Game Development

junior

STEP
IT ACADEMY

Урок 1

Создание первого уровня

➢ **ПЛАН РАБОТЫ.**

Добавление главного персонажа, заднего фона, платформ, статичных противников и монет. Создание параллакс эффекта для заднего фона. Создание второго слоя и добавление объектов TiledBackground, программирование первых событий передвижение героя при помощи поведения Platform, ScrollTo, Solid и событий при касании противников и монет при помощи триггера On collision with another object.

➢ **ДОПОЛНИТЕЛЬНО.**

Изменение параметров поведения Platform при соприкосновении со специальными объектами.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Создание первого уровня

«Construct 3» — конструктор двумерных игр для Windows. Эта платформа позволяет каждому желающему создавать 2D-игры любой сложности и любого жанра, даже не имея навыков программирования.

В *«Construct 3»* создано много популярных игр, и вы о них, наверняка, слышали!

1. **«The Next Penelope»**. Популярная и красочная гоночная игра с оригинальной историей и сложной системой вознаграждения за прохождения уровней (рис. 1).

Рисунок 1

2. **«Angvik»**. Игра о воине, который должен очистить ранее счастливую планету от монстров (рис. 2).

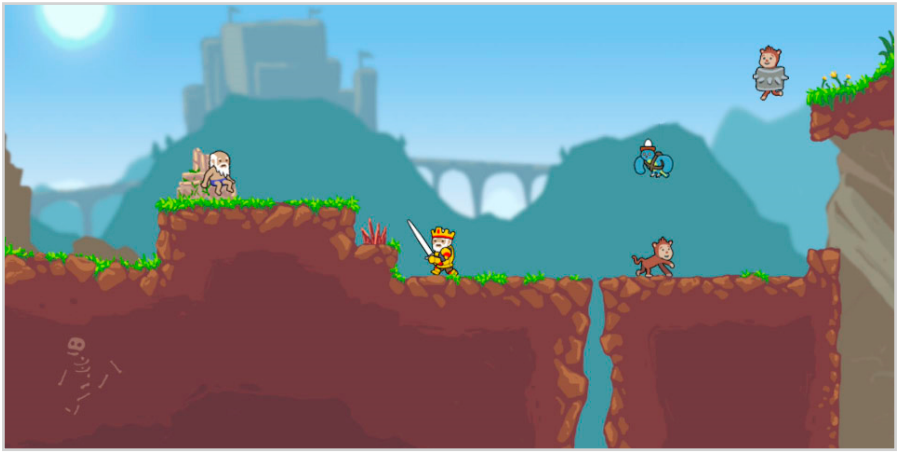

Рисунок 2

3. **«Airscape»**. Это мегапопулярный 60-уровневый платформер, а котором необходимо помочь осьминогу спастись от лап вражеской угрозы (рис. 3).

Рисунок 3

Давайте теперь создадим собственную игру в *«Construct 3»*! Но, прежде чем приступать, посмотрите, какие

красочные платформеры можно создавать с помощью *«Construct 3»* и публиковать на сайтах. Набирайтесь вдохновения и приступим! (рис. 4–6)!

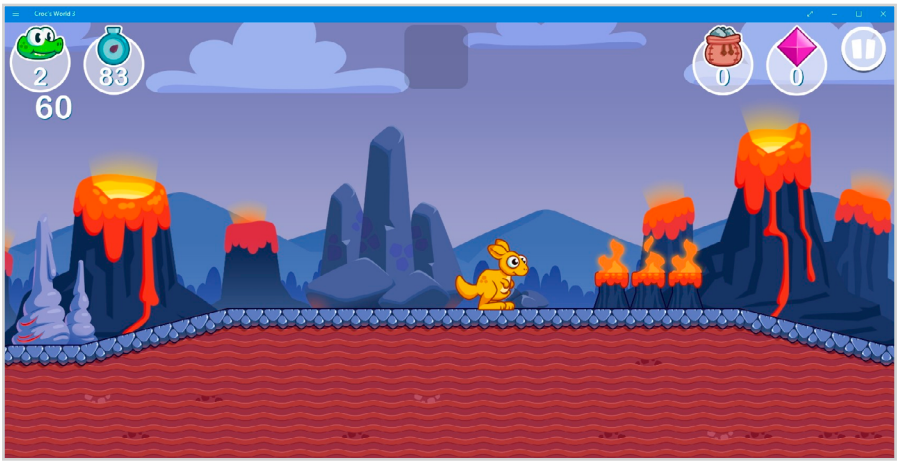

Рисунок 4

Рисунок 5

Рисунок 6

Пришло время реализовать собственный платформер. Переходим по ссылке и открываем программу в браузере. Нажимаем «New Project» (рис. 7).

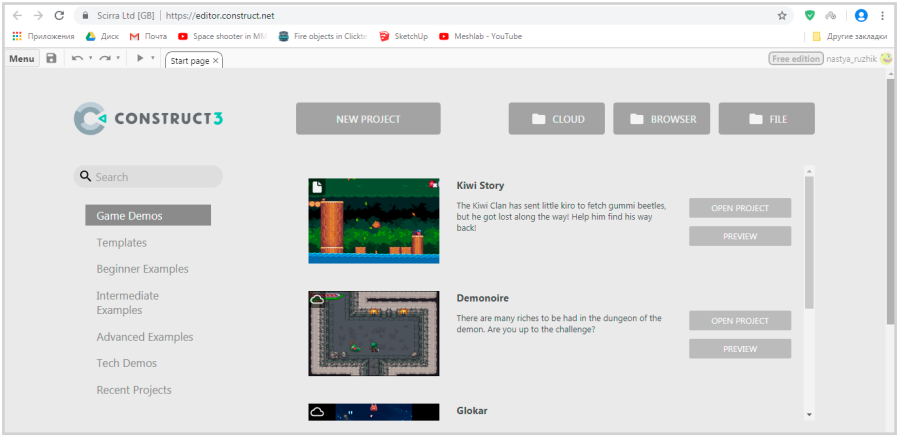

Рисунок 7

Открывается рабочая сцена, на которой мы будем создавать игровую карту (рис. 8).

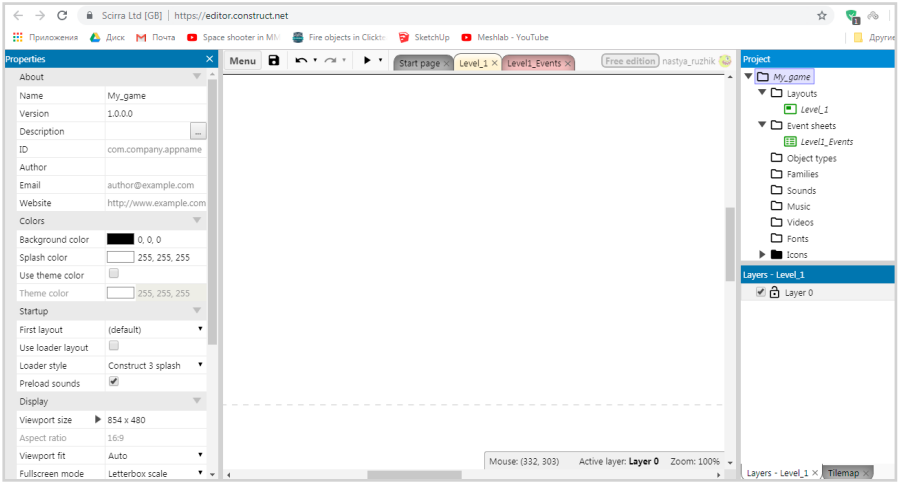

Рисунок 8

Давайте подготовим нашу карту к работе.

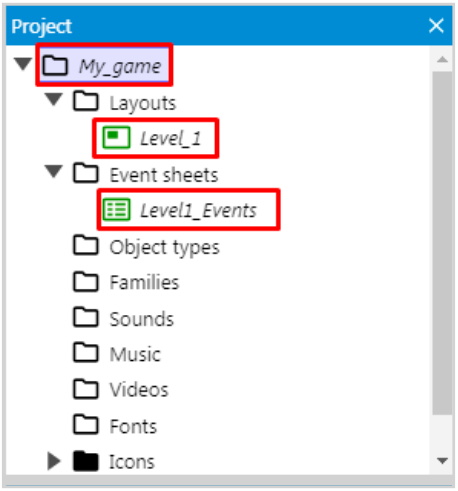

Рисунок 9

Переименуем проект. Во вкладке «Project» нажмем правой кнопкой мыши на папке и выберем «Rename».

Назовем проект «MyGame». Также переименуем «Layout1» в «Level1», а «Event Sheet 1» — в «Level1_Events» (рис. 9).

Теперь настроим размер карты. Для этого выберем «Level 1» и во вкладке «Properties» в параметре «Layout Size» установим значения «3000×600». Включим свойства «Snap to grid» (*Выравнивать по сетке*) и «Show grid» (*Показывать сетку*). Размер сетки устанавливаем «70×70» (рис. 10).

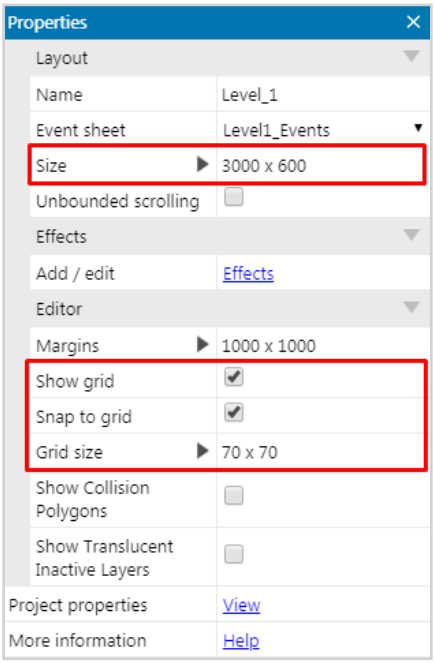

Рисунок 10

Добавим первый игровой элемент. Чтобы это сделать, находим соответствующий файл формата *.png* в проводнике и перетягиваем его на рабочую область (рис. 11).

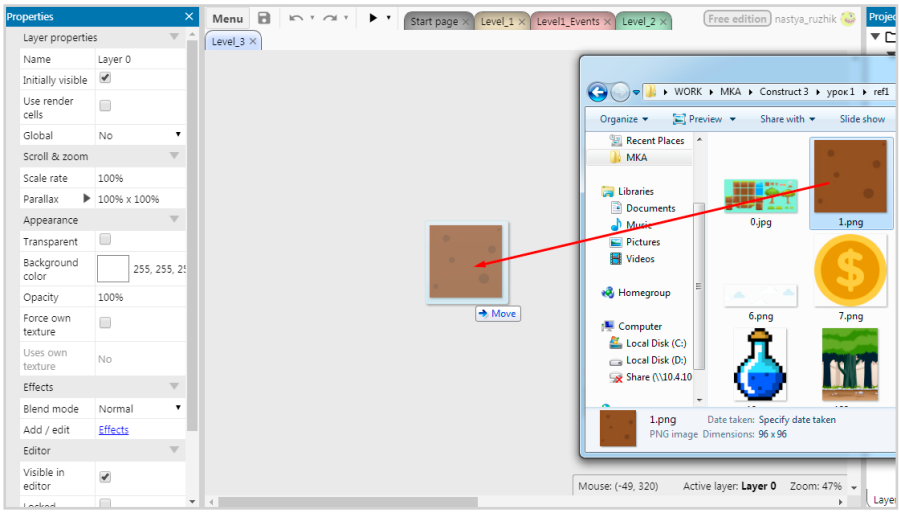

Рисунок 11

Делаем небольшую карту, копируя расположенный на сцене блок (рис. 12).

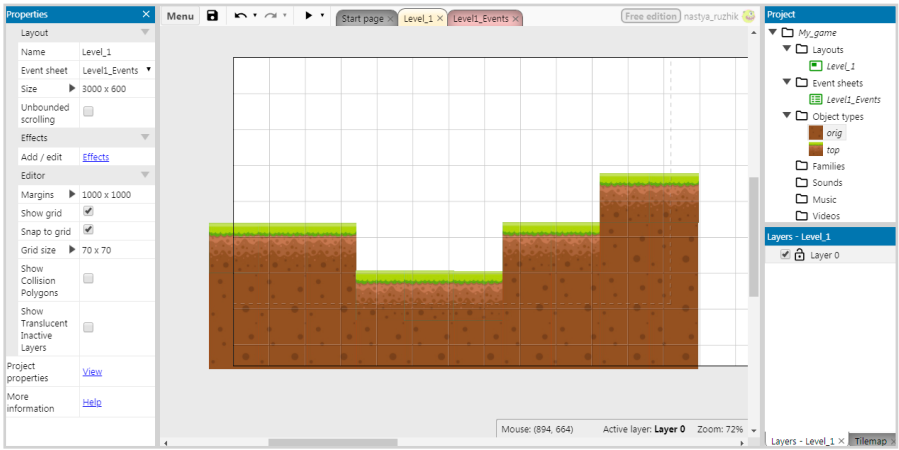

Рисунок 12

Добавляем персонажа по тому же принципу (рис. 13).

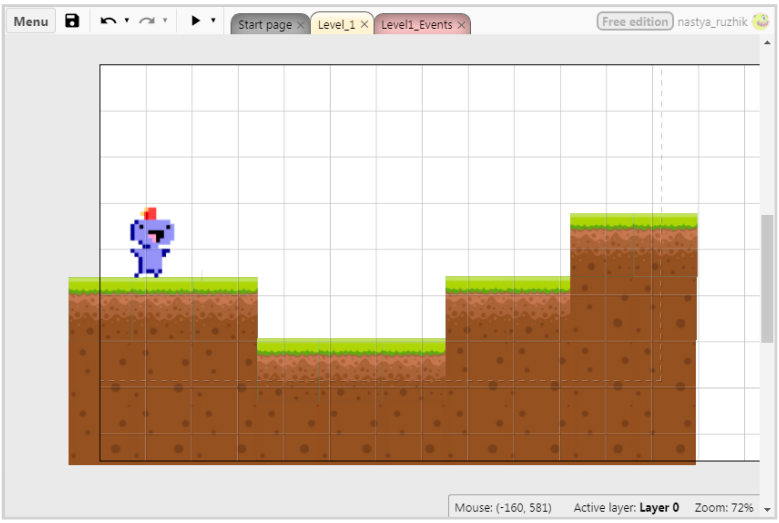

Рисунок 13

Переименовываем его в «Player» (рис. 14).

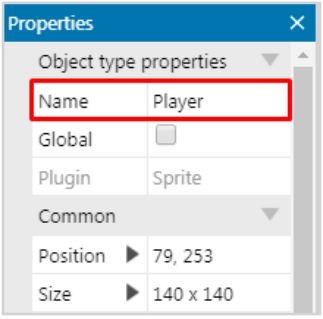

Рисунок 14

Чтобы посмотреть, как игра будет выглядеть в итоге, можно нажать на иконку «Play» в верхнем меню (рис. 15).

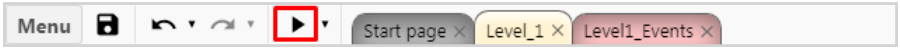

Рисунок 15

Откроется отдельное окно браузера, где можно увидеть готовую картинку (рис. 16).

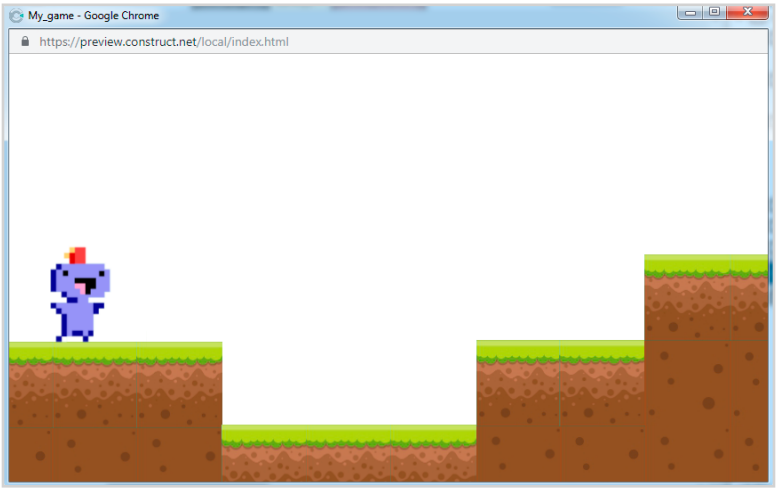

Рисунок 16

Если при воспроизведении игры у вас появилась белая область слева, на панели «Properties» снимите галочку из чекбокса «Unbounded scrolling» (рис. 17–18).

Рисунок 17

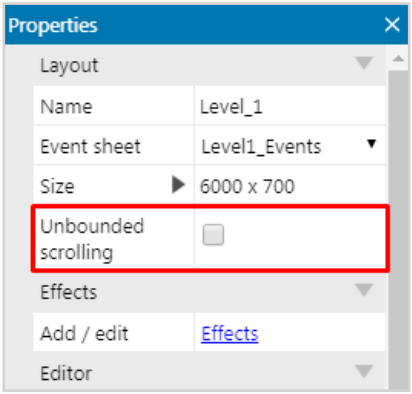

Рисунок 18

Можете попробовать поиграть, понажимать на стрелочки и пробел, но, к сожалению, ничего не произойдет. Почему? Дело в том, что все объекты на сцене являются картинками. Чтобы это исправить, нам необходимо наделить эти картинки особой игровой логикой. В «Construct 3» встроенная игровая логика называется поведением (*Behaviors*). Используя окно свойств, можно добавить поведение на любой игровой объект. Кликаем по «Behaviors» и нажимаем «Add New Behavior» (рис. 19–20).

Перед нами открывается окно, в котором мы видим список заготовленных поведений. Этот список не является полным, дополнительные поведения можно устанавливать, используя разные плагины (рис. 21).

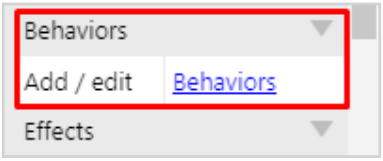

Рисунок 19

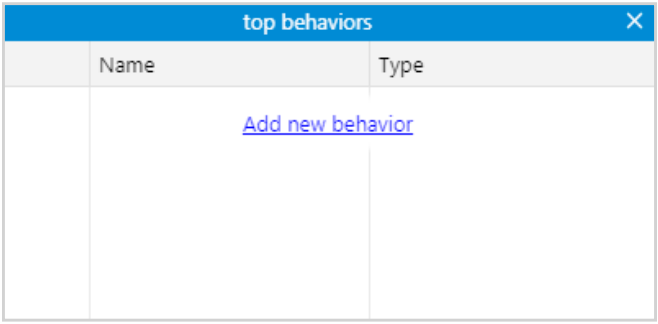

Рисунок 20

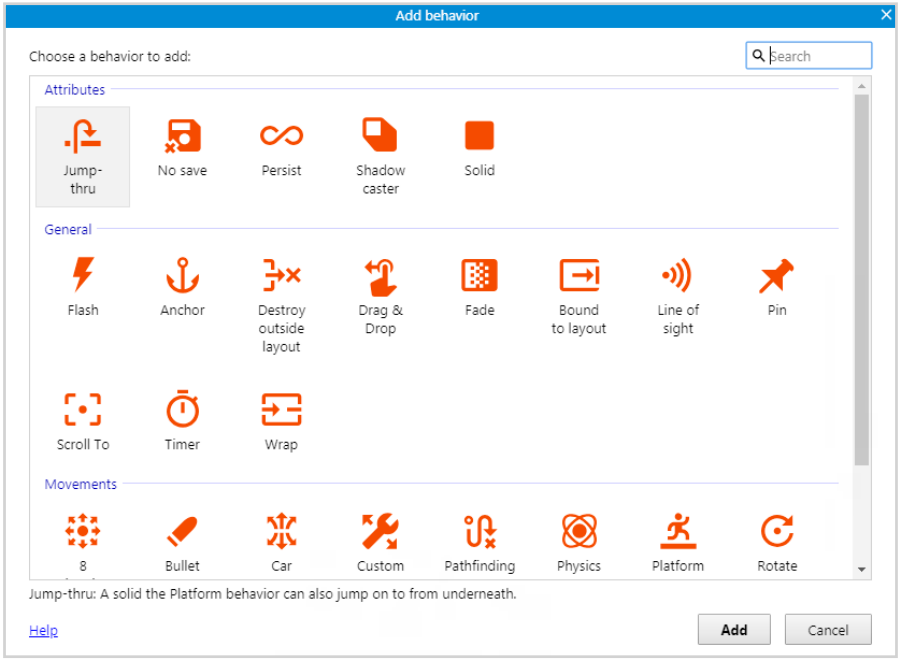

Рисунок 21

Теперь для нашей земли необходимо добавить поведение «Solid» (Solid). Это сделает элемент «твердым», что позволит персонажу ходить по нему.

Добавим поведение персонажу (рис. 22).

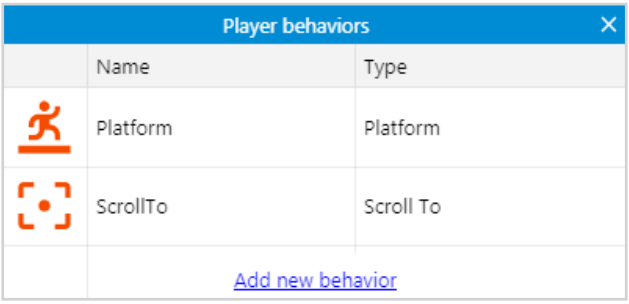

Рисунок 22

Выбираем персонажа и переходим к окну свойств. Здесь можно менять параметры поведения персонажа (рис. 23).

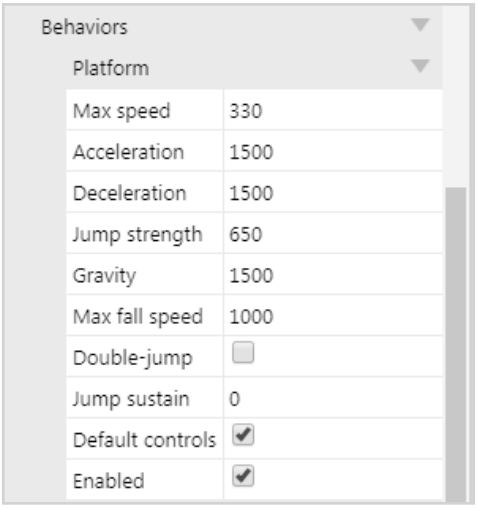

Рисунок 23

Чтобы добавить фон игре, перетягиваем из проводника изображение, кликаем по нему правой кнопкой мыши и выбираем «Z Order => Send to bottom of layer» (рис. 24).

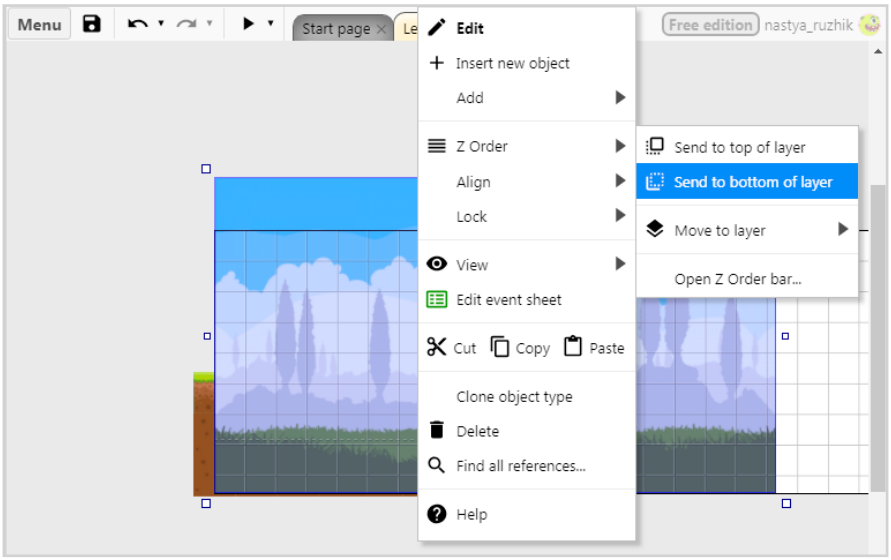

Рисунок 24

Теперь фон не перекрывает игровые элементы (рис. 25).

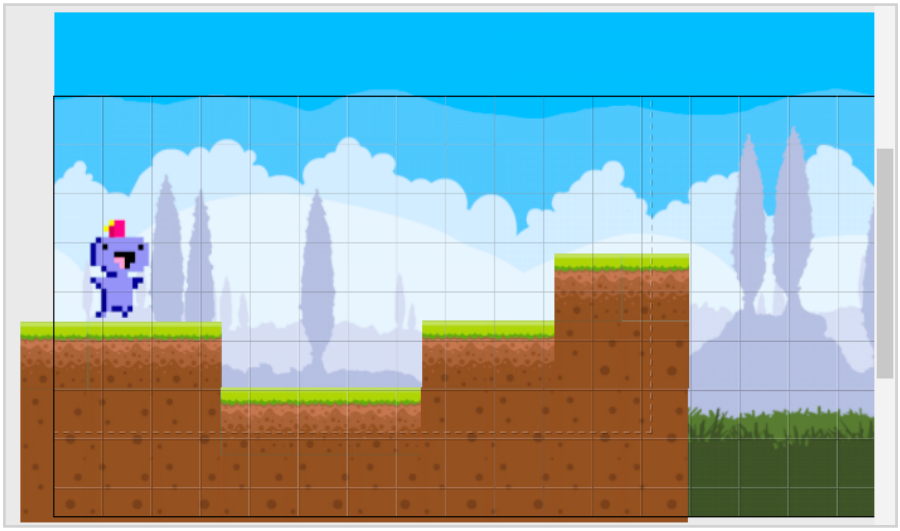

Рисунок 25

Добавим в сцену шипы и монеты (рис. 26).

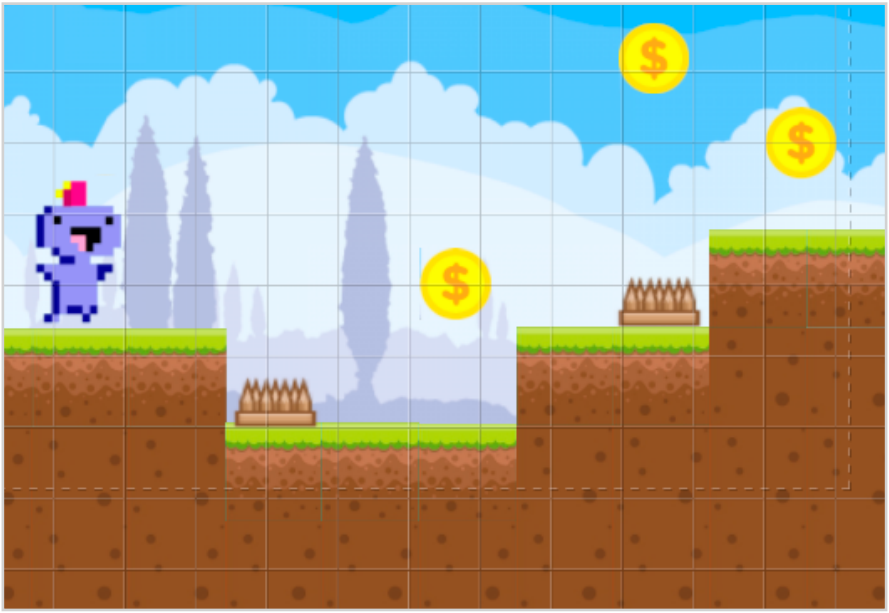

Рисунок 26

При воспроизведении персонаж может немного застревать в текстурах. Это происходить из-за физической границы объекта. Давайте попробуем это исправить. Для этого необходимо дважды кликнуть по персонажу. Увидим следующее окно (рис. 27).

Выбираем инструмент для работы с физической границей объекта — «Edit the collision Polygon» ( ).

Синяя область и есть физической границей нашего объекта. Ее можно изменять, двигая красные вершины и добавляя новые точки (достаточно просто кликнуть по одной из точек для добавления новой по соседству) (рис. 28).

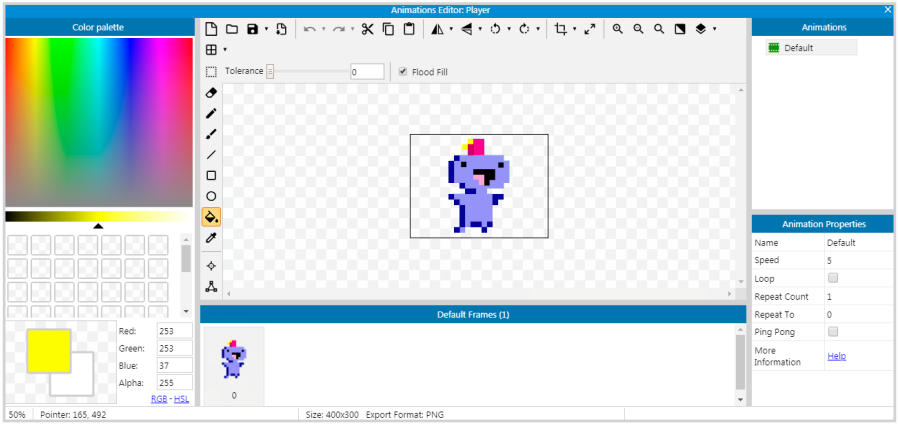

Рисунок 27

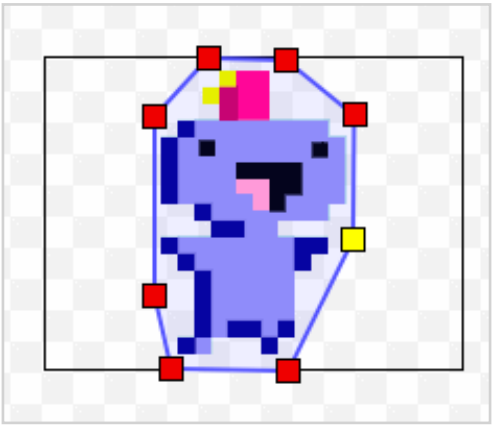

Рисунок 28

Будьте внимательны. Добавление большого количества точек может плохо сказаться на работе вашей игры. Оптимальное количество — не больше 8.

Теперь давайте сделаем квадратную границу нашему персонажу. На любой и точек нажмем правой кнопкой мыши и выберем «Set to bounding box» (рис. 29–30).

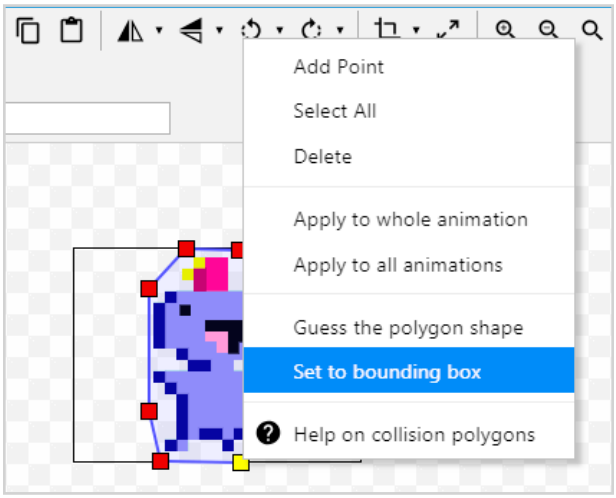

Рисунок 29

Рисунок 30

Давайте добавим основные цепочки событий. Переходим на вкладку «Level1_Events» и выбираем «Add event» (рис. 31).

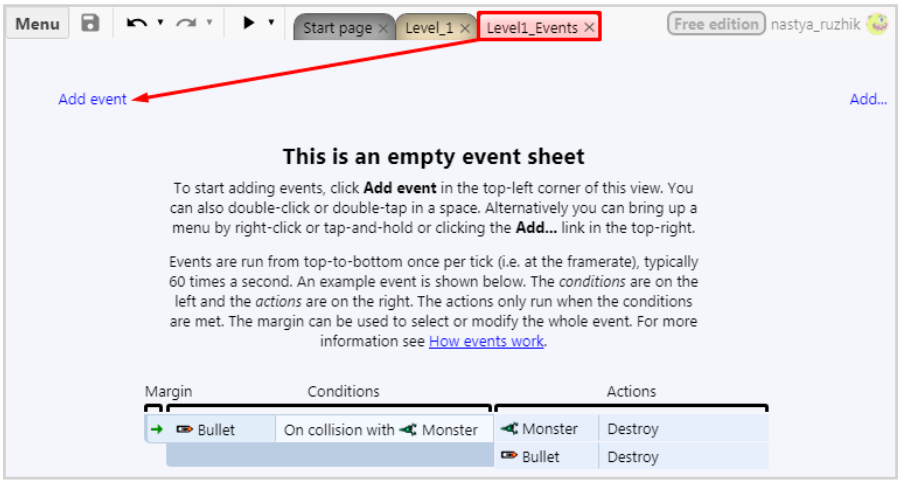

Рисунок 31

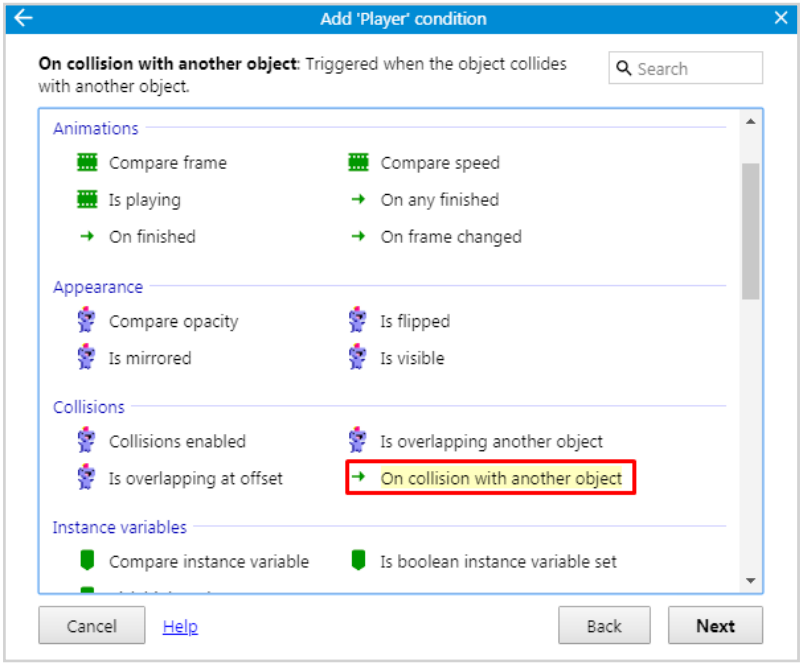

Рисунок 32

Теперь нам нужно выбрать событие касания с другим объектом. В «Construct 3» эта команда называется «On collision with another object» (рис. 32).

Далее нажимаем «Click to choose» и выбираем объект, при соприкосновении с которым будет происходить определенное действие. Это монетка (рис. 33–34).

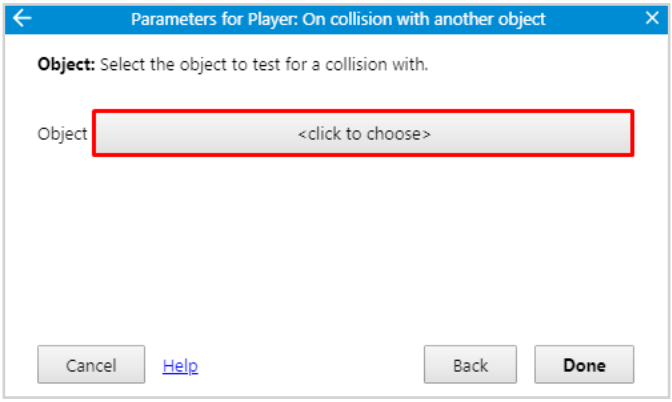

Рисунок 33

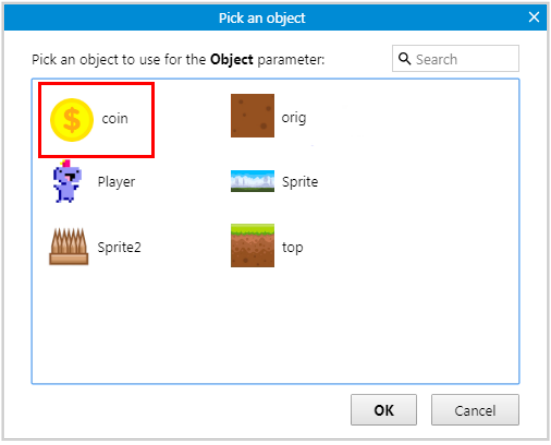

Рисунок 34

Добавляем событие нажав «Add action» (рис. 35).

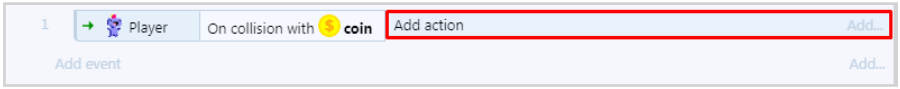

Рисунок 35

Из списка событий выбираем «Destroy» (разрушение/исчезновение) (рис.36).

