ТВОЙ КУРС ИТ для молодежи

5 ПРОСТЫХ ШАГОВ К СОЗДАНИЮ 3D ИГР ВМЕСТЕ С KODU

(Курс для юных разработчиков, учащихся 1-9 классов по знакомству с основами объектно-ориентированного и визуального программирования и созданию 3D игр)

Авторы:

Яникова Н.В., Михеева О.П., Брыксина О.Ф., Останин Я.Е.

2013 г.

Занятия проводятся в ходе учебного трека «Введение в программирование» в рамках проекта «Твой курс: ИТ для молодежи» в рамках инициативы Microsoft YouthSpark

Дорогой друг!

В наше время очень много подростков, молодых людей и даже взрослых (!) увлекаются компьютерными играми. И это понятно! Как это здорово почувствовать себя профессиональным гонщиком, удачливым покорителем новых земель, непревзойденным воином-победителем! Такие игры создают целые команды профессиональных программистов.

Наверное, на сегодняшний день, это самая востребованная профессия! Программисты создают программы и приложения абсолютно для всех людей любого возраста и рода занятий: школьников и студентов, бухгалтеров и юристов, дизайнеров и инженеров, врачей, менеджеров и многих других специальностей.

А хочешь ли ты попробовать себя в роли программиста? Да? Тогда давай начнем с самого интересного - программирования компьютерных игр!

Компания Microsoft создала уникальный по своим возможностям визуальный язык программирования Kodu (рус. Коду), который достаточно прост в изучении, но осваивая который, ты сможешь создавать свои собственные 3D игры и постигать тайны программирования!

Этот язык доступен для детей, но если в вашу команду разработчиков войдут старшие братья, папы или учитель - это будет просто здорово!

На самом деле, Microsoft Kodu Game Lab - это визуальный конструктор, позволяющий создавать трёхмерные игры для персональных компьютеров и игровых приставок XBox без знания языка программирования и элементов компьютерного дизайна!

Двигаясь от простого к сложному, ты откроешь для себя большие возможности данной среды и сделаешь свои собственные игры, в которые смогут играть люди по всему миру. Kodu позволяет понять азы создания настоящих компьютерных программ, поскольку это визуальный язык программирования со своими правилами и секретами. Приятная новость: освоить их все можно всего за **5** занятий!

На каждом занятии рекомендуется просмотр видеофрагментов с комментариями профессионального программиста.

Изучая видеофрагменты или справочные материалы, сразу экспериментируй! Проверяй все возможности Microsoft Kodu Game Lab на практике и ты увидишь, что процесс создания игр становится действительно увлекательным и даже захватывающим! Прояви фантазию, проектируя свой виртуальный мир и выбирая героев, их помощников, врагов, препятствия, населяющих его. Научись задавать управление своим героями, объясни им как взаимодействовать друг с другом в твоем компьютерном мире!

Придумай интересный сюжет для игры и реализуй его с помощью Kodu Game Lab вместе со своими друзьями! Гонки, экшн-стрелялки, квесты, спорт – все это доступно вместе с Kodu.

Сделай первый шаг в программирование: изучай, исследуй, твори! Собери команду единомышленников, общайтесь, создавайте и играйте в собственные игры и делитесь идеями! А мы для тебя подготовили много интересного.

Начнем создавать игры прямо сейчас?

Занятие 1. Знакомство с визуальной средой программирования Kodu: делаем первые шаги

Для справки

Визуальное программирование — способ создания программы для компьютера путём манипулирования графическими объектами вместо написания её текста (кода).

Внимание!

Для установки визуальной (от лат. visualis - зрительный) среды программирования Kodu воспользуйтесь **инструкцией.**

Для начала посмотрите ролик

“Первая игра (обучение)”

Проверь себя!

После просмотра видео ответьте на вопросы:

- Каким образом осуществляется переключение между режимом редактирования и режимом игры?
- Как можно понять что включен режим редактирования?
- Каким образом отображаются в режиме редактирования пути, по которым перемещаются объекты?
- Какие действия необходимо выполнить для создания программы, определяющей последовательность действий объекта?
- Какой вид имеет структура правила в **Kodu**?

Для справки:

Программа в **Kodu** - это набор правил, которые определяют действия объекта.

Для написания правил в Kodu используются два оператора:

When <условие> Do <действие>

- **When** (англ. “когда”, “если”, “в то время как”) - оператор, определяющий условие;
- **Do** (англ. “делать”) - оператор, определяющий непосредственное действие, которое должен выполнить объект при соответствующем условии.

Программа создается для каждого объекта индивидуально!

Упражнение 1. Создание первой программы.

Сюжет игры: Байкер объезжает деревья.

Для успешного выполнения упражнения и создания первой игры, четко следуйте предложенному алгоритму:

- Запустите программу **Kodu**.
- Выберите команду **«Новый пустой мир»** (Empty World). Появится зеленое поле - основа для размещения игровых объектов в мире. Внизу окна размещены иконки, отображающие основные команды программы (рис.1.1).

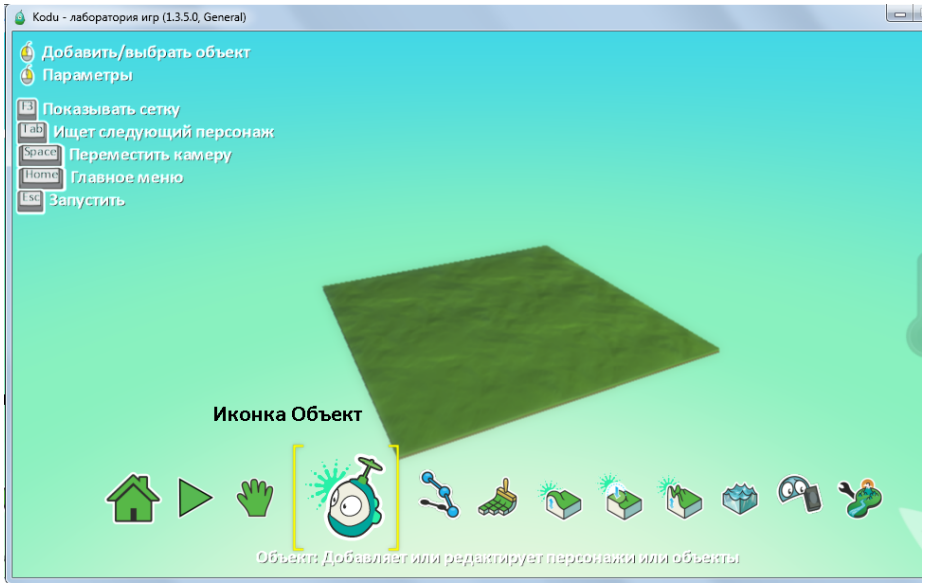

Рис. 1.1. Режим создания мира в Kodu

- Добавьте на поле 3 дерева. Для добавления объектов выберите иконку **«Объект»** и щелкните один раз по полю левой кнопкой мыши. Появится список доступных объектов. Выберите объект «**Дерево»**. Повторите действия ещё два раза.

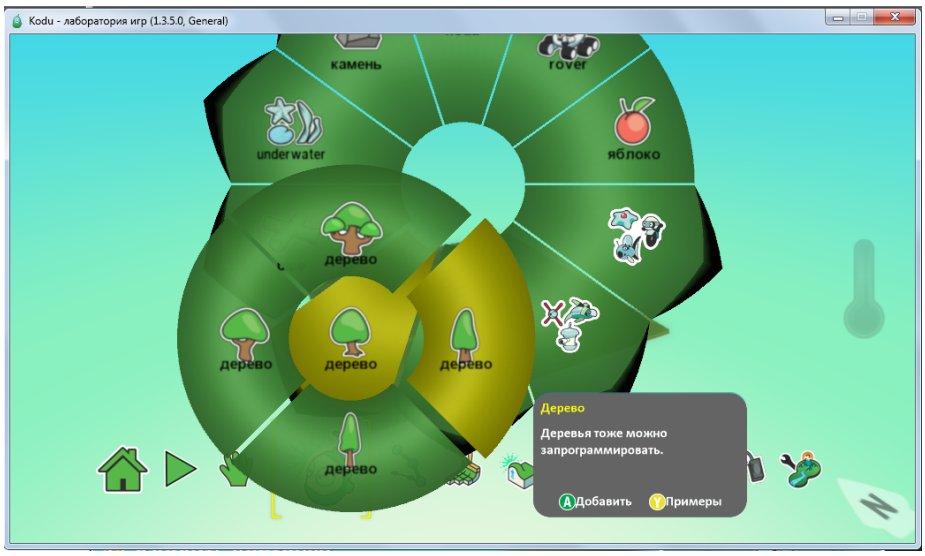

Рис. 1.2. Режим выбора объектов

- Добавьте объект **«Байкер»** и задайте для него программу действий - набор правил. Щелкнув правой кнопкой мыши на объекте **«Байкер»,** вызовите контекстное меню и перейдите в режим создания **Программы**.
- В открывшемся окне кода составьте инструкцию для движения вокруг деревьев. Команды задаются выбором из списка необходимых инструкций. Пример программы приведен на рис. 1.3. При щелчке по карточке (иконка со знаком +) открывается перечень доступных действий, из которых следует выбрать необходимое (подтвердив действия нажатием левой клавиши мыши).

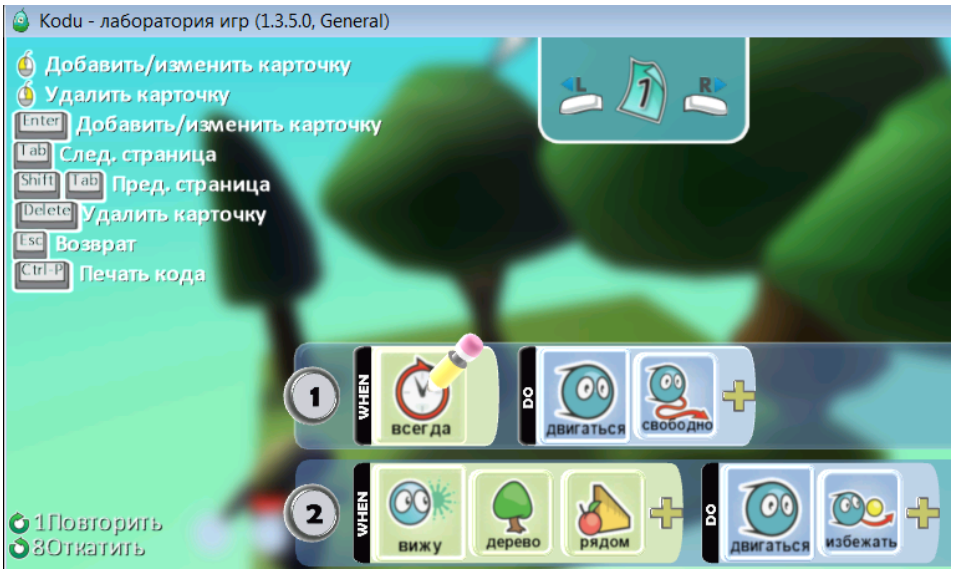

Рис. 1.3. Программа, позволяющая **Байкеру** объезжать деревья

- Запустите программу на выполнение клавишей **Esc**. Понаблюдайте за движением **Байкера**. Если траектория движения не соответствуют поставленной задаче (объект не двигается или не объезжает деревья), то проверьте корректность кода (рис. 1.3).
- Сохраните программу на жестком диске. Для этого перейдите в главное меню (клавиша **Home**) и выберите команду **«Сохранить мой мир»**.

