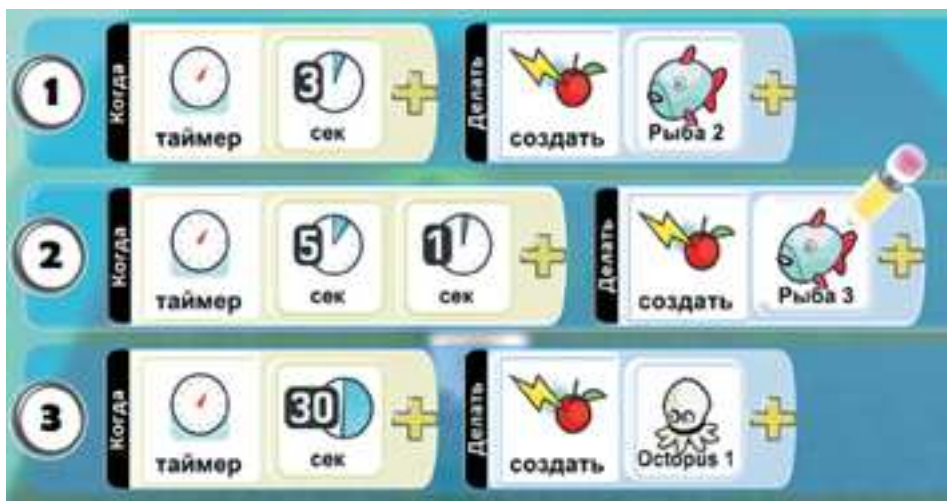
Добавим следующих персонажей:



В установках двух рыб и осьминога (octopus) включим функцию **Родитель**.

В этой игре персонажи с функцией **Родитель** будут появляться периодически, по таймеру, из неприметных глазу объектов. Добавим какой-нибудь неброский объект, из которого будут появляться наши персонажи. Например, таким объектом могут быть **Водоросли** (seagrass).

В программу **Водорослей** запишем строки, в которых по таймеру будут появляться персонажи-враги (будем называть их **Ботами**):



Мы будем играть за рыбку розового цвета. В игре она сможет пройти три этапа взросления: съела **три** маленькие рыбки — подросла, **пять** средних рыб — подросла. На третьем этапе будет засчитываться выигрыш через **10 секунд**.

Предлагаем реализовать игру самостоятельно, чтобы проверить свои знания и навыки в разработке игр.

Ниже представлены программы персонажей, но все же лучше попробовать создать игру самим, а затем сверить свой результат с нашим.

Подсказка 1. У розовой рыбки должно быть три разных поведения, а значит, **три страницы** программы.

Подсказка 2. Чтобы объект подрос, нужно найти карточку **Scale**, находящуюся в пункте **Settings**.