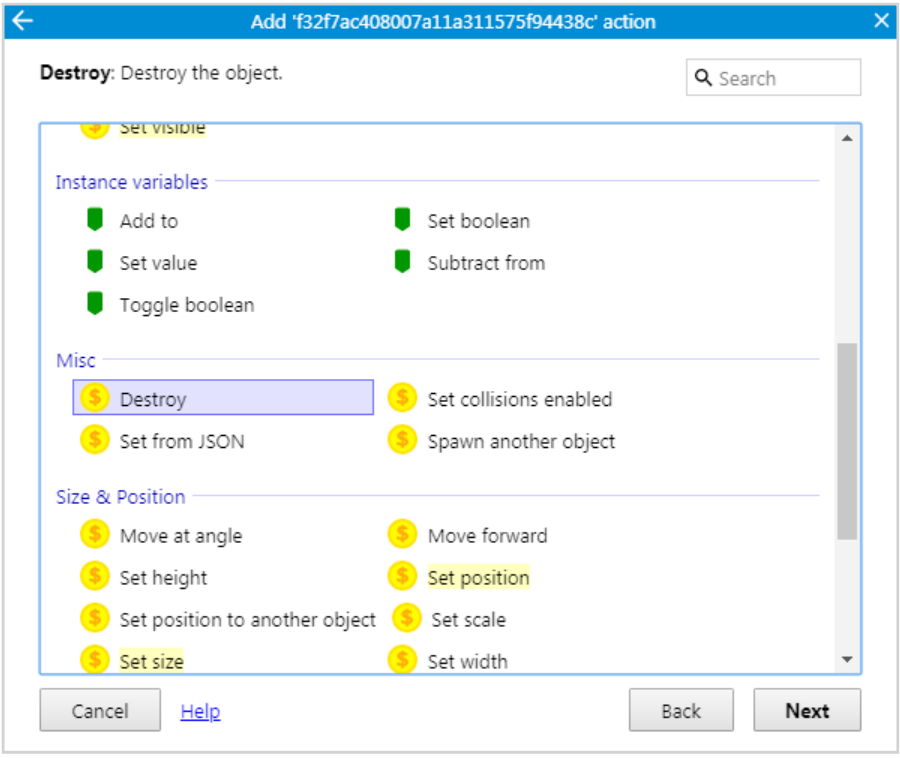

Рисунок 36

Давайте запустим игру. Следует проверить, собираются ли монетки.

Теперь попробуйте самостоятельно написать аналогичную команду, чтобы игра перезапускалась при соприкосновении персонажа с шипами (рис. 37–38).

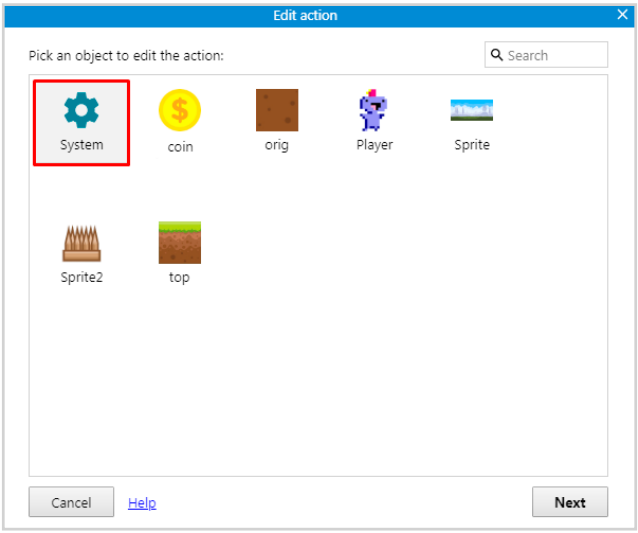

Рисунок 37

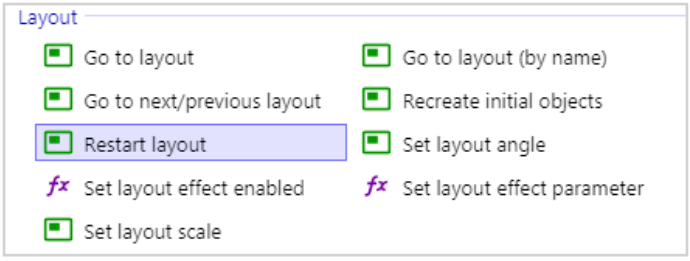

Рисунок 38

Теперь добавим еще один элемент и пропишем для него событие, чтобы игра была интересней. Давайте сделаем так, чтобы, выпив зелье, персонаж увеличивался. Добавляем объект с зельем (рис. 39).

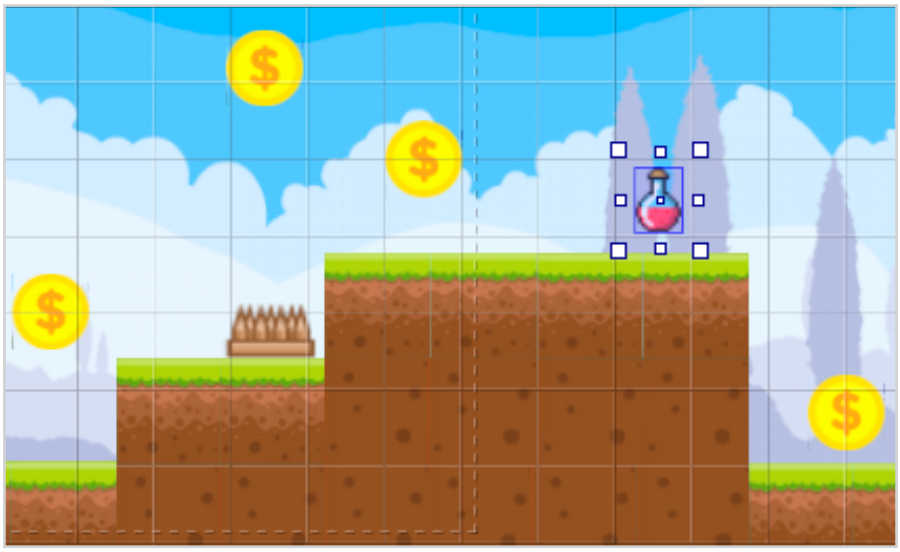

Рисунок 39

Затем добавляем событие и действие во вкладке «Level1_Events» (рис. 40).

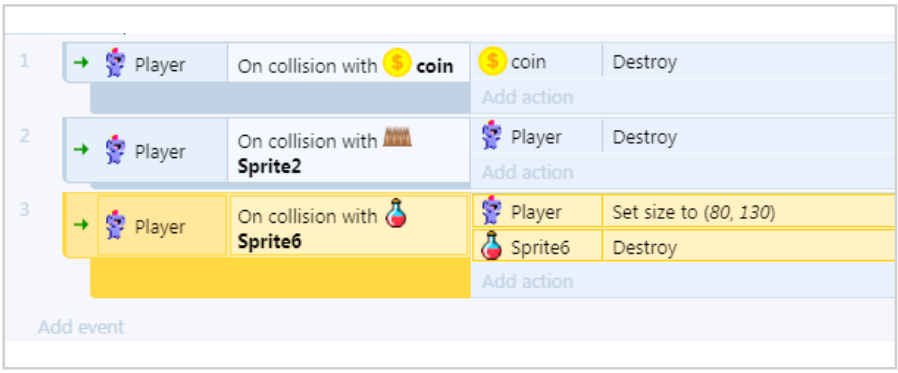

Рисунок 40

Теперь, когда персонаж выпивает зелье, он вырастает (рис. 41–42).

Рисунок 41

Рисунок 42

С событиями и действиями разобрались. Давайте немного обновим окружение персонажа и создадим параллакс эффект для заднего фона. Этот эффект создает иллюзию трехмерного пространства, когда объекты на переднем плане немного двигаются относительно фона.

Для этого создадим новый слой. Кликаем правой кнопкой мыши на панели «Layers» и выбираем «Add layer at top» (рис. 43).

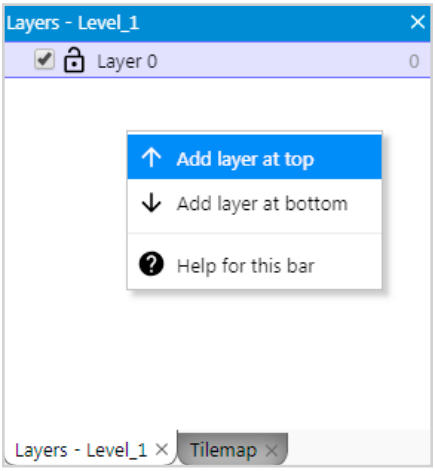

Рисунок 43

На этот слой добавляем элемент с облаками (рис. 44).

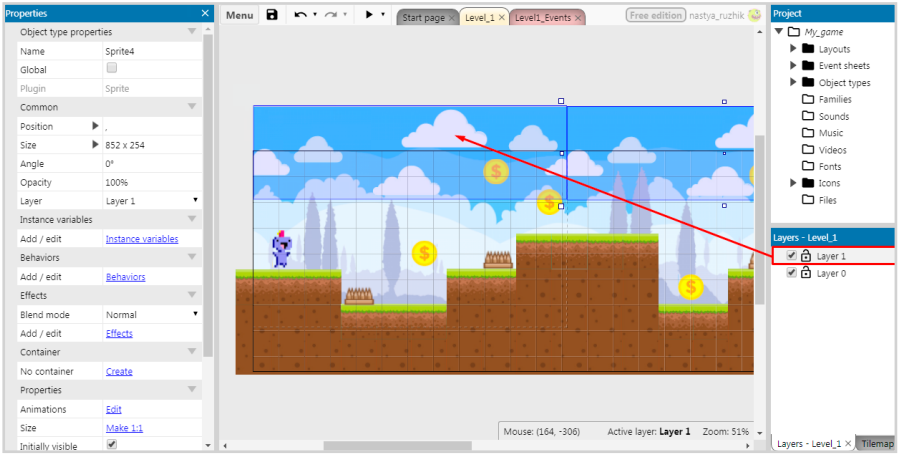

Рисунок 44

Выделяем на панели справа «Layer 1». На панели «Properties» ставим отметку «Transparent» и выставляем значение «Parallax=70×100» (рис. 45).

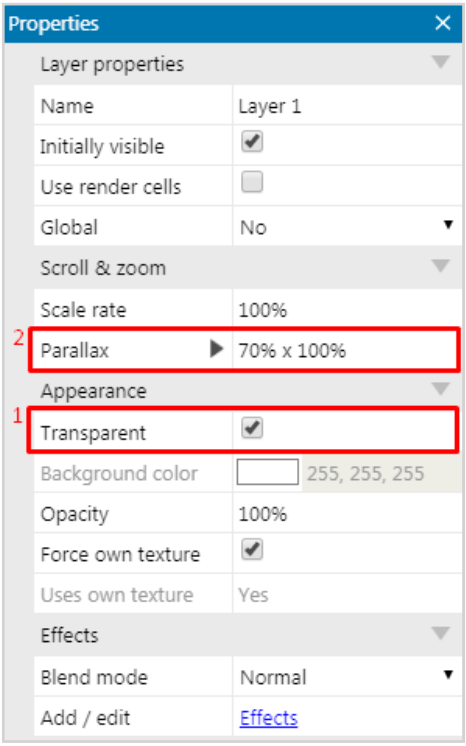

Рисунок 45

Параллакс эффект сработает при воспроизведении игры (рис. 46).

Рассмотрим еще один интересный способ добавить в игровую сцену фон. На этот раз мы воспользуемся объектом «TiledBackground». Кликаем правой кнопкой мыши на рабочей области и выбираем «Insert new object». В появившемся окне находим «TiledBackground» (рис. 47).

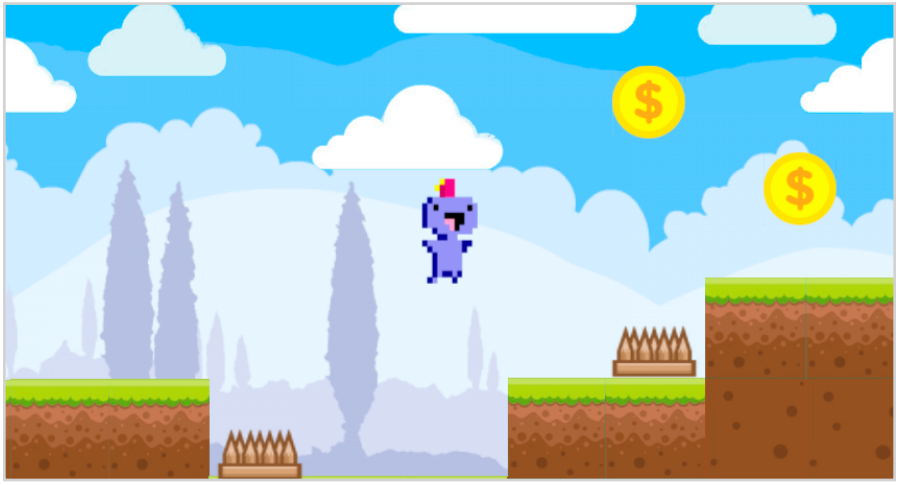

Рисунок 46

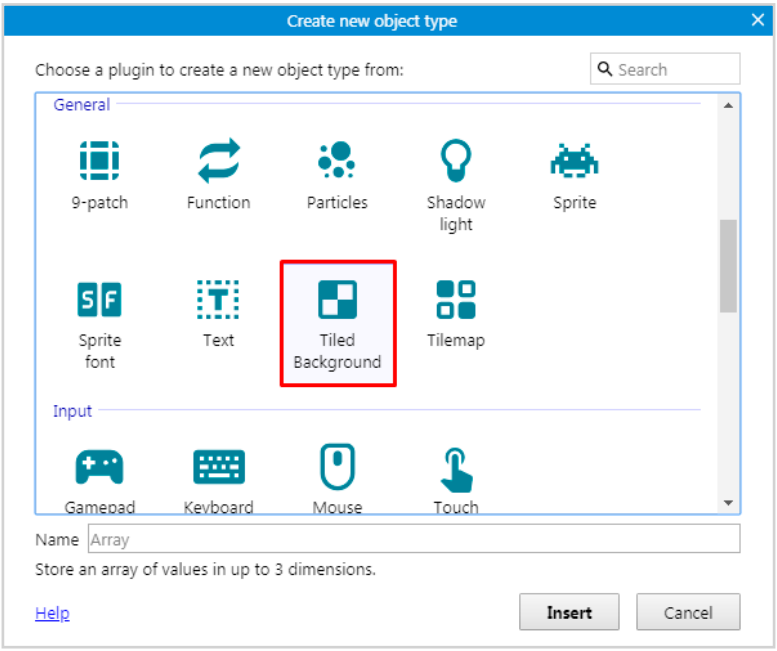

Рисунок 47

Откроется диалоговое окно, в котором можно или самостоятельно нарисовать, или добавить изображение для заднего фона. Кликаем по иконке с папкой, чтобы найти нужный референс (рис. 48).

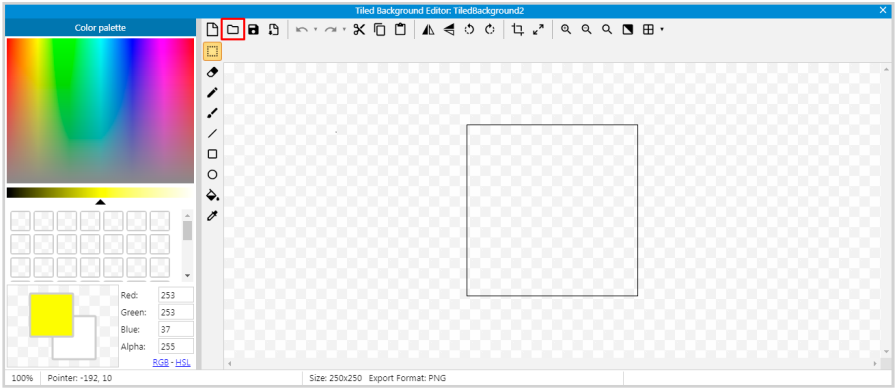

Рисунок 48

В сцене появится добавленная нами плитка (рис. 49).

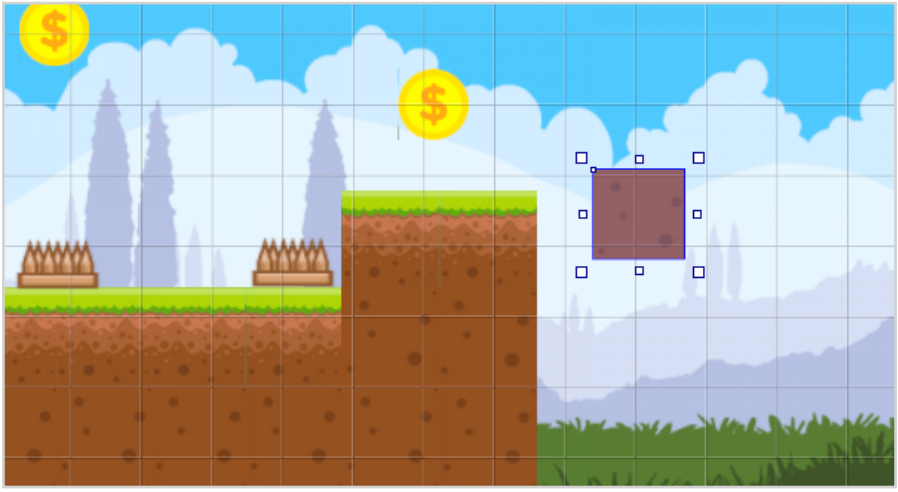

Рисунок 49

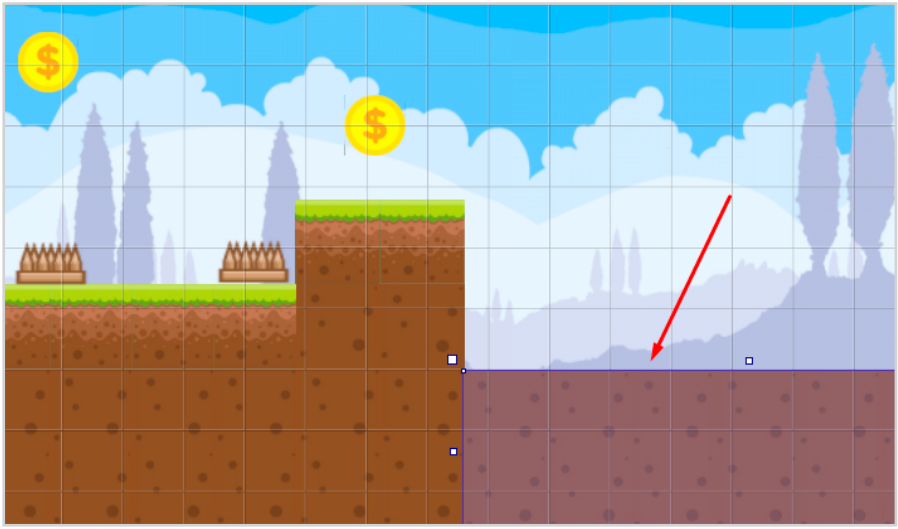

Рисунок 50

Преимущество «TiledBackground» в том, что, когда мы растягиваем объект, орнамент дублируется, а не растягивается. Таким образом, например, можно быстро сделать бэкграунд или платформу для передвижения персонажа (если ей задать поведение «Solid») (рис. 50).

После создания игровой карты, давайте рассмотрим, как можно выгрузить созданную игру. Нажимаем «Menu => Project => Export» (рис. 51).

Выбираем «Web (HTML5)» и нажимаем «Next» (рис. 52).

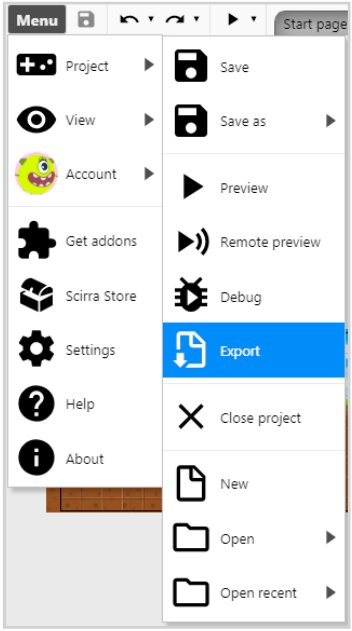

Рисунок 51

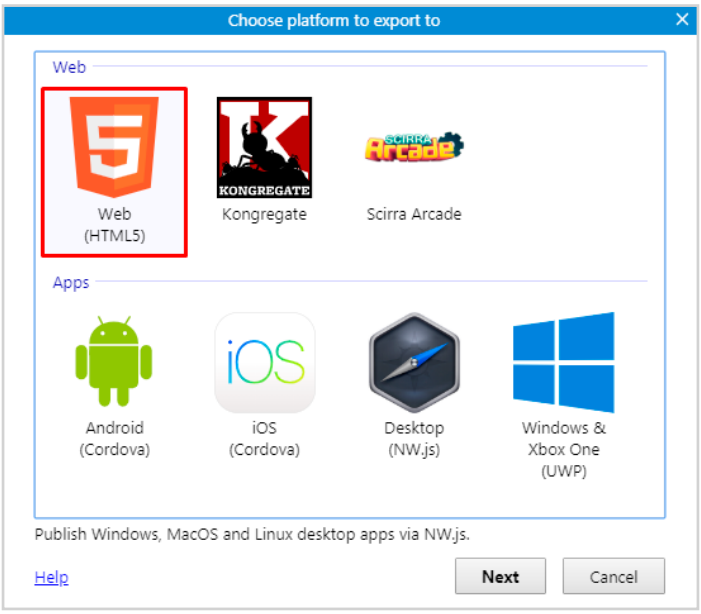

Рисунок 52

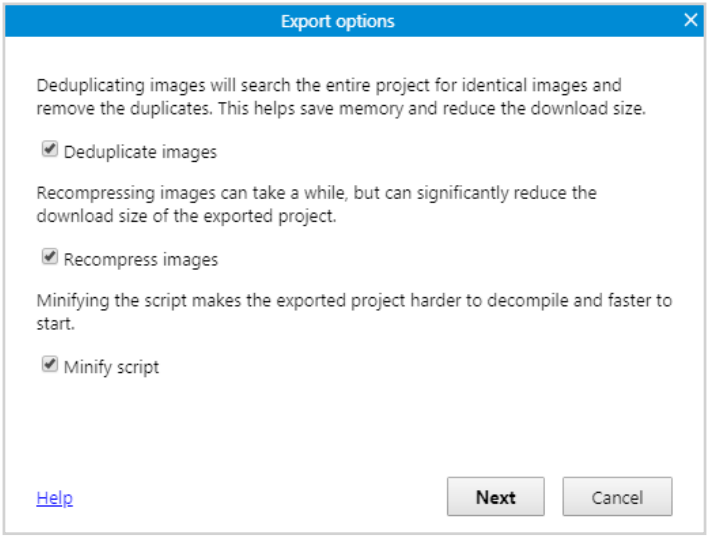

Рисунок 53

В следующем окне отмечаем все чекбоксы и нажимаем «Next» (рис. 53).

После небольшой загрузки, появится окно, в котором сказано, что экспорт завершен. Скачайте архив с файлами по первой ссылке «Download My_game.zip» (рис. 54).

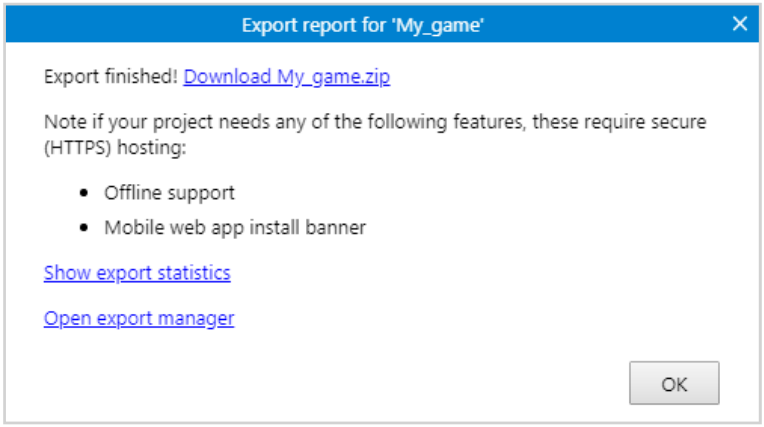

Рисунок 54

Переходим по ссылке и регистрируемся на сайте, где будем публиковать свои игры (рис. 55).

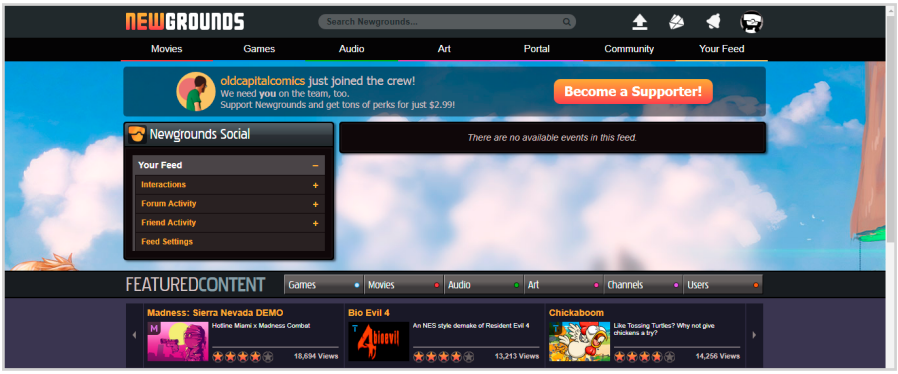

Рисунок 55

Нажимаем на иконку «Upload your creations» и выбираем «Game» (рис. 56).

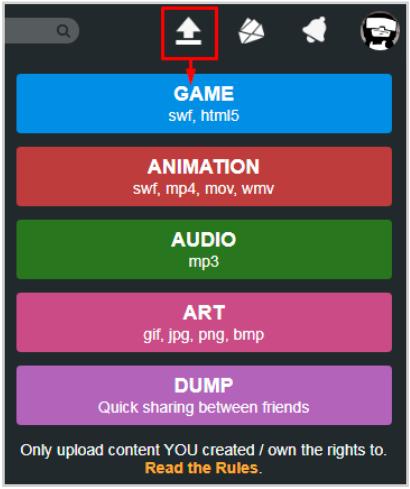

Рисунок 56

В новом окне «New Game Project» указываем название своей игры и нажимаем «Next» (рис. 57).

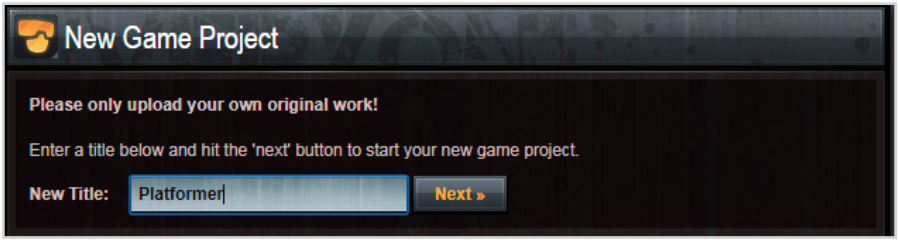

Рисунок 57

В следующем окне необходимо внести детальную информацию о своей игре, добавить скачанный ранее архив, выбрать иконку для игры, жанр и т.д. Внимательно читаем каждую строку и добавляем нужные сведения (рис. 58).

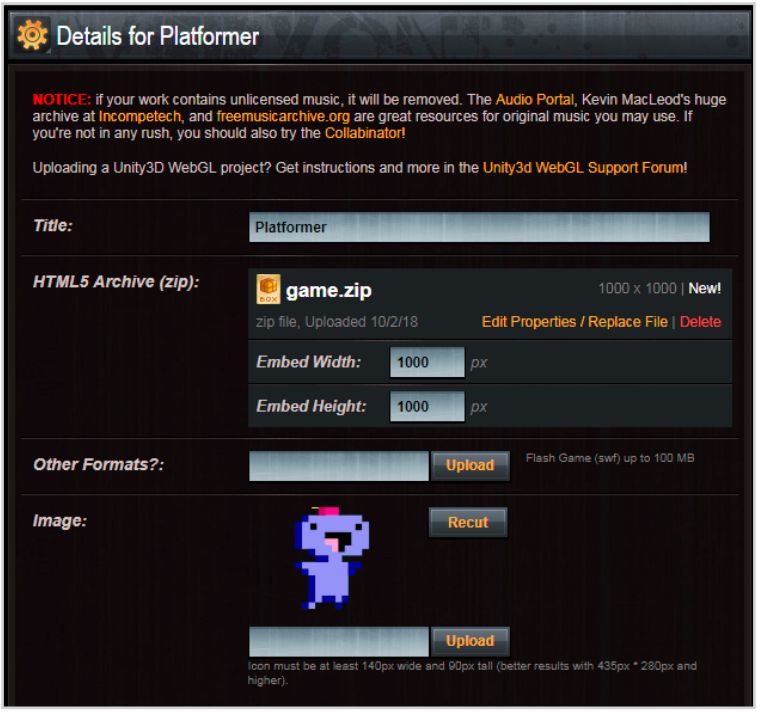

Рисунок 58

В конце этого списка есть две кнопки: «Preview» и «Publish Game». Если вы хотите посмотреть, как выглядит ваша игра до публикации, нажмите первую кнопку. Если же готовы сразу выложить ее на сайт — вторую (рис. 59).

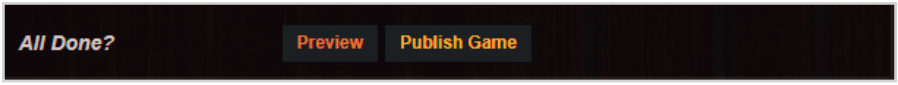

Рисунок 59

Последнее окно — это соглашение пользователя с условиями и правилами публикации игр на сайте. Нажимаем «Submit» (рис. 60).

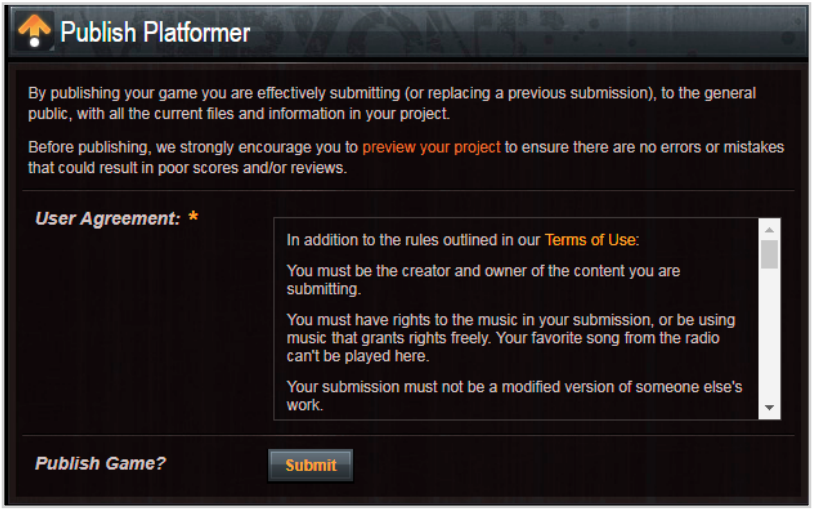

Рисунок 60

Автоматически открывается страница с нашей игрой (рис. 61).

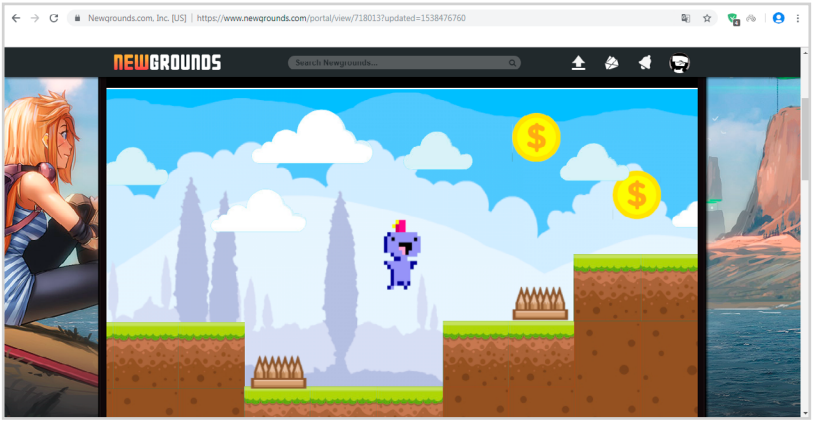

Рисунок 61

Свои проекты вы можете найти на странице «Your Profile» в разделе «Latest Games» (рис. 62).

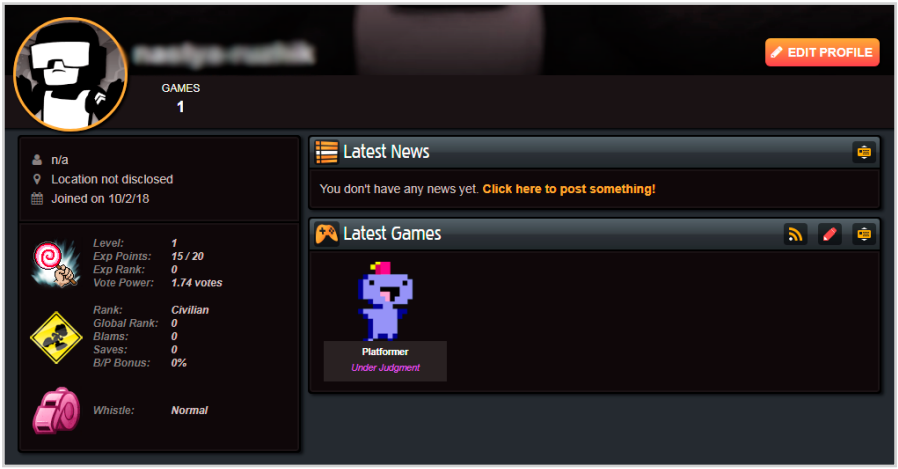
EDIT PROFILE
GAMES
1
n/a
Location not disclosed
Joined on 10/2/18
Level: 1
Exp Points: 15 / 20
Exp Rank: 0
Vote Power: 1.74 votes
Rank: Civilian
Global Rank: 0
Blams: 0
Saves: 0
B/P Bonus: 0%
Whistle: Normal
Latest News
You don't have any news yet. Click here to post something!
Latest Games
Platformer
Under Judgment

Рисунок 62

Урок 2

Добавление переменных и противников

➢ **ПЛАН РАБОТЫ.**

Создание плавающих платформ при помощи поведения sine и jumpthru. Добавление переменных для подсчёта очков (score) и здоровья (health). Добавление текстового поля и спрайтов. Создание папок событий. Добавление аптечки и программирование минимально и максимально пределов здоровья.

➢ **ДОПОЛНИТЕЛЬНО.**

Добавление поведения Flash для паузы между наложениями спрайтов героя и противника с использованием триггера Is overlapping.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Добавление переменных и противников

Для начала немного изменим размер игрового экрана. Для этого переходим на панель «Project properties» и нажимаем «View» (рис. 1).

Рисунок 1

Меняем «Viewport size» на «1500×1000» (рис. 2)

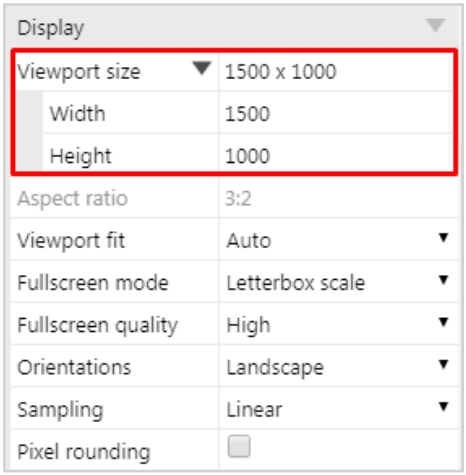

Рисунок 2

Вернемся к первому уровню игры. Добавим пропасть, которую изобразим с помощью воды. Также размещаем платформы (рис. 3).

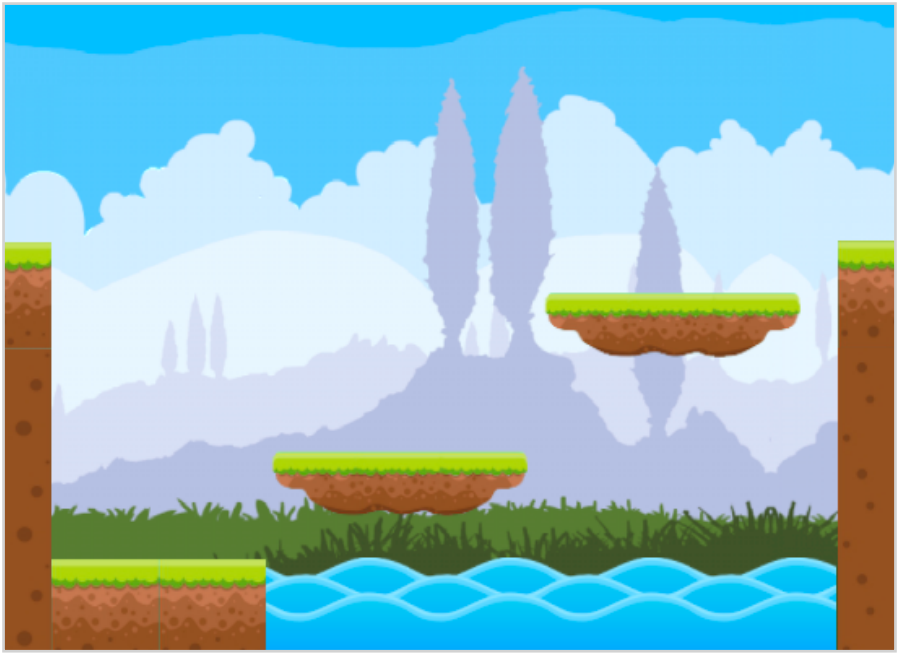

Рисунок 3

Теперь сделаем их плавающими. Кроме поведения «Solid» добавляем еще «Jumpthru». Это значит, что персонаж сможет проходить мимо этого объекта и запрыгивать на него (рис. 4).

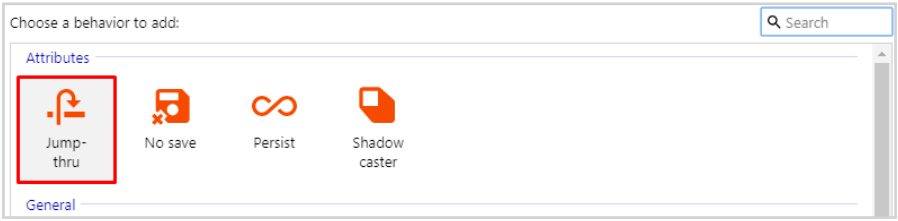

Рисунок 4

Затем добавляем поведение «Sine» (рис. 5).