Задание для самостоятельной работы:

- Попробуйте изменить цвет одного из деревьев, например, на желтый.
- Напишите программу, в которой предусмотрите ситуацию столкновения **Байкера** с деревом.

Вопросы профессионального программиста:

- Попробуй порассуждать, каким образом осуществляется программирование в среде **Kodu**? Что сделали разработчики, чтобы этот процесс стал увлекательным и доступным детям?
- Как ты думаешь, почему разработчики **Kodu** хотят научить детей программировать?
- Ты уже знаешь, какие классы языков программирования выделяют и какое место среди них занимает **Kodu**? Почему его называют визуальным?
- На каких операторах основано программирование в **Kodu**? Подумай, почему? Какие это создает дополнительные возможности?

Упражнение 2. Создание игры для двух игроков.

Сюжет игры: два Байкера поедают яблоки.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Создайте новый мир (**Empty World**). Разместите в нем объекты **Байкер** и **Яблоки**.
- Напишите программу, которая позволит **Байкеру** свободно перемещается по территории мира. Замечая яблоко, **Байкер** должен двигаться к нему (рис. 1.4). Попробуйте создать на территории два близко расположенных яблока. Что может произойти?

Рис. 1.4. Код для перемещения **Байкера**

- Измените правило управления **Байкером**. Для этого следует вырезать из программы строку 1 (щелчок правой кнопкой мыши на номере строки - вырезать), см. рис. 1.5.

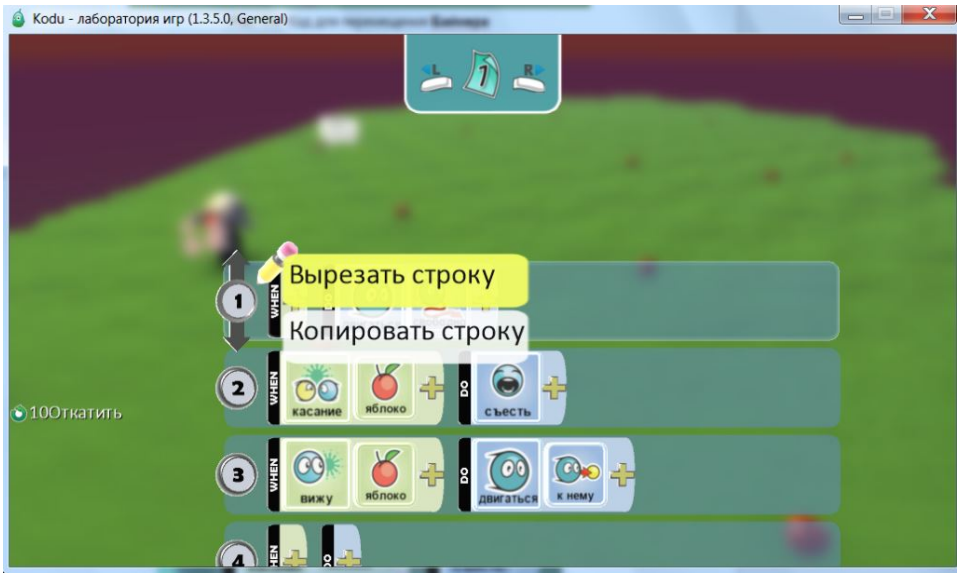

Рис. 1.5. Вырезание строки из программы

Измените правило управления движением **Байкера** на управление при помощи клавиатуры вместо свободного перемещения (“если нажимаются клавиши управления курсором, то **Байкер** должен двигаться”), как это сделано на рис. 1.6.

- Добавьте правило поедания **Яблок** (“когда Байкер коснется яблока, он должен его съесть”). Пример кода приведен на рис. 1.6.

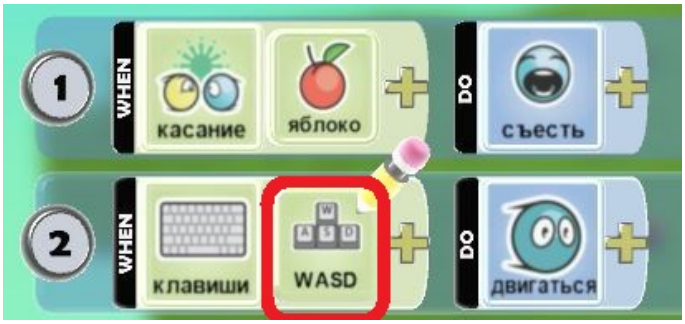

Рис. 1.6. Управление Байкером с клавиатуры

Создайте второго **Байкера**, выполнив операцию копирования (используйте правую кнопку мыши).

- Измените цвет одного из **Байкеров**, чтобы их можно было отличить. Используйте для этого палитру цветов и клавиши управления курсором (рис. 1.7).

Рис. 1.7. Изменение цвета объекта

- Обратите внимание, что при копировании объекта копируется и его программа. Измените правило управления вторым **Байкером** в его программе. Для этого назначьте новые управляющие клавиши, например, стрелки.
- Теперь вы создали игру для двоих участников. Посоревнуйтесь с другом в поедании **яблок**.
- Подумайте, как можно улучшить эту игру.
- Сохраните игру, добавив необходимые теги.

Для справки:

Приучайте себя уточнять значения всех новых слов в Интернете, словарях или энциклопедиях!

Тег, иногда тэг (англ. tag — ярлык, этикетка, бирка; метить) - здесь: ключевое слово, уточняющее действие.

Теги в Kodu используются для того, чтобы объединить игры сходной тематики и лучше ориентироваться при поиске. Например, вы сможете отсортировать игры-приключения или игры, созданные только для xBox. Для созданной вами игры, где байкеры поедают яблоки, подойдут теги **Приключения** или **Гонки** (рис. 1.8).

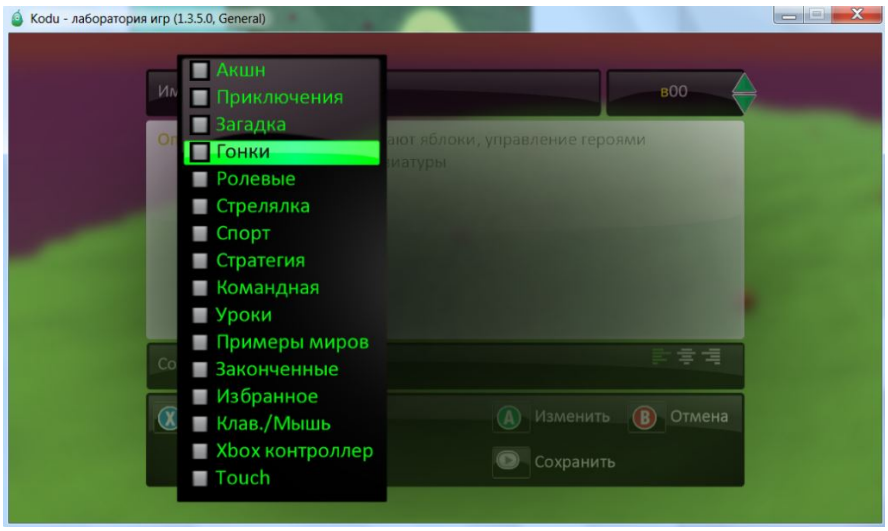

Рис. 1.8. Теги миров Kodu

Задание для самостоятельной работы:

- Сделайте два яблока синего цвета. Измените код **Байкеров** таким образом, чтобы они не ели синих яблок.
- Добавьте объект **Kodu** и запрограммируйте его на поедание только синих яблок.

Вопросы профессионального программиста:

- Какую структуру имеет команда, если на действия исполнителя (например, **Байкера**) налагаются определенные условия?
- Подумай, как правильно сказать “Программа - это алгоритм..” или “Алгоритм - это программа…”.
- Как редактируется программы в среде **Kodu**?
- Можно ли при необходимости поменять порядок выполнения действий исполнителя? Как это можно реализовать?
- В каких случаях программисты используют возможность случайного выбора? Как это может быть реализовано в игре?

Упражнение 3. Создание ландшафтов.

В этом упражнении вы научитесь проектировать вид местности (ее ландшафт), на которой происходят действия игры.

Для справки

Ландша́фт (нем. Landschaft - вид местности, от Land - земля и schaft - суффикс, выражающий взаимосвязь, взаимозависимость).

Рельеф (фр. relief, от лат. relevo — поднимаю) — совокупность неровностей суши, дна океанов и морей, разнообразных по очертаниям, размерам, происхождению, возрасту и истории развития.

Для начала посмотрите ролик

"Инструкции по созданию ландшафта"

Проверьте себя!

После просмотра видео, экспериментируя с местностью в среде **Kodu**, ответьте на вопросы:

- Как выбрать тип ландшафта?
- Как удалить часть ландшафта?
- Как изменить тип кисти для создания территории? А как изменяется тип материала, с помощью которого проектируется ландшафт?
- Как добавить другой вид ландшафта к уже имеющейся территории? А как изменить часть уже существующей местности?
- Какой инструмент служит для создания рельефа?
- Что умеет "волшебная" кисть?
- Какие функции выполняют клавиши **Ctrl**, **Shift** и **пробел** при редактировании территории?
- Каким способом можно дублировать тип местности при создании ландшафта?

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Создайте новый пустой мир (**Empty World**).
- Создайте ландшафт игрового мира в виде зеленой травы, небольших гор и холмов. Для создания зеленого поля выберите инструмент **«Кисть»** для земли и произвольно нарисуйте на игровом поле землю. Цвет земли выберите под номерами **15** или **54**.
- Поднимите отдельные участки земли, сделав из них холмы и горы. Для создания неровностей используйте инструмент «**Неровности»**.
- Добавьте воду в созданный ландшафт при помощи инструмента «**Вода»,** цвет воды под номером **8**.
- Измените цвет неба при помощи инструмента **«Параметры мира»**. Включите волны для воды.

Для справки

Инструмент **«Параметры мира»** позволяет сделать игровой мир **Kodu** более профессиональным. Экспериментируя с различными настройками (включая, отключая или изменяя значения параметров), можно добиться профессиональных эффектов в своей игре.