ПРОГРАММА ДЛЯ РОЗОВОЙ РЫБКИ

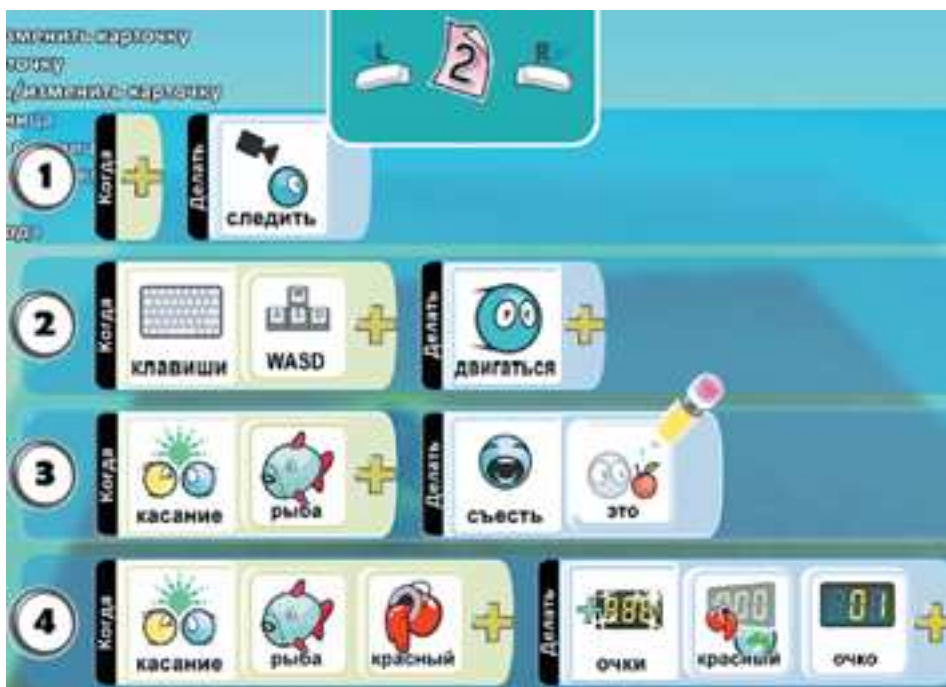
Страница № 1



Окончание Страницы № 1



Страница № 2



Окончание Страницы № 2



Страница № 3



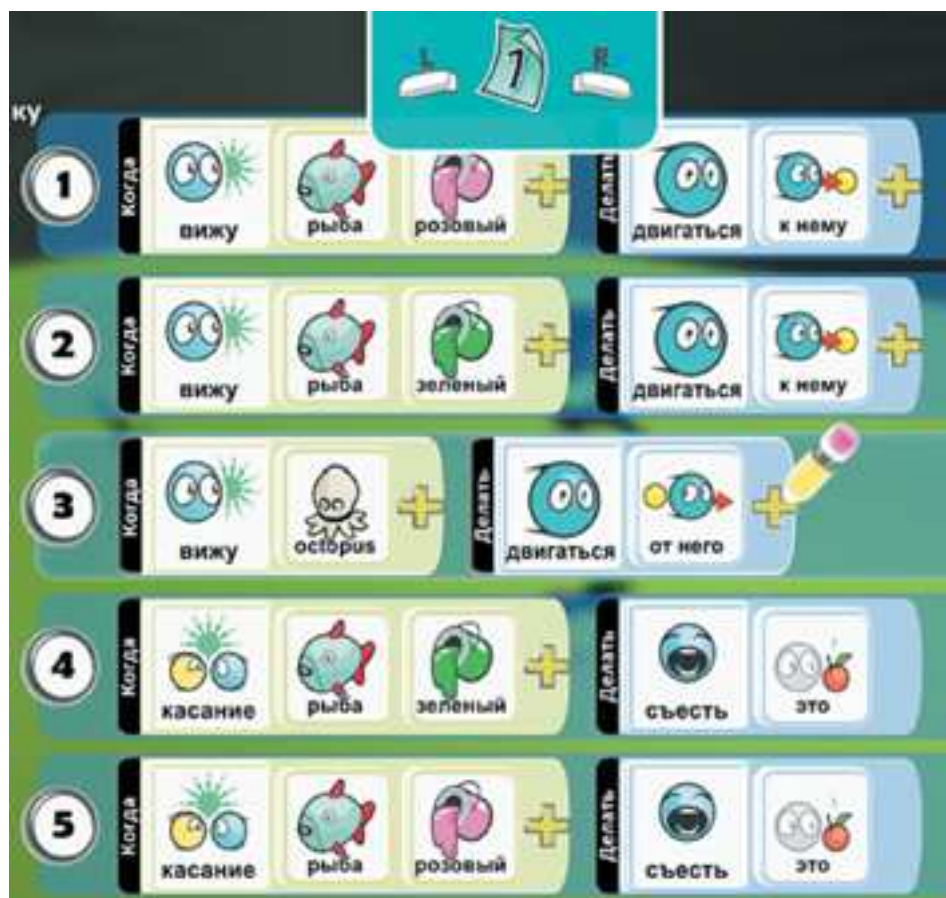
Окончание Страницы № 3



ПРОГРАММА ДЛЯ ЗЕЛЕННОЙ РЫБКИ



ПРОГРАММА ДЛЯ КРАСНОЙ РЫБКИ



ПРОГРАММА ДЛЯ ОСЬМИНОГА



Окончание Программы для осьминога



Кстати!

Для обеспечения более комфортного геймплея (процесса игры) можно зафиксировать положение камеры, сделав «вид сверху». Для этого в **ПАРАМЕТРАХ МИРА** нужно выбрать пункт **Режим камеры** и установить **Фиксированную позицию**. Затем нажать кнопку **Установить камеру**, вручную выставить камеру так, чтобы она «смотрела» на нашу рыбку сверху, и нажать клавишу **ENTER**.

7.3. Задания для самостоятельной работы

1. С помощью изученной функции **Родитель** создайте игру «Свет и тьма», в которой день сменяется ночью, а ночь — днем. Главный персонаж **Коду** днем выглядит обычно, а с наступлением ночи превращается в оборотня (с помощью **Родителя** появляется новый персонаж). Когда наступает день, он возвращается к прежнему виду. В обычном образе **Коду** управляется клавишами **WASD** и его задачей является сбор яблок, которые он несет в дом, за что получает очки «Свет».

В образе оборотня он управляется клавишами-стрелочками, собирает звездочки и относит их в озеро, за что получает очки «Тьма». Кто победит — свет или тьма?

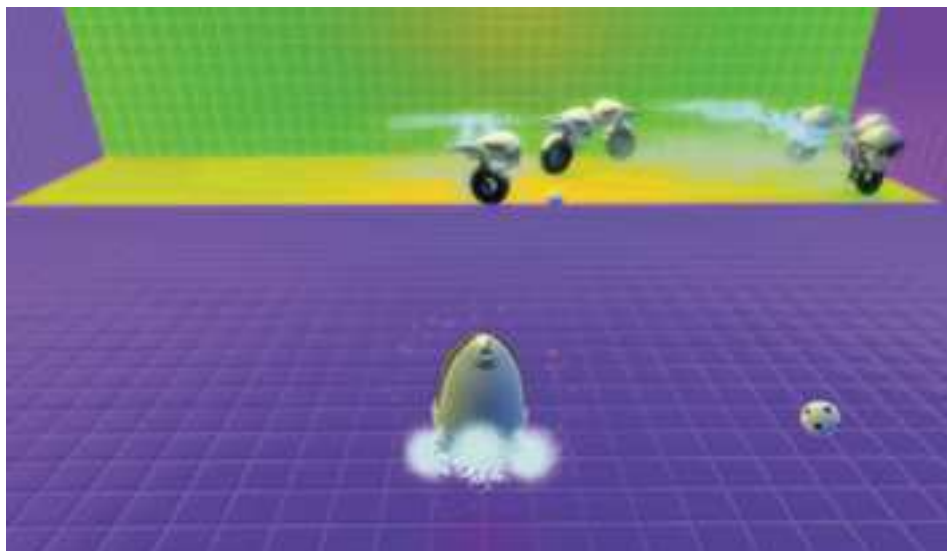
Подсказка. Функцию «подбирания» и «бросания» предметов можно реализовать путем программирования отдельных клавиш и группы команд **Держать**.

2. Разработайте игру «Вышибалы»:

Главный персонаж **Коду** стоит на месте, но может смотреть влево и вправо с помощью стрелок на клавиатуре. У **Коду** есть мяч, который он может пинать с помощью клавиши **Пробел**.

Перед **Коду** есть зона, в которой перемещаются **Байкеры** (появляются в разное время самостоятельно).

Цель игры: попасть мячом в **Байкеров**. При попадании в **Байкера** мяч исчезает, и у **Коду** появляется новый мяч. За **1 минуту** нужно попасть в **10 Байкеров**, тогда в игре засчитывается **Победа**.



3. Разработайте игру «Салки-хваталки».

Главный персонаж — **Байкер**, движется с помощью клавиш; «ловит» рыб клавишей **Пробел**; когда попадает на базу (зону, окрашенную в другой цвет) — отпускает рыбу («роняет»).



Летающие рыбы — персонажи, которые убегают от **Байкера** по суше. В мире каждые **3 секунды** из водорослей появляется **1 рыба**.

Цель игры: поймать всех летающих рыб и отнести на базу. Как только **Байкер** собрал **10 рыб** — в игре засчитывается **Победа**. Если **Байкер** не успел собрать **10 рыб** за **60 секунд** — в игре засчитывается **Прогрыш**.

Тема 8. Кнопки

8.1. «Кликер»

В **Kodu Game Lab** можно создавать кнопки — эти объекты используются самыми разными способами.

Посмотрим, как это выглядит. Создадим новый мир и добавим **Коду** в качестве главного персонажа. Для него запишем следующую программу:

КОГДА + Мышь + Левая + Зеленый (green)

ДЕЛАТЬ + Очки + Зеленый + 01 Очко



Подсказка. Карточка **Green** (зеленая кнопка) находится в пункте **Buttons** (кнопки).

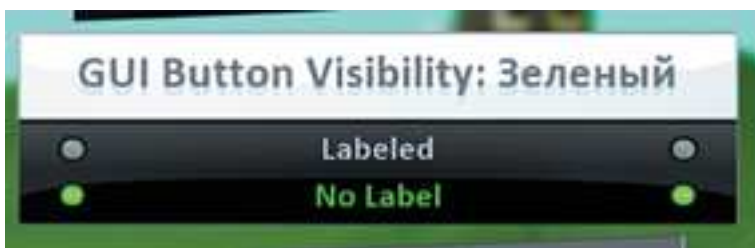
При запуске игры мы увидим большую кнопку, на которой можно кликать *левой* кнопкой мыши. Согласно заданной программе в счетчик (справа) будут добавляться очки.



Да ведь мы только что создали такую популярную игру, как «Кликер»! Интересно, в ней есть смысл?

Кстати!