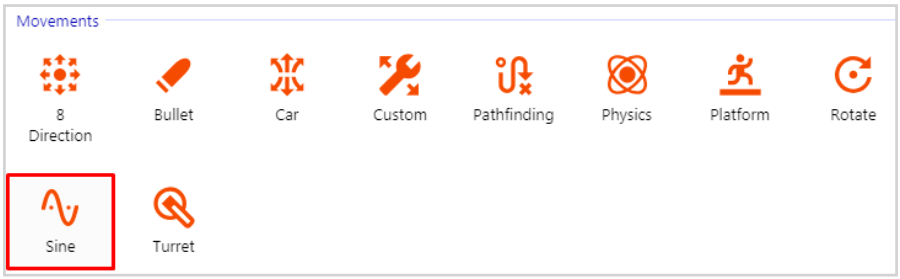

Рисунок 5

На панели справа во вкладке «Behaviors» можно менять параметры поведения, например, направление движения, скорость, интервал и т.д. (рис. 6).

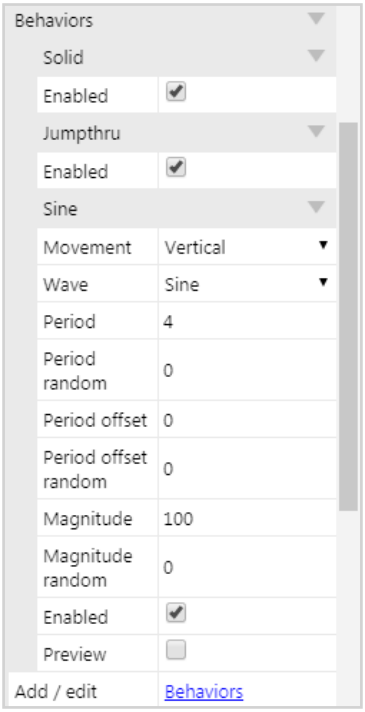

Рисунок 6

Можно задать разные значения платформам, чтобы они двигались с разной скоростью. Это немного усложнит игру (рис. 7).

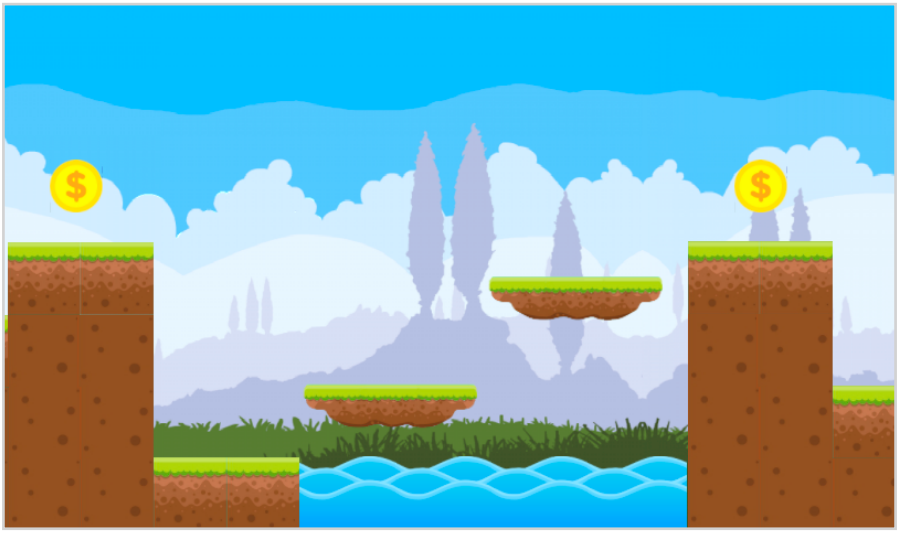

Рисунок 7

Для хранения очков или других данных нам необходимы переменные. Давайте создадим счет очков и жизней персонажа.

Для начала добавляем с помощью «Insert new object» текстовый элемент (Text).

На панели свойств можем изменить наполнение текстового блока, тип, цвет и размер шрифта и т.д. (рис. 8).

В итоге получаем элемент с текстом и размещаем его в верхнем левом углу. Пишем там цифру 0 (рис.9).

Нужно задать текстовому блоку собственную переменную. На панели «Properties» кликаем «Instance variables» (рис. 10).

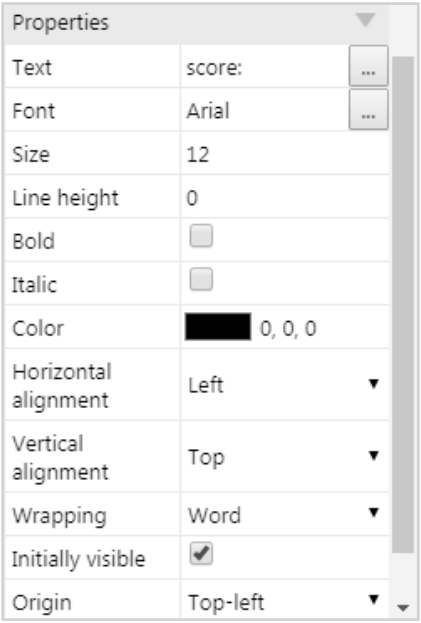

Рисунок 8

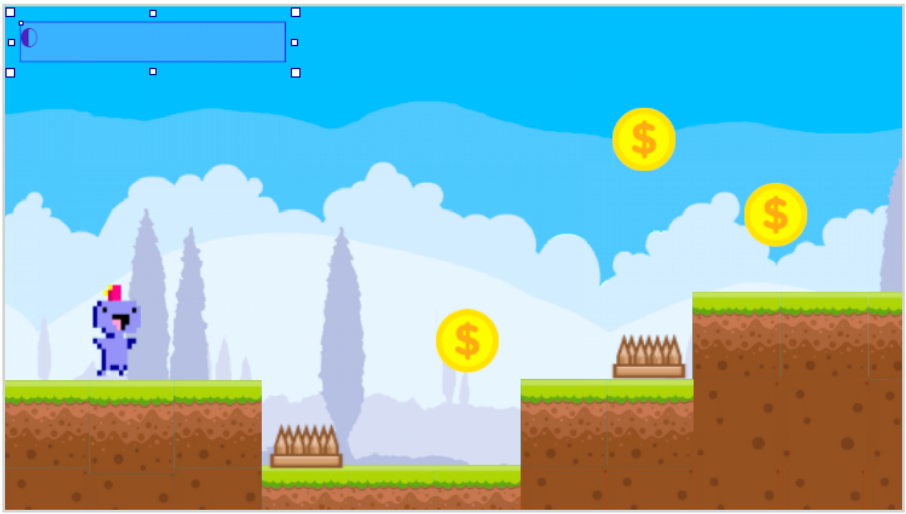

Рисунок 9

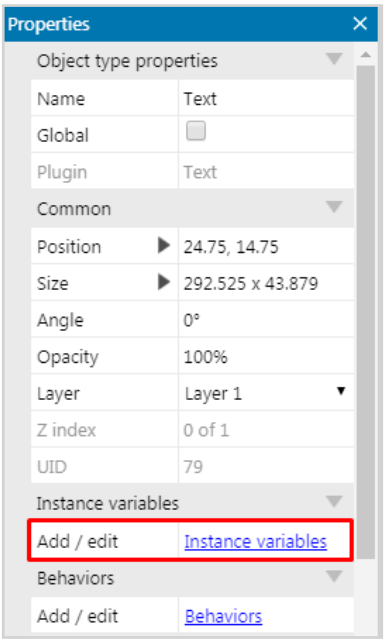

Рисунок 10

Задаем переменной имя, например, тот же «score» и выставляем начальное значение — 0 (рис. 11).

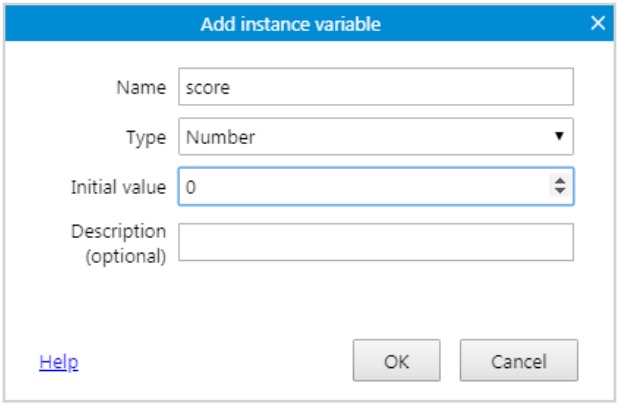

Рисунок 11

Теперь для того, чтобы текст был всегда в фокусе, давайте добавим ему поведение «Anchor». Это поведение привяжет наш объект к углу карты (рис. 12).

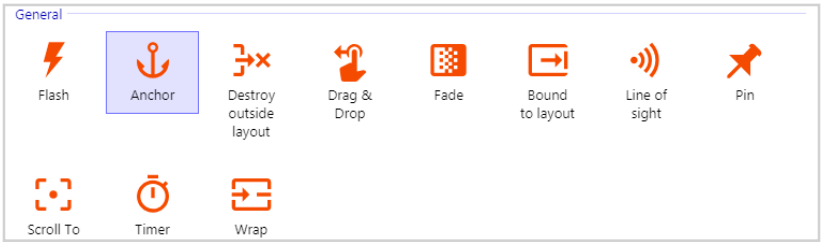

Рисунок 12

Переходим в цепочку событий. У нас уже есть событие, где при соприкосновении персонажа с монетой, второй элемент пропадает. Теперь нужно добавить действие так, чтобы при соприкосновении с монеткой увеличивался счет.

Выбираем «Add action», затем «Text». Из списка действий нужно добавить «Add to» (рис. 13–14).

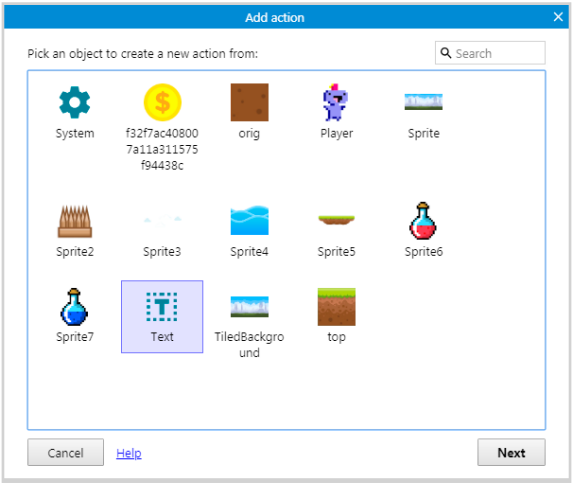

Рисунок 13

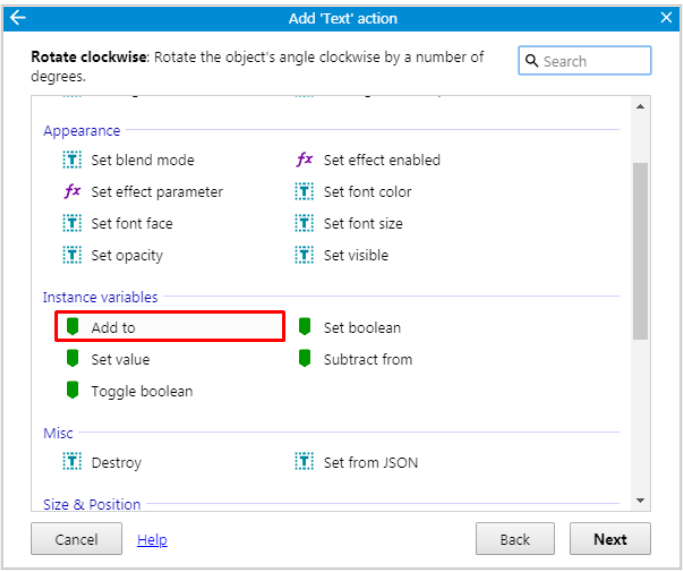

Рисунок 14

В следующем окне нужно установить, сколько очков прибавится при соприкосновении персонажа с монеткой. Оставляем 1 (рис. 15).

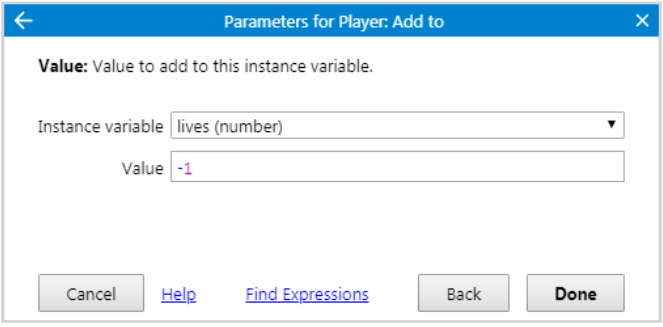

Рисунок 15

Событие мы добавили, но цифры у слова «Score» еще не появились. Для этого нужно добавить еще одно собы-

тие для текста, но в этот раз из списка действий выбрать «Set text» (рис. 16).

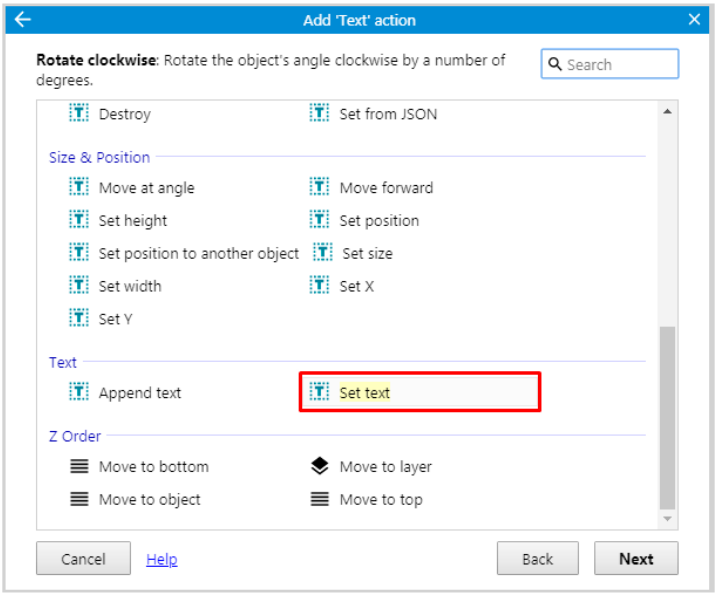

Рисунок 16

Появится следующее окно с текстовым полем. Убираем текст, который выделен красным (рис. 17).

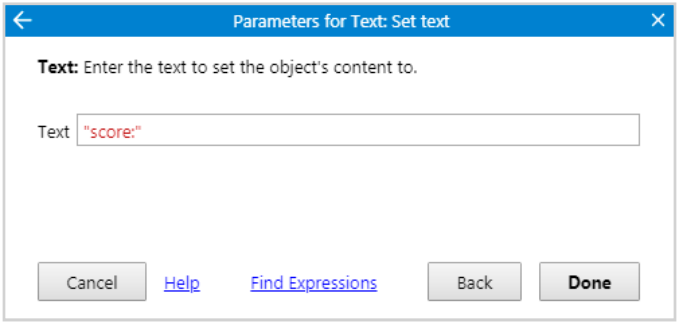

Рисунок 17

Нажимаем внизу окна «Find Expression». После этого появится новое окно «Expression Dictionary», в котором снова выбираем «score» (рис. 18).

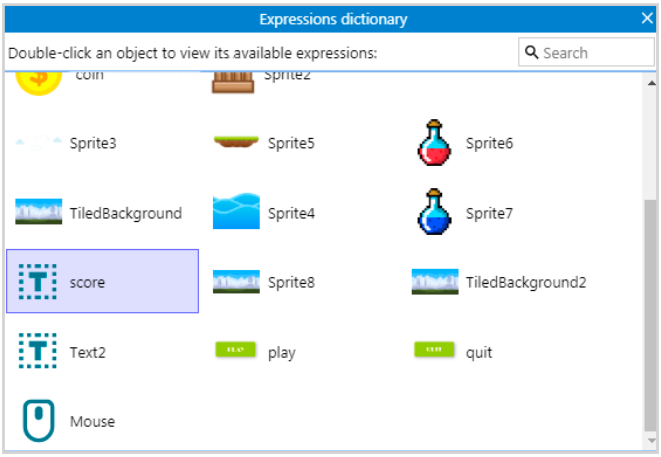

Рисунок 18

Выбираем «score (number) A numeric variable» (рис. 19). В окне с параметрами появляется значение (рис. 20).

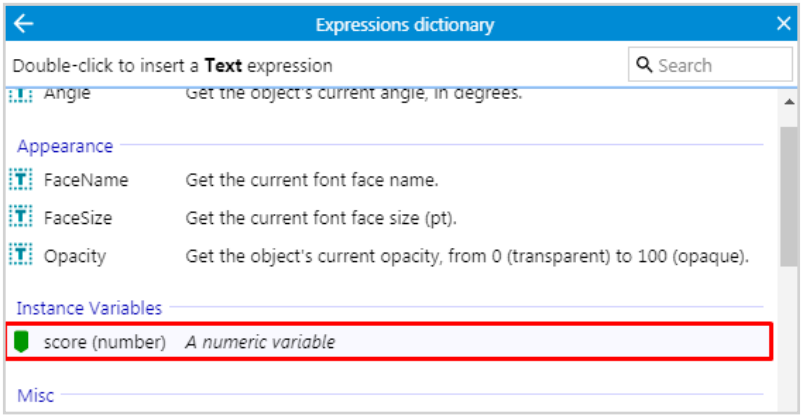

Рисунок 19

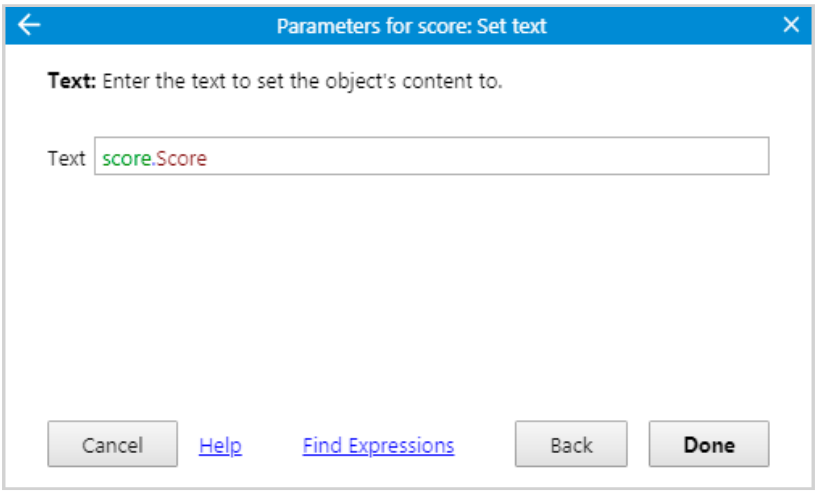

Рисунок 20

Цепочка событий сейчас выглядит следующим образом (рис. 21):

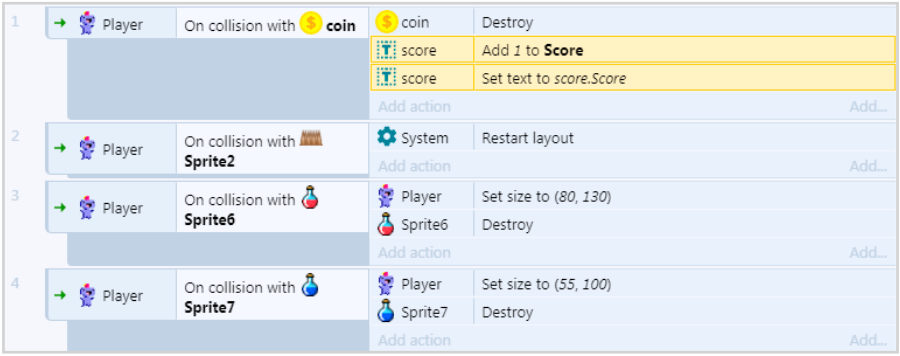

Рисунок 21

Можно также добавить иконку, которая будет отображать накопление монет. Теперь при воспроизведении игры, когда персонаж соприкасается с монеткой, добавляется счет (рис. 22).

Рисунок 22

Осталась одна маленькая деталь. Необходимо прикрепить эти элементы, чтобы они не сдвигались. Для этого добавляем им поведение «Anchor» (рис. 23).

Рисунок 23

Рассмотрим, как добавить персонажу 3 жизни.

Выбираем игрока и добавляем ему «Instance variable». Поскольку это переменная жизней, то выставляем ей значение 3 (рис. 24).

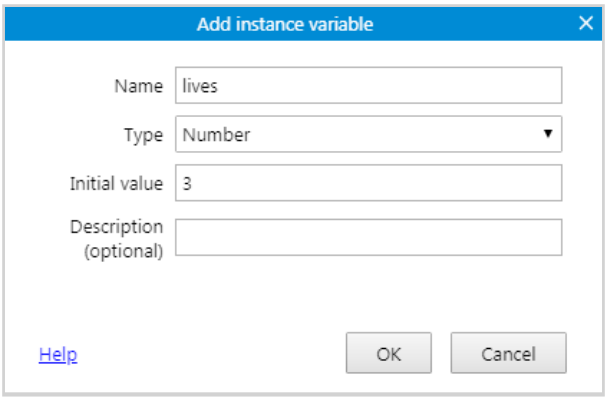

Рисунок 24

Теперь добавляем действие. Добавим его в событие, где игрок попадал на шипы и умирал. Убираем действие «Destroy» (рис. 25).

Рисунок 25

Теперь добавляем «Player => Add to» (рис. 26).

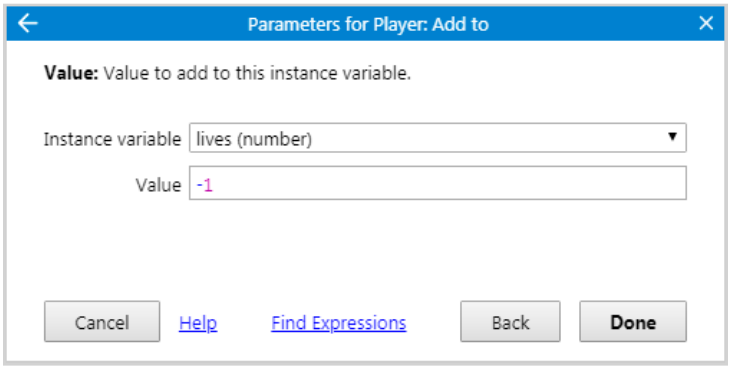

Рисунок 26

Необходимо добавить событие, чтобы, когда жизнь = 0, персонаж умирал и уровень начинался сначала. Для этого выбираем «Player => Compare instance variable» (рис. 27).

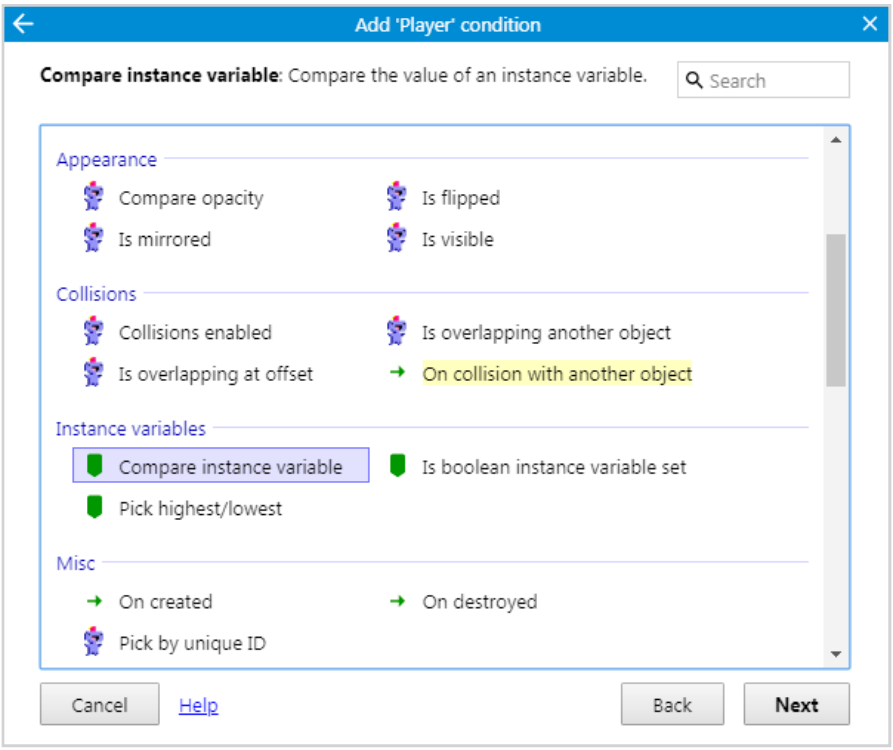

Рисунок 27

В появившемся окне с параметрами выставляем переменную «lives», затем знак ≤ и значение 0 (рис. 28).

Добавляем действие. «Player => Destroy». Затем добавляем «System => Wait» (рис. 29).

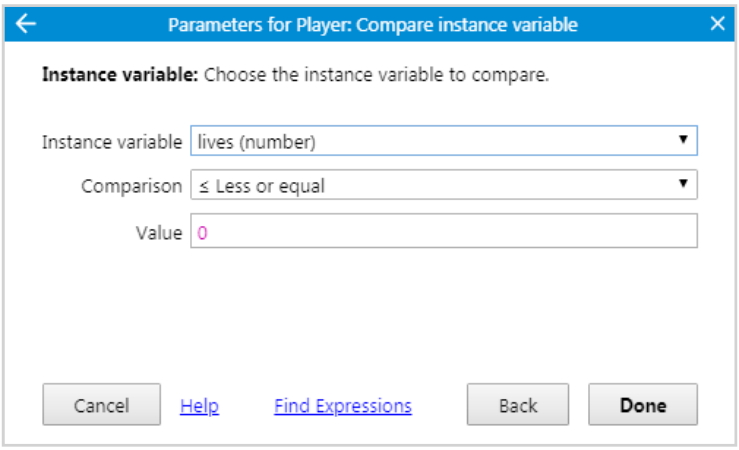

Рисунок 28

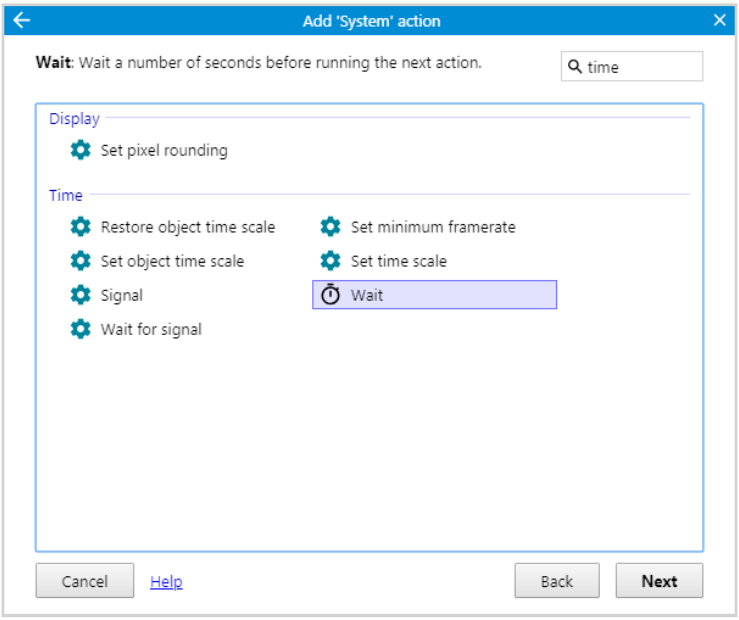

Рисунок 29

Задаем параметр ожидания 1 с (рис. 30).

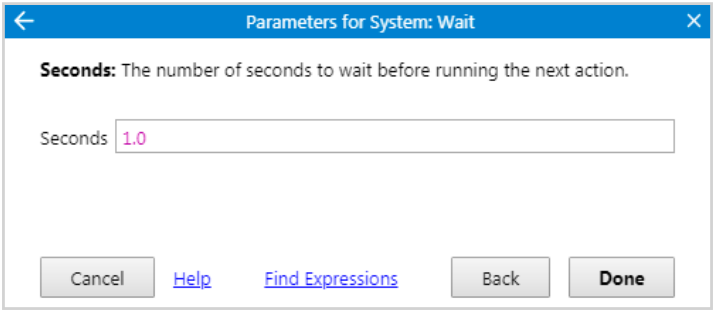

Рисунок 30

Добавим еще одно действие «System => Restart layout».

Теперь, если персонаж трижды попадает на шипы, уровень перезапускается.

Давайте рассмотрим, как сделать так, чтобы игрок тоже видел, сколько жизней у него осталось.

С помощью «TiledBackground» добавляем три сердечка. «Длинна объекта = 120px» (рис. 31).

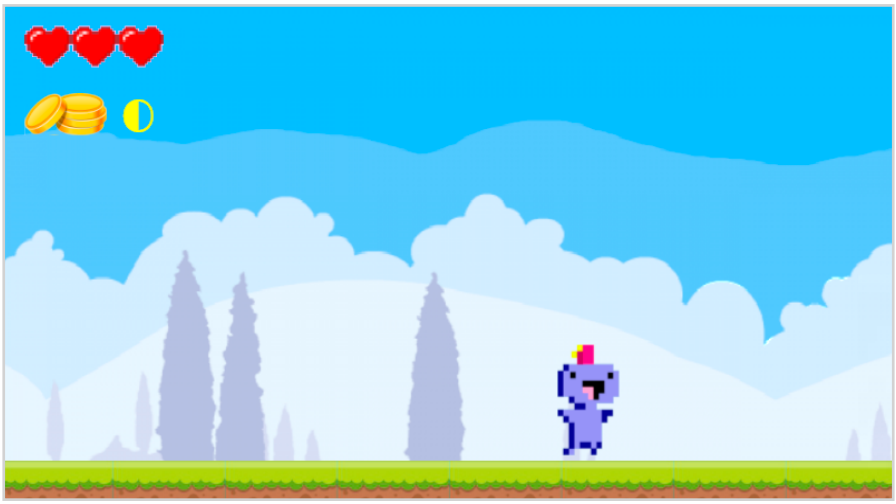

Рисунок 31

Выбираем событие, где персонаж соприкасается с шипами (рис. 32).

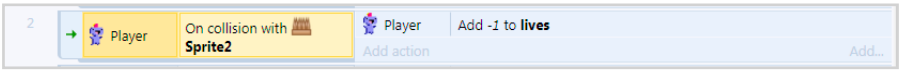

Рисунок 32

Добавим сюда действия для жизней «Lives => Set width» (рис. 33).

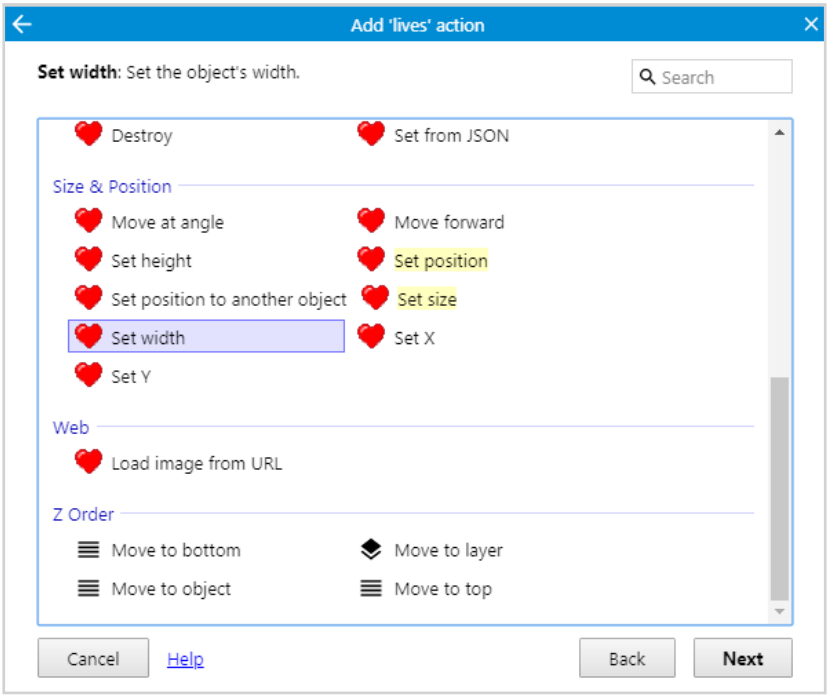

Рисунок 33

В появившемся окне нажимаем «Find Expressions». Снова выбираем объект «Lives», а затем «Get the object's width in pixels» (рис. 34).

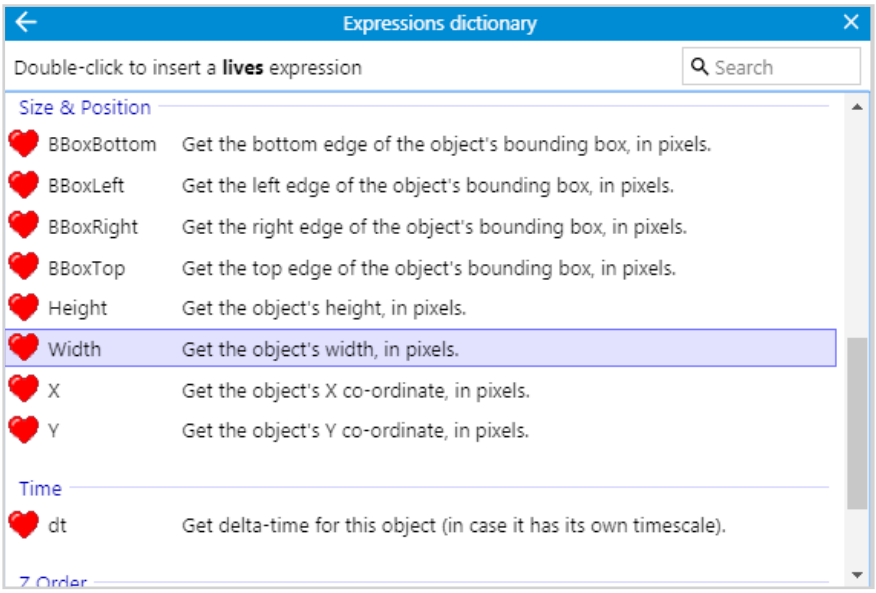

Рисунок 34

К появившемуся параметру добавляем –40, то есть, треть всей длинны объекта (рис. 35).

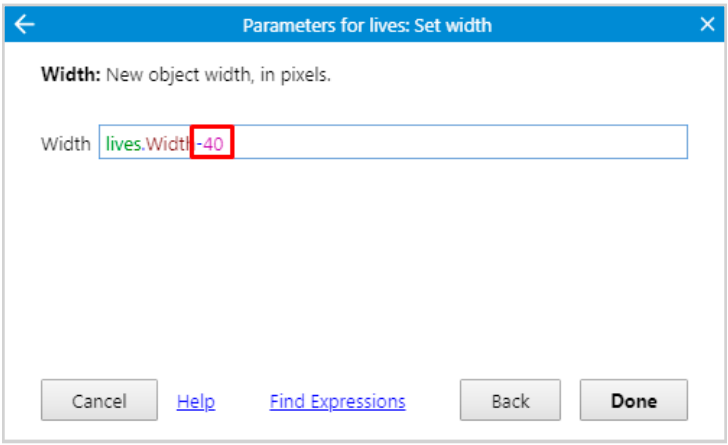

Рисунок 35

Сохраняем результат. Теперь при попадании на шипы, будет пропадать по одному сердечку (рис. 36).

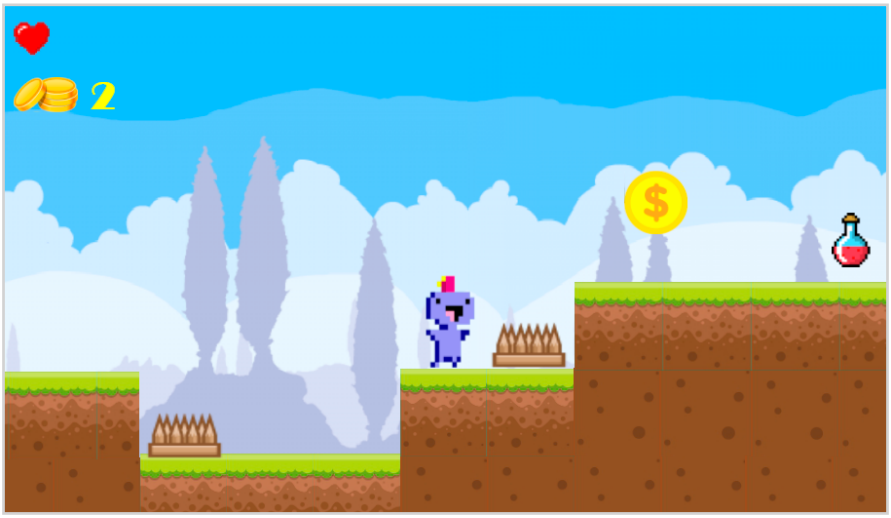

Рисунок 36

Но давайте пойдем и от обратного. Дадим возможность игроку подобрать сердечко и прибавить жизнь.

Создаем новый объект «Insert new object => Sprite» (рис. 37).

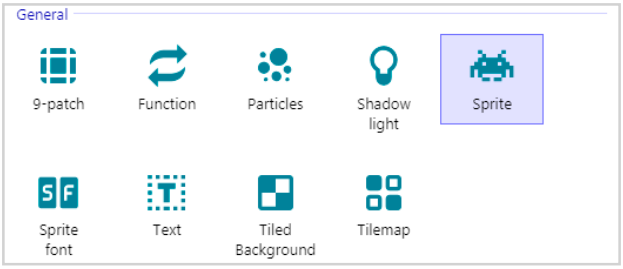

Рисунок 37

Размещаем элемент в сцене (рис. 38).