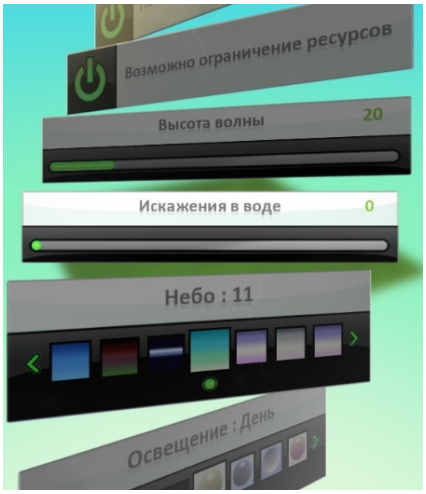

Рис. 1.9. Инструмент Параметры мира

- Нарисуйте облака на небе. Пример ландшафта приведен на рис. 1.10.

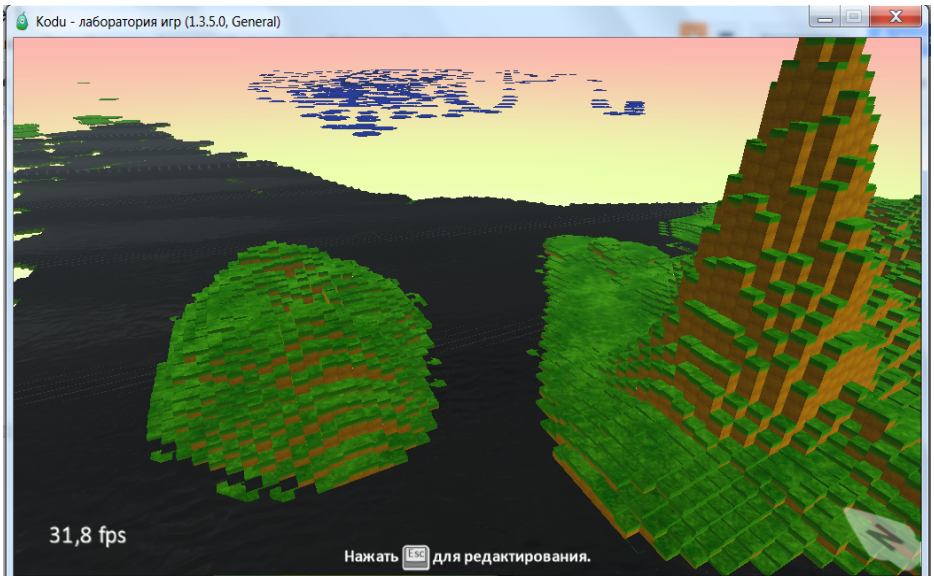

Рис.1.10. Пример ландшафта с водоемами и облаками

- Сохраните мир под именем **«Ландшафт»** и запустите игру на выполнение. Обратите внимание на движение воды и набегающие волны.

Задание для самостоятельной работы:

- Создайте в **Kodu** модель местности, используя кисть и её различные типы, а также увеличение и уменьшение размера кисти.
- Добавьте на созданном рельефе холмы и впадины, используя соответствующий инструмент.
- Постройте стены на моделируемой местности при помощи волшебной кисти.
- Добавьте несколько дорожек на создаваемой территории.
- Измените модель местности, добавив водоемы, парки, здания.

Вопросы профессионального программиста:

- Какого вида модели можно проектировать в Kodu?
- Из каких этапов строится процесс моделирования игры в среде Kodu?
- Вспомните классификацию моделей и приведите примеры реализации различных видов моделей в **Kodu**.
- Докажите, что **Kodu** можно использовать для создания компьютерных моделей объектов, процессов и явлений. Приведите конкретные примеры.
- Вспомните основные этапы моделирования и покажите как они реализуются в процессе проектирования игры.

Практическая работа с миром “Стрельба по рыбам”(Shooting fish)

1. Откройте игровой мир **“Стрельба по рыбам” (Shooting fish).**
2. Добавьте объект **Kodu** на игровое поле.
3. Напишите программу для **Kodu**, с помощью которой можно будет стрелять по рыбам ракетами в разных направлениях.
4. Проведите эксперимент: сравните действие ракет и пулек. Сделайте вывод о том, какое оружие эффективнее.
5. Изучите различные варианты стрельбы.
6. Настройте стрельбу пульками случайных цветов, вылетающих после каждого нажатия на пробел.
7. Пройдите игру, используя стрельбу ракетами. За какое время это у вас получилось?
8. Сохраните созданный мир.

Это интересно...

Первым программистом называют графиню Аду Лавлейс, которая родилась в Лондоне почти 200 лет назад! (10.12.1815). Она составила первую в мире программу для прообраза современного компьютера - аналитической вычислительной машины Чарльза Бебиджа.

Это интересно...

Один из создателей компании MicroSoft по выпуску программного обеспечения Билл Гейтс (родился 28.10.1955) - один из самых богатых людей на планете. Только в период с 1994 по 2010 г. он вложил в свой благотворительный фонд более 28 миллиардов долларов!

В тринадцать лет Билл написал свою первую программу – игру «Крестики-нолики» на языке программирования BASIC. Вот как он об этом вспоминает:

“Я помешался на компьютерах. Пропускал физкультуру. Сидел в компьютерном классе до ночи. Программировал по выходным. Каждую неделю мы проводили там по двадцать-тридцать часов…”

В 17 лет Гейтс, Пол Аллен и Пол Гилберт основали компанию Traf-O-Data. Цель компании была в создании счетчиков для считывания дорожного трафика и составления отчетов для дорожных инженеров. А 26 ноября 1976 года, когда Биллу был 21 год, он основал компанию Microsoft.

Это интересно...

Специалист, занимающийся написанием и корректировкой программ для компьютеров или любых вычислительных устройств, называется программистом. Специальности “программист” учат в средних и высших учебных заведениях. О том, что должен уметь делать программист, можно прочитать в Стандарте специальности.

Это интересно...

Сколько зарабатывают программисты в крупнейших компаниях? Сайт Glassdoor, собирающий анонимную информацию о заработках людей, опубликовал результаты

исследования о зарплатах в компаниях Кремниевой долины. Например, в компании Facebook, среднестатистическая зарплата составляет 110 тысяч долларов в год, плюс около 12 тысяч долларов бонусов.

Блиц-опрос

1. Профессиональные обязанности программиста заключаются в (выбрать все верные ответы):

a. написании программ

b. тестировании программ

c. алгоритмизации поставленной задачи

d. установке и настройке системного ПО

e. описании информационных и математических моделей

f. подготовке графических схем по заданному описанию или эскизу

2. Последовательность инструкций, предназначенных для исполнения устройством управления вычислительной машины, называется:

a. алгоритмом

b. программой

c. формальной моделью

d. тестом

3. Кто был первым программистом?

a. Ада Лавлейс

b. Блез Паскаль

c. Билл Гейтс

d. Никлаус Вирт

4. Сколько цветов содержит палитра Kodu?

a. 7

b. 11

c. 255

d. 125

5. Какие два оператора используются для написания правил в Kodu?

a. When .. Do

b. Begin .. End

c. For .. Do

d. If .. Then

Занятие 2. Новые возможности для перемещения объектов и персонажей - пути. Создание клонов и порождаемых объектов. Опция «Родитель».

На этом занятии ты познакомишься с тем, как создаются пути движения персонажей, научишься создавать клоны и порождения объектов, а также узнаешь, как научить объект “говорить”. Знакомство с опцией **«Родитель»** познакомит тебя с объектно-ориентированным программированием, которое используется для создания сложных программных систем.

Для справки

Объектно-ориентированное программирование - это программирование, в котором основными понятиями являются объекты и классы.

При таком программировании создается класс (например, “комната”), в который входят различные объекты (например, “кухня”). При этом каждый объект имеет общие характеристики, присущие **всему** классу (стены, пол, потолок, окно, дверь), но может приобретать и индивидуальные черты (например: раковина, плита, стол, стул и т.п.).

Для справки

Чтобы изменить линию движения любого персонажа, настроив её так, как это необходимо для игры, надо воспользоваться инструментом **«Путь»** в игровом мире **Kodu**.

Повторение пройденного

При изучении упражнений занятия №1 мы научились создавать коды для объектов игрового мира. Давайте вспомним пройденный материал.

В задании к самостоятельной работе упражнения 2 вам надо было добавить в игровой мир яблоко синего цвета и написать код, который бы запрещал **Байкерам** есть синие яблоки. На рис.2.1 представлен код с ошибкой. Найдите и устраните эту ошибку.

Рис.2.1 Ошибочный код

Упражнение 1. Создание произвольного пути движения игрового объекта.

Сюжет игры: Байкер движется по кругу.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Запустите программу **Kodu**.
- Выберите команду **«Новый пустой мир»** (**Empty World**). Появится зеленое поле - основа для размещения игровых объектов в мире.
- Добавьте объект **Байкер**. Для добавления объекта щелкните левой кнопкой мыши по зеленому игровому полю и выберите объект **Байкер**.
- Нарисуйте круговую траекторию движения **Байкера**. Для создания пути следования объекта выберите инструмент **Путь** и при помощи точек отобразите путь следования **Байкера**, пример на рис. 2.2. Траекторию можно оборвать, нажав на **Esc**.

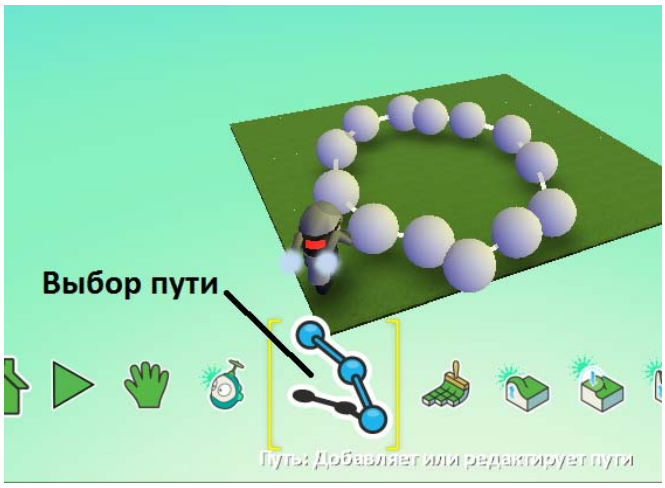

Рис. 2.2. Создание кругового пути следования объекта **Байкер**

Для справки

Траекто́рия — линия в пространстве, представляющая собой множество точек, в которых находилось, находится или будет находиться тело при своём перемещении в пространстве.

- Запрограммируйте движение **Байкера** по созданному пути, пример кода на рис.2.3.

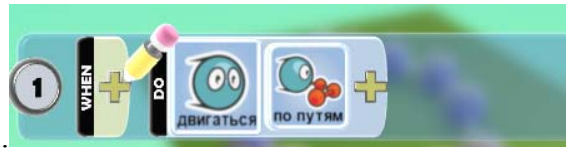

Рис.2.3. Программа для движения объекта по нарисованному пути

- Запустите программу клавишей **Esc** и проверьте её работу. **Байкер** должен ездить по кругу.
- Остановите программу клавишей **Esc**.
- Сохраните программу под именем **Байкер**. Для сохранения программы надо выйти в главное меню (клавиша **Home**) и выбрать пункт **«Сохранить мой мир»**.