Ранее мы научились записывать имя счетчику. С кнопками происходит примерно та же история. В **ПАРАМЕТРАХ МИРА** найдем функцию **GUI Button Visibility**: [Цвет кнопки].



При нажатии на пункт **Labeled** (подписанный) появится строка ввода, где можно записать название, например **Кнопочка**:



Теперь при запуске игры на кнопке появляется надпись:



Встречаются игры, в которых действие, происходящее при нажатии кнопки, не всегда заметно. Тогда можно использовать **скрытый счетчик**, о существовании которого игрок не сможет узнать в течение игры.

8.2. Скрытый счетчик

Изменим программу **Коду** на следующую:



Здесь после карточки **Очки** мы выбрали пункт меню **World Scores** и в нем выбрали букву **A**. В чем отличие **World Scores** от **Private Scores**?



Оба лепестка отвечают за создание скрытого счетчика. Но скрытым он является лишь для игрока, персонажи в игре знают значение на счетчике и легко ориентируются по невидимому счетчику. Если выбрать лепесток **World Scores**¹, то счетчик будет виден всем

¹ World Scores (англ.) — мировые очки (общедоступный скрытый счетчик).

персонажам и объектам игры. Если же выбрать **Private Scores**¹, то счетчик будет виден только тому персонажу, в программе которого этот счетчик был создан.

Рассмотрим пример игры, в которой можно задействовать скрытый счетчик. В ней мы зададим кнопке условие: *при нажатии менять день на ночь и наоборот*.

1. В новом мире добавим объект, например **Облако**. Для него напомним следующую программу, в которой для наглядности сначала применим видимый счетчик:



Так как кнопка будет «знать» всего два действия («включить» день или ночь), то и счетчик должен иметь только два значения. Если считать значения любого счетчика по порядку: 0, 1, 2, 3, 4, ..., то первые два значения — это 0 и 1. Таким образом, значение 2 в этой игре существовать не должно и после цифры 1 должна снова идти цифра 0. В результате получится набор значений: 0, 1, 0, 1, 0, 1, ..., и так до бесконечности.

2. Чтобы превратить значение 2 в значение 0 напишем следующую строку программы:

КОГДА + Счет + Синий + Равен (equals) + 02 Очка
ДЕЛАТЬ + Установить счет (set score) + Синий +
+ 0 Очков



¹ Private Scores (англ.) — личные очки (приватный скрытый счетчик).

Таким образом, как только счетчик захочет «перепрыгнуть» на значение **2**, программа вернет его к нулю.

Кстати!

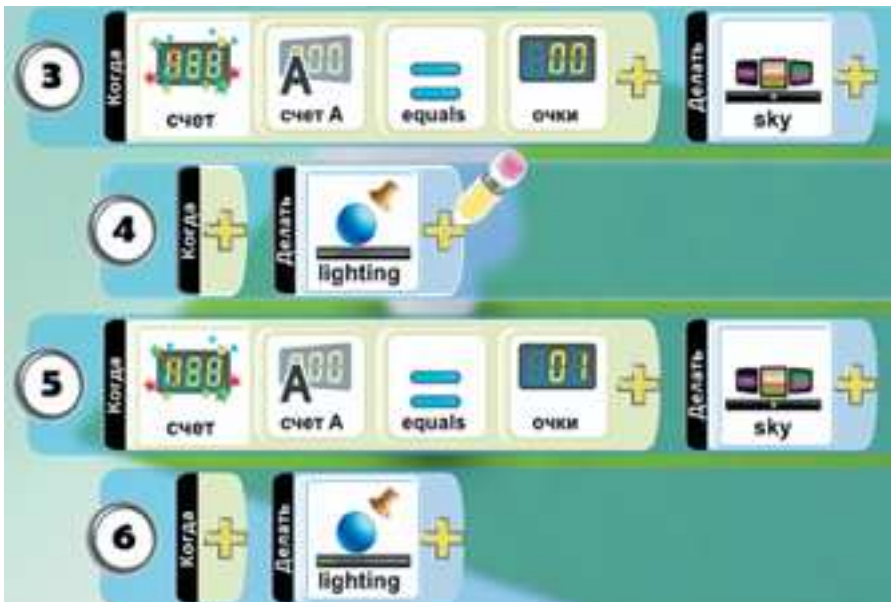
Аналогично будет работать карточка **Сброс** вместо карточки **Установить счет (set score)**.

Запустим игру и проверим, что получилось.

Отлично, в счетчике мы имеем два значения. Теперь изменим программу так, чтобы счетчик работал в скрытом виде:



Осталось настроить смену дня и ночи (когда счет равен **0**, то в мире наступает день, когда равен **01**, то наступает ночь):



Совет. Не настраивайте скрытый счетчик в программе главного персонажа игры, потому что программы главных персонажей и без счетчика могут быть достаточно громоздкими. Лучше задействовать под эти настройки отдельный объект.

8.3. Задания для самостоятельной работы

1. Добавьте персонажей в получившуюся игру, продумайте действия и цель игры самостоятельно.

2. Создайте игру «Аквариум-2», в которой при нажатии **кнопки** в аквариум будут падать яблоки, камни и другие объекты, напоминающие корм для рыбок. Остальные условия игры останутся такими же, как и в первой версии.

3. Попробуйте создать игру «Кликер», целью которой является нажатие кнопки на скорость. Придумайте правила игры, например «Успей сделать 100 нажатий за 30 секунд» или «Нажимай на кнопку, пока персонаж не дошел до красной зоны». Проявите фантазию.

Тема 9. Телепортация

Скрытый счетчик, кроме подсчета нажатий на кнопки, можно использовать для реализации других интересных и полезных вещей. Создадим демо-игру «Телепорт», в которой персонаж сможет перемещаться между двумя островами через замки, выполняющие роль порталов.

9.1. Телепорт

Приступим! В новом мире создадим два острова и на них разместим два **Зámка**: красный и синий.

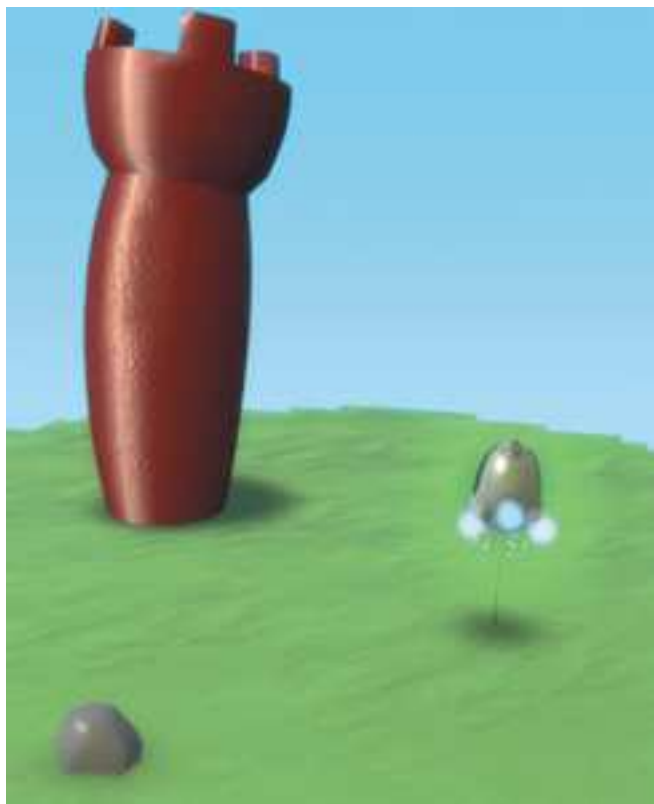


Замки будут перемещать персонажа, который зайдет в них. Телепортация будет происходить в два этапа:

- 1) персонаж зашел в **Красный замок** (исчез при касании);
- 2) персонаж вышел из другого, **Синего замка** (Замок создал его заново).