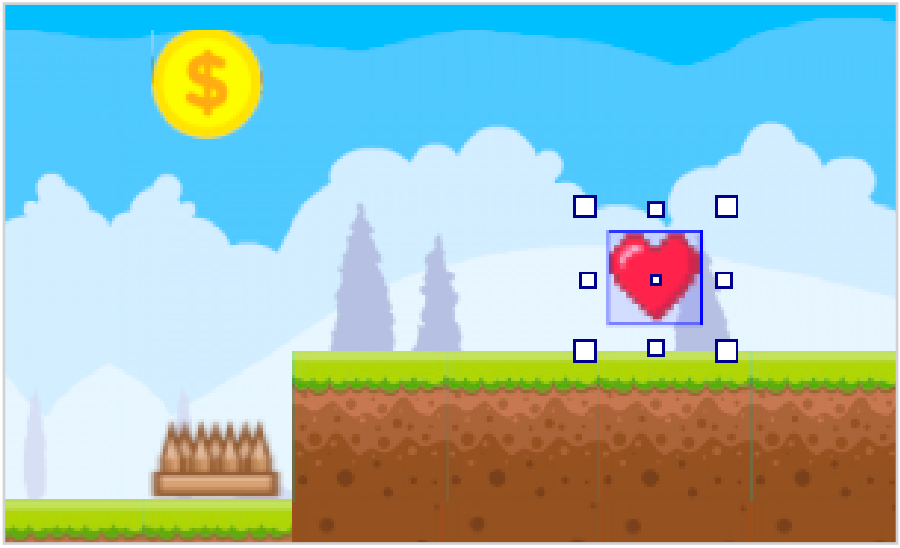

Рисунок 38

Теперь нужно добавить событие, чтобы при соприкосновении персонажа с сердечками, у первого добавлялись жизни. «Add event => Player => On collision with => heart» (рис. 39).

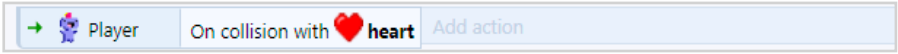

Рисунок 39

«Add action => Player => Add to». Добавляем одну переменную (рис. 40).

Добавим еще одно действие. «Lives => Set width» (рис. 41).

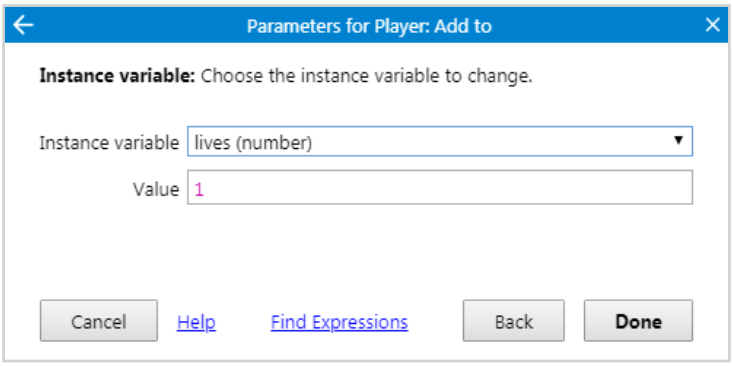

Рисунок 40

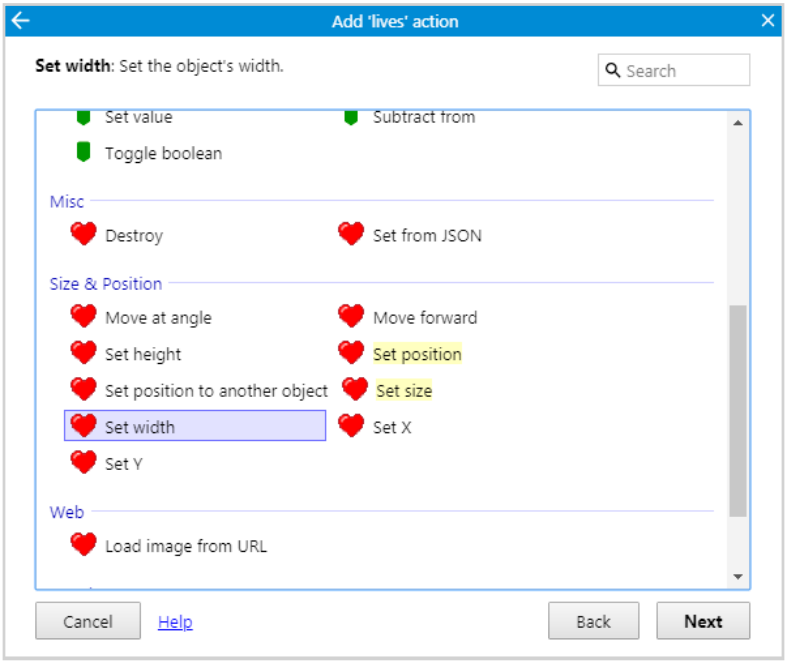

Рисунок 41

В появившемся окне нажимаем «Find expression» (рис. 42).

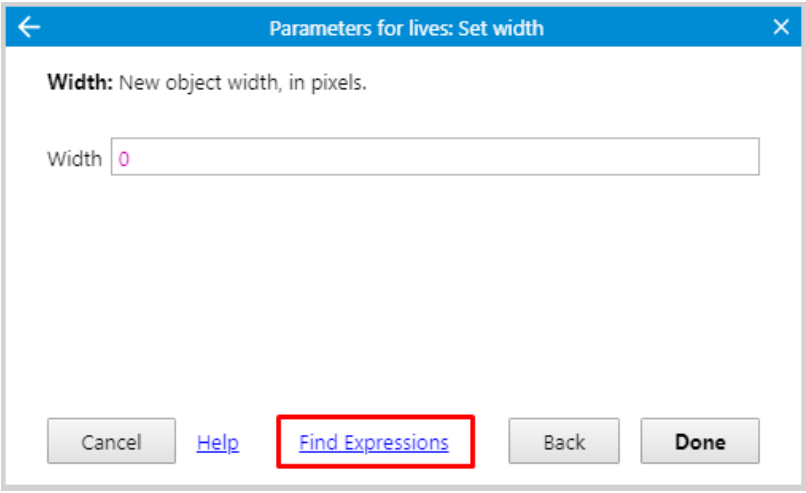

Рисунок 42

Из списка снова выбираем «lives», а затем «Get the object's width, in pixels» (рис. 43–44).

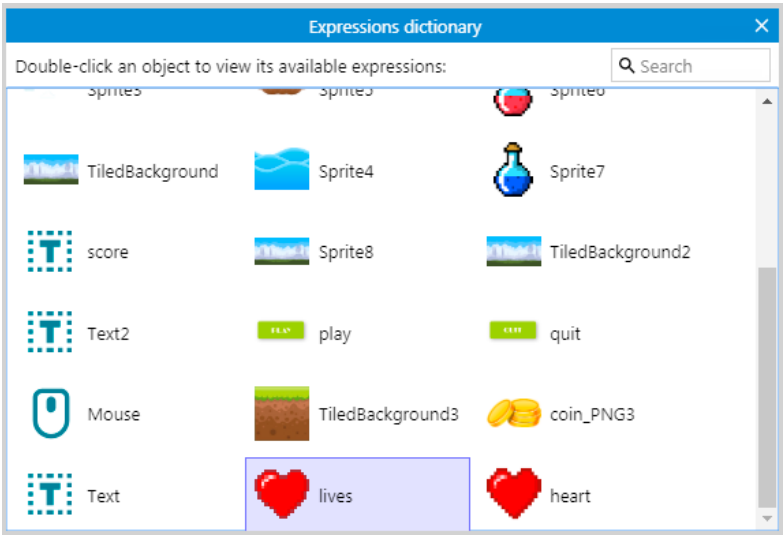

Рисунок 43

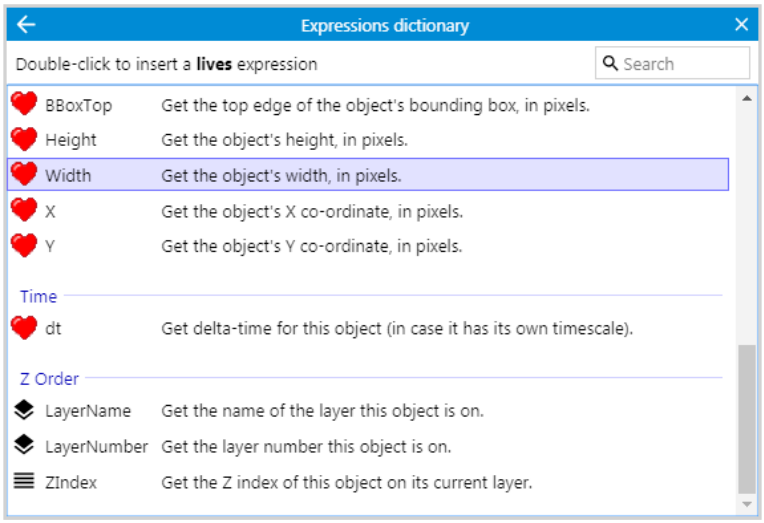

Рисунок 44

В появившемся окне добавляем +40. По сути, мы создаем действие обратное к потере жизни персонажа (рис. 45).

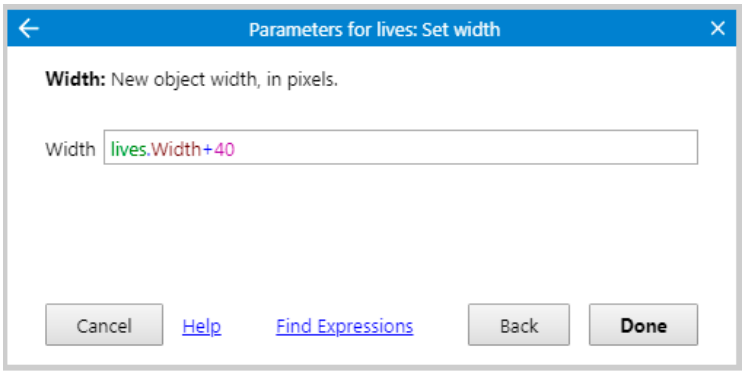

Рисунок 45

Добавим еще одно действие «heart => Destroy», чтобы сердечко добавляло жизнь, а потом исчезало (рис. 46).

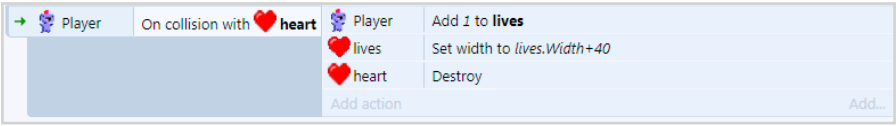

Рисунок 46

Вот и все, теперь персонаж может терять или подбирать жизни.

Теперь нужно ограничить возможность игрока подбирать жизни, если у него есть максимальное их количество.

Для этого добавляем событие «Add event => Lives => Compare width». Выставляем максимальное значение длинны — «120 px» (рис. 47).

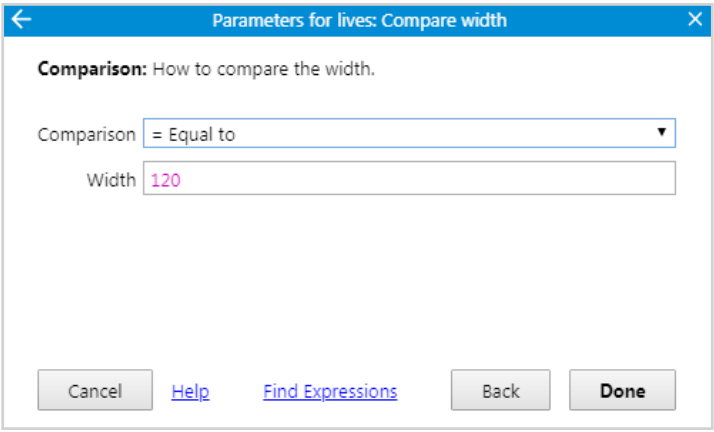

Рисунок 47

Добавляем действие для объекта «Heart => Set collisions enable => Disable» (рис. 48).

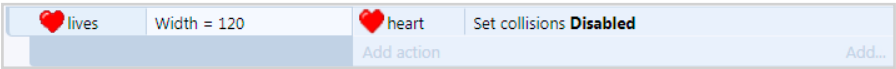

Рисунок 48

После этого нужно добавить событие, чтобы в случае, когда длинна жизней меньше 120, игрок смог подбирать жизнь. Копируем предыдущий ивент. Затем меняем «Width=120» на «Width<120». Затем меняем действие с «Disable» на «Enable» (рис. 49).

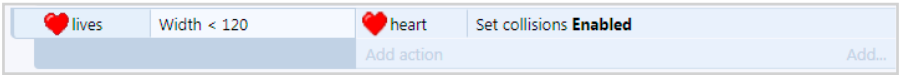

Рисунок 49

В «Construct 3» также есть возможность добавлять другие папки событий. Для этого на панели «Project» справа кликаем правой кнопкой мыши по папке «Event Sheets» и выбираем «Add event sheet» (рис. 50).

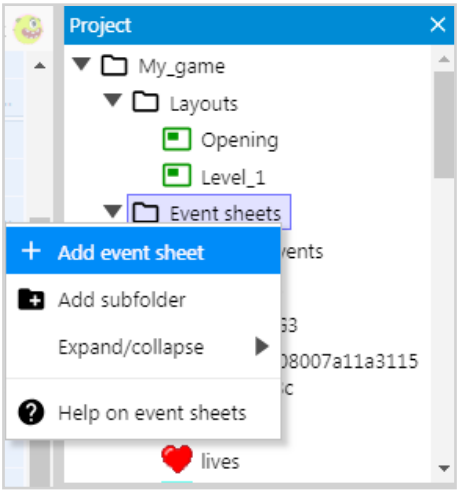

Рисунок 50

Эта функция крайне полезна, когда мы будем создавать другие уровни («Layouts»). Тогда для каждого уровня можно будет прописывать отдельную цепочку событий.

Урок 3

Создание сложных патрулирующих противников и препятствий

➢ **ПЛАН РАБОТЫ.**

Программирование патрулирующих противников (поведение sine) и нанесения урона главному герою. Добавление преследующих врагов. Использование эффекта Camera Shake. Дополнительное задания: Создание эффекта заморозки и возгорания персонажа при наложении на специальные объекты.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Создание сложных патрулируюших противников и препятствий

Итак, у нас уже есть базовые элементы игры — платформа, персонаж, преграды. Мы научились работать со слоями и добавлять поведения объектам, добавить и программировать счет монет и жизней.

Теперь можно совершенствовать некоторые элементы игры, чтобы она была более увлекательной, а уровень усложнялся с каждым этапом. Что мы можем сделать? Например, добавить патрулирующего врага. Для этого применяем «Insert new object => Sprite». Выбираем или рисуем врага и добавляем его в игровую карту (рис. 1).

Рисунок 1

Добавляем поведение «Platform» и «Sine» (рис. 2).

Рисунок 2

В параметрах поведения меняем настройки передвижения и интервал (рис. 3).

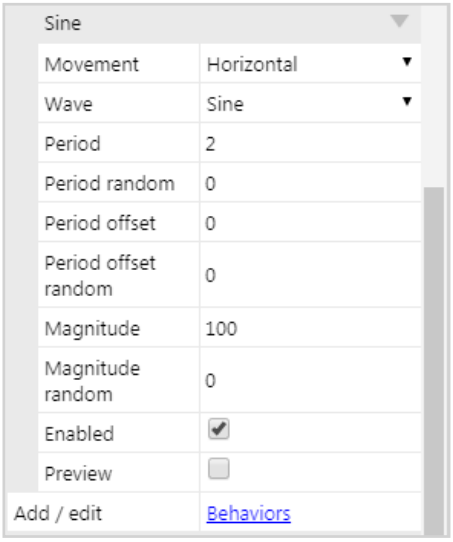

Рисунок 3

Теперь добавим событие, чтобы патрулирующий враг причинял вред игроку. Для этого нужно добавить игроку поведение «Flash» (рис. 4).

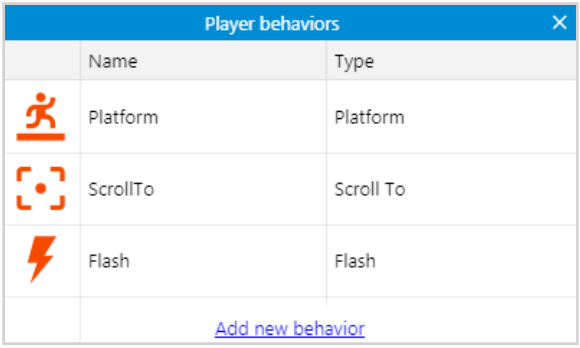

Рисунок 4

Переходим к цепочке событий. «Add event => Patrol => Is overlapping another object» (рис. 5).

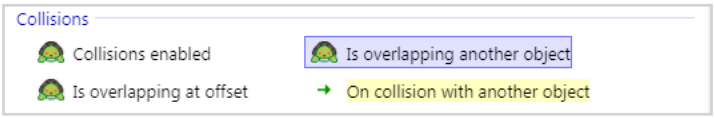

Рисунок 5

Выбираем объект «Player» (рис. 6).

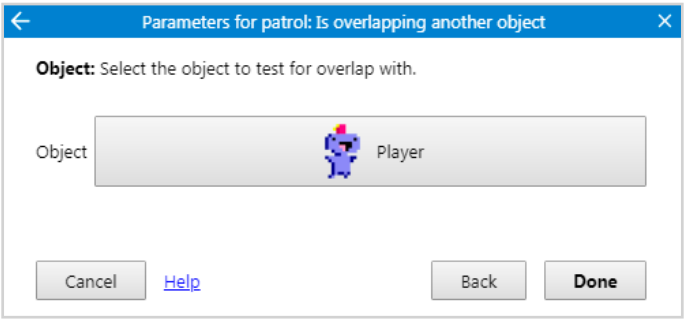

Рисунок 6

Добавляем к только что созданному событию «Add another condition» (рис. 7).

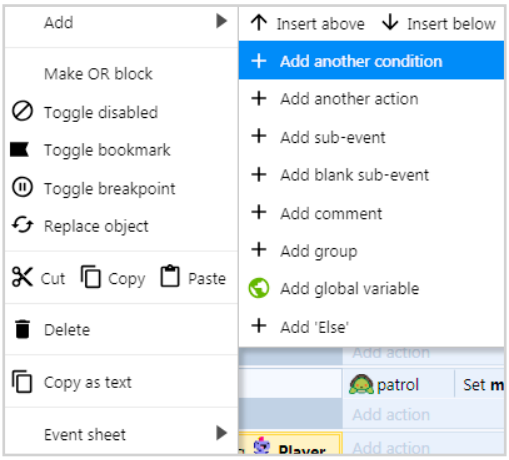

Рисунок 7

Выбираем «Player» и ищем триггер «Is flashing» (рис. 8).

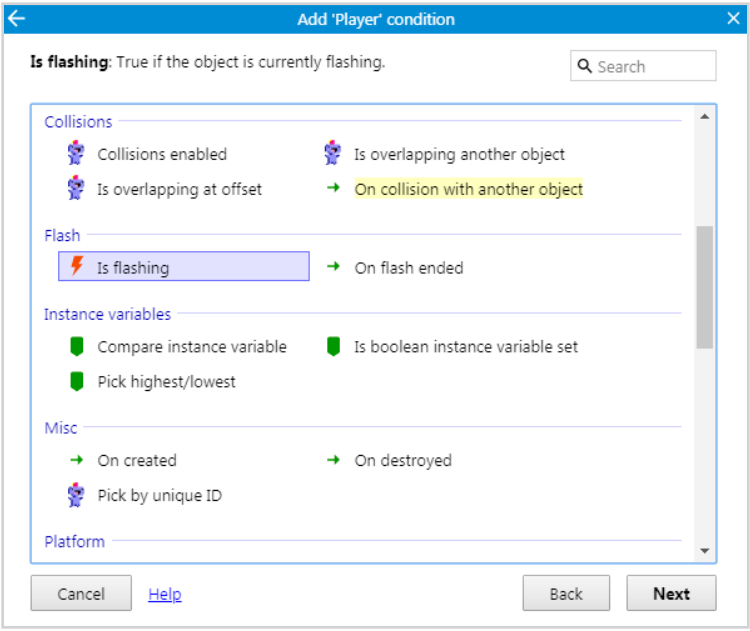

Рисунок 8

Затем инвертируем его в цепочке событий. Кликаем по нему правой кнопкой мыши и выбираем «Invert» (рис. 9).

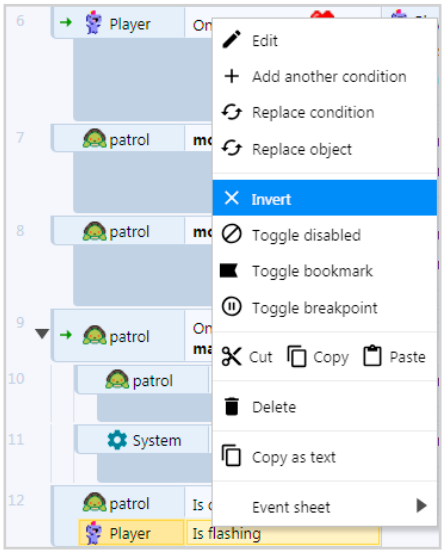

Рисунок 9

И добавляем событие: «Player => Flash». Значение «Flash» выставляем следующим образом: «On» Time=0,1», «Off» time=0,1», «Duration=2,0» (рис. 10).

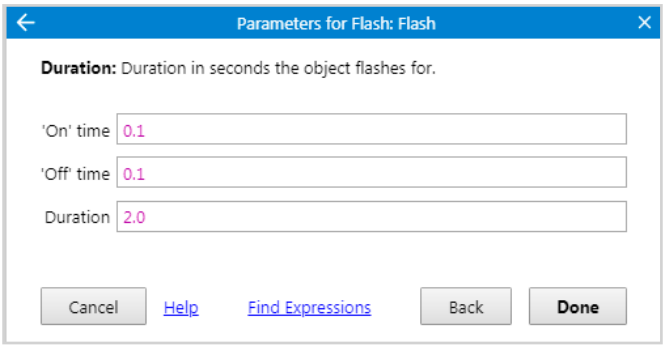

Рисунок 10

Добавим еще одно событие, чтобы игрок терял одну жизнь при соприкосновении с врагом. «Add action => Lives». Нажимаем «Find Expression», дважды кликаем по «lives» и выбираем «Get object's width with pixels».

В строке «Width» появится значение, к которому нужно дописать –40 (длинна одного сердечка) (рис. 11).

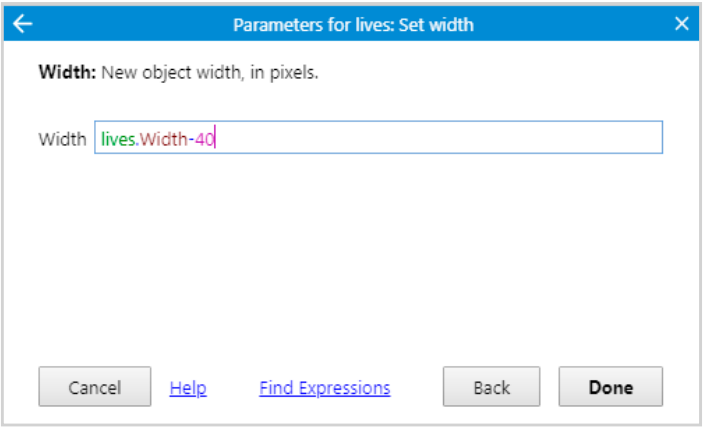

Рисунок 11

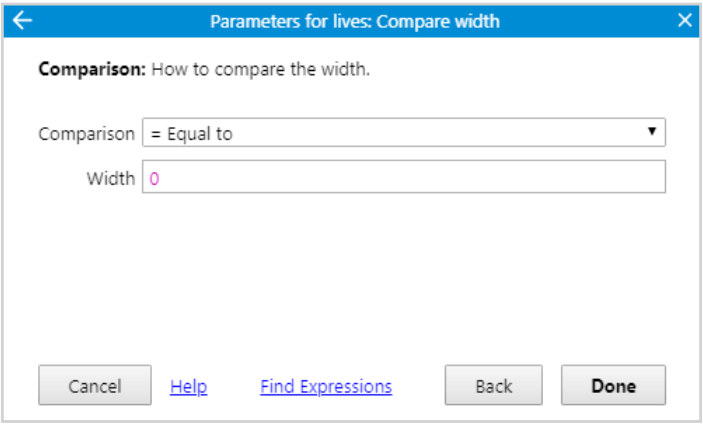

Рисунок 12

Теперь нужно добавить одно событие, которое всегда будет перезапускать уровень, если игрок потеряет последнюю жизнь. «Add event => Lives => Compare width». Выставляем длину 0 px (рис. 12).

Затем добавляем действие «System => Restart layer» (рис. 13).

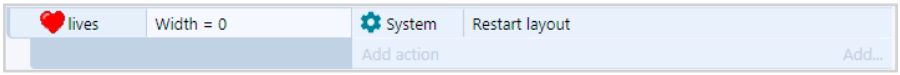

Рисунок 13

Давайте немного разнообразим врагов и создадим второго противника, но на этот раз пускай он преследует игрока.

Добавляем спрайт и задаем ему поведение «Platform» (рис. 14).

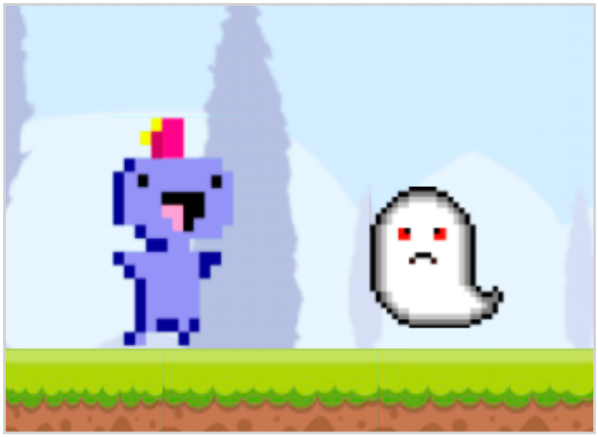

Рисунок 14

На панели «Properties» убираем отметку «Default controls» (рис. 15).

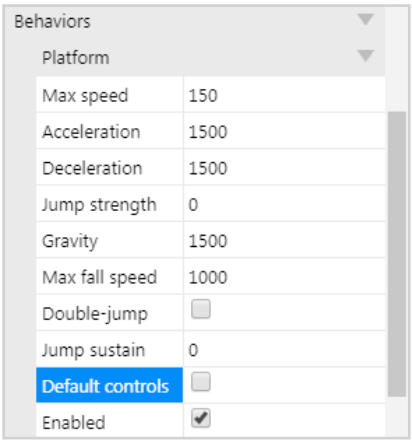

Рисунок 15

Добавляем событие. «System => Compare two values». В появившемся окне выставляем следующие значения:

- «Firsr value=Player.X»
- «Comparison=<Less than»
- «Second value=follow.X» (рис. 16)

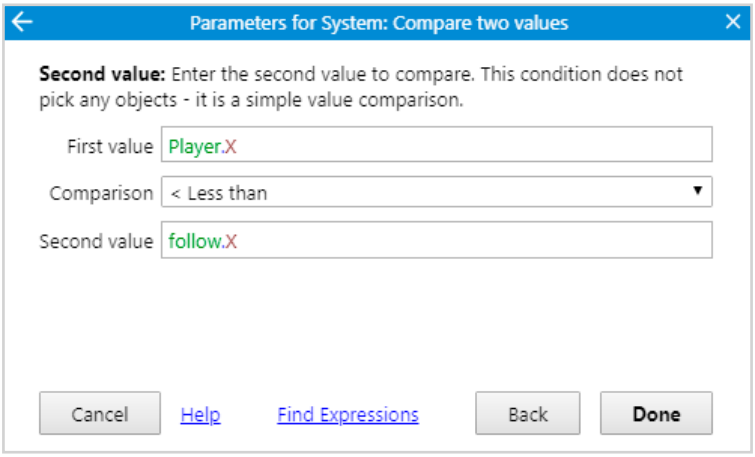

Рисунок 16

Теперь добавим действие. «Follow => Simulate control => Left». То есть, когда позиция врага меньше, чем игрока, будет происходить симуляция нажатия клавиши влево, и первый будет догонять второго (рис. 17).

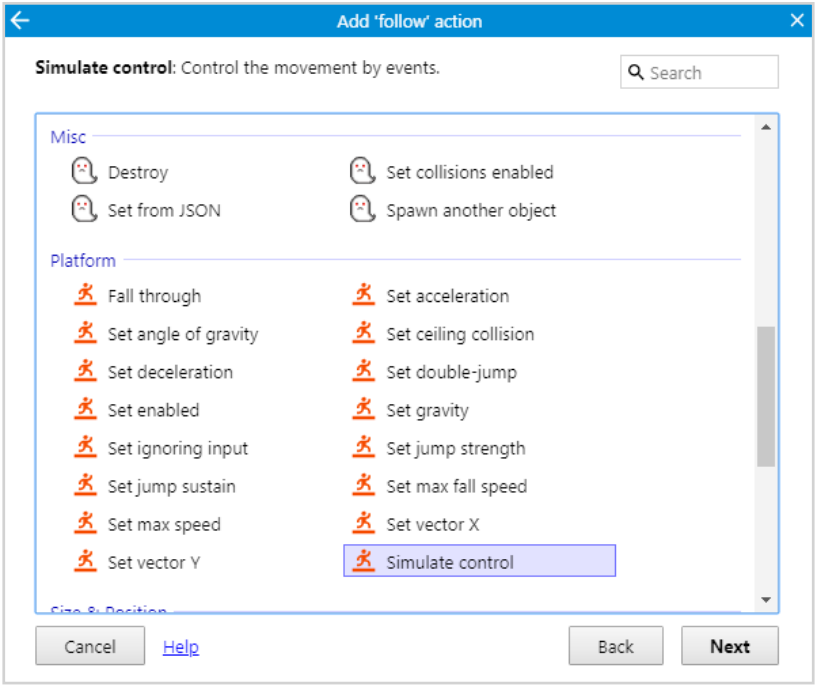

Рисунок 17

Копируем ивент и выставляем противоположные значения (рис. 18).

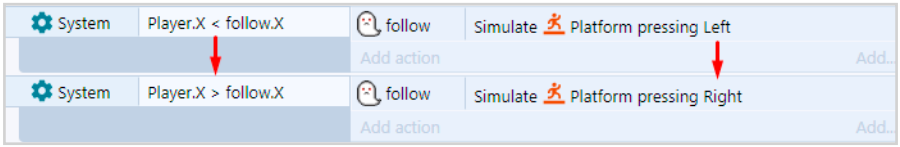

Рисунок 18

Теперь наш враг постоянно преследует игрока (рис. 19).

Рисунок 19

Добавим еще одно событие. Нужно, чтобы враг наносил игроку урон, в результате получения которого пользователь бы терял жизнь, а также срабатывал бы эффект дрожания камеры. «Add event => Player => Shake» (рис. 20).

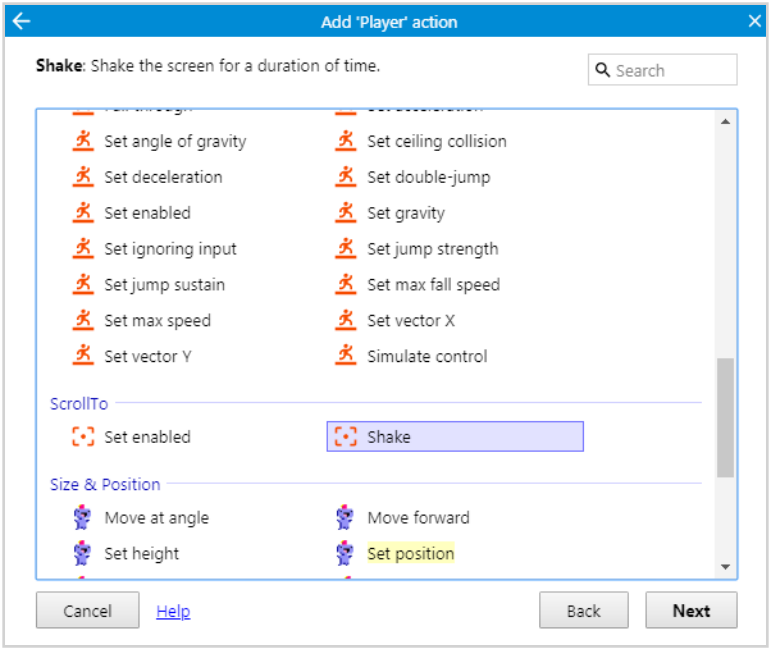

Рисунок 20

В появившемся окне выставляем время и амплитуду дрожания камеры (рис. 21).

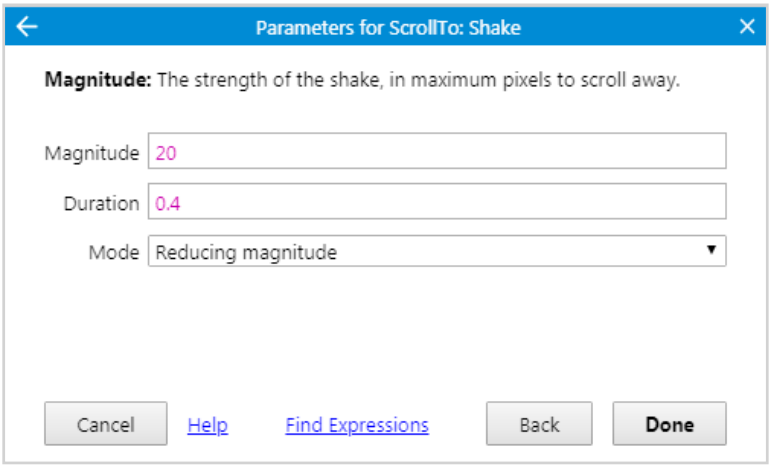

Рисунок 21

Затем добавляем событие потери жизней. В цепочке это выглядит следующим образом (рис. 22).

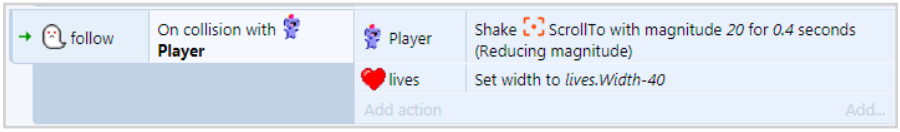

Рисунок 22

Теперь нужно ограничить движение врага. Для этого воспользуемся маркером. Добавляем новый спрайт. Его цвет на данный момент не имеет значения, потом мы сделаем его прозрачными. Задаем имя «marker». Нужно поставить созданный блок слева от врага, чтобы, пока наш игрок проходит начало уровня, преследователь не упал в яму (рис. 23).

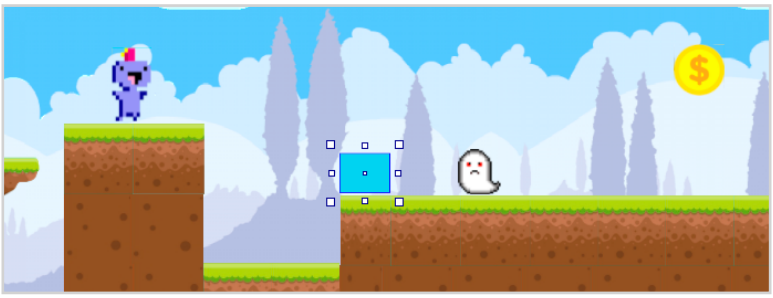

Рисунок 23

Задаем объектам поведение «Solid» (рис. 24).

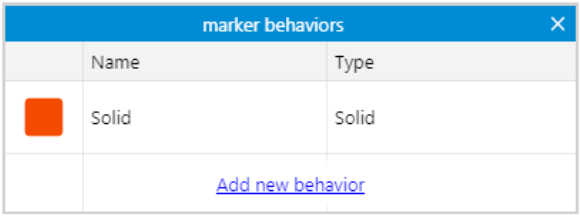

Рисунок 24

Во вкладке «Properties» снимаем отметку в чекбоксе «Initially visible». Соответственно, в проекте маркеры остаются видимыми, но при запуске игры их не видно (рис. 25).

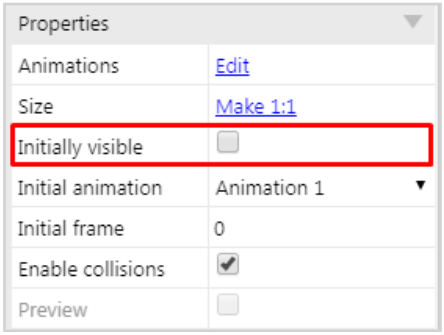

Рисунок 25

Сейчас маркер останавливает движение врага. Необходимо сделать так, чтобы наш игрок смог проходить через эти маркеры. Добавляем новое событие. «Player => On collision with => Marker» (рис. 26).

Рисунок 26

И добавляем действие «Marker => Set Enabled». Выставляем значение «State=Disable» (рис. 27).

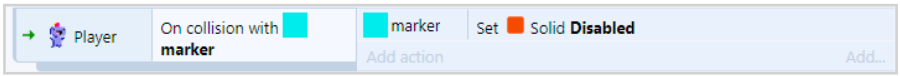

Рисунок 27

Это означает, что при соприкосновении игрока и маркера, маркер теряет поведение «Solid».

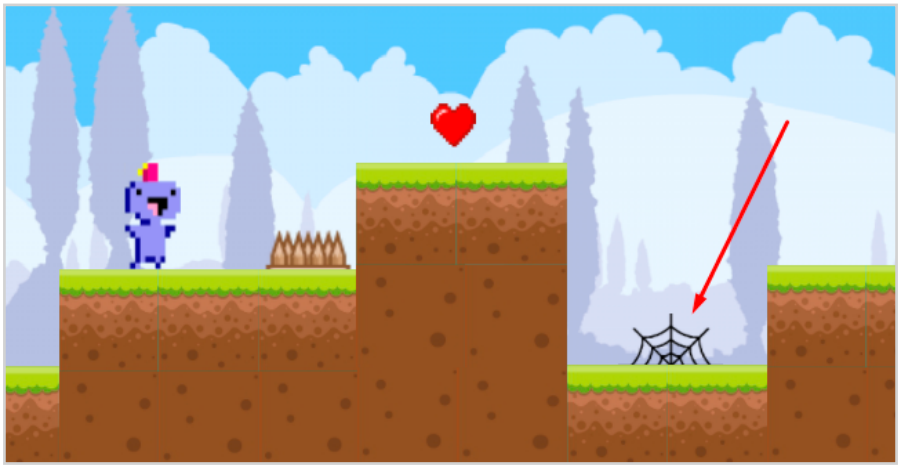

Рисунок 28

Преследующий враг у нас уже есть, но этого все еще мало, чтобы сделать игру увлекательной. В карту нужно добавить больше преград. Давайте рассмотрим, как создать эффект заморозки игрока при соприкосновении со специальными объектами. Добавим спрайт с паутиной и назовем его «web» (рис. 28).