Задание для самостоятельной работы:

- Добавьте ещё одного **Байкера**, измените его цвет, например, на красный.
- Нарисуйте путь его движения и обязательно измените цвет пути, например, на красный.
- Напишите программу, в которой **Байкер** движется по красному пути.
- Напишите код, где в случае столкновения **Байкеров** друг с другом они говорят «Извините!». Примеры пути и кода на рис. 2.4 и 2.5. Код для столкновения следует написать для обоих **Байкеров**!

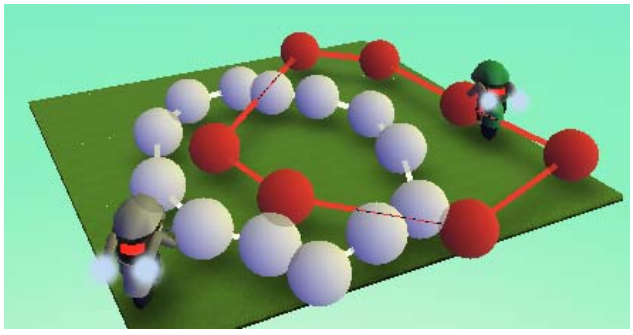

Рис. 2.4. Создание пути движения для двух объектов **Байкер**

Рис. 2.5. Код программы, в которой при столкновении **Байкеров** произносятся извинения

Дополнительное задание:

- Выполнив задания этого занятия, поэкспериментируйте с игрой **Tutorial 02**. Проверьте, сможете ли вы составить программу, позволяющую управлять движением и стрельбой мотоцикла с помощью клавиатуры и мыши.
- Изучите также мир **3D Flare Paths**, содержащий интересные графические эффекты и примеры представления трехмерных объектов.
- Пример путей, поднятых над поверхностью, вы можете увидеть в игре **Rock Fight v09**. Чтобы играть в эту игру, ее необходимо изменить, обеспечив ее работоспособность при отсутствии контроллера. Кроме того, необходимо изменить начальный экран, отобразив на нем сведения о клавишах, используемых для управления игрой.

Внимание!

Чтобы изменить содержимое начального экрана, выберите на главной панели инструментов значок гаечного ключа и измените параметр **Start Game With: World Description** («Отображать при начале игры: описание мира»).

Анализируя игры и играя в них, **подумайте**, каким образом используются пути и наклонные поверхности и как они влияют на активность действий в игре.

Упражнение 2. Создание клонов объектов.

Сюжет игры: три **Аэростата** стреляют по **Самолету**.

Для справки

Аэростат (возду́шный шар) — летательный аппарат легче воздуха, использующий для полёта подъёмную силу заключённого в оболочке газа (или нагретого воздуха) с плотностью меньшей, чем плотность окружающего воздуха (согласно закону Архимеда).

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Создайте новый мир (**Empty World**). Разместите в нем объекты **Самолет** и **Аэростат**.
- Напишите для **Аэростата** код, выполняющий следующие действия: когда **Аэростат** замечает вдали **Самолет**, звучит громкая музыка. Приближаясь к **Самолёту**, **Аэростат** начинает светиться.
- Составьте программу, позволяющую управлять **Аэростатом** с помощью клавиатуры. Пример программы приведен на рис.2.6, строки 1 - 3.
- Сохраните мир под именем **«Самолет».**

- Запустите программу на выполнение клавишей **Esc**. Подведите **Аэростат** к **Самолету**, послушайте музыку, которая была выбрана для игры.

Рис. 2.6. Код, управляющий работой Аэростата

- Создайте эскадрилью **Аэростатов** из трех объектов. Для этого щелкните на уже имеющемся **Аэростате** правой кнопкой мыши и выберите пункт «**Копировать**», затем поместите указатель мыши туда, где надо создать новый **Аэростат**, щелкните правой кнопкой мыши и выберите пункт **«Вставить (Аэростат)»**. Повторите вставку так, чтобы на поле оказалось три **Аэростата**.
- Запустите игру и убедитесь, что все **Аэростаты** управляются и двигаются одновременно.
- Добавьте код, позволяющий каждому **Аэростату** выстреливать **Импульсы**, рис.2.6, строка 4. Сколько раз нужно изменить код?
- Сохраните игру, запустите её и убедитесь в работоспособности.

Для справки

Клонирование — синоним копирования любых объектов.

Для начала посмотри ролик “Опция Родитель”.

Проверь себя!

После просмотра видео, экспериментируя с местностью в среде **Kodu**, ответьте на вопросы:

- Что позволяет опция **Родитель**?
- Для чего предназначен инструмент, изображенный на рис. 2.7?

Рис. 2.7. Для чего нужен этот инструмент?

- Какие параметры (опции) объектов можно изменять?
- Как получить информацию по опциям?
- По каким признакам можно понять, что для персонажа включена опция Родитель?
- Как сделать действие дочерним? что это означает?

Упражнение 3. Создание порожденных объектов.

Сюжет игры: новая эскадрилья **Аэростатов** стреляет по самолету.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Откройте созданный в упражнении 2 мир **Самолет**.
- Добавьте на игровое поле ещё одну группу **Аэростатов**, созданных при помощи технологии порождения. Для этого добавьте новый **Аэростат**, поменяйте его цвет, например, на желтый.
- Переместите созданный **Аэростат** за пределы игрового поля. Данный объект не будет виден в процессе игры, а нужен только для того, чтобы создавать новые объекты **Аэростат**.
- Задайте созданному **Аэростату** параметр **Родитель**. Для этого щелкните правой кнопкой мыши по **Аэростату** и выберите пункт «**Изменить установки**». Активируйте параметр **Родитель**. Запустите программу на выполнение клавишей **Esc** и убедитесь, что новый **Аэростат** не виден. Остановите программу и перейдите в режим редактирования (клавиша **Esc**).
- Создайте три **Аэростата** в новой эскадрилье. Для этого скопируйте созданный родительский **Аэростат** (правая кнопка мыши), а затем трижды вставьте объект **Аэростат** в любое место игрового поля. Если поместить указатель мыши на один из **Аэростатов**, между исходным и новым объектом появится пунктирная линия, показывающая, что эти **Аэростаты** входят в цепочку порождаемых объектов (рис.2.8).

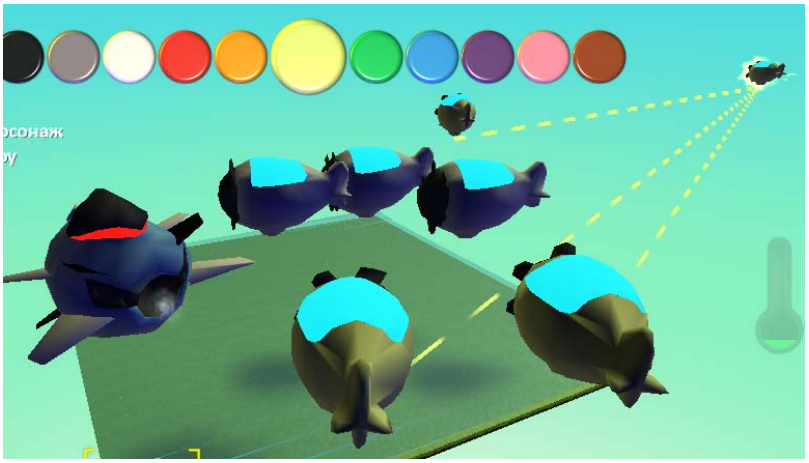

Рис.2.8. Порожденные объекты **Аэростат** связаны с родительским объектом

- Напишите код для родительского **Аэростата**, заставляющий объекты двигаться к **Самолету**. Обратите внимание, что теперь код надо написать только один раз (для родительского объекта), в порождаемых объектах все строки кода добавляются автоматически.
- Сохраните программу, запустите её на выполнение, убедитесь в её работоспособности.

Задание для самостоятельной работы:

Напишите код, позволяющий издавать звук взрыва при столкновении **Аэростатов** с **Самолетом.**

Упражнение 4. Опция **Родитель**

Сюжет игры: Каждые 4 секунды объект **Камень** создает новый объект **Летающая рыба**. **Камень** стреляет ракетами по появляющимся рыбам.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Загрузите программу **Kodu**;
- Загрузите мир **Shooting Fish**.
- Очистите мир от объектов.
- Создайте объект **Летающая Рыба** и измените её установки, включив опцию **Родитель**.

- Добавьте объект **Камень** и задайте ему действия таким образом, чтобы каждые 4 секунды он создавал объект **Летающая Рыба**. Используйте таймер и опцию **Родитель**. Пример кода приведен на рис. 2.9.

Рис. 2.9. Пример кода для создания объекта Рыба

- Создайте программу для Kodu, благодаря которой он сможет стрелять по появляющимся рыбам ракетами.
- Измените установки Kodu так, чтобы перезарядка ракетами происходила за минимальное время.
- Сохраните мир.

Блиц-опрос

1. Сколько различных путей можно нарисовать и запрограммировать в игре?

a. 5

b. 7

c. 11

d. 256

2. Какое действие следует выбрать для того, чтобы в группе одинаковых объектов можно было изменять код каждого объекта отдельно?

a. порождение

b. копирование

c. удаление

d. перемещение

3. Где находится опция «Родитель»?

a. в меню «**программа**»

b. в меню «**изменить установки**»

c. в главном меню **«редактировать мир»**

4. Какое действие следует выбрать для того, чтобы в группе одинаковых объектов можно было изменять код всех объектов одновременно?

a. порождение

b. копирование

c. удаление

d. перемещение

5. При выборе действия «сказать» опции Полный экран происходит...

a. остановка игры и вывод запрограммированного текста на экран;

b. продолжение игры, запрограммированный текст выводится в левом верхнем углу экрана;

c. продолжение игры, запрограммированный текст выводится на весь экран;

d. остановка игры, запрограммированный текст произносится персонажем.

Вопросы профессионального программиста:

Итак, получив первый опыт программирования в Kodu, постарайтесь ответить на вопросы:

В чем преимущества визуального программирования? Сформулируйте свое определение “Среда визуального программирования - это…”.

Обсудите с друзьями, какие задачи стояли перед разработчиками этой системы. Попробуйте создать перечень таких задач. Например:

a. Создать коллекцию ландшафтов
b. Запрограммировать возможность выбирать ландшафт из предлагаемого перечня и использовать его при моделировании территории
c. Создать коллекцию...

Это интересно...

Операционная система Windows 98 содержит 18 миллионов строк кода, а Windows XP (2002 год выпуска) уже 40 миллионов строк кода.

Это интересно...

Самая дорогостоящая в плане разработки игра называется ShenMue. Она была создана для Sega Dreamcast и обошлась разработчикам в 20 миллионов долларов.

Это интересно...

Крупнейшим игровым издателем в мире является компания Electronic Arts, в которой работает почти 5000 человек и которая ежегодно издает игр на сумму 3 миллиарда долларов.