Нам предстоит «сказать» **Замку**, чтобы тот создал персонажа, поэтому понадобится функция **Родитель**.

Итак, сначала на первом острове нам нужно создать персонажа (пусть у нас это будет **Коду**) и включить для него функцию **Родитель**. А как же его создать? А вот как: «скажем» любому не приметному объекту, например **Камню**, создать **Коду** в начале игры:



Когда **Коду** войдет в **Красный замок**, сигнал об этом должен поступить **Синему замку**. Для этого будем использовать скрытый общедоступный счетчик.

ПРОГРАММА КАМНЯ



ПРОГРАММА КОДУ

Коду будет управляться с помощью клавиш **WASD**. При касании **Красного замка** скрытый счетчик устанавливается в единицу (одно очко), одновременно с этим персонаж становится невидимым (дочерняя строка программы):



ПРОГРАММА СИНЕГО ЗАМКА

Синий замок постоянно следит, установлен ли скрытый счетчик в единицу. Как только это происходит, он создает нового **Коду**.



9.2. Задания для самостоятельной работы

1. Реализуйте обратную телепортацию.
2. Реализуйте телепортацию между тремя **Замками**.
3. Создайте игру «Лабиринт телепортов», в которой игрок сможет пройти лабиринт только в том случае, если в правильном порядке войдет в нужные порталы.

Тема 10. Переключение между персонажами

Встречаются игры, в которых нужно управлять сразу несколькими персонажами. Удобнее всего это делать, когда есть некий **переключатель**.

10.1. Футбол

Реализуем игру «Футбол», где один игрок будет играть за целую команду сразу, а переключать персонажей можно будет с помощью кнопки. Для реализации нам снова пригодится *скрытый счетчик*.

Начнем! Создадим футбольное поле, выделим зоны, которые будут являться воротами, построим стены.



Цель игры: реализовать игру между двумя командами по пять игроков. Для начала настроим команду персонажей, за которую будет играть сам игрок, например пять **Коду** красного цвета (**Коду1**, **Коду2** и т. д.). Создадим одного из пяти персонажей красного цвета — **Коду1**, мяч с функцией **Родитель** и невидимый (в установках функция **Невидимка**) летающий объект (**Тарелку**), который будет отвечать за скрытый счетчик и кнопку переключения, а в целом будет выполнять функцию судьи на поле.



ПРОГРАММА НЕВИДИМОЙ ТАРЕЛКИ

Если мяч исчезает (забит гол или он улетел за пределы поля), тарелка создает новый мяч. Кроме того, в тарелке запрограммирован невидимый счет в кнопке синего цвета: если одна из команд набрала **5 очков**, то в игре соответственно засчитывается выигрыш или проигрыш.



ПРОГРАММА МЯЧА

Если мяч попал в ворота одной из команд, он исчезает и дает очко команде противника.



ПРОГРАММА КОДУ 1

Страница № 1

По умолчанию красный **Коду1** не двигается и ждет выбора кнопки. Как только кнопка указала **0** очков, включается поведение **Коду1**.



Страница № 2

Персонаж двигается с помощью клавиш, захватывает мяч, если касается его, и при нажатии клавиши «пробел» пинает мяч. Если кнопка перешла на другое значение (игрок решил сменить **Коду** через нажатие кнопки), **Коду1** возвращается в начальную позицию и становится неподвижен (поведение по умолчанию).



10.2. Задания для самостоятельной работы

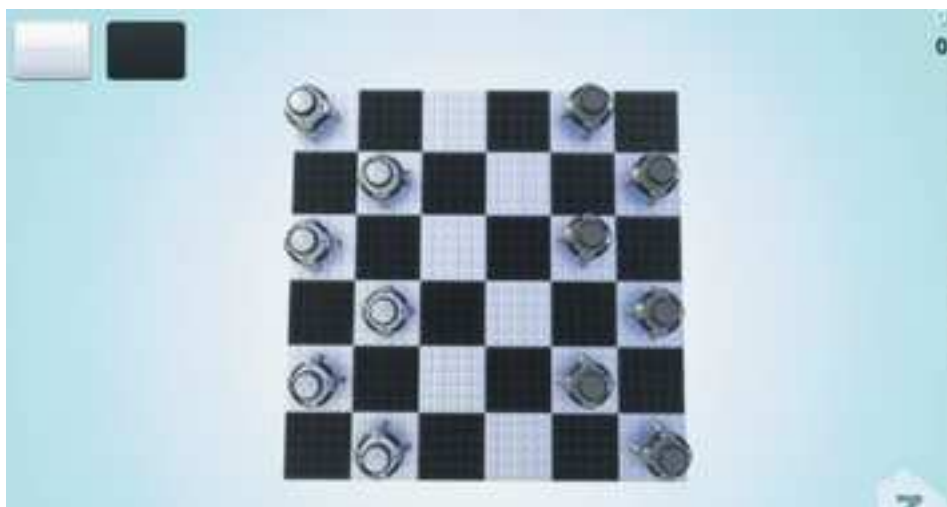
1. Самостоятельно настройте остальных игроков в игре «Футбол» по аналогии с запрограммированным **Коду1**. Сделайте так, чтобы неактивные персонажи,

которыми не управляет игрок, не стояли на месте, а тоже участвовали в игре самостоятельно.

2. Реализуйте игру с управлением от двух игроков за одним компьютером.

Совет. Для управления двумя игроками зафиксируйте камеру просмотром «Сверху вниз».

3. Создайте игру «Шашки» с шахматным полем 6×6 клеток и переключением всех персонажей от двух кнопок. В качестве шашек используйте **Тумбы**. Шашка «съедает» другую, если касается ее.