Добавляем событие. «Web => On collision with => Player» (рис. 29).

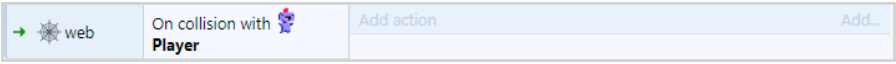

Рисунок 29

Теперь добавим действие. «Player => Set enable» (рис. 30).

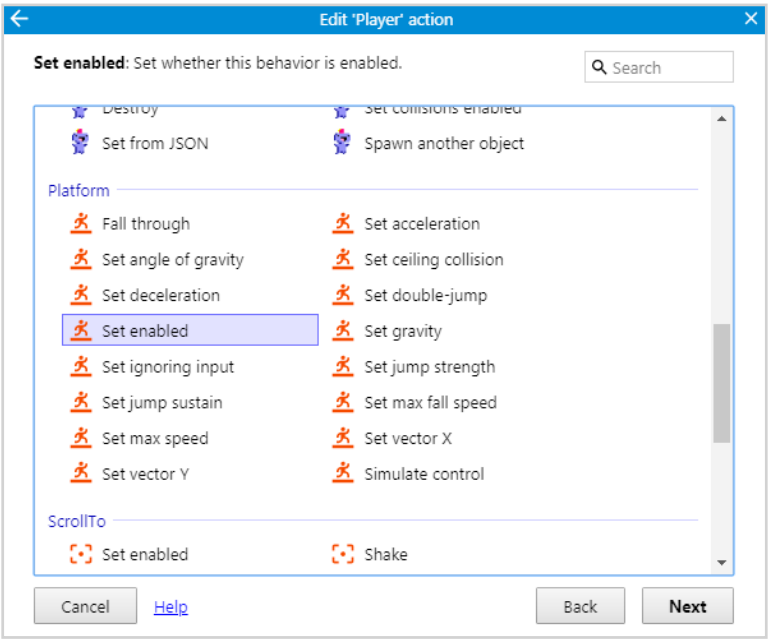

Рисунок 30

Нужно изменить цвет игрока на время заморозки. Для этого нужно добавить анимацию. Кликаем дважды по игроку. На панели справа добавляем новую анимацию, щелкнув правой кнопкой мыши и выбрав «Add new animation». Называем ее «frost» (рис. 31).

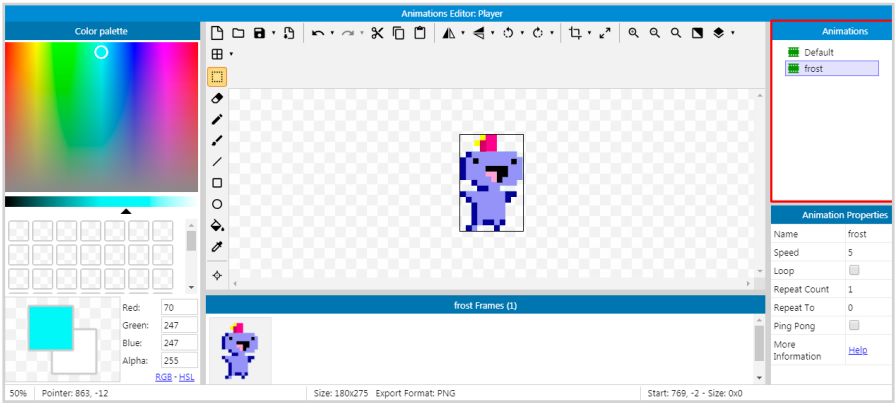

Рисунок 31

Копируем персонажа и закрашиваем его другими цветами (рис. 32).

Рисунок 32

Теперь добавим это действие. «Player => Set animation». В появившемся окне прописываем «Animation=«frost» (рис. 33).

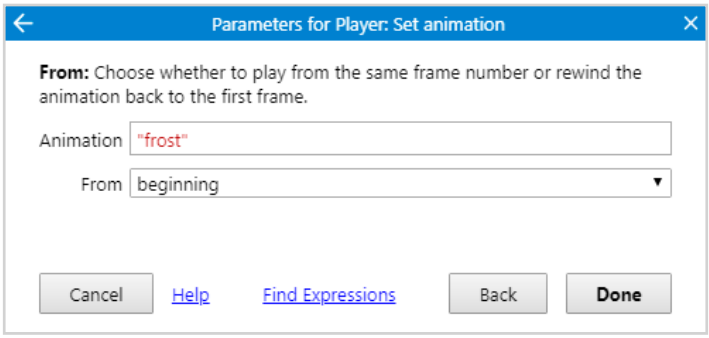

Рисунок 33

Добавляем новое действие, из-за которого персонаж застынет на одном месте на 2 секунды. «System => Wait». В окне с параметрами выставляем время, на которое персонаж будет заморожен (рис. 34).

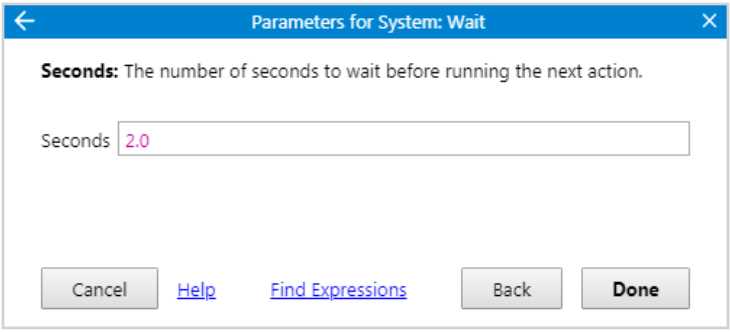

Рисунок 34

Следует также добавить действие, которое разморозит персонажа и позволит ему двигаться. «Player => Set Enable => Enable» (рис. 35).

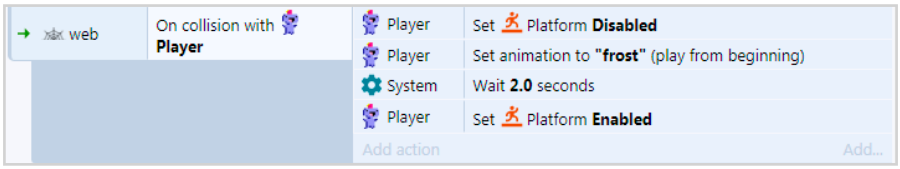

Рисунок 35

После этого возвращаем персонажу исходную анимацию. «Player => Set animation => Default» (название исходной анимации персонажа) (рис. 36).

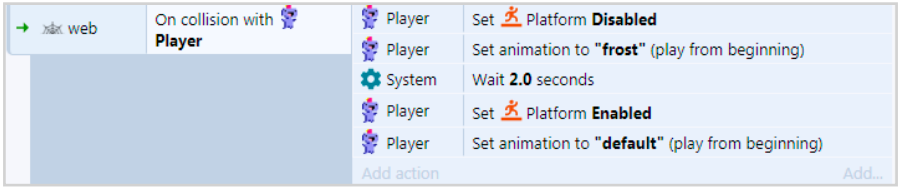

Рисунок 36

Теперь при попадании на паутину, персонаж меняет цвет, застывает на 2 секунды, а потом возвращается к стандартному цвету и продолжает двигаться (рис. 37).

Рисунок 37

Если персонаж может заморозится на несколько секунд, то давайте рассмотрим и противоположную ситуацию. Почему бы не добавить элементы, от соприкосновения с которыми, игрок будет воспламеняться? Рассмотрим, как создать анимацию возгорания. Размещаем новый спрайт на игровой карте (рис. 38).

Рисунок 38

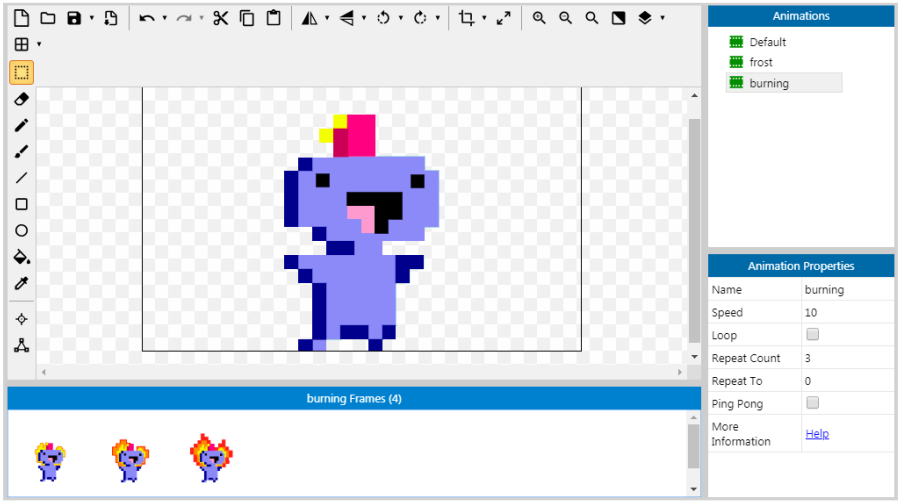

Рисунок 39

Создаем анимацию возгорания персонажа, которую называем «burning». На панели справа задаем параметры повторения и скорости воспроизведения (рис. 39).

Переходим в цепочку событий. Повторяем предыдущее событие, только меняем объект паутины на огонь, а анимацию заморозки на возгорание (рис. 40).

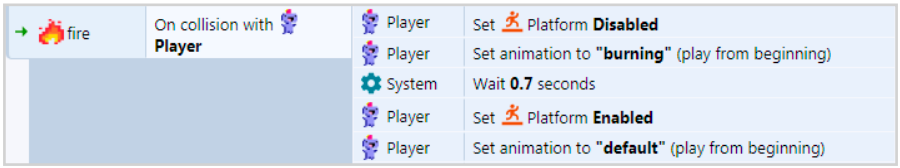

Рисунок 40

Теперь попадая на огонь, игрок воспламеняется (рис. 41).

Рисунок 41

В это же событие можно добавить потерю жизни от соприкосновения с огнем (рис. 42).

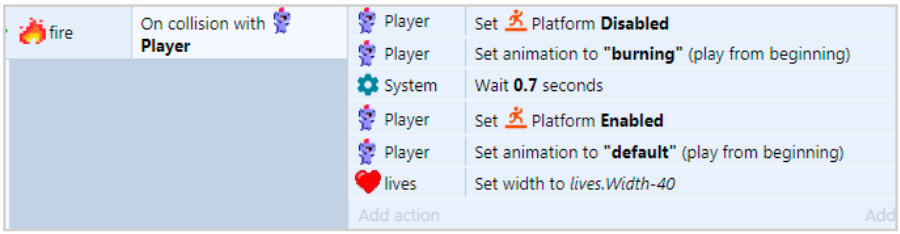

Рисунок 42

Итак, мы создали несколько новых видов объектов, которые помогут разнообразить уровень. Теперь пройти игру становится все сложнее и сложнее. С каждым новым добавленным элементом наша игра становится более захватывающей.

Посмотрите, какие элементы врагов и препятствий можно добавить в игровую карту, тем самым усложняя прохождение уровней (рис. 43–45).

Рисунок 43

Рисунок 44

Рисунок 45

В свободное время рассмотрите кейсы по созданию игр на «Construct 3» по ссылке. Обратите внимание на то, какие враги и преграды представлены в играх, а также как они запрограммированы.

Урок 4

Создание уровней игры, телепортов и точек сохранения игры (чекпоинтов)

➢ **ПЛАН РАБОТЫ.**

Добавление второго и третьего уровней через создание новых layout и event sheet. Эффект кривой обучения. Добавление невидимых спрайтов для переходов между уровнями и телепортации персонажа. Создание локальной булевой переменной для отслеживания состояния перехода к чекпоинту и локальной числовой переменной для порядкового номера чекпоинта.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Создание уровней игры, телепортов и точек сохранения игры (чекпоинтов)

У нас уже есть достаточно элементов игровой карты, чтобы сделать его отдельным уровнем (рис. 1).

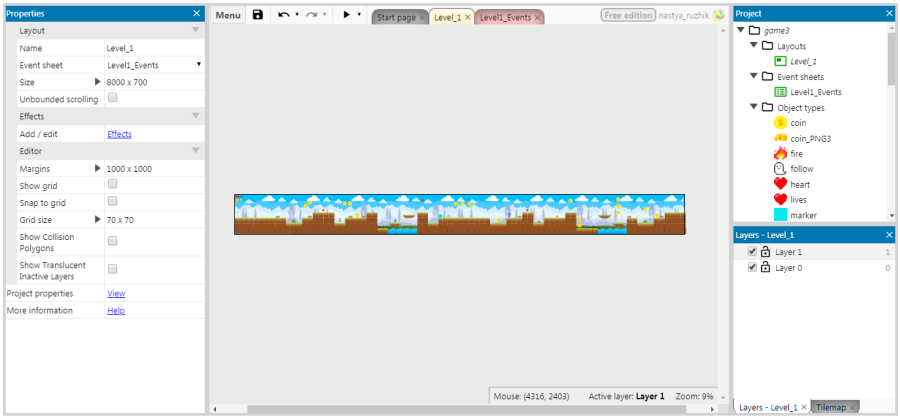

Рисунок 1

Обратите внимание на одну особенность игрового уровня. При его воспроизведении персонаж может выходить за рамки «frame». Чтобы это отключить необходимо всего лишь добавить игроку дополнительное поведение — «Bound To Layout» (рис. 2).

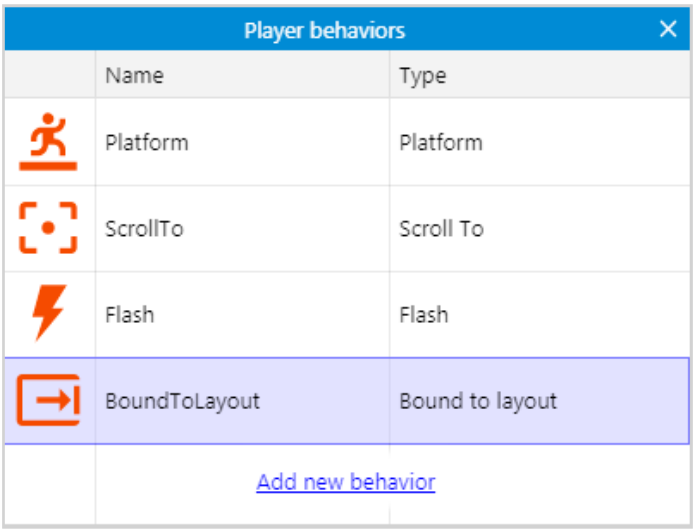

Рисунок 2

Рассмотрим, как добавить следующие «layout». Для этого на панели «Project» справа кликаем правой кнопкой мыши по папке «Layouts» и выбираем «Add layout» (рис. 3).

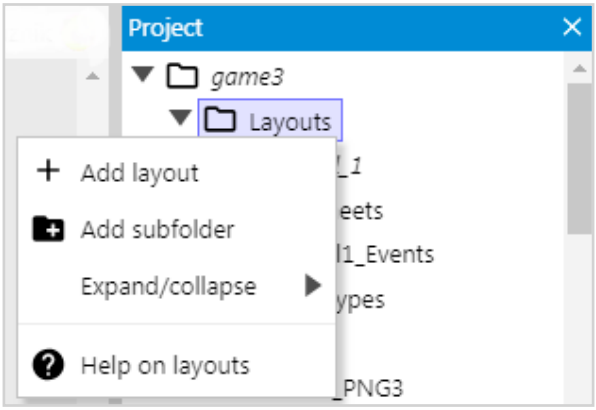

Рисунок 3

Появится диалоговое окно с вопросом: «Хотите добавить и страницу с цепочкой событий для этого «layout»?». Выбираем «Add event sheet» (рис. 4).

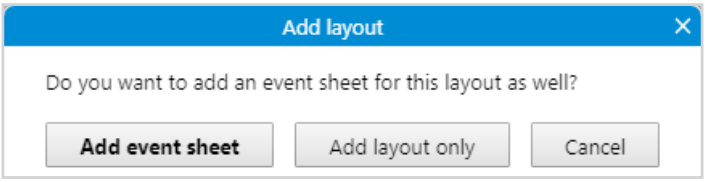

Рисунок 4

На панели «Project» появился новый «layout» и страница событий (рис. 5).

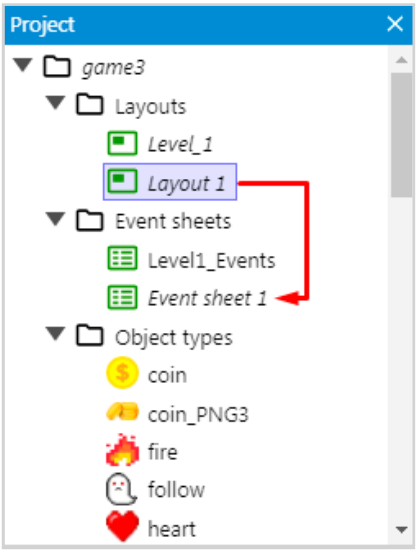

Рисунок 5

Давайте поменяем их названия, чтобы было проще с ними работать. Назовем их «Level_2» и «Level2_pEvents» (рис. 6).

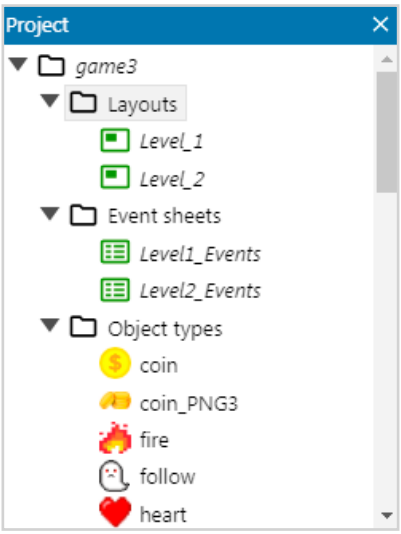

Рисунок 6

Если кликнуть по пустому рабочему пространству, на панели «Properties» слева отобразится информация о данной игровой карте. Здесь указано название уровня и подвязанный к нему список событий. Если вы хотите создать несколько уровней, оставив для них одни и те же цепочки событий, то в поле «Event Sheet» выбираем нужный список (рис. 7).

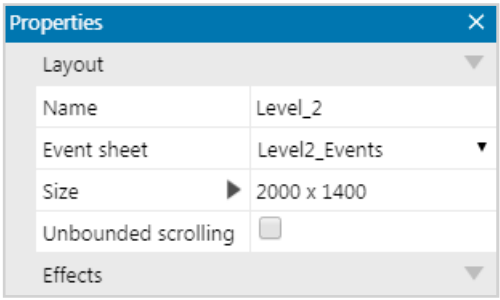

Рисунок 7

Добавим фон с помощью «Tiled Background», чтобы, при переходе из вкладки первого уровня на вкладку второго, визуально быстрее ориентироваться, на каком уровне мы находимся (рис. 8).

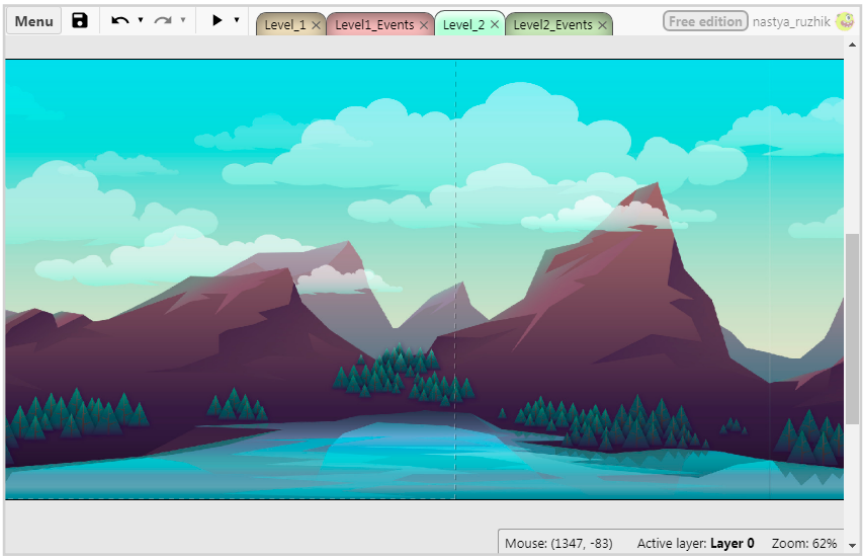

Рисунок 8

Таким же способом добавляем 3 уровень с помощью «Add layout» (рис. 9).

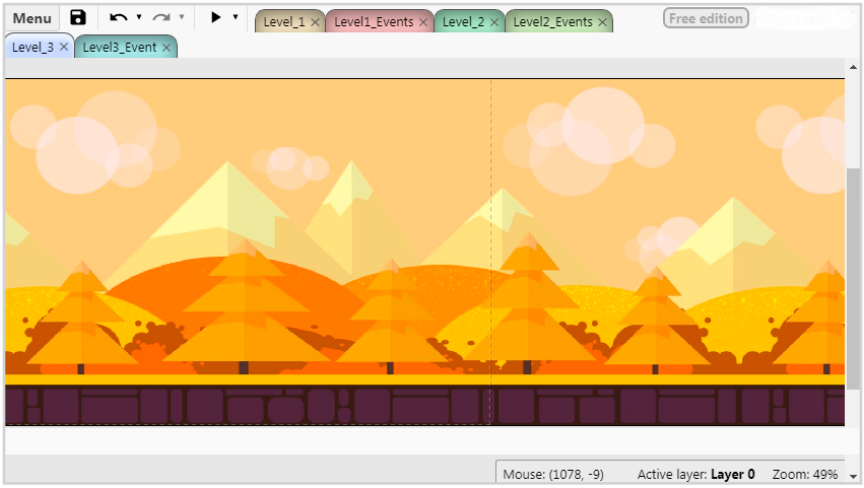

Рисунок 9

После того, как мы создали несколько уровней, давайте разберемся, что такое кривая обучения. По сути, это функция зависимости навыка игрока от времени проведенного в игре. Ее можно изобразить в виде небольшого графика (рис. 10).

Рисунок 10

Суть кривой в том, что уровни должны создаваться с нарастающей сложностью, увеличением количества врагов и препятствий.

То есть, на первых уровнях игрок знакомится с игровой картой, персонажами, узнавая, как игрок взаимодействует с окружением. Затем происходит игровой прогресс. Это случается по той причине, что мы уже привыкаем к игровому окружению, действиям и событиям, которые появляются на уровнях. А дальше прогресс замедляется, так как появляются новые вводные. Соответственно, игрок снова пытается разобраться, как с ними взаимодействовать. Это усложняет прохождение уровня.

У кривой обучения есть две задачи: убедить разработчиков, что на игрока не сваливалось слишком много нововведений, и, наоборот, что в игре не будет длинных этапов без новых элементов, чтобы игрок не заскучал.

Если уже мы заговорили об игровых уровнях и их усложнении, давайте рассмотрим, как сделать переход на следующий «level». Для этого необходимо добавить невидимые спрайты. Добавляем элемент, закрасив его в какой-нибудь цвет и назовем его «teleport» (рис. 11).

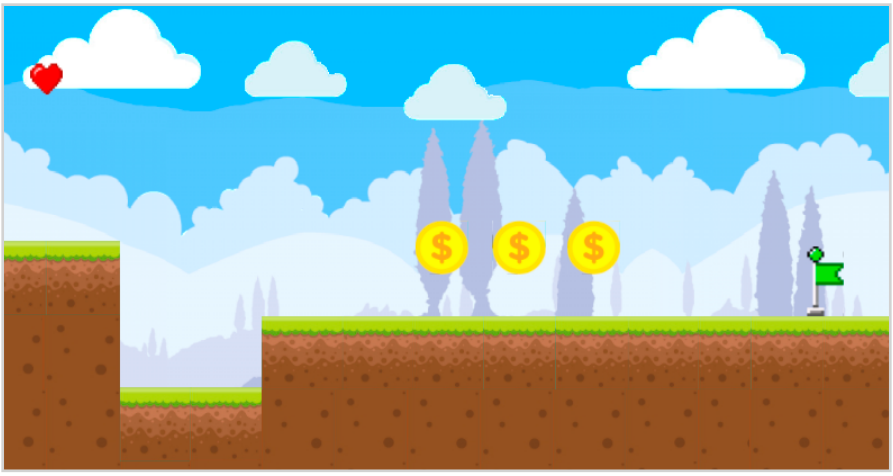

Рисунок 11

Переходим к цепочке событий. «Player => on collision with teleport» (рис. 12).

Рисунок 12

Добавляем действие: «System => Go to layout» (рис. 13).

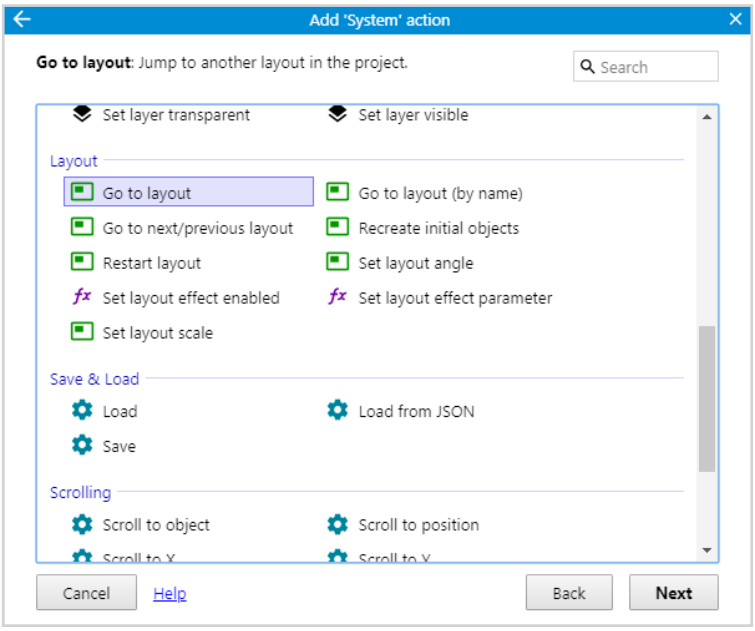

Рисунок 13

В окне параметров выбираем второй уровень (рис. 14).

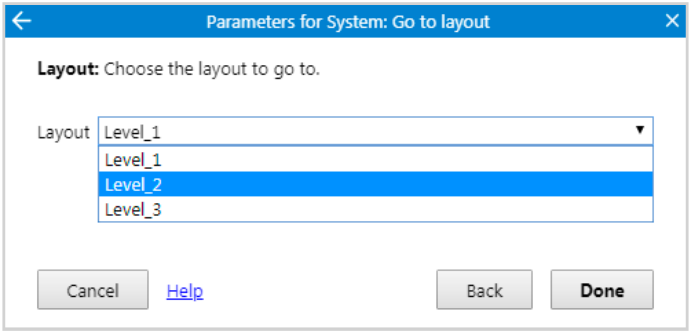

Рисунок 14

Теперь в самом конце платформы размещен телепорт на второй уровень (рис. 15).

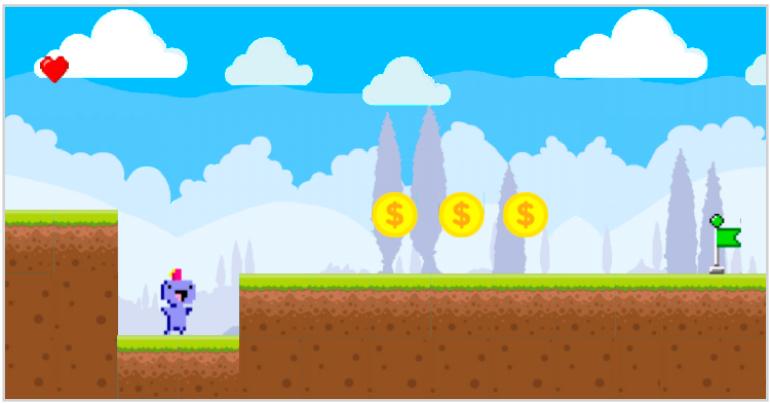

Рисунок 15

Чтобы сделать телепорт невидимым, выделяем спрайт и на панели «Properties» убираем галочку из чекбокса «Initially visible» (рис. 16).

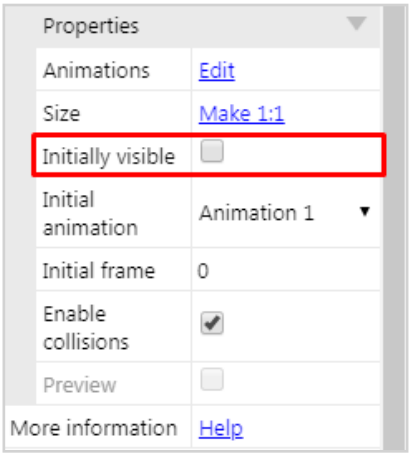

Рисунок 16

Теперь при попадании персонажа на этот невидимый спрайт, мы автоматически попадаем на второй уровень (рис. 17).

Рисунок 17

Теперь следует создать локальную булевую переменную для отслеживания состояния перехода к чекпоинту, а также локальную числовую переменную для порядкового номера чекпоинта.

Чекпоинт — это точка сохранения текущего состояния игры. Рассмотрим, как это работает в «Construct 3». Добавим новый спрайт в виде флажка и назовем его «checkpoint» (рис. 18).

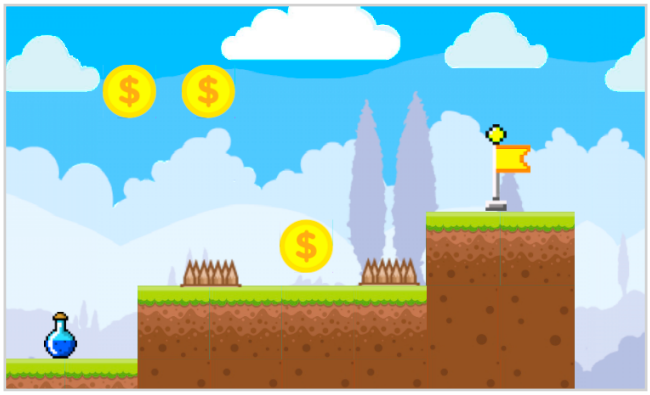

Рисунок 18

Добавляем ему несколько переменных. Одну называем «check» и задаем ей тип «Boolean», вторую называем «number» и задаем «тип Number» (рис. 19–20).

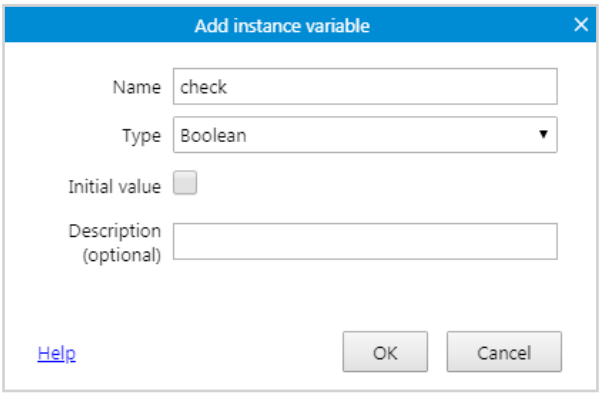

Рисунок 19

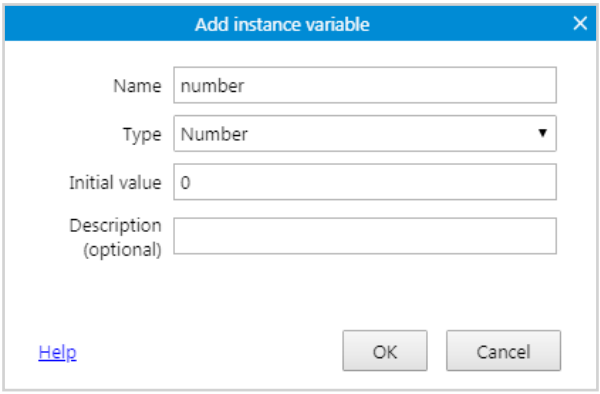

Рисунок 20

Теперь необходимо прописать события. Переходим к цепочке, кликаем правой кнопкой мыши по пустому пространству и выбираем «Add global variable» (рис. 21).

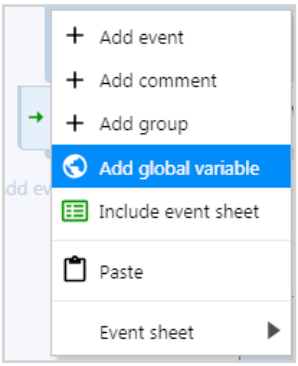

Рисунок 21

Называем ее «resp», задаем тип «Number» и начальное значение «Initial value=1» (рис. 22).

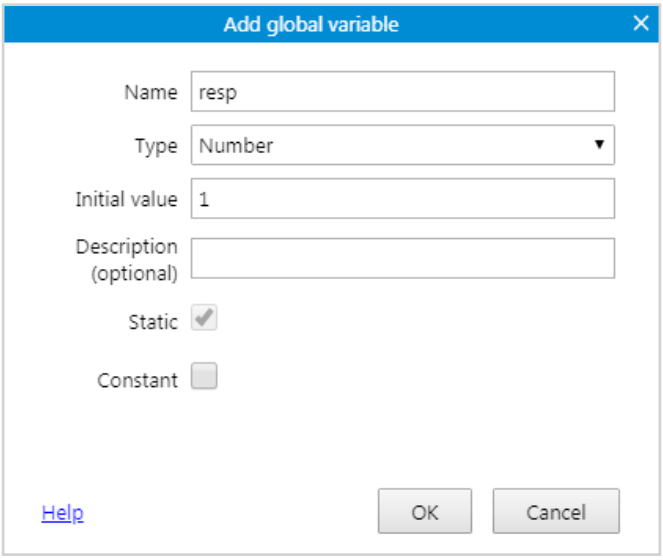

Рисунок 22

Глобальная переменная отображается вверху списка перед всеми событиями (рис. 23).

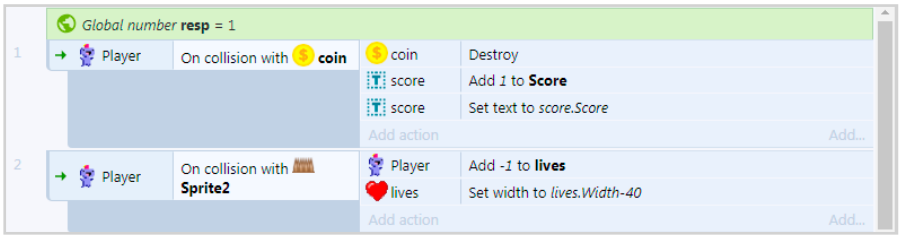

Рисунок 23

Добавляем «event: Player => on collision with another object => checkpoint» (рис. 24).

Рисунок 24

Кликаем по событию правой кнопкой мыши и выбираем «Add another condition» (рис. 25).

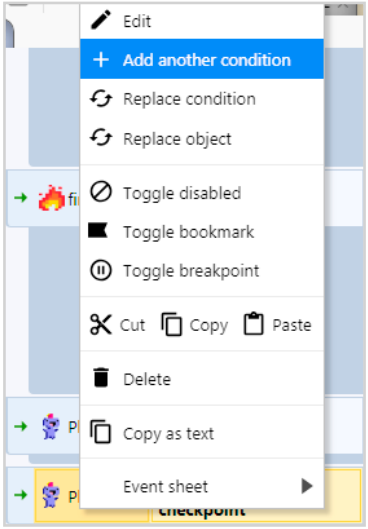

Рисунок 25

Выбираем «checkpoint => Is Boolean instance variable set» (рис. 26).

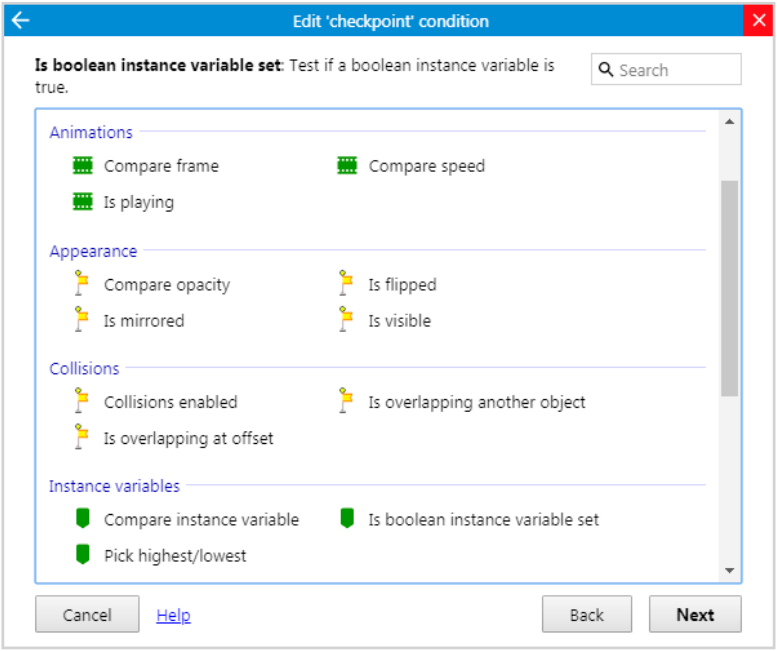

Рисунок 26

Затем выбираем «check» и нажимаем «Done» (рис. 27).

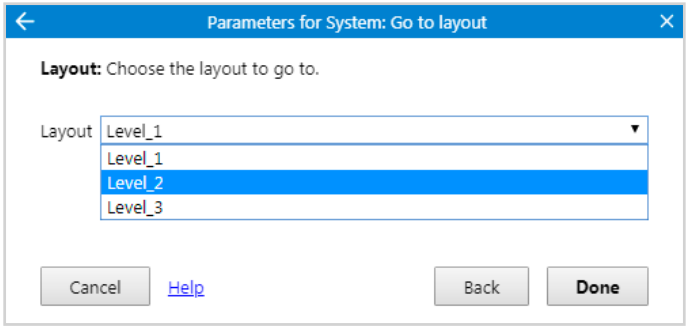

Рисунок 27

Инвертируем созданное условие (рис. 28).