Занятие 3. Знакомство с визуальной средой программирования Kodu: подсчёт баллов, индикатор здоровья, объект таймер

На этом занятии ты познакомишься с тем, как заставить игровой объект выполнять команды в определенные моменты времени, программировать характеристики и поведение персонажей, начислять им баллы за определенные действия, отслеживать успех действий в виде индикатора здоровья.

Использование таймера, индикатора здоровья и функции подсчета очков позволит придать игре профессиональный вид.

Для справки

Та́ймер (англ. *timer - счетчик времени, от time*: время) - в программировании называется объект, возбуждающий событие по истечении заданного промежутка времени. Событием является посылка сообщения, вызов функции, установка параметров объекта и т.д.

Упражнение 1. Назначение времени действия игрового объекта

Сюжет игры: Завод выпускает мячи каждые 5 секунд и монеты каждые 10 секунд.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Загрузите программу Kodu
- Создайте новый пустой мир
- Добавьте объект **Завод**. При необходимости уменьшите размеры объекта. Уменьшение размеров объекта выполняется через выбор команды *Уменьшить размер* в меню объекта (открывается щелчком правой кнопки мыши по объекту).
- Напишите код для объекта **Завод**, согласно которому каждые 5 секунд Завод производит футбольный мяч, а каждые 10 секунд - монеты. Для написания кода щелкните правой кнопкой мыши по объекту **Завод** и выберите команду **Программа**. В открывшемся окне введите программу, содержащую команды (пример приведен на рис. 3.1):

  When Таймер - 5 сек **Do** действия - запуск мяч.

  When Таймер - 10 сек **Do** действия - создать - монета.

Рис. 3.1. Пример кода с использованием опций таймера и подсчета баллов

- Запустите программу на выполнение клавишей **Esc**. Обратите внимание на отличие действий **Запуск** и **Создать**. Действие Запуск характеризуется более динамичным действием по созданию объекта.
- Сохраните созданный игровой мир под именем **Завод**. Для сохранения нажмите клавишу Home и в главном меню выберите команду **«сохранить мой мир»**.

Задание для самостоятельной работы:

- Попробуйте добавить код по выпуску снарядов каждые 60 секунд.
- Добавьте объект **Спутник** и сделайте так, чтобы он запускал объекты **Звезда** в интервале от 0 до 5 секунд. Пример кода приведен на рис. 3.2.

Рис. 3.2. Пример кода, когда таймер срабатывает случайным образом в интервале между 0 и 5 секундами

Для справки

Генератор псевдослучайных чисел (ГПСЧ, англ. *Pseudorandom number generator, PRNG*) — алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно равномерному).

Упражнение 2. Начисление баллов за действия объекта

Сюжет игры: За каждый мяч **Завод** получает **1** балл, за выпуск монеты - **5** баллов.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Загрузите программу **Kodu**.
- Откройте созданный в упражнении 1 игровой мир **Завод**. Для загрузки мира выберите команду **Загрузить мир** в главном меню. Переход в главное меню через кнопку **Home**.
- Добавьте код для подсчета баллов. Пример кода приведен на рис.3.1 в строках 2 и 4. В строке 5 записана команда остановить игру если будет нажата клавиша **Пробел**:

  When клавиша Пробел **Do** конец.
- Запустите программу на выполнение клавишей **Esc**. Примерный результат игры приведен на рис. 3.3. Нажмите клавишу **Пробел** для выхода из игры.

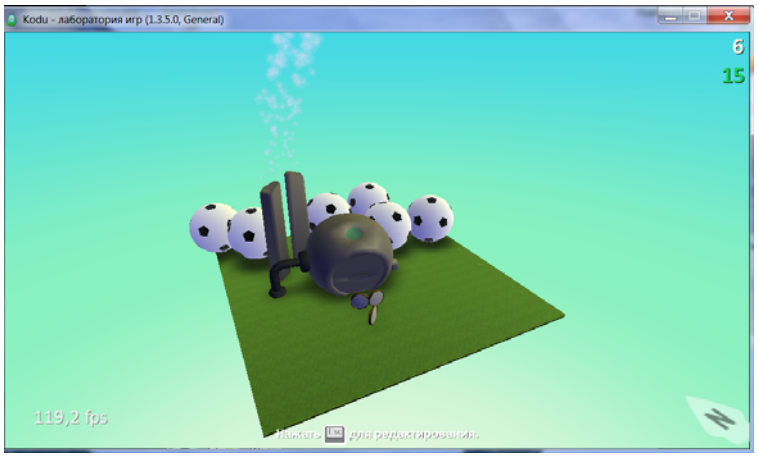

Рис. 3.3. Пример игры по выпуску мячей и монет и начисления баллов за работу

- Сохраните игру.

Задание для самостоятельной работы:

- Попробуйте изменить код программы так, чтобы за каждый мяч начислялось 10 баллов, а за каждый снаряд 20.
- Напишите программу, которая позволяет остановить игру, если набрано 100 баллов.

Задание для самостоятельной работы:

- В игре **Shoting Fish** задайте в программе объекта **Камень** строку, позволяющую производить счёт в игре (не забудьте сделать её дочерней).
- В программе для объекта **Камень** задайте условие окончания игры: появление в мире более пяти рыб.
- Добавьте в программе объекта **Летающая рыба** строку, отвечающую за вычитание одного очка из общего счёта, когда рыба теряет жизнь.
- Проведите эксперименты с разными условиями **счёта.**
- Сохраните игру

Для начала посмотри ролик "Таймеры и подсчёт очков в Kodu Game Lab"

Проверь себя!

1. Каким образом можно назначить увеличение/уменьшение счёта в игре?
2. Зачем может использоваться таймер? Как назначить измерение случайных интервалов времени?
3. Как сделать действие дочерним и каково основное свойство дочерних действий?

Дополнительное задание: Используя видео, создайте игру по предложенному сюжету. Объект **Тарелка**, двигаясь свободно по путям, создаёт красные яблоки с интервалом в 5 с, а также зелёные, через случайный интервал времени (в интервале от 10 до 20 с). **Байкер** поедает красные яблоки, за что получает одно очко. При съедании зелёного яблока **Байкер** уменьшает счёт на 5 очков. Сохраните созданную игру.

Упражнение 3. Использование индикатора уровня жизни.

Сюжет игры: Объект **Байкер** теряет здоровье при поедании зелёных яблок. При низком уровне здоровья объект **Байкер** светится красным.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

1. Откройте игру, созданную в упражнении 2 занятия 1. Сделайте так, чтобы **Байкер** терял здоровье при поедании зелёных яблок. При низком уровне здоровья добавьте красное свечение объекта **Байкер**.
2. Используя меню **«изменить установки»**, включите для объекта **Байкер** опцию **«показать жизни»** (рис. 3.4).

Рис. 3.4. Изменение установки **Показать жизни**

Здесь же, в меню опций, установите максимальное количество жизней - **5.**

3. При выходе из меню **«изменить установки»**, вы увидите над объектом **Байкер** зелёную полосу, которая и является индикатором здоровья (рис. 3.5). Проверьте, остаётся ли индикатор видимым в режиме игры.

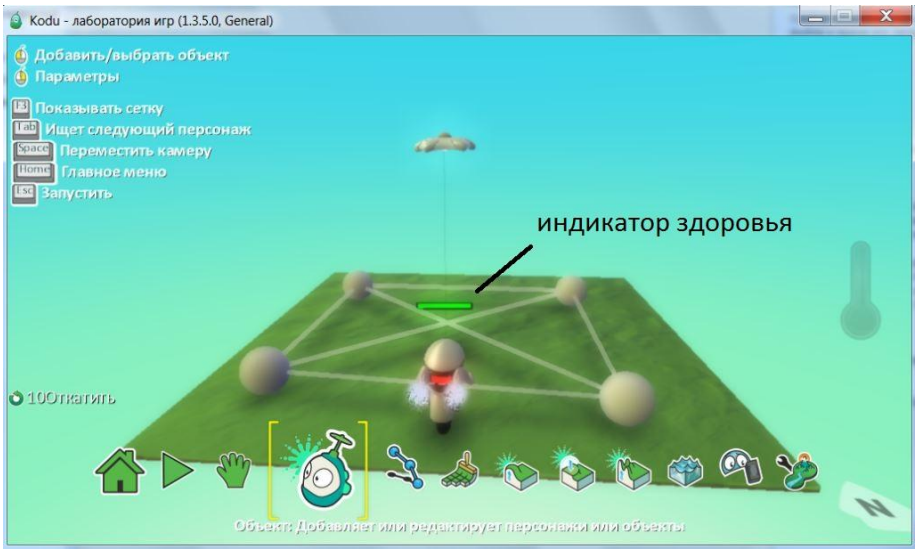

Рис. 3.5. Индикатор здоровья

4. В код объекта **Байкер** добавьте строку, уменьшающую количество жизней на одну при поедании зелёных яблок (рис. 3.6). Учтите, что ранее было запрограммировано поедание яблок при их касании, поэтому для условия используем **«касание»**. Для действия воспользуйтесь карточкой **«ущерб»,** которая находится в команде **«сражение»**.

Рис. 3.6. Строка кода, отвечающая за уменьшение уровня жизни

5. Запустите игру на выполнение и проследите за изменениями, происходящими с индикатором здоровья при поедании **Байкером** зелёных яблок (рис 3.7).

Рис. 3.7. Уменьшение уровня жизни

6. Добавьте свечение **Байкера**, сигнализирующее о критическом уровне жизни, используя соответствующий код в программе (рис. 3.8).

Рис. 3.8. Строка кода, программирующая свечение Байкера

7. Сохраните игру.

Задание для самостоятельной работы

- Придумайте систему средств, увеличивающих и уменьшающих возможности персонажа в вашем игровом мире или в игровом мире другого учащегося.
- Создав систему, подумайте, как можно использовать участки земли, объекты и других персонажей.
- Чтобы повысить сложность, попробуйте ограничить продолжительность увеличения или уменьшения энергии с помощью таймера.

Блиц-опрос

1. **Для подсчета очков в Kodu используется инструмент**
    a) таймер
    b) счет
    c) геймпад
    d) клавиши

2. **Для программирования времени выполнения команды в Kodu используется инструмент**
    a) таймер
    b) счет
    c) геймпад
    d) клавиши

3. **Какое наибольшее время можно выставить в игре при помощи объекта Таймер?**
    a) 30 секунд
    b) 60 секунд
    c) 90 секунд
    d) 30 минут

4. **У каких объектов Kodu имеется действие «съесть»?**
    a) Rover
    b) Kodu
    c) Байкер
    d) Яблоко

5. **В каком секторе кода находится команда остановки игры «конец»?**
    a) Перейти
    b) Игра
    c) Setting
    d) Действия

Вопросы профессионального программиста:

- Выясните, для чего в программировании используется генератор случайных чисел?
- Найдите в Интернете информацию о том, какое отношение к программированию имеет город Монте-Карло? Подсказка: это связано со случайными числами.

Это интересно...