Тема 11. Переход на новый уровень, собственный проект

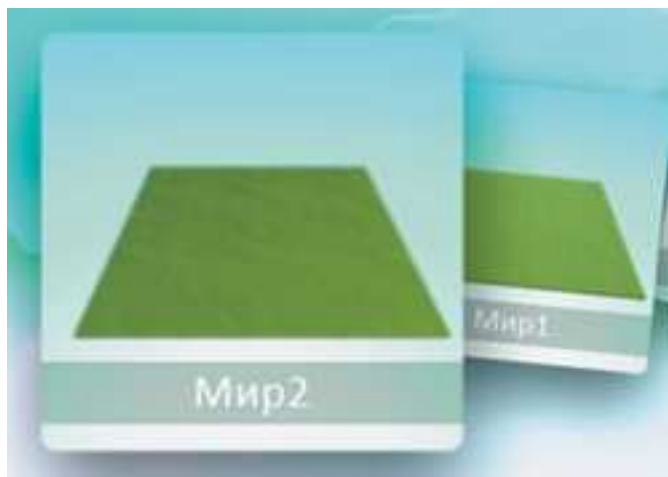
Чаще всего мы встречаем игры, которые состоят из нескольких уровней, и в **Kodu Game Lab** это тоже предусмотрено.

11.1. Игры с несколькими уровнями

Создадим два новых мира, в которых рассмотрим действие функции **Next Level** (следующий уровень). Суть нашей демо-игры будет состоять в том, что **Коду** находит монетку и, когда касается ее, переходит в другой мир, где попадает в сокровищницу.

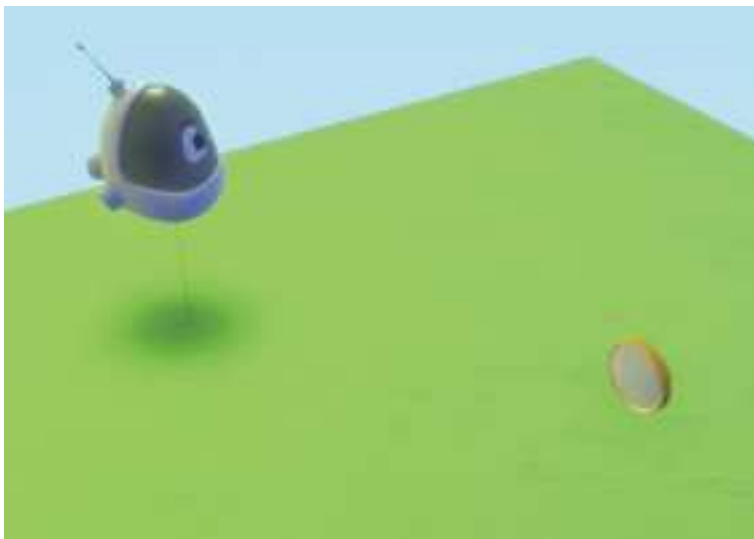
Приступим!

Для функции **Next Level** важно, чтобы оба мира были уже созданы и сохранены. Поэтому зайдём в **Новый Пустой мир** и сохраним его как «**Мир1**». Затем еще раз создадим **Новый Пустой мир** и сохраним его как «**Мир2**».



Нестрашно, если при сохранении миры будут пустыми.

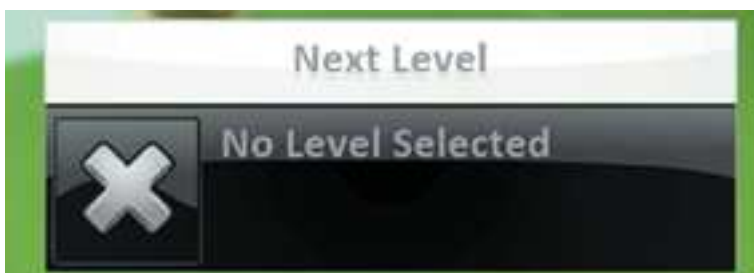
Зайдем в **Мир1**, создадим в нем **Коду** и монетку:



Программа **Коду** будет выглядеть следующим образом:



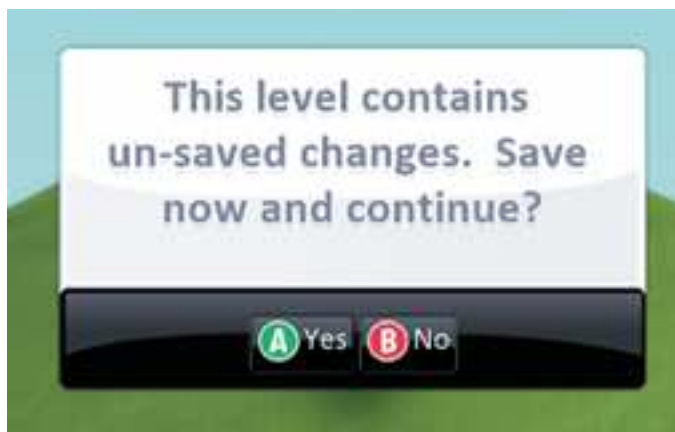
При выборе карточки **Next Level** откроется окно выбора мира для перехода:



Нажатием на «крестик» мы попадем в **Загрузку локально сохраненных миров**. Здесь следует выбрать **Мир2** и нажать **Attach** (прикрепить).

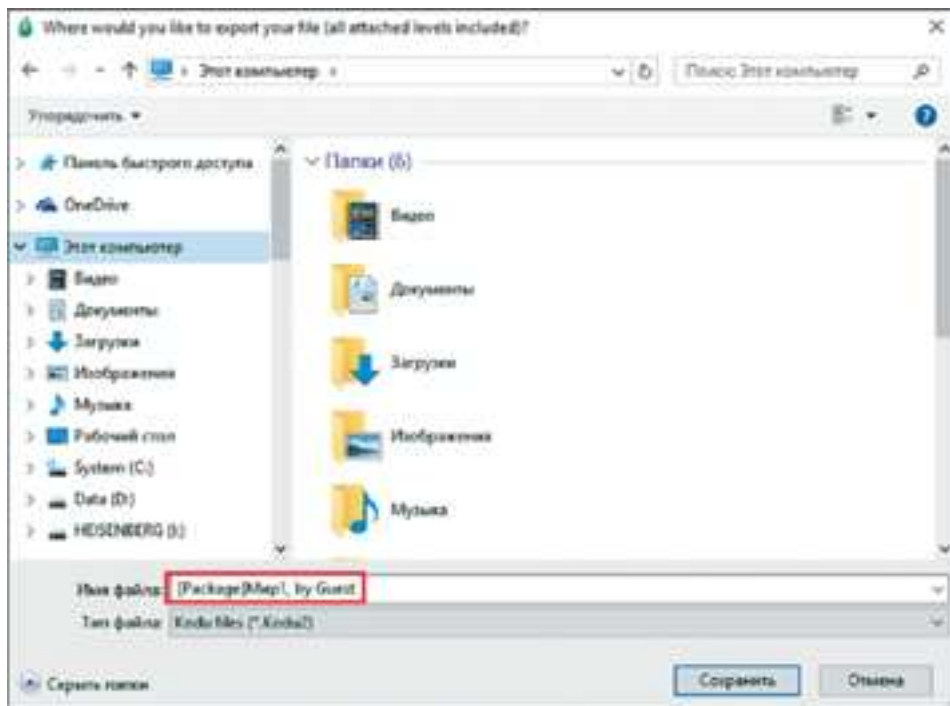


Вроде бы все готово, но для функции **Next Level** обязательно нужно протестировать игру до конца. При первом тестировании программа будет запрашивать подтверждение о всех переходах.