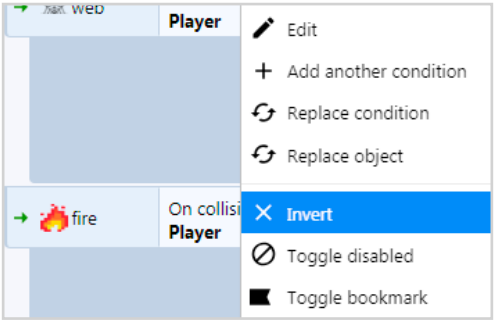

Рисунок 28

Теперь добавляем событие «Checkpoint Set Boolean »(рис. 29).

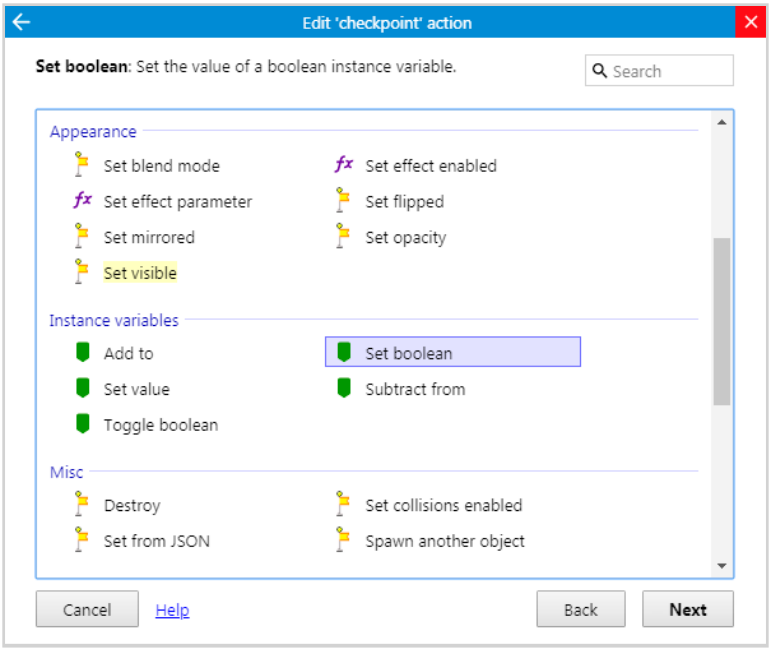

Рисунок 29

В появившемся окне параметров выставляем «check» и «True» (рис. 30).

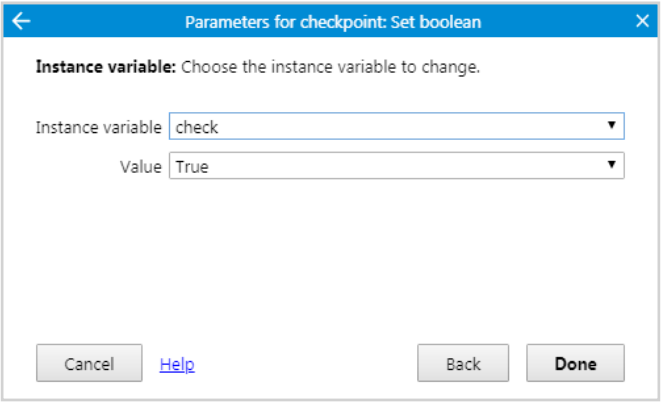

Рисунок 30

«System => Set value». В появившемся окне выбираем «Find Expression» (рис. 31).

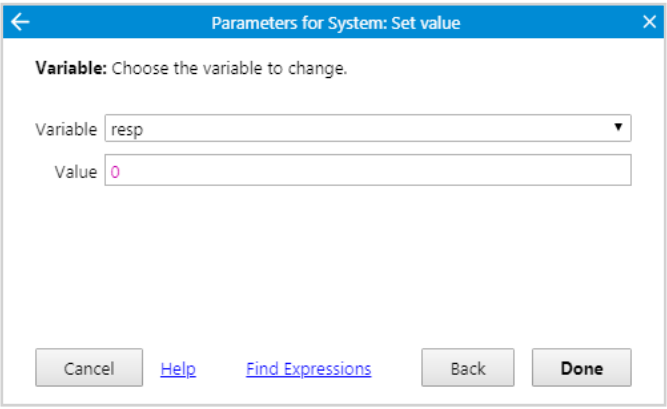

Рисунок 31

Затем выбираем «checkpoint» и «A numeric variable» (рис. 32).

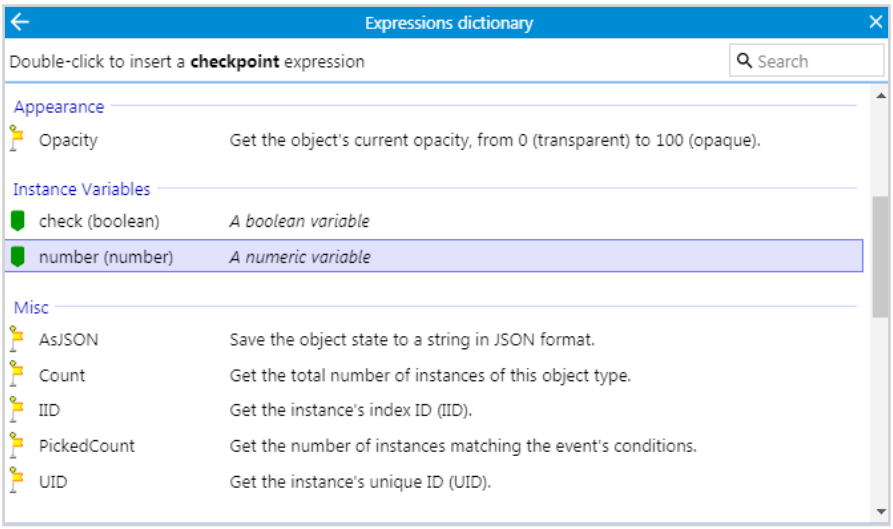

Рисунок 32

В окне параметров сразу появится значение «Value» (рис. 33).

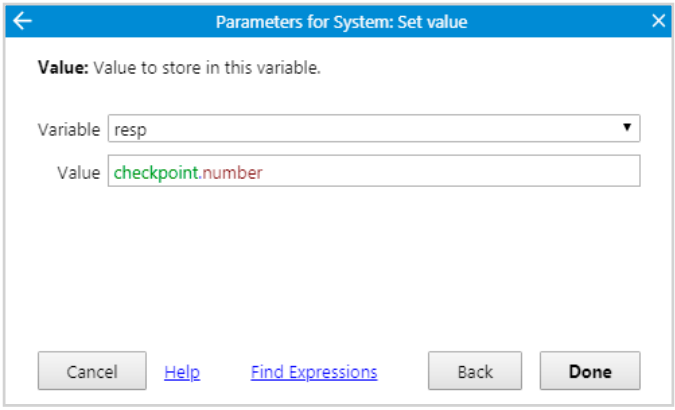

Рисунок 33

Добавляем еще одно действие, чтобы, при пересечении персонажем чекпоинта, второй становился прозрач-

ным. «Checkpoint => Set opacity». В параметрах поставим 40 (рис. 34).

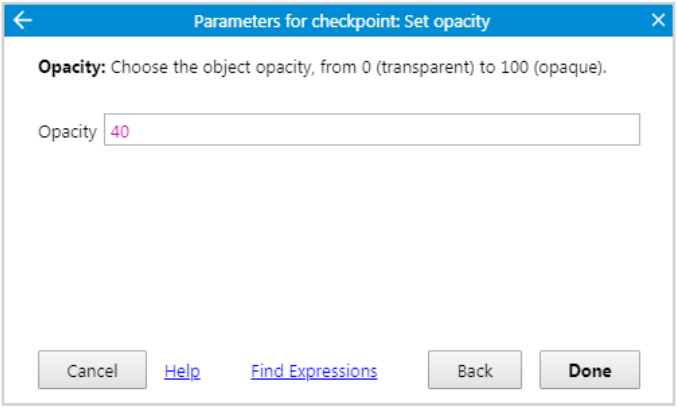

Рисунок 34

Точка сохранения готова. Теперь добавляем в цепочку событий группу. Правая клавиша мыши — «Add group». Называем ее «resp» (рис. 35).

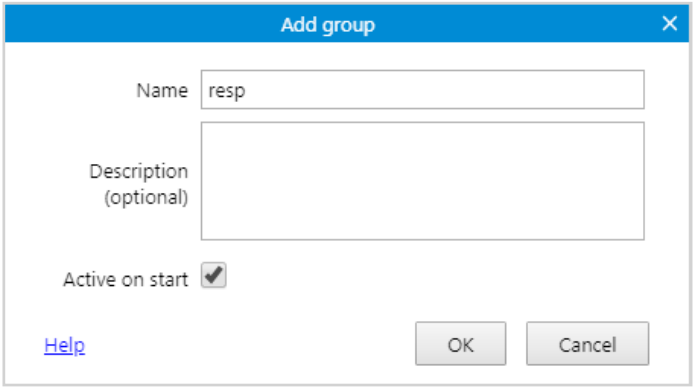

Рисунок 35

Добавляем «sub-event». «Checkpoint => Compare instance variable» (рис. 36).

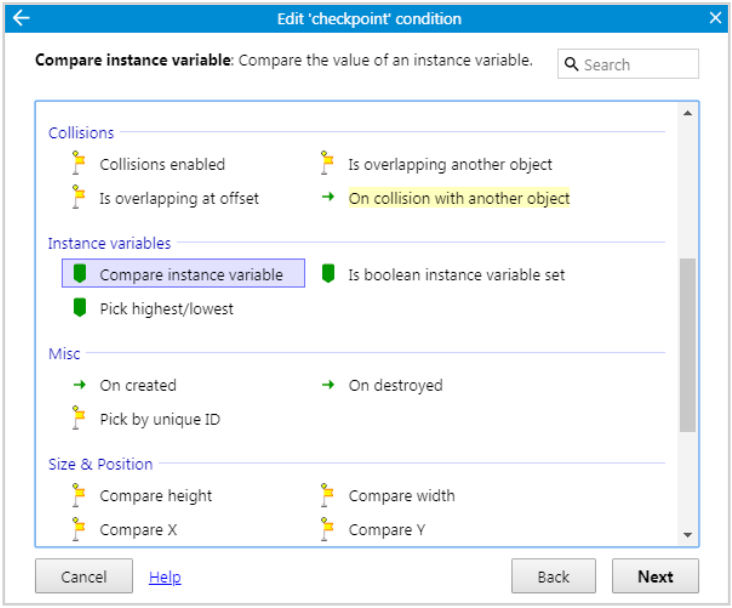

Рисунок 36

В окне параметров выставляем глобальную переменную «resp» в поле «Value» (рис. 37).

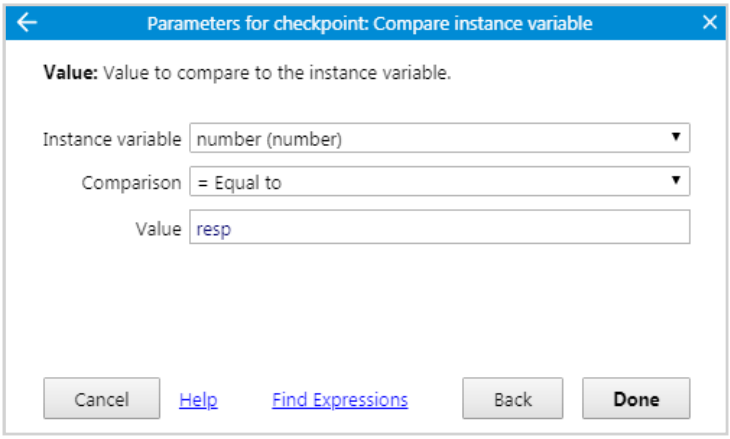

Рисунок 37

Теперь нужно добавить действия. «Player => Destroy».

Затем «System => Create object». Выбираем игрока и выставляем координаты его размещения. По «X» и по «Y» это будет место «Checkpoint» (рис. 38).

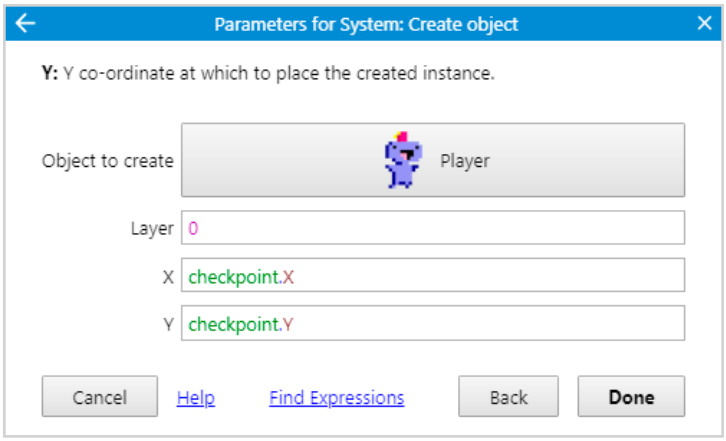

Рисунок 38

Добавляем еще одно действие. «System => Set group». В окне параметров прописываем название группы (у нас это «resp») и выставляем статус «Deactivate» (рис. 39).

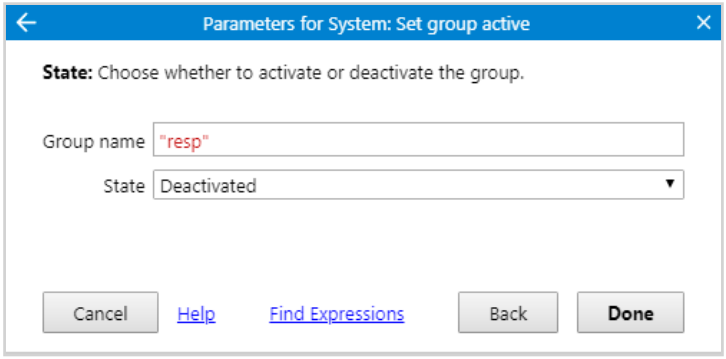

Рисунок 39

Для проверки можно также добавить кнопку, при нажатии которой игрок будет «респауниться». Прописываем это в событиях (рис. 40).

Рисунок 40

И добавляем действие. «System => Set group active». В окне параметров указываем группу «resp» и статус «Activated» (рис. 41).

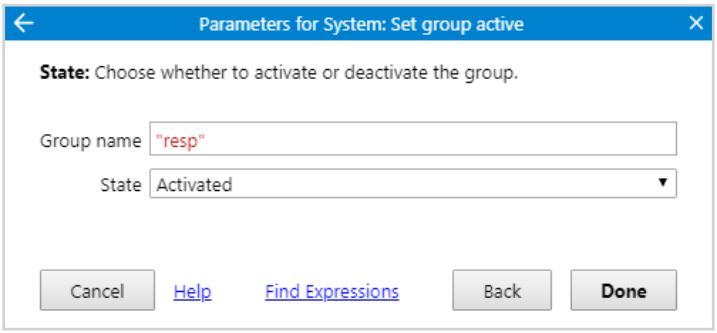

Рисунок 41

В цепочке событий это выглядит следующим образом (рис. 42).

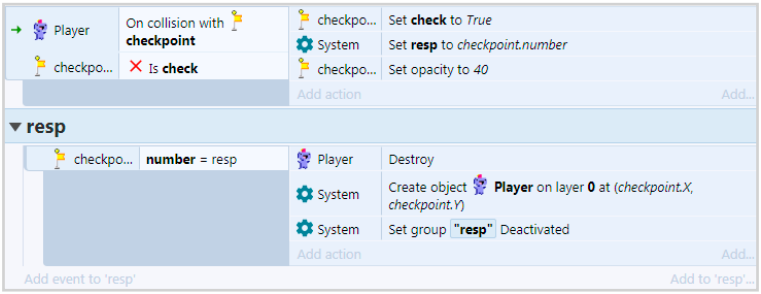

Рисунок 42

Осталось сделать так, чтобы персонаж, после потери 3-х жизней, оказывался у чекпоинта, а не в самом начале уровня. Для этого находим событие, где жизни=0 и добавляем вместо «restart layout» новое действие. «Player => Set position to another object» и выбираем наш «checkpoint» (рис. 43).

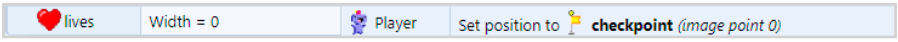

Рисунок 43

И здесь же добавляем действие «Lives => Set width => 120 px», чтобы при появлении возле чекпоинта у игрока снова было 3 жизни (рис. 44).

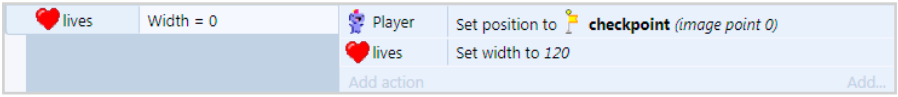

Рисунок 44

Теперь, когда игрок теряет все жизни после прохождения чекпоинта, уровень не запускается с самого начала, а персонаж начинает игру с точки сохранения (рис. 45).

Рисунок 45

Рассмотрите, какие интересные чекпоинты и телепорты добавляют в популярных платформерах.

1. «Splasher »*(checkpoint)* (рис. 46).

Рисунок 46

2. «King of Egypt GX »*(checkpoint)* (рис. 47).

Рисунок 47

3. «Super Cat Girl »*(телепорт)* (рис. 48).

Рисунок 48

Бонусное задание!

Давайте добавим одну хитрость в создаваемую нами игру — скрытый бонусный уровень! Для этого создаем еще один «layout» вместе с «event sheet »и называем их «bonus level» и «bonus level_Events» соответственно (рис. 1).

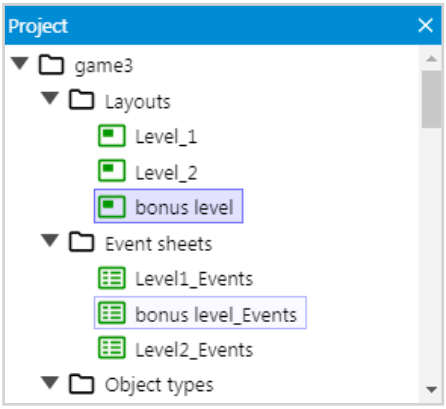

Рисунок 1

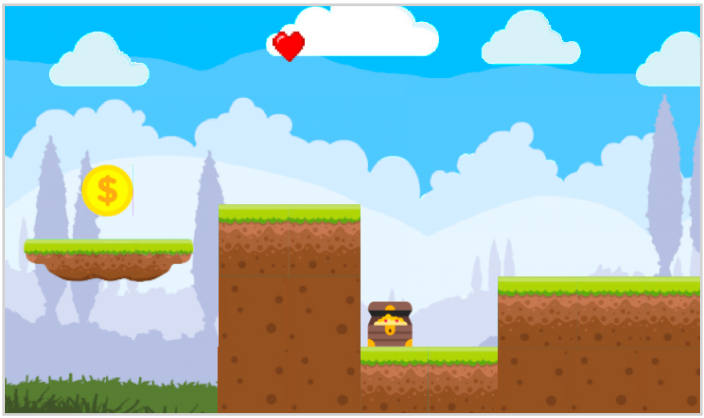

Рисунок 2

Теперь добавим спрайт в конец игровой карты. В нашем случае это сундук с драгоценностями, потому что уровень бонусный (рис. 2).

Сразу пропишем, что игрок при попадании на этот спрайт перенесется на бонусный уровень. Событие: «Player => on collision with => bonus_teleport». Действие: «System => go to layout => bonus level» (рис. 3).

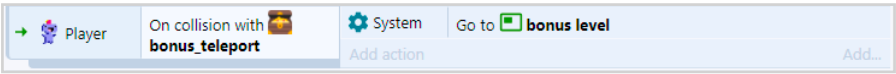

Рисунок 3

Но поскольку уровень бонусный, не каждый игрок сможет на него попасть. Нужно поставить условие, при котором телепорт будет неактивен. Например, сделаем так, чтобы игрок смог увидеть и перейти на бонусный уровень только в том случае, если на его счету более 10 монет.

Выделяем спрайт и делаем его невидимым, сняв отметку «Initial visible» (рис. 4).

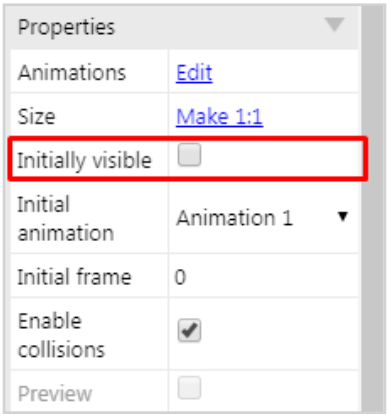

Рисунок 4

Прописываем событие. «Score => Compare Instance variable» (выставляем значение — 10). Затем действие: «bonus_teleport => Set collisions => Disabled». Теперь, если счет меньше 10, то переход будет невидимым и не активированным (рис. 5)

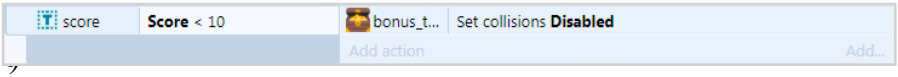

Рисунок 5

Следующее событие — «Score => Compare Instance variable» (выставляем значение ≥ 10). И действие, чтобы переход стал видимым и активировался. «Bonus_teleport => Set visible» и «bonus_teleport => Set collisions => Enabled» (рис. 6).

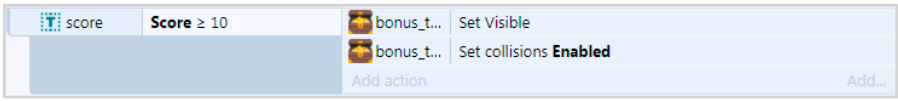

Рисунок 6

Переходим к бонусному уровню. Размещаем на нем игрока, счет жизней и монет, а также сами монеты (рис. 7).

Рисунок 7

В конце игровой карты снова ставим «bonus_teleport», чтобы перейти на следующий уровень (рис. 8).

Рисунок 8

Переходим к цепочке событий бонусного уровня. Прежде всего нужно прописать события для персонажа и монет. Они дублируются с первым уровнем (рис. 9).

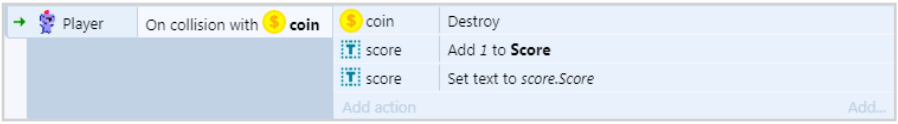

Рисунок 9

После этого в список событий бонусного уровня добавляем переход на следующий уровень при соприкосновении игрока с сундуком (рис. 10).

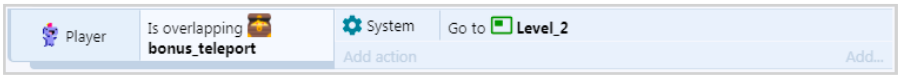

Рисунок 10

Осталось исправить небольшой нюанс. Счет очков обнуляется на каждом уровне, а нам нужно запрограммировать игру так, чтобы он продолжался при переходе. Для этого выбираем объект с цифровым значением очков и ставим на панели «Properties» отметку «Global» (рис. 11–12).

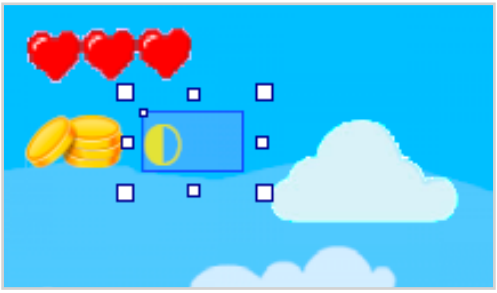

Рисунок 11

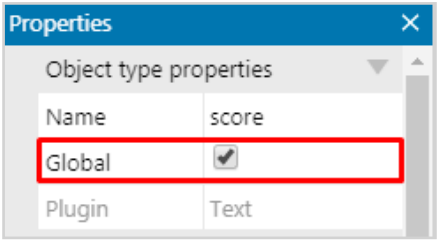

Рисунок 12

На всех последующих «Layouts» просто удаляем объект с цифровым показателем. Он должен остаться только на первом уровне (рис. 13–14).

Рисунок 13

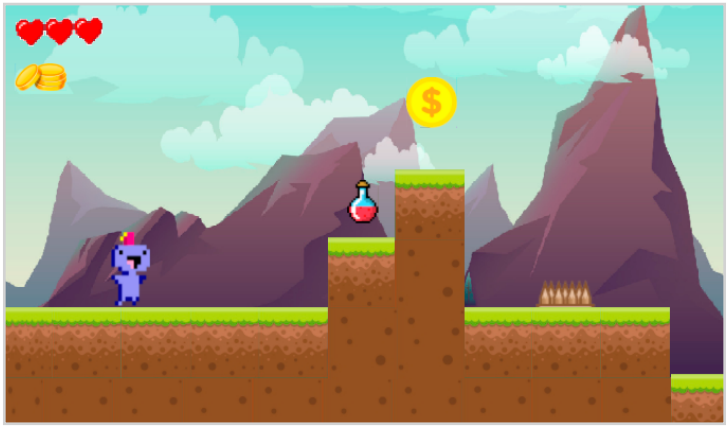

Рисунок 14

При воспроизведении игры, этот показатель будет распространяться на все уровни сразу. На старте нового уровня будет отображаться конечный результат предыдущего (рис. 15).

Рисунок 15

Теперь игра стала гораздо интереснее! У нас есть несколько уровней, бонусный «layout», чекпоинты и телепорты. В свободное время вы можете развивать игровые карты, добавлять больше препятствий и врагов, а также прописывать цепочки событий, тем самым увеличивая сложность уровней.

Посмотрите, какие бонусные уровни добавлены в популярные платформеры.

1. «Super Mario» (рис. 16).

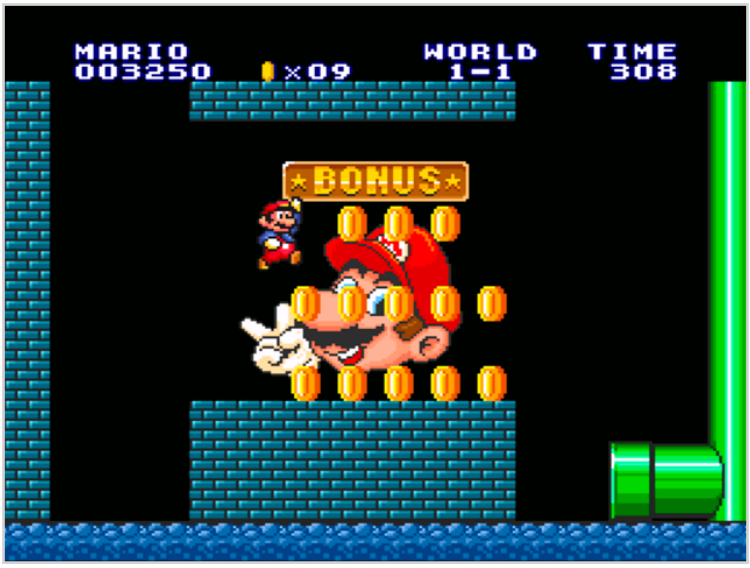

Рисунок 16

2. «Magic Rampage» (рис. 17).

Рисунок 17

3. «Super KOK» (рис. 18).

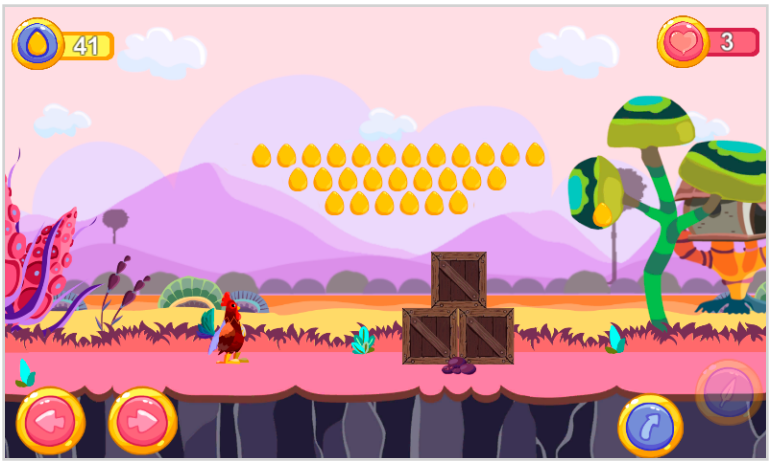

Рисунок 18

Урок 5

Анимация главного героя

➢ **ПЛАН РАБОТЫ.**

Создание покадровой анимации главного персонажа для состояний: спокойствие, бег, прыжок, падение. Применение параметров анимаций loop и ping pong. Создание подсобытий для анимации после приземления. Добавление плагина клавиатура для разворота персонажа. Анимация монет.

➢ **ДОПОЛНИТЕЛЬНО.**

Создание анимации противников.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Анимация главного героя

Давайте создадим новую интересную игру, в которой реализуем анимации различных состояний игрока!

Итак, сперва зададим параметры игровой карте, а также добавим «Tiled Background», платформы и главного героя (рис. 1).

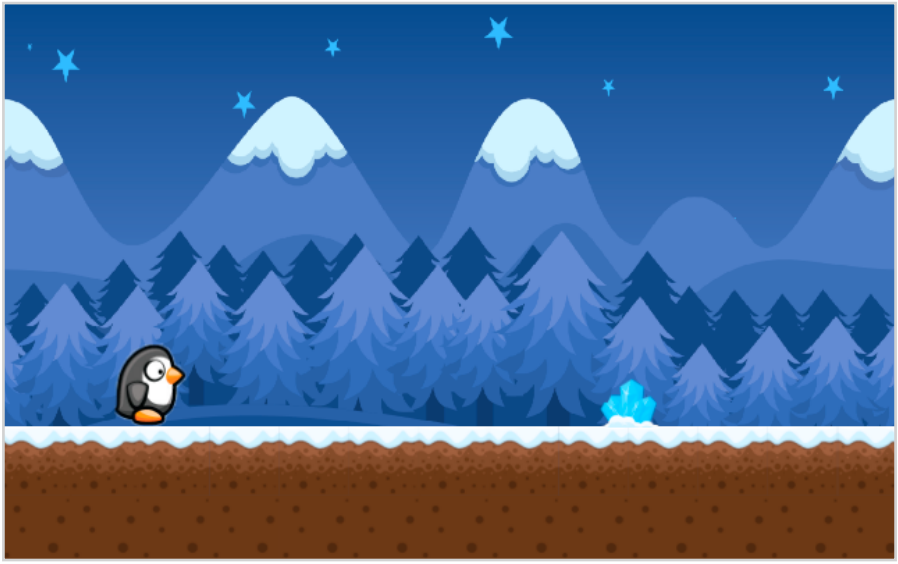

Рисунок 1

Рассмотрим, как можно анимировать игрока. Для этого дважды кликаем по нему, после чего откроется окно редактирования.

В окне «Animations» кликаем правой кнопкой мыши и выбираем «Add Animation». Задаем ей имя «Run» (рис. 2).

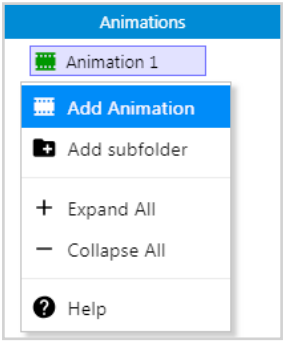

Рисунок 2

Анимация в «Construct 3» »— это поочередное воспроизведение нескольких фреймов. В окне редактирования персонажа анимацию можно нарисовать самостоятельно, или же добавить ее из папки, если у вас есть референсы. Кликаем правой кнопкой мыши на поле «Run Frames => Import Frames => From Files» (рис. 3).

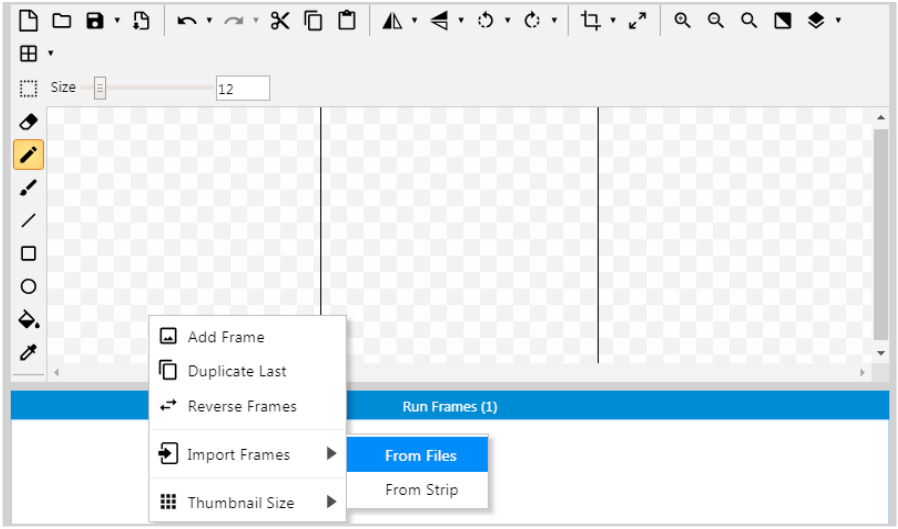

Рисунок 3

Выбираем в проводнике необходимые изображения и добавляем их. Все фреймы и их последовательность отображается в специальном поле (рис. 4).

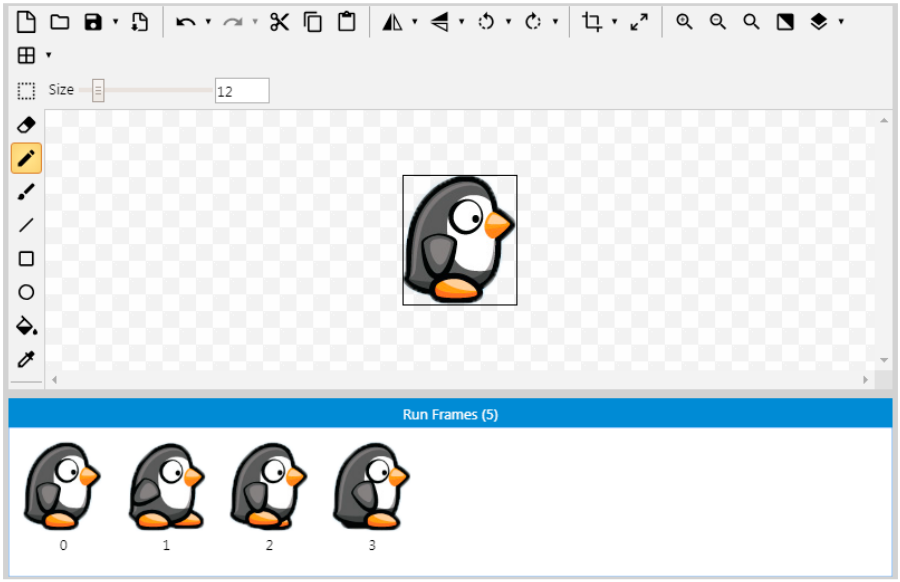

Рисунок 4

Закрываем окно редактирования персонажа и переходим к странице событий. Здесь нужно запрограммировать персонажа, чтобы при передвижении воспроизводилась созданная нами анимация.

Добавляем событие «Player => On moved». Затем сразу добавляем действие «Player => Set Animation». В окне параметров в кавычках указываем название анимации («Run»), которую мы создали в окне редактирования персонажа (рис. 5).

В цепочке событий действие выглядит достаточно просто (рис. 6).

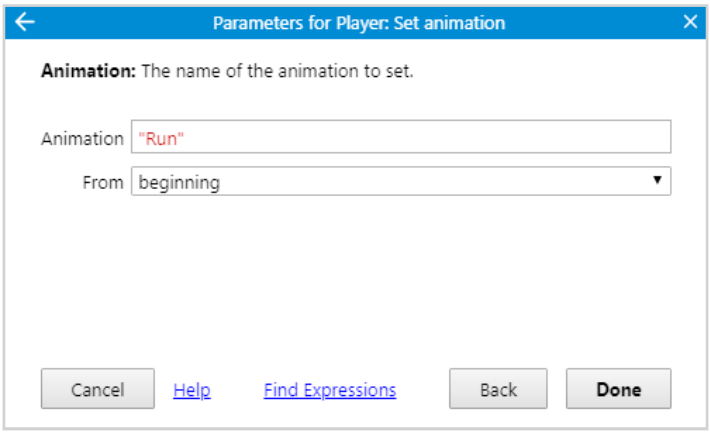

Рисунок 5

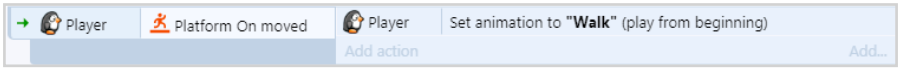

Рисунок 6

А при воспроизведении игры, персонаж уже анимирован и передвигается, перебирая лапками (рис. 7–8).

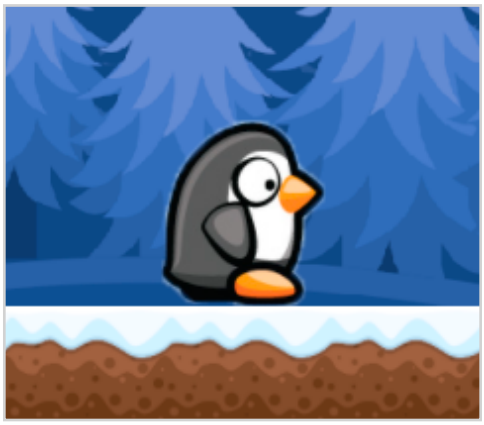

Рисунок 7

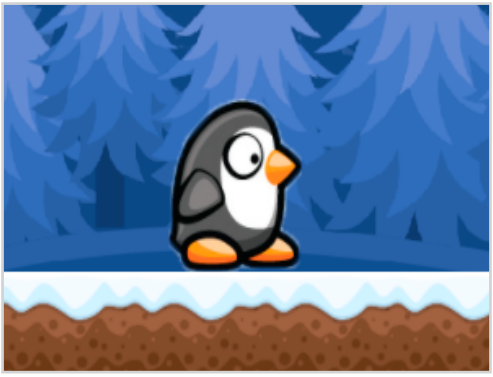

Рисунок 8

По такому же принципу добавляем анимацию прыжка и называем ее «Jump» (рис. 9).