История *компьютерных игр* начинается в 1947 году. В 1947 году Томасом Т. Голдсмит-младшим и Эстл Рей Манном был создан ракетный симулятор — самая ранняя из известных интерактивных электронных игр.

Это интересно...

Построенный специально для съемок "Военных игр" компьютерный центр NORAD, стал самым дорогостоящим искусственным съемочным помещением своего времени и обошелся в 1 миллион долларов. Всего на съемки фильма ушло 12 миллионов, в прокате "игры" собрали 74 миллиона.

Это интересно...

С 1982 года, когда был изобретен Тетрис, игра была продана 40 миллионами копий по всему миру.

Это интересно...

В Microsoft работает общей сложностью 72 тысячи сотрудников (75% мужчин, 25% женщин), годовой оборот компании только от сетевых проектов составляет 44 миллиарда долларов.

Занятие 4. Использование страниц в Kodu Game Lab. Создание уникальных историй и персонажей

На этом занятии ты познакомишься с тем, как использовать в игре несколько страниц, это пригодится тебе для того, чтобы назначить персонажу более продвинутые и сложные свойства и правила поведения. Использование многостраничного сценария в Kodu позволит повысить привлекательность игры и вывести ее на новый уровень.

Также мы рассмотрим правила, по которым создаются игры.

Для начала посмотри ролик

Видео “Использование страниц в Kodu Game Lab”

Проверь себя!

После просмотра видео ответьте на вопросы:

- Зачем используются страницы в игре?
- С помощью каких клавиш можно удалить/вставить код на страницу?
- Как происходит перемещение между страницами кода?

Упражнение 1. Работа с несколькими страницами

Сюжет игры: Завод выпускает сначала мячи, потом монеты, а потом ракеты.

Для успешного выполнения упражнения и создания игры, четко следуйте предложенному алгоритму:

- Загрузите программу **Kodu**.
- Откройте мир **Завод**, созданный в упражнениях 1-2 занятия 3.
- Измените код персонажа **Завод**, запрограммировав его на выпуск мячей каждую секунду. По прошествии 5 секунд перейти на страницу 2, пример кода приведен на рис. 4.1.

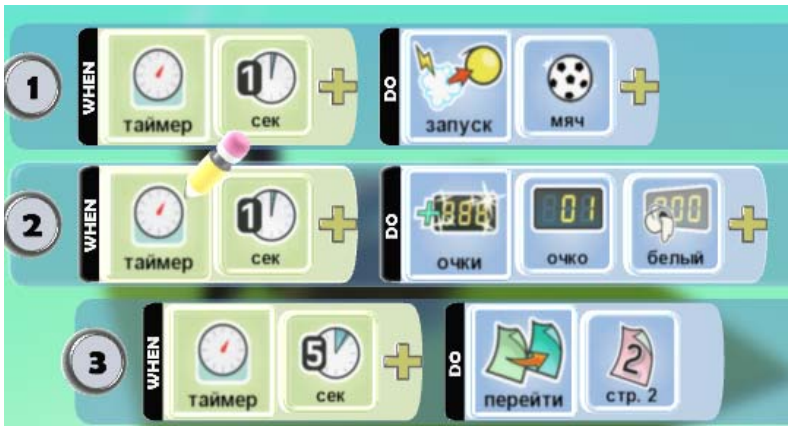

Рис. 4.1. Кодирование страницы 1 на выпуск мячей

.

- На странице 2 создайте код, позволяющий выпускать монеты каждый 10 секунд. А через 30 секунд запрограммируйте перейти на страницу 3. Пример кода приведен на рис.4.2.

Рис. 4.2. Кодирование страницы 2 на выпуск монет

- На странице 3 добавить код для выпуска ракет.
- Запустить программу на выполнение. Посмотреть результаты. Сохранить мир.

.

Задание для самостоятельной работы:

- Измените код так, чтобы переход на следующую страницу происходил не по времени, а при накоплении определенного количества баллов.
- Измените игру **Shooting Fish** так, чтобы через 10 секунд после появления объект **Летающая Рыба** направлялся к объекту **Kodu**, и при их встрече происходило бы окончание игры. Проверьте функциональность вашей игры.

Вопросы профессионального программиста:

- Подумайте, с какой целью программисты используют подпрограммы (вспомогательные алгоритмы)?
- Как реализуется возможность создания и вызова подпрограмм в Kodu?
- Какие дополнительные возможности вы получаете, используя подпрограммы?

Блиц-опрос

1. Сколько страниц находится в Kodu?

a) 1

b) 5

c) 9

d) 12

e) 20

2. Какой команды нет в основном меню работы со страницами?

a) Вырезать страницу

b) Вставить страницу

c) Переключить страницу

d) Копировать страницу

3. Какая команда отсутствует в контекстном меню (вызывается правой кнопкой мыши) работы со строками?

a) Вырезать строку

b) Вставить строку

c) Удалить строку

d) Копировать строку

Упражнение 2. Создание игры по предложенному сценарию

Вы познакомились с основными компонентами среды программирования **Kodu**. Но для создания собственной компьютерной игры нужна не только среда программирования, важно еще знать и общие правила создания игр.

Разделитесь на группы по 2-3 человека и попробуйте смоделировать деятельность команды по разработке проекта игры.

Итак, пробуем свои силы?! Ваш первый заказ:

Спортивный теннисный клуб планирует познакомить учеников начальных школ вашего города с игрой в теннис. Они наняли вас для проектирования игры, в целом основанной на правилах тенниса, чтобы увлечь ребят теннисом и повысить популярность этого вида спорта.

Для справки

Теннис, или **большой теннис** — вид спорта, в котором соперничают либо два игрока («одиночная игра»), либо две команды, состоящие из двух игроков («парная игра»). Задачей соперников является при помощи ракеток отправлять мяч на сторону соперника так, чтобы тот не смог его отразить не более чем после первого падения мяча на игровом поле на половине соперника.

Вам предоставили информацию и обзорный тур по тренировочным помещениям клуба. Итак, клуб выдвигает несколько требований для игры:

- два игрока;
- красочность;
- подпрыгивающий/отскакивающий объект (мяч), по которому ударяют два игрока, перебрасывая его друг другу;
- возможность вести счет.

Шаг 1. Итак, сюжет (идея и смысл) игры понятен. Для того, чтобы выполнить заказ, вам сначала понадобится сделать раскадровку игры с текстом.

Для справки

Раскадровка — процесс создания предварительной визуализации кадров. История раскадровки берет свое начало в эпоху немого кино.

Создайте на листах бумаги раскадровку, отражающую видеоигру в теннис. На различных листах должны последовательно отображаться снимки экрана, соответствующие различным этапам игры: с начала (экран названия) и до конца (экран победы).

Шаг 2. Создайте виртуальный мир, соответствующий сюжету игры:

- спроектируйте теннисный корт, выбрав его размеры и нанеся соответствующую разметку;
- подумайте, какие объекты можно добавить еще на территории корта и за его пределами (сетка, место для судьи на корте, трибуны, деревья за трибунами и т.п.).

Для справки

Теннисный корт (англ. *court*, от лат. *cohors* — огороженное место) — ровная, прямоугольная площадка для игры в теннис. Длина корта — 23,77 м, ширина — 8,23 м

(для одиночной игры) .

Шаг 3. Создайте игровые объекты: персонажей-теннисистов и мяч. Обратите внимание на их свойства и подумайте, как вы будете их использовать в процессе игры.

Шаг 4. Создайте код, определяющий правила поведения игроков и объекта:

- задайте способ (клавиши) управления и перемещения игроков по полю;
- напишите программу, задающую поведение мяча;
- опишите условия изменения счёта и запрограммируйте эту ситуацию.
- реплики героев и звуки

Шаг 5. Сделайте (в соответствии с имеющейся раскадровкой) экраны начала и окончания игры.

Шаг 6. Протестируйте игру в паре. Предложите поучаствовать в тестировании своим друзьям. Внесите необходимые правки в код. При необходимости осуществите корректировку ландшафта.

Шаг 7.Проведите презентацию игры и представьте разработку каждой группы. Выскажете своё мнение о том, что вам особенно понравилось в представленных проектах, предложите конкретные шаги по доработке.

Шаг 8. Сравните ваш подход и подходы ваших друзей к программной реализации поставленной задачи по следующим параметрам:

- оформление виртуального мира и его соответствие поставленной задаче;
- прозрачность правил игры;
- корректность поведения игроков;
- алгоритм подсчета очков (работы счетчика);

Выберите наиболее удачные решения написания кода, аргументируя свой выбор.

Вопросы профессионального программиста:

- Какие возможности объектно-ориентированного программирования вы уже использовали?
- Продолжите фразу “Объектно-ориентированное программирование позволяет…” Обсудите свои выводы с друзьями!

Это интересно...

Компьютерные игры часто создаются на основе фильмов и книг; есть и обратные случаи. С 2011 года компьютерные игры официально признаны в США **отдельным видом искусства**.

Это интересно...

Геймдизайнер — специалист, разрабатывающий правила, стиль и дизайн компьютерных игр. Роль геймдизайнера аналогична роли постановщика задачи в обычном программировании и режиссёра в кино. Профессия появилась в конце 1980-х годов.

Это интересно...

Раздел математики под названием **теория игр** изучает оптимальные стратегии в играх. Под игрой понимается процесс, в котором участвуют две и более стороны, ведущие борьбу за реализацию своих интересов. Каждая из сторон имеет свою цель и использует некоторую стратегию, которая может вести к выигрышу или проигрышу — в зависимости от поведения других игроков. Теория игр помогает выбрать лучшие стратегии с учётом представлений о других участниках, их ресурсах и их возможных поступках.

Это интересно...

Существует профессия **сценарист компьютерных игр**. Известный сценарист канадец Дрю Карпишин создает сценарии для видео и компьютерных игр.

Занятие 5. Разработка своей оригинальной игры от “А” до “Я”

Для справки:

Проект (от лат. *projectus — брошенный вперед, выступающий, выдающийся вперёд*) — это работа, планы, мероприятия и другие задачи, направленные на создание нового продукта.

В нашем случае проектом будет созданная игра. Какая игра? Это решать вам! Но постарайтесь, чтобы она была интересна не только вам, но и вашим друзьям, родителям, близким, которые будут в нее играть. Вы можете посмотреть примеры удачных игр, созданных в Kodu другими разработчиками на сайте http://www.kodugamelab.com/

Итак, изучив все возможности **Kodu**, приступаем к проектированию **своей** игры от “А” до “Я”!

Советы профессионального программиста:

Конечно, программировать игру можно в одиночку. Но практика показывает, что наиболее удачными являются проекты (продукты), разработанные командой профессионалов с различной специализацией: сценаристов, дизайнеров, тестировщиков и непосредственно программистов! Важно не только собрать, но и правильно распределить роли. Это этого зависит и настроение, и вклад каждого из участников, и самое главное - результат!

Поэтому мы рекомендуем пригласить для совместной работы над игрой одноклассников или друзей. А может быть, родителей?