Согласившись на все изменения, мы перейдем в **Мир2**, и игра будет иметь название **[Package]** (пакет, упаковка). Это означает, что выбранные миры объединены.

Теперь достаточно выгрузить на флешку один из миров **[Package]**, и все объединенные уровни сохранятся автоматически.



Совет. Если вы планируете создать разные уровни в одном и том же мире, то следует сначала оформить один мир (ландшафт и прочую детализацию). После этого, не создавая персонажей, сохранить игру под разными именами (сколько планируется уровней, столько сделать и сохранений).

11.2. Задания для самостоятельной работы

Пришло время собрать все знания воедино и реализовать их в собственной большой игре!

КРИТЕРИИ ДЛЯ СОЗДАНИЯ СВОЕГО ПРОЕКТА

1. Составьте **Описание** игры, которую вы будете создавать в среде **Kodu Game Lab**. Когда проект будет готов, включите этот текст в «Описание» при сохране-

нии игры (пункт **Сохранить мой мир**). Это будет своего рода инструкцией для тех, кто еще не знает идеи и сюжета игры.

2. Выбрать жанр игры: приключение, сражение, аркада или имитация существующей игры.

3. Описать цель игры.

4. Описать объекты и персонажей, которые будут созданы в игре.

5. Описать, что делают объекты и персонажи вашей игры и как они это делают.

6. Описать, при каких условиях засчитывается проигрыш, а при каких — победа (или переход на новый уровень).

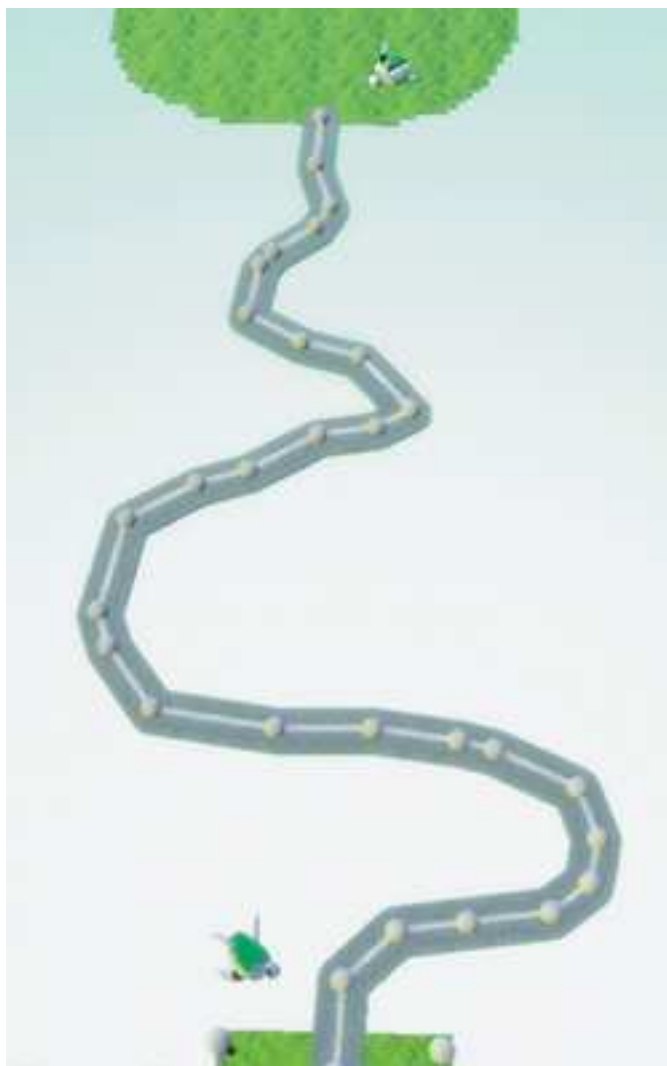
7. Продумать несколько уровней игры (минимум два).

12. Темы проектов для самостоятельной работы

12.1. «Пройди на время»

Игра-квест, в которой главный персонаж должен пройти миссии на время. Создайте минимум три миссии. Например:

- Не упади в пропасть: с помощью дорог придумать сложный для прохождения маршрут.



- Попади в мишень 3 раза: создать мишень, можно движущуюся, в которую надо попасть пулями.



- Не ошибись с выбором: создать для игрока выбор между действиями: выбрал верный вариант — прошел дальше, выбрал неверный — не прошел.



12.2. «Портал»

Игра-квест, где игрок перемещается между мирами, комнатами или другими локациями. Тематика игры может быть свободной.





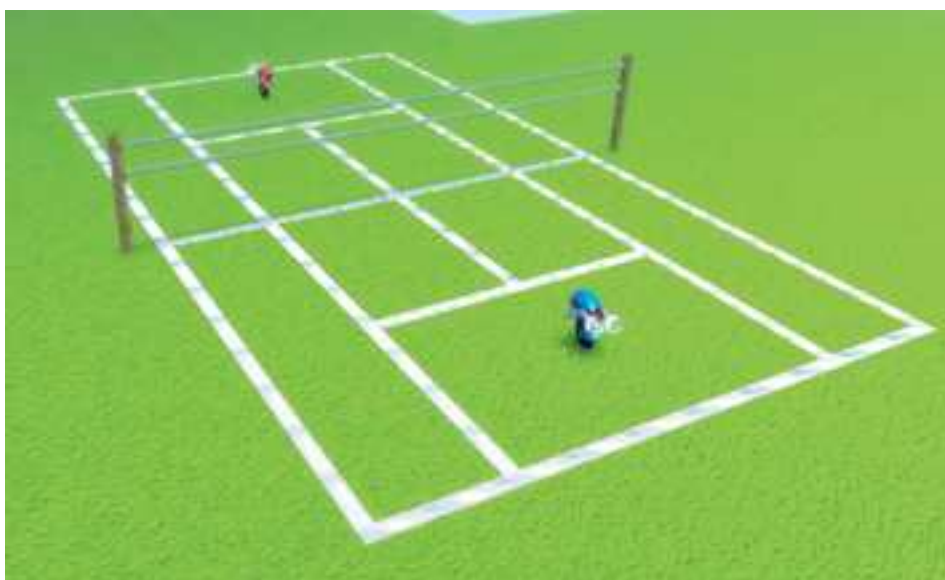
12.3. «Ловушка на ловушке»

Создайте поле и нарисуйте на нем разноцветные клетки с помощью **Кисти земли**. Составьте для игрока маршрут и окружите его ловушками. Если игрок наступает на клетку с ловушкой, игра прекращается, но маршрут остается неизменным. Игрок, начиная новую игру, должен помнить, где он попал в ловушку. Так методом «проб и ошибок» он в конце концов доберется до финиша. Количество возможных попыток нужно ограничить.