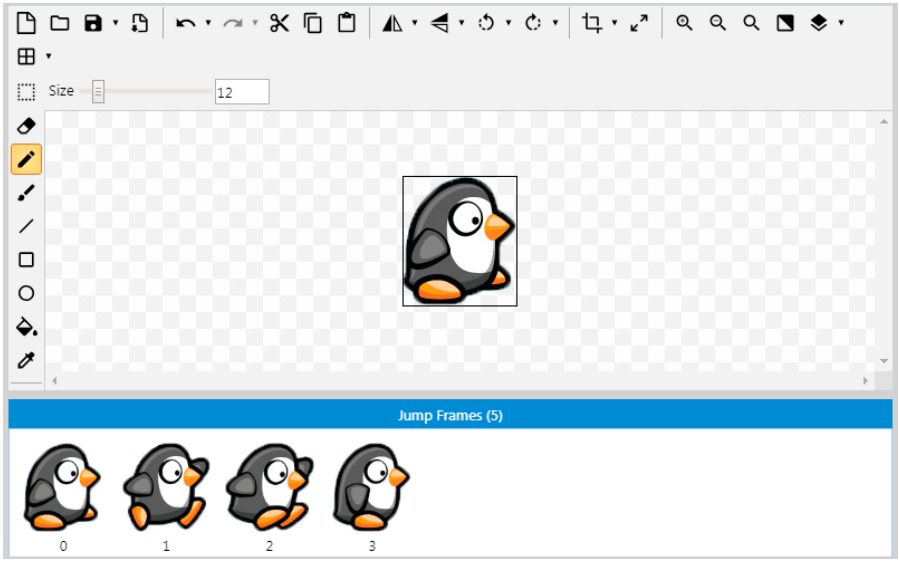

Рисунок 9

Снова прописываем событие. На этот раз «Player => On jump». Затем добавляем действие «Player => Set An-

imation». В окне параметров в кавычках указываем название анимации «Jump» (рис. 10).

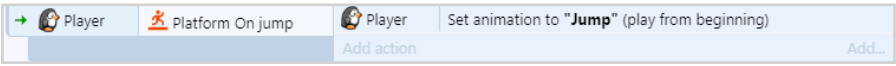

Рисунок 10

Прыжок игрока анимирован! При воспроизведении он расставляет крылья и двигает лапками (рис. 11).

Рисунок 11

Добавим еще одну анимацию — «Fall». Она будет воспроизводиться при падении пингвина (рис. 12).

Обязательно прописываем для падения событие «Player => On fall». Затем добавляем действие «Player => Set Animation». В окне параметров в кавычках указываем название анимации «Fall» (рис. 13).

Теперь при падении игрок немного будет отклоняться назад (рис. 14).

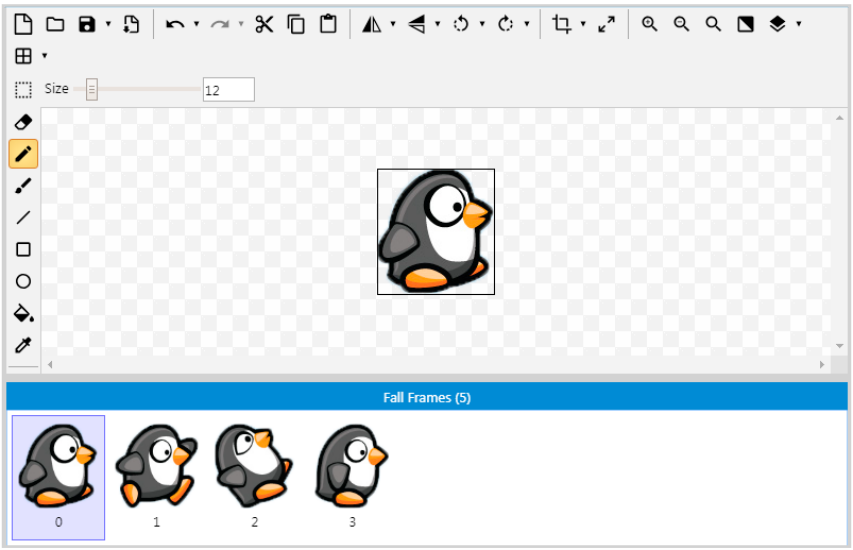

Рисунок 12

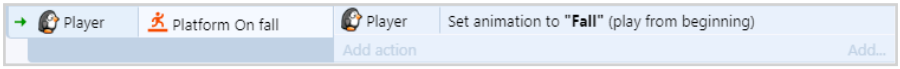

Рисунок 13

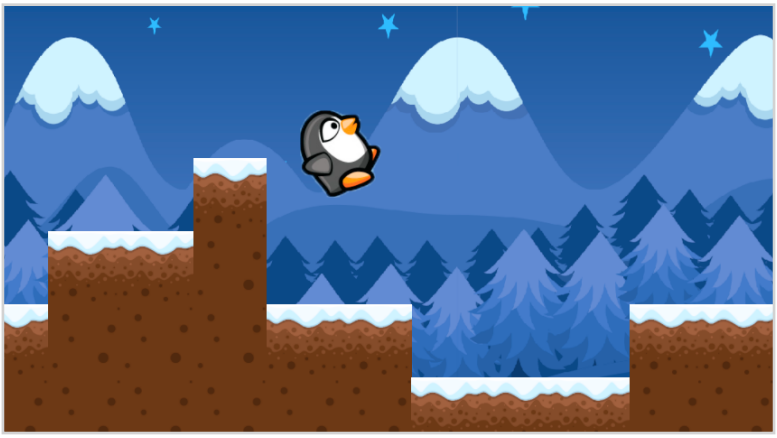

Рисунок 14

Также в списке событий следует добавить анимацию для статического положения при остановке персонажа: «Player => On stopped» и «Player => On landed». Для действий в этих ивентах выбираем анимацию «Stand» (это дефолтная анимация, которая создается любому спрайту при добавлении на игровую карту) (рис. 15).

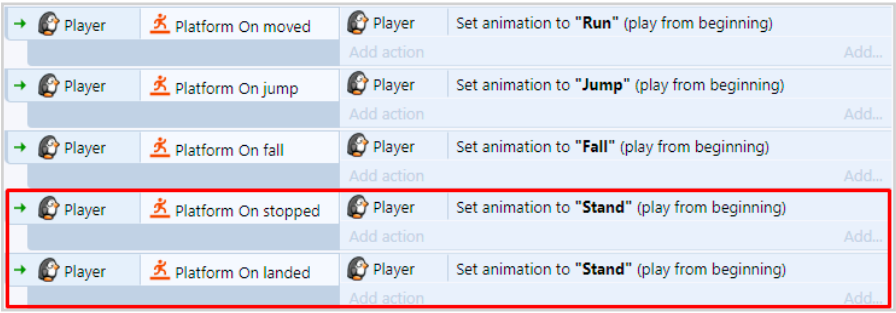

Рисунок 15

«Landed» — это анимация приземления игрока. Давайте немного усовершенствуем ее с помощью подсобытий. Кликаем правой кнопкой мыши и выбираем «Add sub-event» (рис. 16).

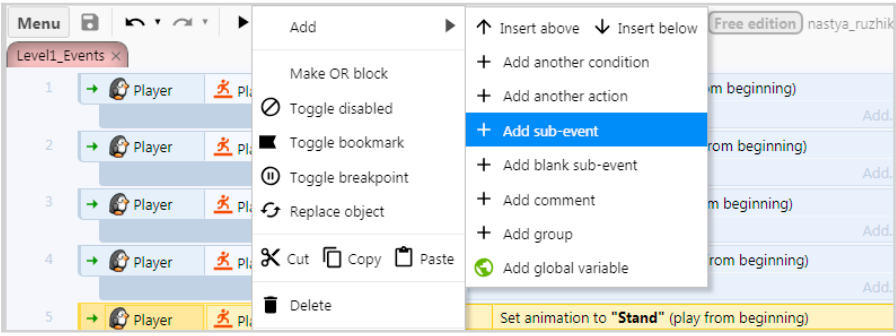

Рисунок 16

Первое подсобытие «Player => Platform is moving». Добавляем действие «Player => Set animation», затем в появившемся окне параметров вписываем анимацию «Run».

Создаем второе подсобытие «Player => Platform is moving». Инвертируем его, потому что действие, которое мы добавим, должно воспроизводиться, когда игрок статичен. Затем добавляем действие «Player => Set animation», в параметрах вписываем анимацию «Stand» (рис. 17).

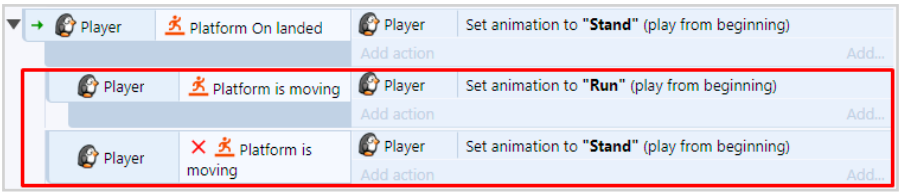

Рисунок 17

Соответственно, при приземлении игрока, если ему задали действие бега, проигрывается анимация «Run». А если после приземления игроку не задали действие, проигрывается анимация «Stand».

Это основная анимация персонажа. Ее значения сейчас выставлены по умолчанию, но их можно изменить или настроить. Для этого переходим в редактор игрока. Справа внизу есть панель «Animation Properties». Это все настройки, которые можно изменить в воспроизведении фреймов (рис. 18).

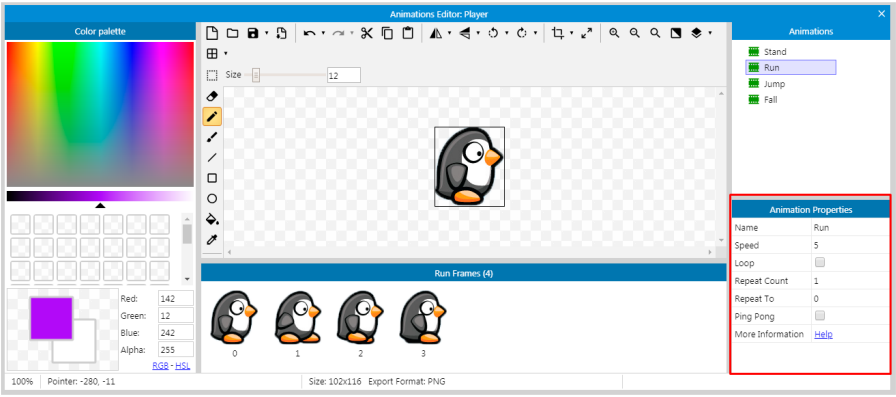

Рисунок 18

Например, выбираем анимацию «Run» (рис. 19).

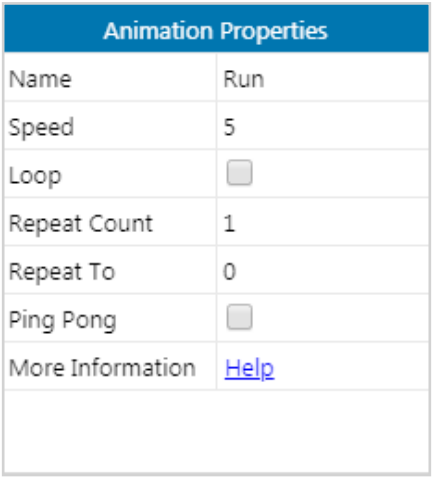

Рисунок 19

Здесь есть возможность увеличить/уменьшить скорость воспроизведения фреймов, а также изменить количество повторений анимации за одно воспроизведение.

Обратите внимание, есть две строки с чекбоксами. «Loop» и «Ping Pong».

- «Loop» — постоянное повторение анимации.
- «Ping Pong» — поочередное изменение анимации.

Теперь пришло время добавить поворот игрока. То есть, положение игрока должно соответствовать стороне, куда он направляется. Для этого в «Construct 3» используют плагин клавиатуры.

Кликаем по пустому пространству на игровой карте, выбираем «Insert new object», затем «Keyboard» (рис. 20).

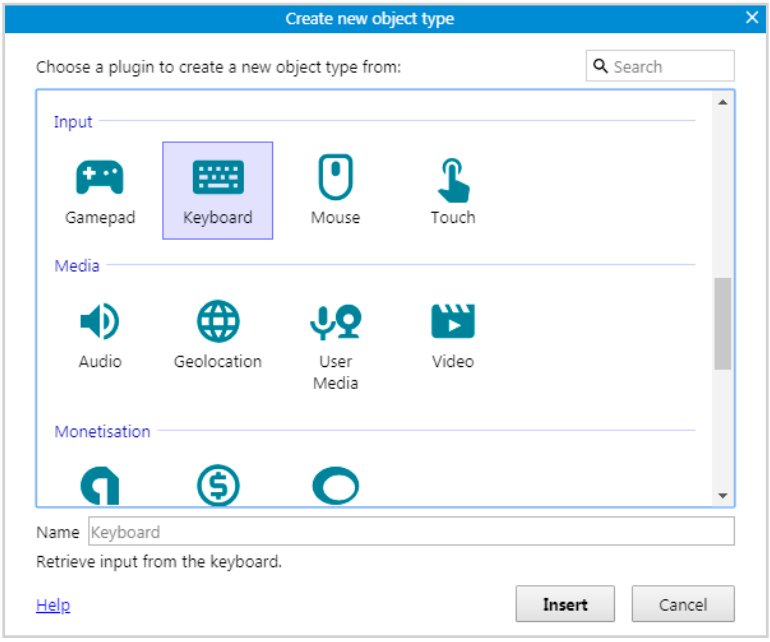

Рисунок 20

Он не появляется на рабочей сцене. Его можно найти на панели «Project» в папке «Object types» (рис. 21).

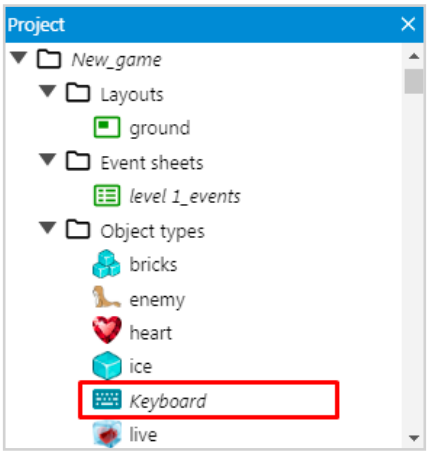

Рисунок 21

Когда плагин добавлен в проект, можно прописывать события с использованием клавиатуры. Переходим в «Event Sheets». «Add event => Keyboard => On key pressed». В появившемся окне выберите клавишу, затем нажмите ее на клавиатуре, и она автоматически отобразится в параметрах (рис. 22). К примеру, нажмите на клавиатуре левую стрелочку.

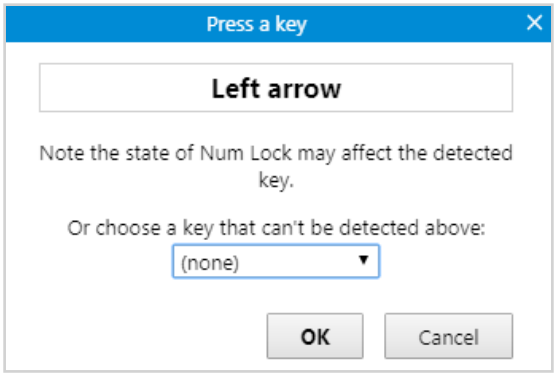

Рисунок 22

В цепочке событий это выглядит так (рис. 23):

Рисунок 23

Теперь добавим действие, которое будет запускаться при нажатии на левую стрелочку. «Add action => Player => Set mirrored» (рис. 24).

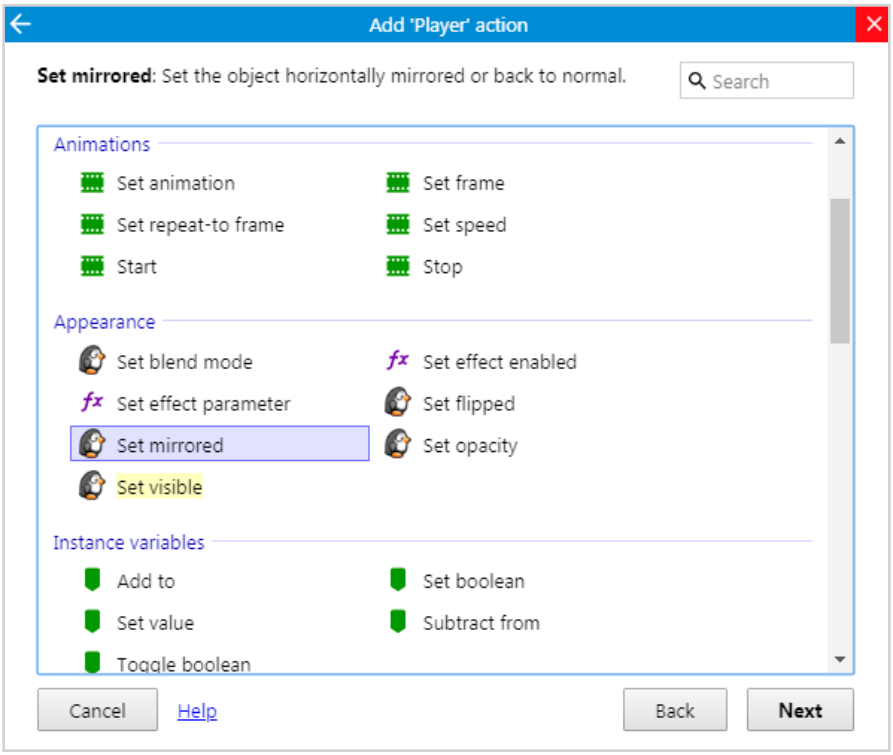

Рисунок 24

В окне с настройками устанавливаем позицию «Mirrored», которая и отзеркалит игрока при нажатии на левую стрелочку (рис. 25).

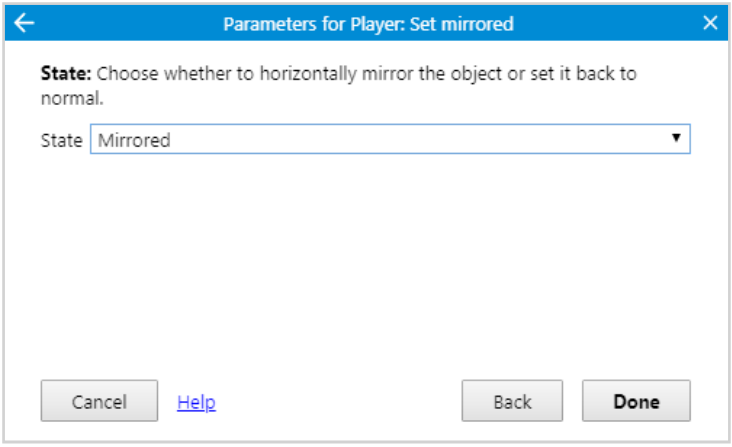

Рисунок 25

Теперь наш персонаж может разворачиваться! Но, снова нажав на правую стрелку, ничего не происходит. Игрок все еще смотрит влево (рис. 26).

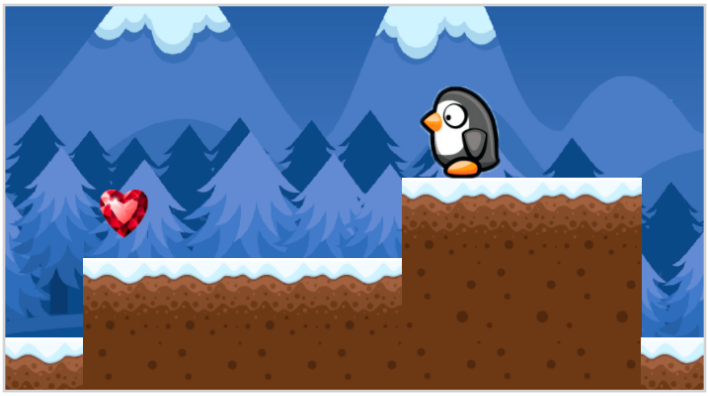

Рисунок 26

Нам необходимо добавить еще одно событие, чтобы персонаж мог разворачиваться в обе стороны. Второе событие должно быть противоположным первому: при

нажатии на правую стрелку, активируется параметр «Not Mirrored» (рис. 27).

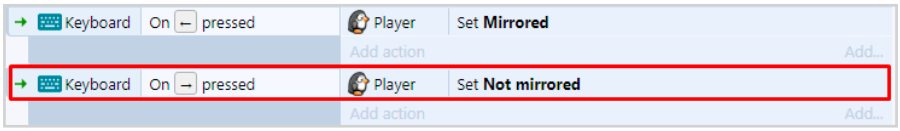

Рисунок 27

Теперь при перемещении наш персонаж разворачивается как в правую, так и в левую сторону (рис. 28–29).

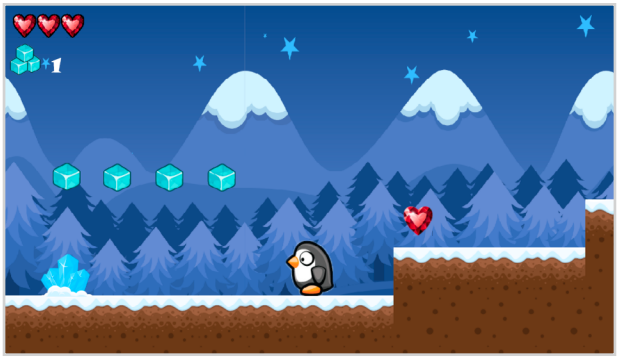

Рисунок 28

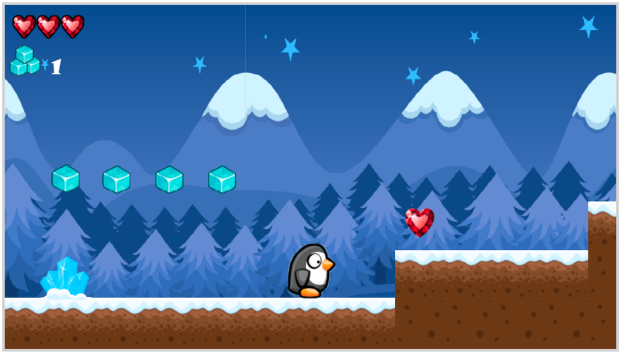

Рисунок 29

Давайте добавим еще немного анимации. На этот раз усовершенствуем льдинки, которые собирает пингвин. Переходим в окно редактирования элемента «ice». Добавляем новую анимацию и называем ее «turn_around» (рис. 30).

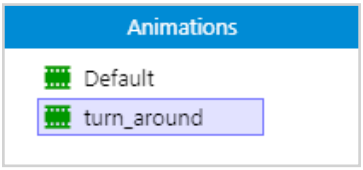

Рисунок 30

Добавляем все необходимые фреймы в окно «turn_around Frames» (рис. 31).

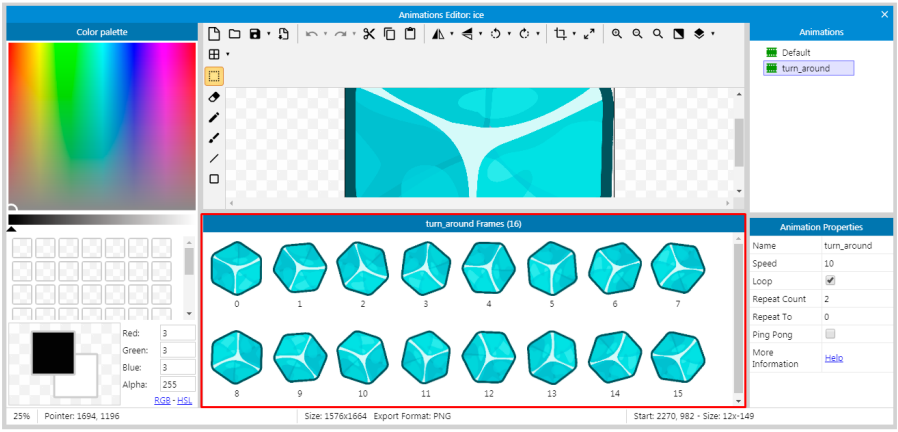

Рисунок 31

На панели «Animation Properties» увеличиваем скорость воспроизведения анимации, задаем количеству повторений значение 2 и ставим галочку возле функции безостановочного повторения «Loop» (рис. 32).

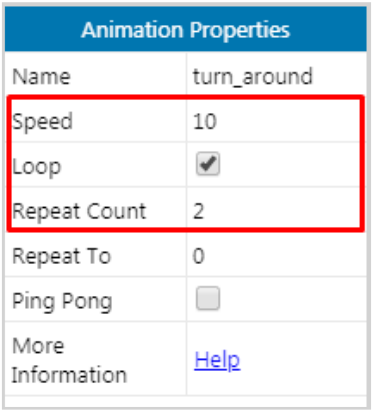

Рисунок 32

Возвращаемся к игровой карте, выделяем льдинку и на панели слева в разделе «Properties» выбираем созданную для этого объекта анимацию (рис. 33).

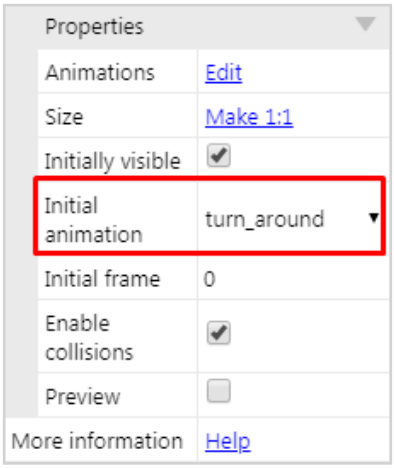

Рисунок 33

Теперь при воспроизведении игры льдинки будут вращаться по часовой стрелке (рис. 34).

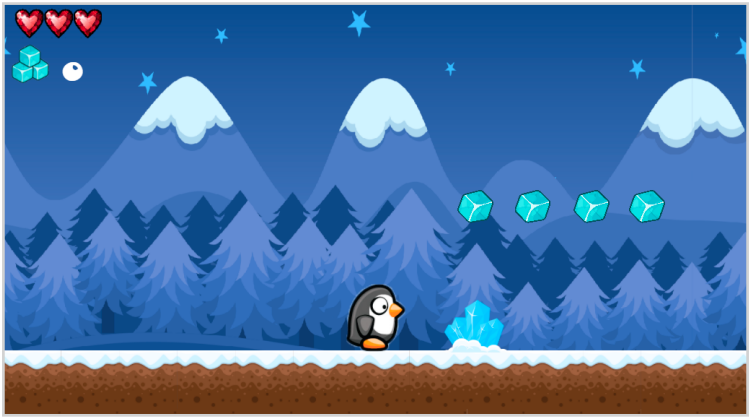

Рисунок 34

Мы создали игру с более сложной анимацией игрока! Однако вы можете добавить еще больше анимированных объектов в игровую карту, что сделает ее более «оживленной».

Посмотрите, какую интересную анимацию можно добавлять объектам в платформерах:

1. «Super Sonik» (рис. 35).

Рисунок 35

2. «Braid game» (рис. 36).

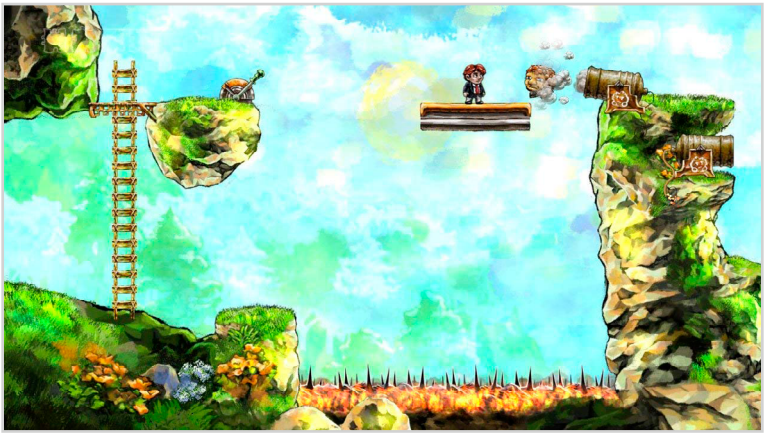

Рисунок 36

3. «Fez game» (рис. 37).

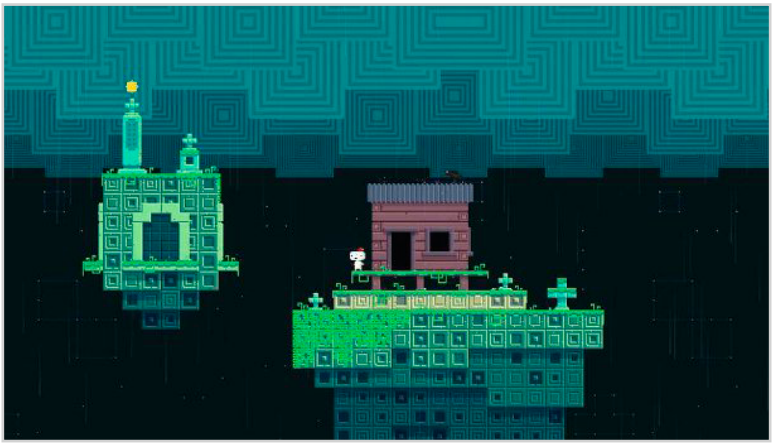

Рисунок 37

Урок 6

Стрельба, увеличение и уменьшение персонажа

➢ **ПЛАН РАБОТЫ.**

Добавление объекта для стрельбы персонажа при нажатии клавиши пробел. Программирование смерти противников при попадании пули, программирование исчезновение пули по таймеру. Добавление объектов для апгрейда параметров персонажей и изменения размера. Программирование анимации усиленного персонажа.

➢ **ДОПОЛНИТЕЛЬНО.**

Настройка Particles при стрельбе.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Стрельба, увеличение и уменьшение персонажа

Мы создаем крутую игру с переходами на разные уровни, анимацией и врагами.

Но давайте еще немного усовершенствуем один из уровней и добавим игроку возможность стрелять, чтобы уничтожать врагов. Для начала добавим еще одного врага, назовем его «octopus_enemy» и сделаем так, чтобы игрок смог его победить (рис. 1).

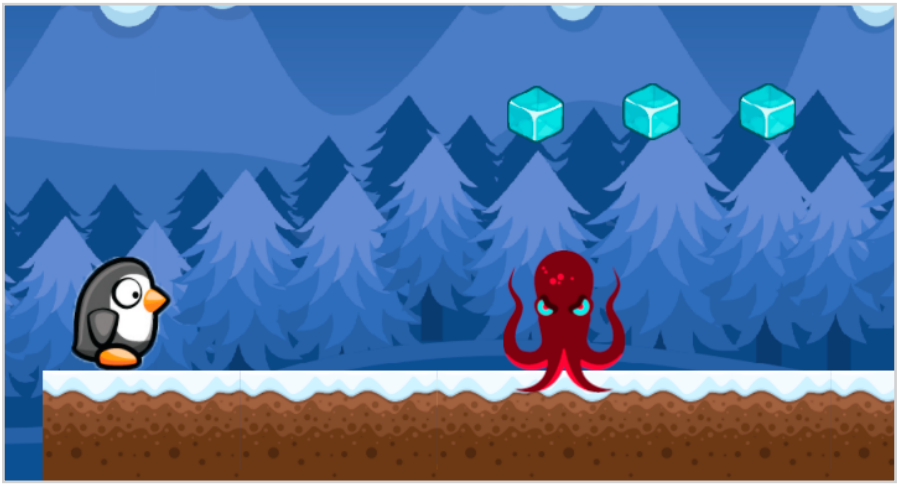

Рисунок 1

Для начала создадим снежный шар, которым пингвин будет атаковать противника. Добавляем новый спрайт и задаем ему поведение «Bullet» (рис. 2).

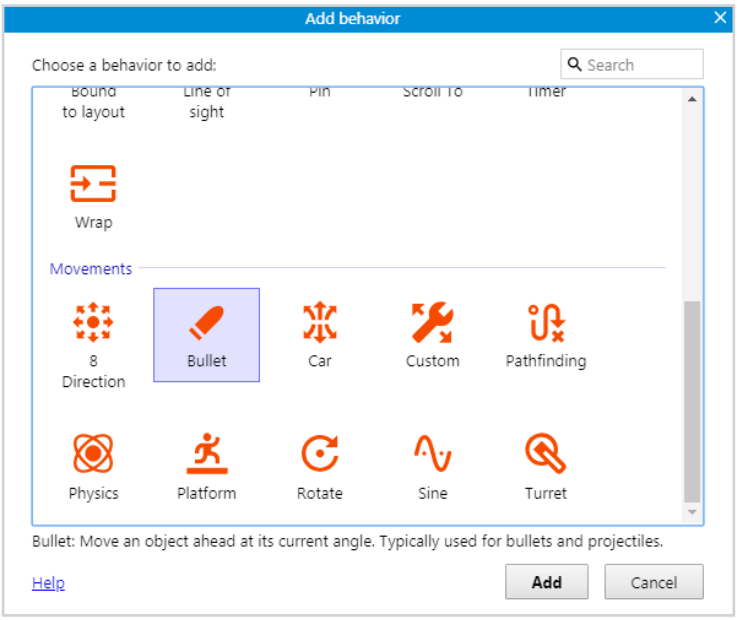

Рисунок 2

Заходим в окно редактирования персонажа. На панели слева кликаем правой кнопкой мыши и выбираем «Add a new image point» (рис. 3).

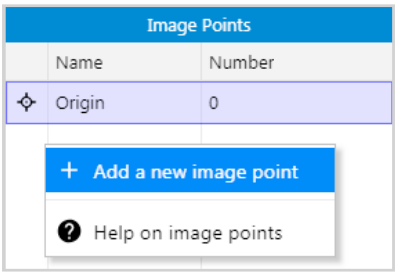

Рисунок 3

Перемещаем точку в зону, где будет появляться снежный шар при стрельбе (рис. 4).

Рисунок 4

Переходим в цепочку событий. Добавляем действие «Keyboard => On key pressed», затем нажимаем пробел, чтобы задать эту клавишу для выстрела (рис. 5).

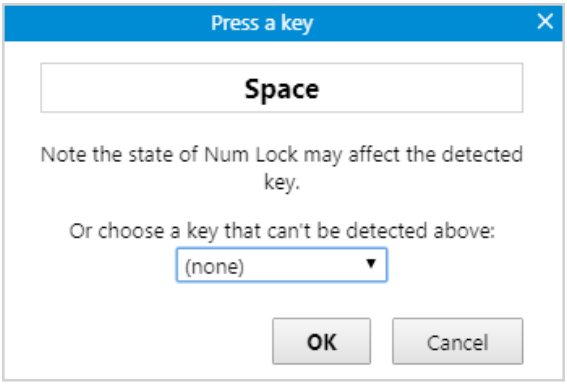

Рисунок 5

Добавим действие «System: Create object». Выбираем «snowball» и параметры его появления. В нашем случае это «Layer 0» и координаты точки, которую мы ранее создали (рис. 6).

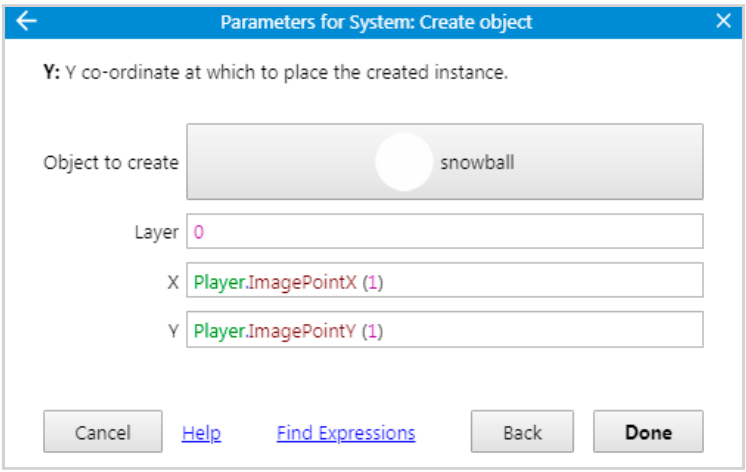

Рисунок 6

Теперь, при воспроизведении игры и нажатии на пробел, наш пингвин бросает во врагов снежки (рис. 7).

Рисунок 7

Что же эти снежки могут сделать врагу? Конечно же могут сразу его уничтожить. Для этого добавляем событие «Octopus_enemy => is overlapping another object => snow-

ball». И выбираем действие «Octopus_enemy => Destroy» (рис. 8).

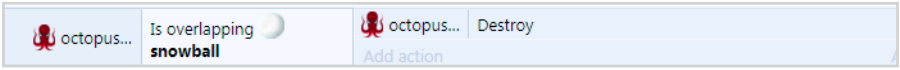

Рисунок 8

Давайте немного усложним игру: для уничтожения врага нужно попасть по нему несколько раз. С каждым попаданием он будет становиться прозрачнее, и будет уничтожен, когда совсем исчезнет.

Добавляем событие «Octopus_enemy => is overlapping with object => snowball». Затем действие, которое будет происходить при таком условии. В нашем случае, враг будет терять «Opacity». То есть, выбираем «Octopus_enemy => Set opacity». Кликаем по «Find Expression» и выбираем «Get the object's current opacity, from 0 to 100» (рис. 9).