На этапе обсуждения обязательно обсудите роль каждого из участников вашей команды. Проведите “**мозговой штурм**”, а результаты записывайте в Протокол “мозгового штурма”

Для справки

Метод **мозгового штурма** (мозговой штурм, мозговая атака, англ. brainstorming) — оперативный метод решения проблемы на основе стимулирования творческой активности, при котором участникам обсуждения предлагают высказывать как можно большее количество вариантов решения, в том числе самых фантастичных. Затем из общего числа высказанных идей отбирают наиболее удачные, которые могут быть использованы на практике.

Шаг 1. Для начала выберите жанр игры. Что это будет за игра? Уточним, какие бывают жанры. Выбирай…

Для справки:

Жанр определяется целью игры. Игра может принадлежать как к одному, так и к нескольким жанрам.

Приключенческая игра (Adventure) — игра, обладающая полноценным литературным сюжетом, и игрок в процессе игры сам раскрывает все перипетии этого сюжета.

Ролевая игра (RPG — англ. *Role Playing Game*) — игра, отличительной особенностью которой является наличие у персонажей определённых навыков и характеристик, которые можно обрести, а впоследствии развивать, выполняя какие-либо действия.

Компьютерный симулятор (Simulator) — игра, полностью имитирующая какую-либо область реальной жизни, например, имитация управления гоночным автомобилем или самолётом.

Головоломка (Puzzle) — игра, полностью или более чем наполовину состоящая из решения различных логических задач и головоломок.

Образовательная игра — игра, включающая в себя элементы обучающих программ, которые подаются через сам игровой процесс и, благодаря повышению интереса к ним в связи с необычным антуражем, впоследствии хорошо запоминаются.

Забавы — игры, в основном рассчитанные на детей, где психологическое впечатление от происходящего на экране гораздо важнее самого процесса игры — например, вид лопающихся пузырьков.

Результат этого шага - разработка **ключевой идеи** игры. Вы должны **в общих чертах** представлять, **где** будут происходить игровые действия, **какими объектами** будет управлять играющий и с какой **целью**.

Шаг 2. Проектируем сюжет игры. Заглянем в словарь...

Для справки:

Сюжет (от фр. subjet — «предмет») — в литературе, театре, кино и играх — ряд событий (последовательность сцен, актов), происходящих в определенном порядке и выстроенных для (зрителя, игрока) .

Советы профессионального разработчика игр:

Проектирование сюжета, естественно, потребует:

- придумать и создать эскиз на бумаге как будет выглядеть виртуальный игровой мир (ландшафта и его объектов: вода, дороги, деревья, горы и т.д.), в котором будут развиваться события;
- придумать героев (персонажей, объекты), управление которыми происходит в процессе игры (развития сюжета) или которые будут действовать самостоятельно. Заметим, что сюжет игры, во многом будет определяться возможностями тех объектов, которыми будет управлять играющий.

В любой игре используют такие элементы как:

- Основной персонаж/персонажи (главные герои).
- Основной противник или препятствия, которые противодействуют достижению цели игры (победе). Например, препятствием может стать условие или ограничение, например, по времени.
- Второстепенные персонажи.

Рис. 5.1. Идея сюжета игры

Сюжет игры предполагает описание последовательности действий, которые происходят в процессе игры с главным героем/персонажем. Обязательно сначала сформулируйте и запишите идею сюжета вашей игры, подберите соответствующее название.

Для того, чтобы сюжет игры был реализуемым, надо хорошо изучить возможности управляемых объектов и персонажей. Постарайтесь ответить на вопросы:

- Какую миссию будут выполнять главные герои (персонажи, объекты)?
- Нужны ли в вашей игре дополнительные герои и объекты? С какой целью?
- Будут ли персонажи и объекты, препятствующие достижению цели?
- Что каждый из героев умеет/будет делать (роль каждого из них в сюжете)?

- Каковы свойства и характеристики выбираемых объектов и как они будут использоваться в процессе игры?
- При каких условиях и в каких обстоятельствах будут использоваться те или иные свойства (возможности) персонажей (объектов)?
- Как будет осуществляться управление персонажами (объектами)?

Итак, на этом шаге вы должны:

- придумать ландшафт и выбрать управляемые объекты;
- изучить/освежить в памяти возможности управляемых персонажей (объектов);
- выбрать варианты поведения персонажей согласно сюжету.

Шаг 3. Детализируем цель игры.

Советы профессионального разработчика игр:

Очевидно, что сюжет игры должен быть ориентирован на какой-либо результат. Каждый играющий в вашу игру должен понимать, что он должен сделать, каков должен быть результат.
Как правило, цель игры (а значит, и победа игрока!) во многом определяется ее жанром. Например, игрок должен по сюжету:

- набрать как можно больше баллов (очков);
- то или иное действие выполнить за ограниченный период времени (например, лабиринт надо пройти за 2 минуты);
- прийти к финишу первым;
- собрать/создать/открыть/разрушить/освободить/ те или иные объекты, персонажей, информацию и т.п.
- соревноваться с другим персонажем, управляемым вторым игроком.

Результат этого шага - определить цель проектируемой вами игры и миссию игровых персонажей (героев, объектов) в процессе достижения этой цели.

Шаг 4. Создаём раскадровку и описываем правила игры.

Советы профессионального разработчика игр:

Вам потребуется описать правила для каждого объекта и персонажа, который встретится нам по ходу сюжета. Здесь также можно предусмотреть реплики персонажей и другие свойства и варианты поведения: сияние, исчезание, движение, звуковой сигнал, выражение эмоций, захват предметов и т.д.

Пример правила: Объект **Рыба** исчезает при столкновении с объектом **Снаряд**

Шаг 5. Создаем игровой мир, ландшафт, атмосферу согласно нашей раскадровке и сюжету.

Советы профессионального разработчика игр:

Такие вещи как цвет неба, текстура ландшафта, наличие или отсутствие воды влияют на восприятие игроком сюжета и его отношение к игре. Вид игрового мира должен соответствовать сюжету и жанру игры.

Пример: Если действие приключенческой игры происходит в дремучем лесу, то уместно выбрать тёмный цвет неба (а не ярко-голубой!).

Шаг 6. Добавляем персонажей. Программируем действия героев (персонажей, объектов) согласно разработанным правилам и сюжету игры.

Шаг 7. Проводим тестирование игры.

Советы профессионального разработчика игр:

Важно проводить тестирование сразу, по мере создания кода на каждом из этапов разработки в соответствии с раскадровкой - так вам будет проще отследить ошибки в своем коде и быстро их поправить.

Когда игра будет готова, пройдите ее от начала и до конца. Во время прохождения обращайте внимание не только на корректность поведения героев, но и на детали. Постарайтесь сделать что-то против заданных вами правил! В процессе разработки используйте метод “защита от дурака”.

Для справки

Защи́та от дурака́ — защита предметов пользования (в особенности, техники), программного обеспечения и т. п. от очевидно неверных действий человека, как при пользовании, так и при техническом обслуживании или изготовлении.

Также важно в процессе тестирования пройти игру всеми возможными путями, перебрав все варианты.
Проверьте, реализована ли цель игры и соответствует ли она обозначенному жанру.

Шаг 8. Презентация игры.

Советы профессионального разработчика игр:

Советую для презентации игры привлечь как можно большую аудиторию. Это могут быть ваши друзья, одноклассники, родственники. Вы можете провести презентацию игры в социальной сети. Во время презентации обязательно обоснуйте, чем ваша игра отличается от других и почему люди должны захотеть играть в эту игру?

- Расскажите о сюжете и правилах вашей игры, что вас вдохновило на ее создание.
- Представьте свою команду разработчиков (кто за что отвечал при разработке), отметьте вклад каждого участника.
- Запустите игру и предложите поиграть в нее всем желающим. Поинтересуйтесь мнением. Обратите внимание на замечания и пожелания, возникшие у игроков трудности. Это поможет вам усовершенствовать игру и привлечь новых игроков. Это самый большой успех программиста!

Шаг 9. Поделитесь (загрузите) своей игрой в сообществе Kodu и на сайте “ИТ для молодёжи” .

Дорогие друзья!

Вот и закончились 5 занятий по освоению Kodu. Вы сделали первые шаги в программировании и получили важные навыки по созданию 3D-игр. Теперь вы знаете и умеете:

- разрабатывать сюжет и стратегию игры;
- проектировать виртуальный мир, создавать и редактировать модель местности;
- управлять объектами, создавая программный код;
- привязывать действия объектов к определенному времени;
- использовать звуковое сопровождение в игре;
- программировать счет очков и др.

Эти навыки пригодятся вам в дальнейшем для освоения языков программирования. Надеемся, что вы поняли, что перед вами открыты все пути: можно освоить другие языки программирования и попробовать себя в разработке мобильных приложений для телефонов и планшетов, создать онлайн-сервис, написать программы для себя и своих друзей.

Помните, что навыки программирования помогают вам самоутвердиться, они способствуют развитию твоего мышления, памяти, внимания, умения формулировать и формализовать свои мысли и действия. Программируя, вы учитесь ставить себя на место игрового персонажа, прогнозировать развитие событий. Это очень пригодится в жизни и работе для достижения своих целей!

Но самое важное, что вы работали **вместе**, как настоящая к**оманда программистов**, выполняя общий итоговый проект. Вы делились идеями, обсуждали сюжет и способы реализации игры, приходили к единому мнению. Может, не всегда это было легко, но умение договариваться - это тоже очень важный профессиональный навык.

Учитесь везде и всегда! Ведь вы сможете менять мир, в котором мы живем, делать его лучше и интереснее! Создавайте и творите! Успехов вам!

Вам понравилось создавать игры в Kodu? Обязательно поделитесь мнением о пройденном курсе, заполнив небольшую анкету. Это займет всего 2 минуты.

ПРИЛОЖЕНИЯ

Инструкция по установке и работе с Kodu

Перед установкой программы ОЗНАКОМЬСЯ с требованиями к ПК:

1. ПК должен работать под управлением ОС Windows.
2. ПК должен поддерживать графику DirectX9.
3. ПК должен поддерживать Shader Model версии 2.0 или более поздней.

Определение версии DirectX

- Для Windows 7 и Windows Vista: в меню «Пуск» введите в поле поиска строку Cmd.

  Для Windows XP: в меню «Пуск» выберите пункт «Выполнить» и введите команду Cmd.
- Выполните в открывшемся окне команду Dxdiag. Версия DirectX будет указана на первой вкладке.

Если на компьютере не установлено ПО .NET Framework версии 3.5 и XNA Game Studio версии 3.1, программа установки предложит установить эти продукты.