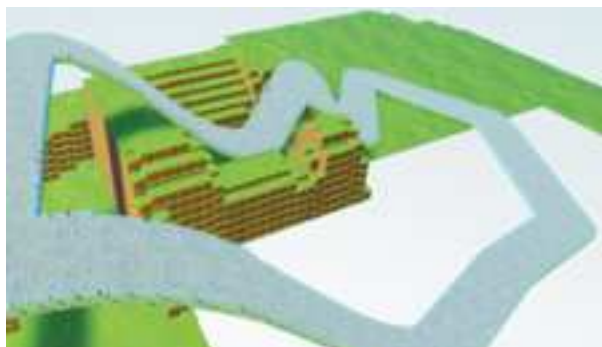
12.4. «Теннис»

Игра для одного или двух игроков. Реализуйте игру в большой теннис, где два персонажа будут ловить мяч и отбрасывать его обратно друг другу. Игра должна следовать основным правилам игры в теннис.



12.5. «Американские горки»

С помощью путей, дорог и стен, а также рельефа земли создайте аттракцион, где при запуске игры персонаж будет проходить виражи и крутые повороты, как если бы вы действительно прокатились на аттракционе.



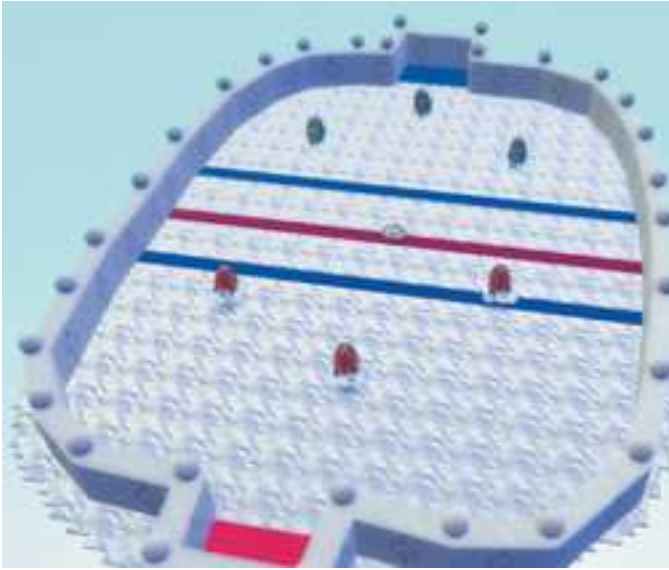
12.6. «Суперквест»

Игра-квест, которая должна начаться предысторией и закончиться интересным финалом. Персонаж будет проходить квестовые задания, передвигаясь на разных видах транспорта, участвуя в разных миссиях. В качестве основной идеи можно взять сюжет игры **Uncharted**.



12.7. «Хоккей»

Создайте игру для одного или двух игроков. За основу возьмите созданный проект «Футбол». Игра должна следовать основным правилам игры в хоккей.



12.8. «Google Game»

Знаете ли вы игру, которая запускается при нажатии клавиши «Пробел» в окне браузера **Google Chrome**? Игру, в которой динозавр бежит и перепрыгивает через препятствия? Попробуйте создать аналогичную игру, где персонаж **Байкер** (вид камеры «сбоку») перепрыгивает препятствия. Постепенно скорость передвижения увеличивается. Если **Байкер** коснулся препятствия, он проиграл.



12.9. «Чудеса в лесу»

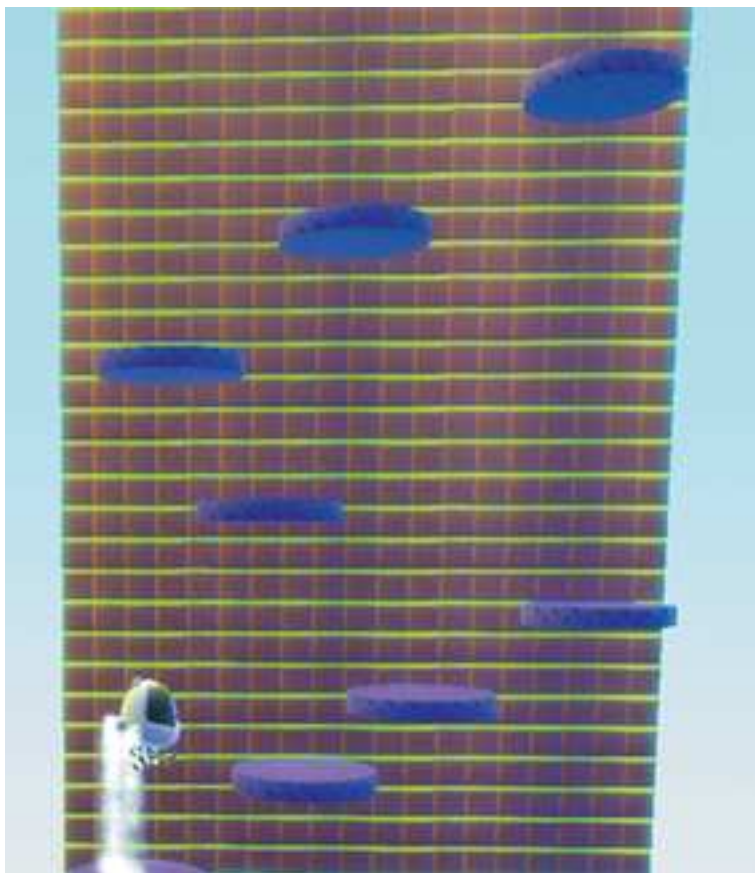
Создайте игру в жанре RPG¹, в которой персонаж перемещается по большой территории, проходит различные миссии и увеличивает свой опыт игры, получая различные награды, умения и навыки.



12.10. «Попрыгунчик»

Создайте игру, где персонаж постоянно движется вверх, прыгая по платформам. На платформу запрыгнуть непросто. Если не попал на платформу (упал), игра начинается заново. Проект должен напоминать популярную игру Doodle Jump.

¹ RPG (от англ. Role-Playing Game) — жанр игр, в которых игрок принимает участие, примеряя на себя определенную роль, как в традиционных настольных играх.



Заключение

Поздравляем тебя! Ты проделал огромную работу и теперь тебя смело можно называть настоящим Программистом и Разработчиком игр!

Надеемся, что ты весело и с пользой провел время в нашем увлекательном путешествии по миру программирования и разработки игр вместе с **Kodu Game Lab!**

Однако это только начало пути! Впереди еще много интересного, сложного, удивительного! Стоит попробовать программировать игры на **Scratch**. Зная этот язык, можно принимать участие в олимпиаде по программированию.

Для повышения уровня своей квалификации тебе нужно изучить и текстовые языки программирования высокого уровня, такие как Python, C++, C#, Java. Настоящему разработчику игр не обойтись без технологий 3D-моделирования. А если ты хочешь стать известным на весь мир, то тебе, как минимум, понадобится собственный сайт, на котором ты сможешь познакомить жителей нашей планеты со своими достижениями и идеями!