Загрузка Kodu с сайта разработчика

- Kodu является бесплатно распространяемым программным обеспечением (ПО). Для того чтобы установить программу на ПК, её необходимо скачать с сайта разработчика http://fuse.microsoft.com/Kodu (рисунок 1). Информация на сайте представлена на английском языке, но современные браузеры (программы для просмотра веб-страниц) позволяют автоматизировать перевод на русский язык. В контекстном меню веб-страницы (открывается при щелчке по телу веб-страницы правой кнопкой) выбрать команду ***«Перевести на русский».***
- На главной странице сайта выбрать ссылку **Download it here for FREE**, (рисунок 2).
- На сайте разработчика программы щелкнуть по кнопке **Download, (**рисунок 3).
- Браузер предлагает выбрать действия - выполнить программу или сохранить её. Программа Kodu занимает 199 МБайт. При выборе команды ***«выполнить»*** программа автоматически начинает установку. При выборе команды

«***сохранить»*** программа сохраняется на жесткий диск компьютера, что позволит установить ее на других ПК без обращения к Интернету.

- Выбрать команду «**сохранить как»**, указав место для хранения файла, (рисунок 4). Информацией о том, что файл данного типа может нанести вред компьютеру можно пренебречь, так как программа скачивается с официального сайта корпорации Microsoft, которая ответственно подходит к вопросам обеспечения информационной безопасности своих пользователей.
- Дождаться загрузки файла KoduSetup.msi на жесткий диск компьютера. Время ожидания зависит от скорости работы интернета.
- Найти сохраненный программный файл и двойным щелчком по имени файла запустить его на ПК.
- В процессе инсталляции (установки) файла могут появиться сообщения о том, что некоторые компоненты не установлены (рисунок 5). Следует выполнить их инсталляцию. Информация, приведенная на рисунке 5, может отличаться, так как она зависит от того, какие компоненты были установлены на компьютере ранее. В данном примере приложение .NET Framework 3.5 было установлено ранее, а утилита XNA Framework 3.1 требует установки. Для установки следует щелкнуть по кнопке **«click here to download»** и загрузить необходимые для работы ресурсы.
- Загруженный файл с утилитами следует найти (обычно он хранится в папке «Загрузки») и установить (инсталлировать) приложение на ПК.
- После установки всех необходимых для работы приложений NET Framework 3.5 и XNA Framework 3.1, продолжить процесс

  инсталляции, обязательно отметив флажок о присоедини к лицензионному соглашению «**I accept the terms in the License Agreement»** (рисунок 6).
- После успешной инсталляции на рабочем столе и системной кнопке «Пуск» появятся иконки для запуска программы.

Процесс инсталляции Kodu можно понаблюдать на **видео-уроке “Установка Kodu”**

Рисунок 1. Главная страница ресурса Kodu Game Lab

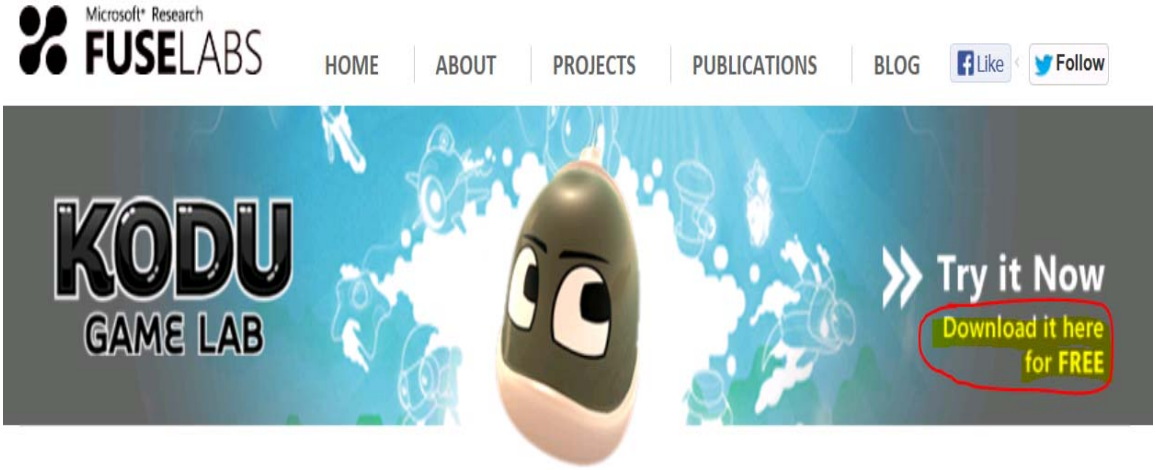

Рисунок 2. Ссылка для скачивания программы

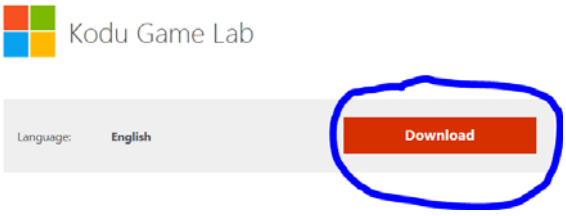

Рисунок 3. Выбор опции для сохранения программы на ПК

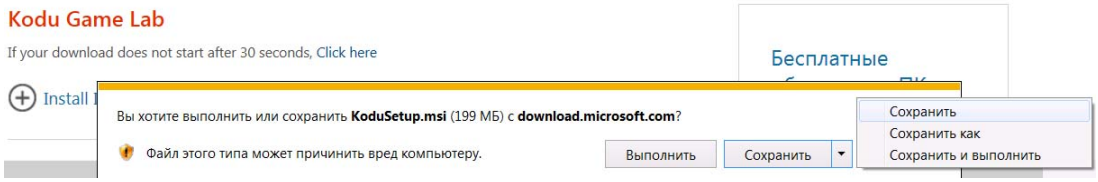

Рисунок 4. Выбор команды для сохранения программы на ПК

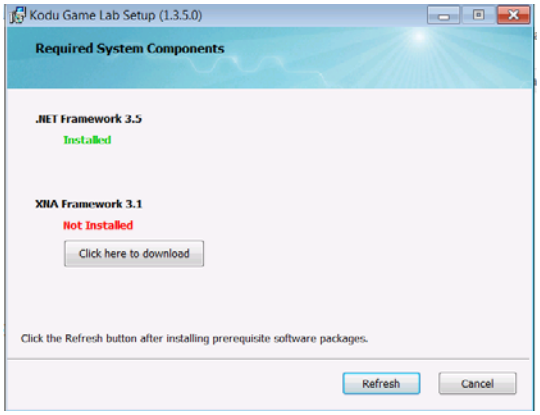

Рисунок 5. Информация о доступности необходимых приложений

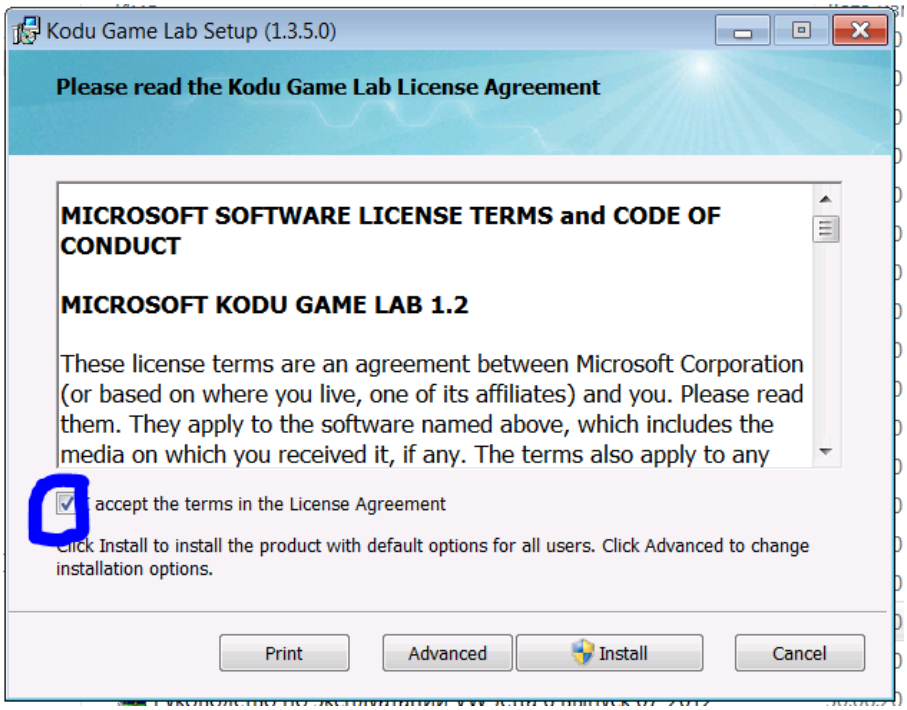

Рисунок 6. Лицензионное соглашение

Оптимизация Kodu для работы на ПК

Если изображение в игре движется плавно, значит видеоадаптер обеспечивает достаточную частоту смены кадров. Для комфортной игры необходимо, чтобы частота смены кадров составляла не менее 20 кадров в секунду.

Если при получении команд Kodu движется рывками, воспользуйтесь приведенными ниже рекомендациями, чтобы повысить удобство игры.

1. **Определение частоты смены кадров**

Запустите приложение **«Configure Kodu Game Lab»** из меню программ на ПК. Появится панель инструментов настройки Kodu. Установите флажок «Show Frames per Second» («Показывать число кадров в секунду»).

При следующем запуске Game Lab на экране будет отображаться количество кадров в секунду (Frames per Second, fps). Отображение числа кадров в секунду можно включать и отключать в любой момент, устанавливая или снимая флажок Show Frames per Second в панели инструментов настройки.

2. Повышение частоты смены кадров

- Если в панели инструментов настройки выбран параметр Advanced (Shader Model 3) («Модель построителя текстуры версии 3»), выберите вместо него параметр Standart (Shader Model 2) («Модель построителя текстуры версии 2»).
- Снимите флажок Fullscreen («Полноэкранный режим»).
- Выберите в раскрывающемся списке Resolution («Разрешение») меньшее разрешение. При этом уменьшится детализация изображения, и приложение будет потреблять меньше ресурсов процессора.
- Снимите три флажка визуальных эффектов: Glow («Свечение»), Smoothing («Сглаживание») и Animation («Анимация»).

✓ ПРИМЕЧАНИЕ ДЛЯ ПОЛЬЗОВАТЕЛЕЙ НОУТБУКОВ. Некоторые ноутбуки со встроенными видеоадаптерами не поддерживают ряд возможностей Kodu, но позволяют использовать основные средства Kodu и играть в игры.

ПРИЛОЖЕНИЕ ПРОТОКОЛ (Занятие 5)

Протокол “мозгового штурма”

Шаг	**Результат**
Опишите жанр игры	
Придумайте название вашей игры	
Сформулируйте и запишите идею сюжета вашей игры	
Опишите виртуальный мир: в каких условиях (ландшафт, объекты и т.п.) будут происходить действия игры?	
Какие игровые персонажи или объекты вы планируете использовать? Назовите главных и второстепенных персонажей, возможных противников и т.п.	
Опишите характеристики игровых объектов. Какие из их свойств будут использоваться в игре?	
Детализируйте цель проектируемой игры	
Выполните раскадровку (выполните на отдельных листах и приложите к протоколу!)	
Опишите правила для каждого объекта на каждом этапе игры в соответствии с раскадровкой	

Итак, переходим к написанию кода!

Успехов вам!