Продолжай заниматься, совершенствовать свои навыки и развиваться в самых различных направлениях, в частности в программировании.

Наша серия «Школа юного программиста» поможет тебе в этом!

Успехов, дорогой друг! Еще встретимся!

Оглавление

| | |
|--|-----------|
| От автора | 3 |
| Благодарности | 4 |
| Введение | 5 |
| Загрузка и установка Kodu Game Lab | 6 |
| Тема 1. Запуск Kodu Game Lab. Создание мира | 9 |
| 1.1. Главное меню | 9 |
| 1.2. Инструменты | 10 |
| 1.3. Параметры мира | 21 |
| 1.4. Сохранение мира | 22 |
| 1.5. Сохранение мира на диске компьютера (экспорт) | 23 |
| 1.6. Задания для самостоятельной работы | 26 |
| Тема 2. Начинаем программировать. Простые условия ... | 28 |
| 2.1. Первая программа. Движение | 28 |
| 2.2. Задания для самостоятельной работы | 36 |
| 2.3. Игра «Гонки» | 36 |
| 2.4. Задания для самостоятельной работы | 42 |
| Тема 3. Игры в жанре «Сражение» | 43 |
| 3.1. Коду против Замка | 43 |
| 3.2. Игра «Утром спасение» | 50 |
| 3.3. Задания для самостоятельной работы | 54 |
| Тема 4. Счетчики | 56 |
| 4.1. Часы, прямой отсчет времени | 56 |
| 4.2. Часы, обратный отсчет времени | 57 |
| 4.3. Задания для самостоятельной работы | 60 |
| Тема 5. Дороги и стены | 62 |
| 5.1. Подробнее о путях | 62 |
| 5.2. Наследование. Родительские и дочерние действия .. | 65 |
| 5.3. Отрицание | 66 |
| 5.4. Задания для самостоятельной работы | 67 |
| Тема 6. Страницы программ..... | 69 |
| 6.1. Меняем поведение персонажей | 69 |
| 6.2. Игра «Арканоид» | 72 |
| 6.3. Задания для самостоятельной работы | 79 |

| | |
|--|------------|
| Тема 7. Возможности функции Родитель | 80 |
| 7.1. Задания для самостоятельной работы | 82 |
| 7.2. Игра «Рыбки» | 83 |
| 7.3. Задания для самостоятельной работы | 90 |
| Тема 8. Кнопки | 93 |
| 8.1. «Кликер» | 93 |
| 8.2. Скрытый счетчик | 95 |
| 8.3. Задания для самостоятельной работы | 98 |
| Тема 9. Телепортация | 99 |
| 9.1. Телепорт | 99 |
| 9.2. Задания для самостоятельной работы | 101 |
| Тема 10. Переключение между персонажами | 102 |
| 10.1. Футбол | 102 |
| 10.2. Задания для самостоятельной работы | 105 |
| Тема 11. Переход на новый уровень, собственный проект | 107 |
| 11.1. Игры с несколькими уровнями | 107 |
| 11.2. Задания для самостоятельной работы | 110 |
| 12. Темы проектов для самостоятельной работы | 112 |
| 12.1. «Пройди на время» | 112 |
| 12.2. «Портал» | 113 |
| 12.3. «Ловушка на ловушке» | 114 |
| 12.4. «Теннис» | 115 |
| 12.5. «Американские горки» | 116 |
| 12.6. «Суперквест» | 116 |
| 12.7. «Хоккей» | 117 |
| 12.8. «Google Game» | 117 |
| 12.9. «Чудеса в лесу» | 118 |
| 12.10. «Попрыгунчик» | 118 |
| Заключение | 120 |

Минимальные системные требования определяются соответствующими требованиями программ Adobe Reader версии не ниже 11-й либо Adobe Digital Editions версии не ниже 4.5 для платформ Windows, Mac OS, Android и iOS; экран 10"

Электронное издание для дополнительного образования

Серия: «Школа юного программиста»

Астахова Ксения Ивановна

СОЗДАЕМ ИГРЫ С KODU GAME LAB

Под редакцией В. В. Тарапаты

Для детей среднего школьного возраста

Ведущий редактор *Т. Г. Хохлова*

Ведущие методисты *Н. Н. Самылкина, А. А. Салахова*

Художники *В. А. Прокудин, Я. В. Соловцова*

Технический редактор *Т. Ю. Федорова*

Корректор *И. Н. Панкова*

Компьютерная верстка: *Е. Г. Ивлева*

Подписано к использованию 13.11.18.

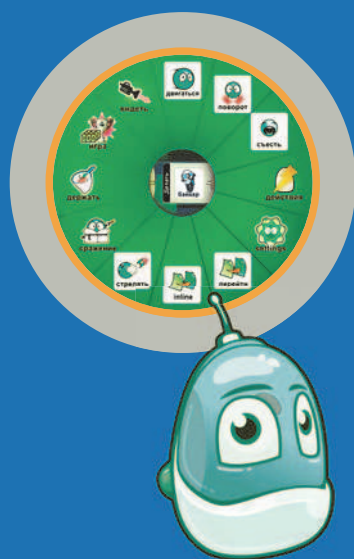
Формат 155×225 мм

Издательство «Лаборатория знаний»

125167, Москва, проезд Аэропорта, д. 3

Телефон: (499) 157-5272

e-mail: info@pilotLZ.ru, <http://www.pilotLZ.ru>



Программирование – это грамотность XXI века!

Книги новой серии «Школа юного программиста» издательства «Лаборатория знаний» построены на методике поэтапного обучения программированию. Следуя этой методике, любой желающий, от школьника до студента вуза, сможет научиться писать программы, разрабатывать мобильные приложения и компьютерные игры и даже освоить технологии машинного обучения и нейросетей.

В серию войдут следующие учебные пособия:

- «Учимся вместе со Scratch: программирование, игры, робототехника» (5–6 классы)
- «Scratch 2.0: от новичка к продвинутому пользователю. Пособие для подготовки к Scratch-Олимпиаде» (1–11 классы)
- «Творческие задания в среде Scratch. Рабочая тетрадь для 5–6 классов»
- «Scratch 2.0: творческие работы на вырост. Рабочая тетрадь для 7–8 классов»
- «Создаем игры с Kodu Game Lab» (4–5 классы)
- «Python для начинающих – от основ до ООП и приложений» (7 класс)
- «Олимпиадное программирование на Python» (7–8 классы)
- «С# – новый учебный курс программирования от основ до продвинутого уровня» (8–9 классы)
- «Android-разработка: мобильные приложения» (8–9 классы)
- «Web-разработка: создай свой идеальный сайт. Обучаемся тонкостям HTML, HTML5, CSS3, SQL, PHP, JavaScript» (8–10 классы)
- «Основы искусственного интеллекта и нейросетей» (10–11 классы, студенты) и другие.