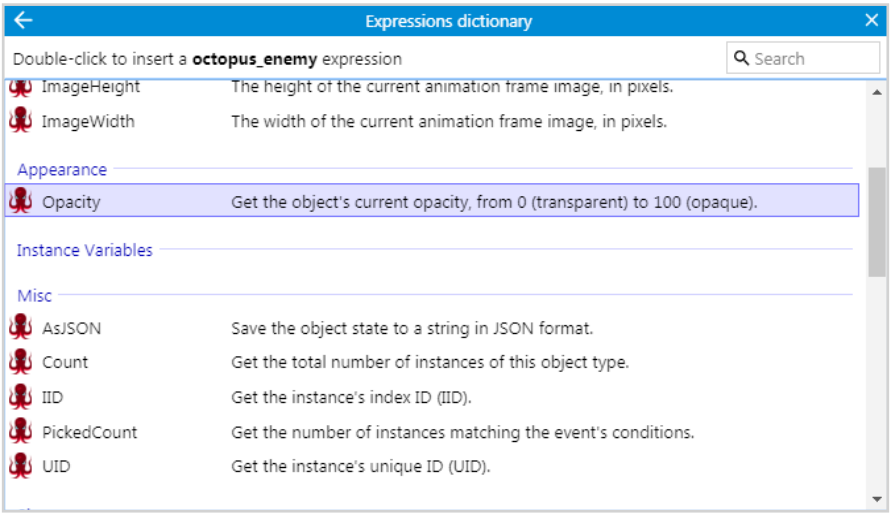

Рисунок 9

В окне параметров появится значение, к которому нужно дописать, например, –3. Это значит, что при попадании снежного шара во врага, он будет терять 5 значений насыщенности (рис. 10).

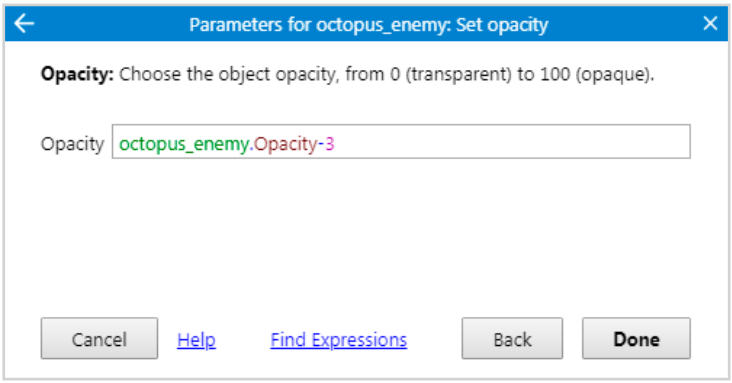

Рисунок 10

Заданные значения будут отображены в цепочке событий (рис. 11).

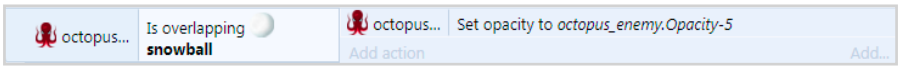

Рисунок 11

Но, если мы хотим, чтобы враг не уничтожался сразу, удаляем событие, в котором указано, что, при попадании снежка, осьминог исчезает. Вместо него следует прописать, что враг умрет, когда его видимость («Opacity») будет ровна 0.

Добавляем событие «Octopus_enemy => Compare opacity => ≤ Less or equal 0». Действие остается прежним: «Octopus_enemy => Destroy» (рис. 12).

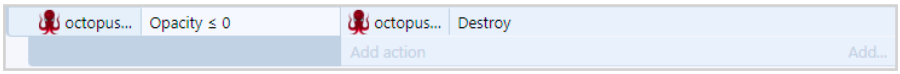

Рисунок 12

Теперь за несколько ударов игрок может уничтожить врага. В нашем случае, осьминог исчезает за 4 удара (рис. 13).

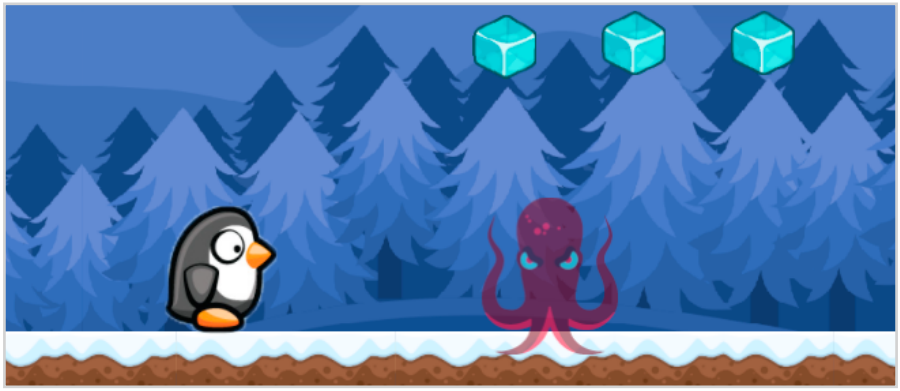

Рисунок 13

Теперь добавим событие, чтобы, при соприкосновении с врагом, игрок получал урон. На предыдущих уроках с патрулирующим врагом мы уже рассматривали, как реализовать это. Повторим событие по тому же принципу (рис. 14).

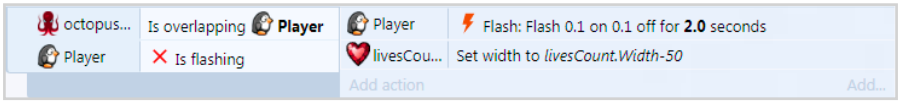

Рисунок 14

Осталось добавить цепочке событий только один нюанс. Сейчас игрок стреляет только вправо, и, при по-

вороте, снежные шары улетают туда же. Необходимо развернуть выстрел. Сначала создадим еще один объект «snowball2» и задаем ему поведение «Bullet».

Добавляем новое событие «Keyboard => Key is down => Space». После этого добавляем «Add new condition». Выбираем «Player => is mirrored». То есть, при нажатии пробела, наш персонаж сможет стрелять в обе стороны (рис. 15).

Рисунок 15

Добавляем «sub-event». И к нему прикрепим несколько действий:

1. «Player => Spawn another object». Выбираем «snowball2», оставляем «level=0» и «image point=1» (то есть, игрок будет создавать второй снежный шар на слое 0 с точкой появления 1, которую мы создали).
2. «Snowball2 => Set angel => 180» (разворачиваем угол второго снежного шара на 180°).
3. «Snowball => Destroy» (уничтожаем первый снежный шар, чтобы он не вылетал вместе со вторым) (рис. 16).

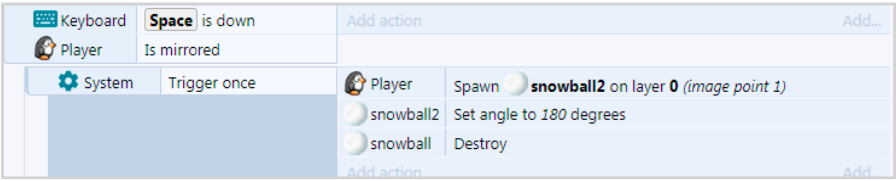

Рисунок 16

Давайте немного усовершенствуем стрельбу и добавим исчезновение снежного шара по таймеру. Для этого

на игровой карте выбираем объект снежка и добавляем ему поведение «Fade» (рис. 17).

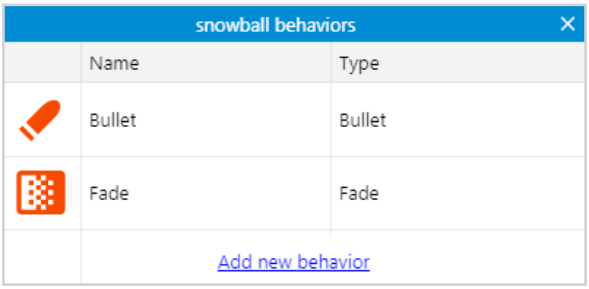

Рисунок 17

Теперь на вкладке «Properties» справа нужно выставить необходимые параметры исчезновения и уничтожения снежного шара.

В нашем случаем необходимо установить значение для «Fade out time». Пускай наш снежный шар исчезает через 1 секунду. Ставим отметку «Destroy» и «Enabled» (рис. 18).

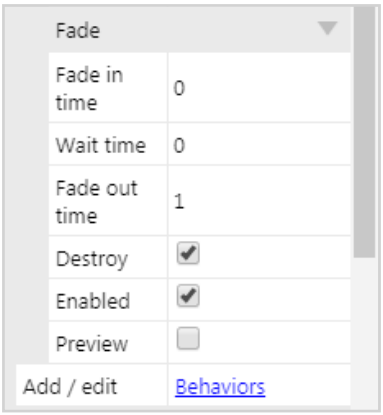

Рисунок 18

Теперь снежок исчезает и уничтожается через секунду после того, как пингвин его бросает.

В предыдущих уроках мы уже добавляли объекты, которые изменяли размер нашего игрока. Давайте разместим и в этой игровой карте такие же объекты, но в виде рыбок (рис. 19).

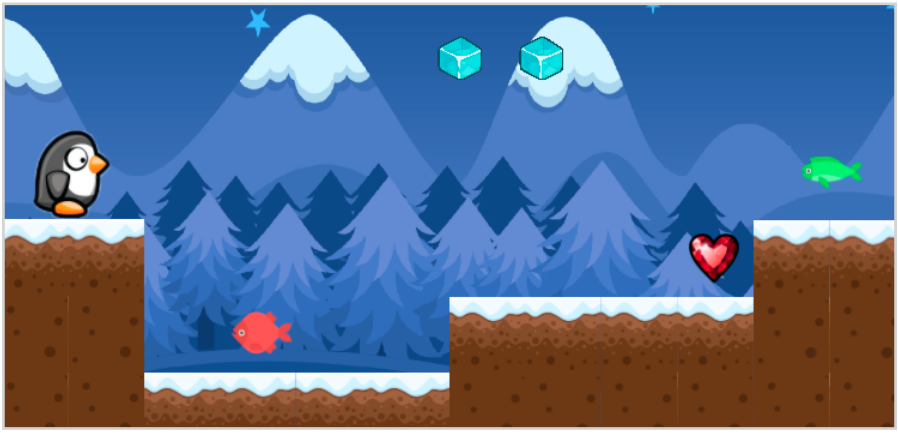

Рисунок 19

Переходим в цепочку событий и добавляем увеличение размера игрока после поедания красной рыбки, и уменьшение — после зеленой. Не забываем, что рыбки должны исчезать после того, как игрок меняет размер (рис. 20).

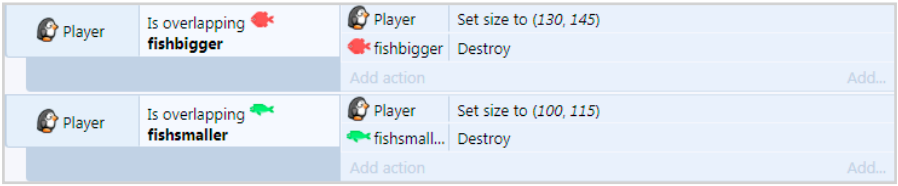

Рисунок 20

Также к таким объектам можно добавлять увеличение скорости, изменение силы тяжести и т.д (рис. 21).

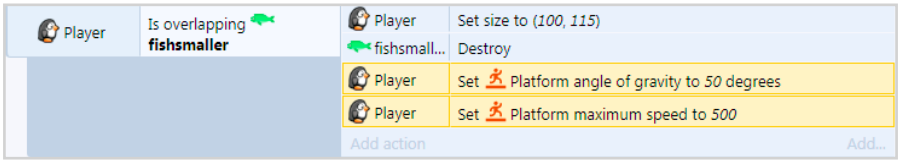

Рисунок 21

Давайте теперь рассмотрим еще один вариант программирования анимации усиленного персонажа. Например, добавим элемент, после соприкосновения с которым у нашего персонажа увеличится сила прыжка. Пускай это будет мороженное (рис. 22).

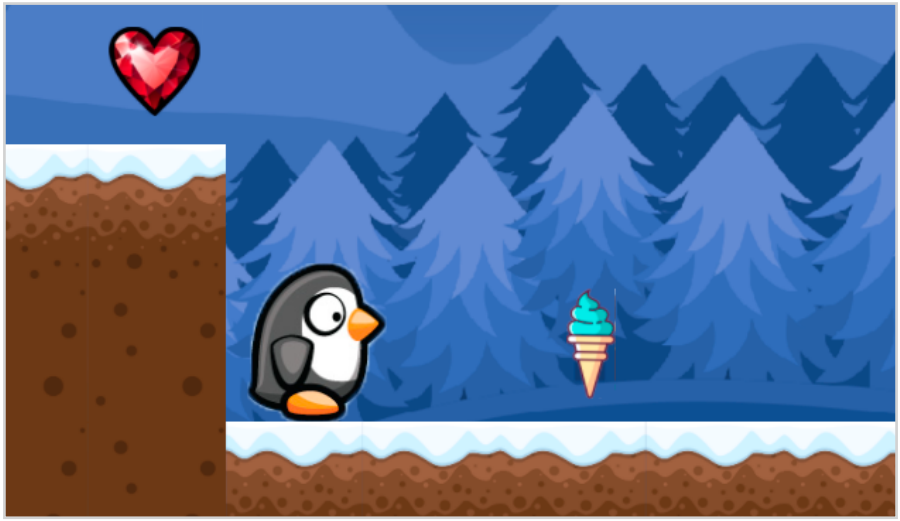

Рисунок 22

В цепочке событий добавляем «Add event => Player => Is overlapping another object => icecream». Прописываем

действие, чтобы установить силу прыжка 1000, а затем «icecream => Destroy» (рис. 23).

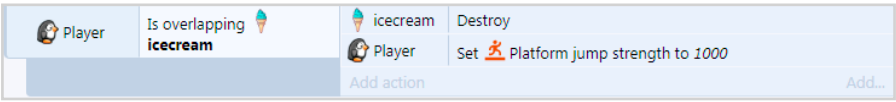

Рисунок 23

При стрельбе в «Construct 3» также можно использовать настройку «Particles», которая создает визуальный эффект хаотичного перемещения маленьких изображений.

Давайте рассмотрим, как она выглядит, и как ее можно применять в игре. Добавляем новый элемент «Particles» на игровую карту (рис. 24).

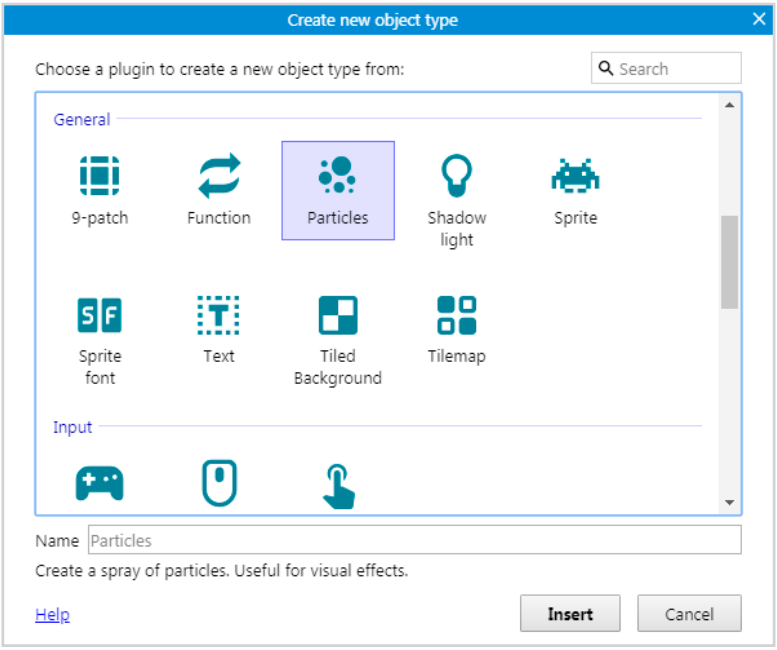

Рисунок 24

Появится окно редактирования изображения. Здесь необходимо нарисовать фигуру, которая будет применяться. У нас это может быть обычная белая точка (рис. 25).

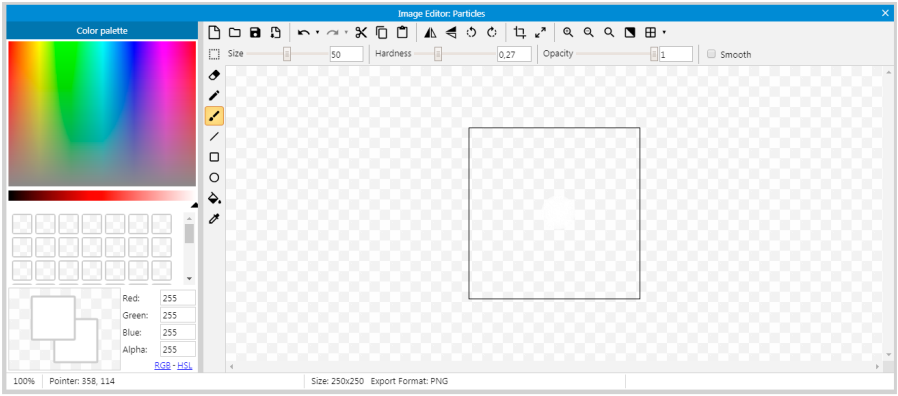

Рисунок 25

После закрытия окна на рабочем пространстве появится угол с нарисованным нами объектом внутри (рис. 26).

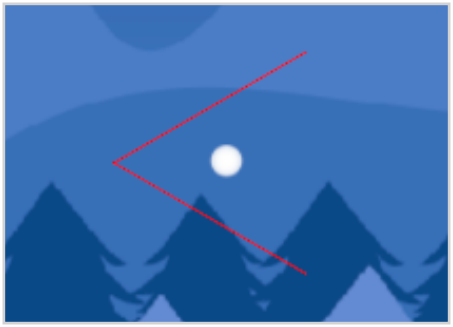

Рисунок 26

Затем выставляем необходимые параметры на панели «Properties», «Type=One-shot», чтобы, при попадании во

врага, эффект срабатывал только один раз, а не воспроизводился постоянно (рис. 27).

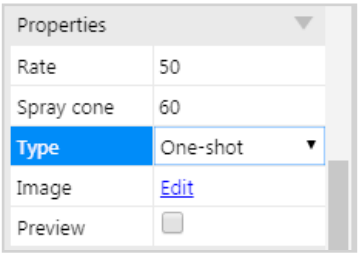

Рисунок 27

Затем выставляем «Spray cone» на 360. Это значит, что распыление будет распространяться на 360°. На рабочей сцене пропадет угол возле нарисованной точки и появится линия. Также можно уменьшить «Rate» до 30 (рис. 28–29).

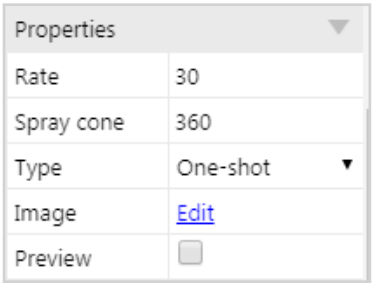

Рисунок 28

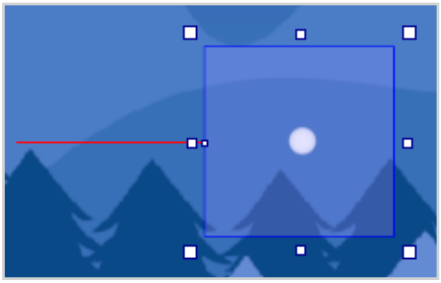

Рисунок 28

Если сейчас воспроизвести игру, то в том месте, где находится элемент «Particles», вы увидите небольшой взрыв из белых точек (рис. 30).

Немного уменьшим скорость взрыва («Speed») и ускорение («Acceleration») до значения 100 (рис. 31).

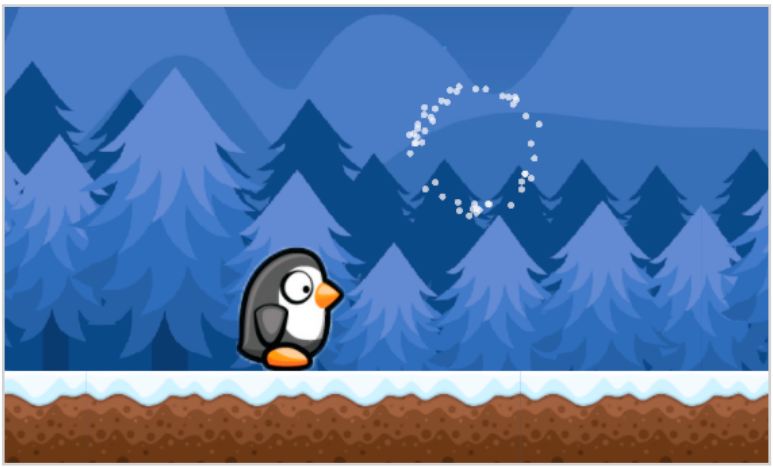

Рисунок 30

Initial particle properties	
Speed	100
Size	32
Opacity	100
Grow rate	0
X randomiser	0
Y randomiser	0
Speed randomiser	0
Size randomiser	0
Grow rate randomiser	0
Particle lifetime properties	
Acceleration	100
Gravity	0
Angle randomiser	0

Рисунок 31

Теперь переходим в цепочку событий. Выбираем «Snowball => On created» (рис. 32).

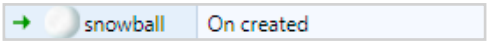

Рисунок 32

И добавляем действие, которое будет происходить в этот момент. «Snowball => spawn another object => particles» (рис. 33).

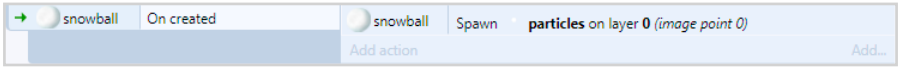

Рисунок 33

По такому же принципу добавляем эффект «Particles для snowball2» (рис. 34).

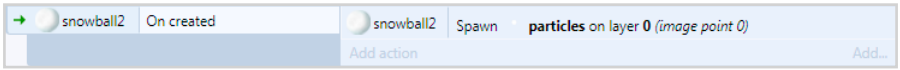

Рисунок 34

Теперь, при вылете снежного шара, будет воспроизводиться эффект «Particle», и создастся впечатление, что снежок собирается из множества снежинок, а потом летит в сторону врага (рис. 35).

Таким образом мы максимально усложнили анимацию игры, усовершенствовали игрока, добавив ему возможность сражаться с врагами.

В свободное время постарайтесь добавить больше элементов, которые помогут игроку победить всех врагов и добраться до следующего уровня.

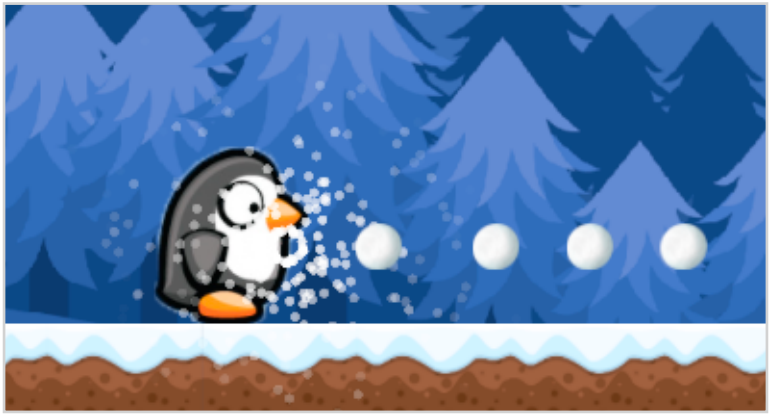

Рисунок 35

Если вы хотите усложнить уровень, добавьте больше врагов, в также увеличьте количество способов их уничтожения.

Для вдохновения посмотрите, какие крутые элементы стрельбы и взрывов можно добавить в игру.

1. «Metal Shooter» (рис. 36).

Рисунок 36

2. «Velocity 2X» (рис. 37).

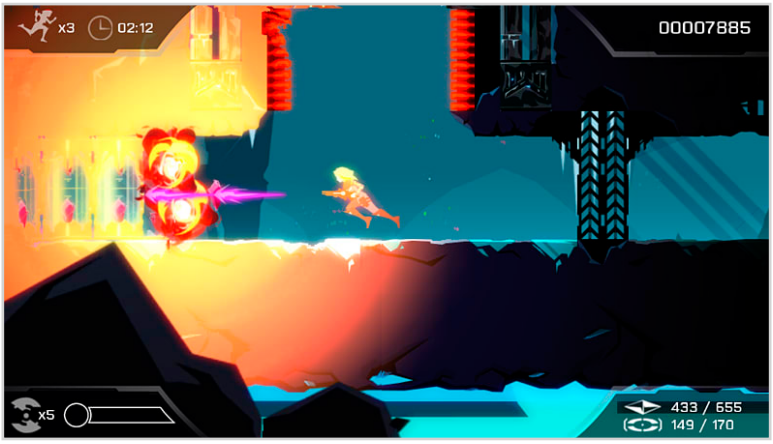

Рисунок 37

3. «20XX» (рис. 38).

Рисунок 38

Урок 7

Создание финального супербосса

➢ **ПЛАН РАБОТЫ.**

Создание нового layout и event sheet для финального уровня. Добавление персонажу и противнику полосок жизней, программирование урона персонажа и босса. Создание случайных скоростей передвижения и выстрела для босса. Программирование телепортации относительно главного героя.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Создание финального супербосса

У нас уже есть несколько уровней новой игры. Давайте теперь добавим финальный игровой «layout», где состоится схватка игрока с супербоссом.

Создаем новый «layout», добавляем фон, игрока и врага (рис. 1).

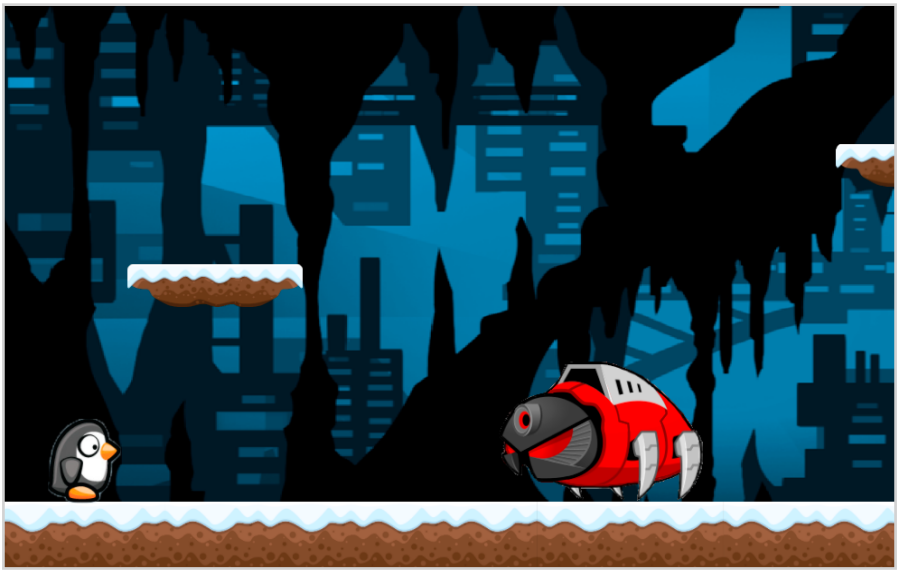

Рисунок 1

Добавим игроку анимацию. Можем продублировать ее с предыдущего уровня. Теперь пингвин анимирован во время бега, прыжков и падений (рис. 2).

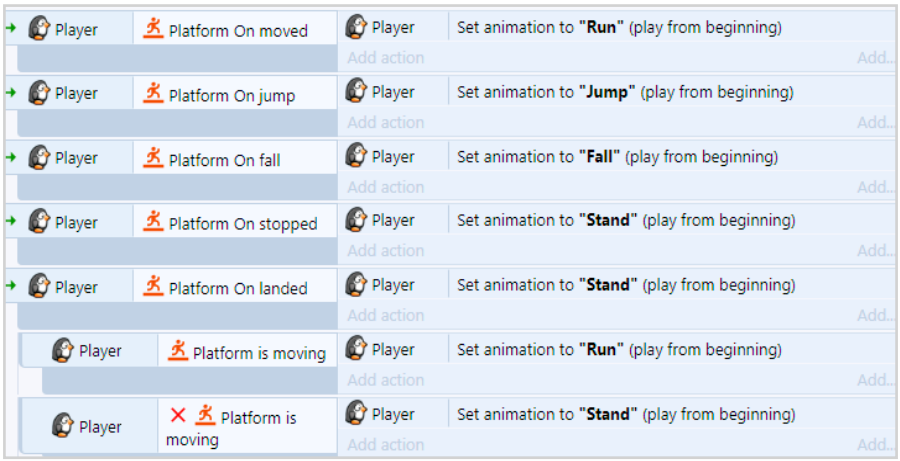

Рисунок 2

Продублируем изменение позиции персонажа при нажатии стрелочек на клавиатуре (рис. 3).

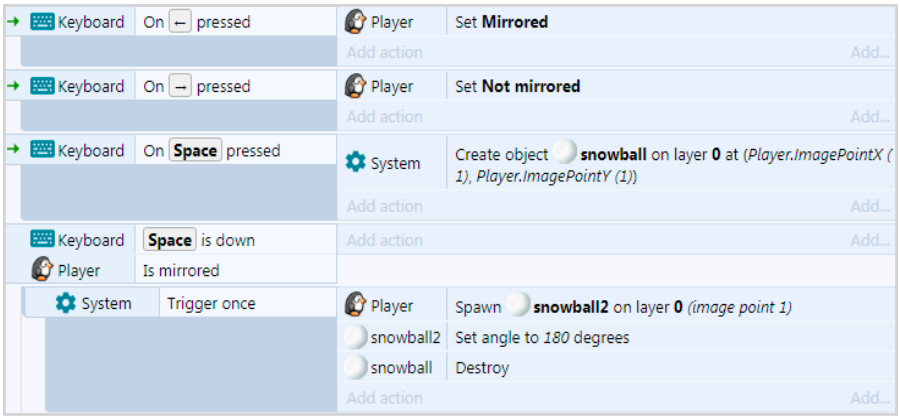

Рисунок 3

Чтобы эти элементы в цепочке событий вам не мешали, можно добавить их в папку, а затем свернуть ее (рис. 4).

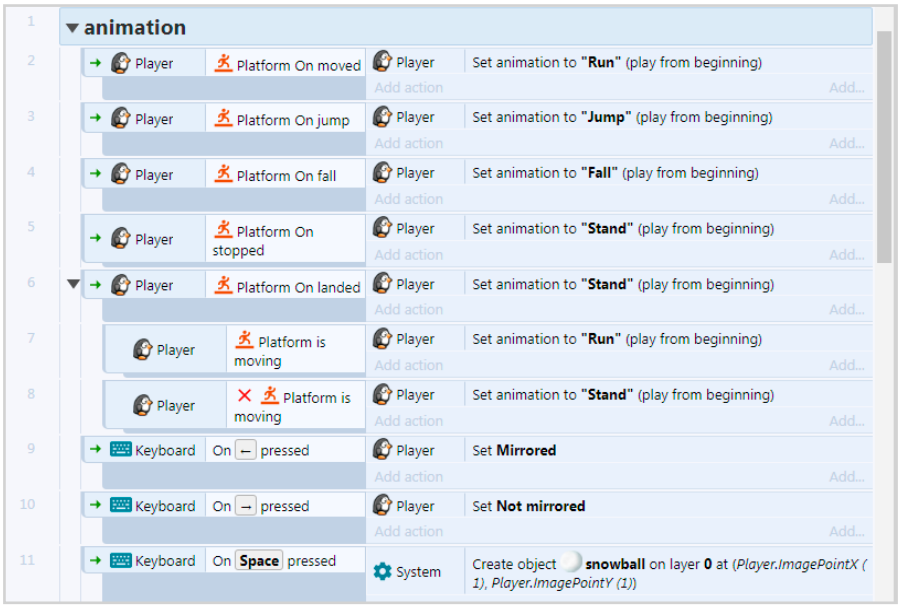

Рисунок 4

Теперь создадим спрайт в виде линии, который будет отображать здоровье игрока. Поверх этой линии можно добавить спрайт, который будет обрамлением для «health-bar» (рис. 5).

Рисунок 5

Переходим к цепочке событий и добавляем глобальную переменную, которую называем «health», после чего задаем ей постоянную длину 100 (рис. 6).

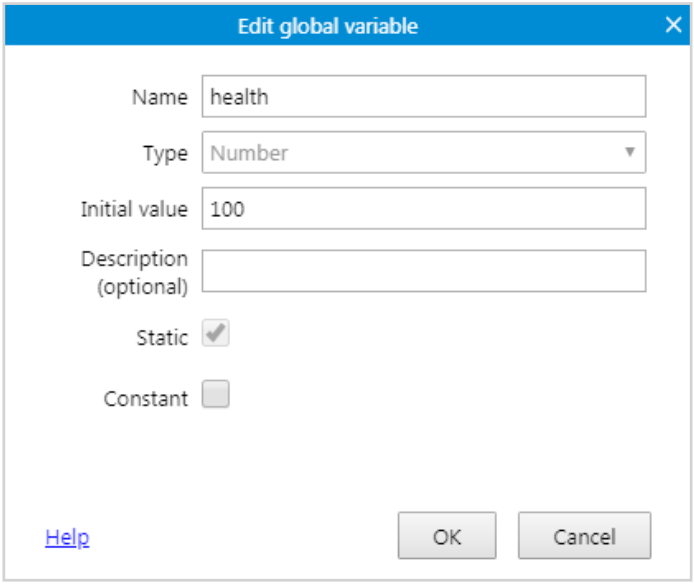

Рисунок 6

Глобальная переменная отображается вверху цепочки событий (рис. 7).

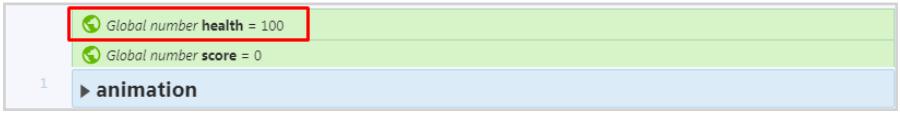

Рисунок 7

После этого нам необходимо добавить события, чтобы, при соприкосновении со врагом, игрок вспыхивал на несколько секунд и терял 5 пунктов «healthbar». «Add event => Player => is overlapping another object => enemy». Сразу же можем добавить «another condition => Player => is flashing» и инвертировать событие. Добавляем действие «Player => Flash => 2.0sec» (рис. 8).

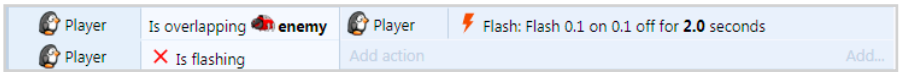

Рисунок 8

Теперь добавим действие, чтобы игрок терял жизнь. «healthbar => set width». Кликаем по «Find Expression», выбираем «healthbar» и вписываем –5 (рис. 9).

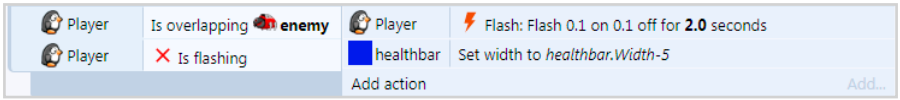

Рисунок 9

Не забудьте добавить условие уничтожения игрока. В случае, если «healthbar» будет равен 0 или меньше, уровень перезапускается (рис. 10).

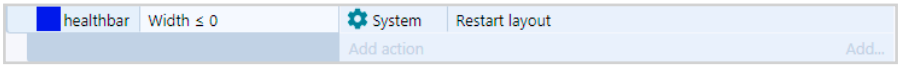

Рисунок 10

По тому же принципу создаем «healthbar_enemy» для врага (рис. 11).

Рисунок 11

Дублируем действия, только теперь по отношении к врагу. Однако в качестве объекта, который наносит ему урон, выбираем снежные шары (рис. 12).

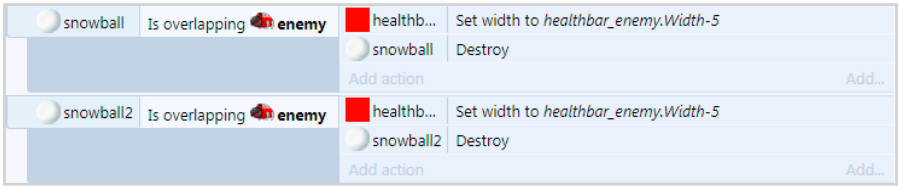

Рисунок 12

Как и в случае с игроком, враг тоже будет уничтожен, если его жизнь будет ≤0. Добавим это в цепочку событий (рис. 13).

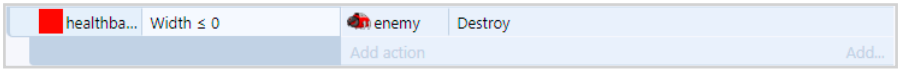

Рисунок 13

Запрограммируем выстрел для босса. Сначала добавим спрайт с пулей и зададим ему поведение «Bullet» (рис. 14).

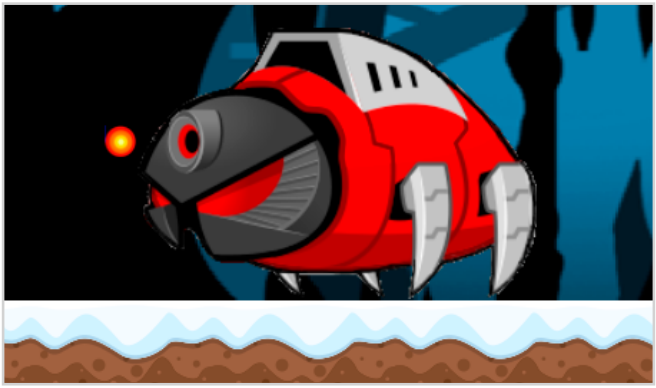

Рисунок 14

Заходим в редактор изображения врага, где добавляем еще одну «Image Point». Выставляем ее в том месте, где будет появляться пуля (рис. 15).

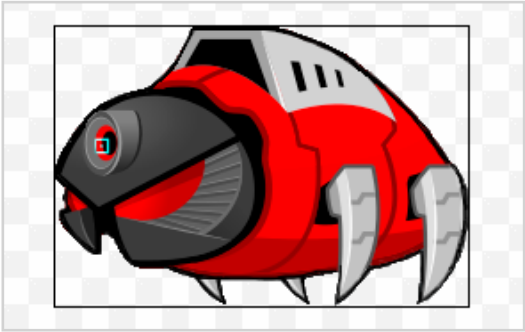

Рисунок 15

Переходим к цепочке событий. «Add event => System => Every X seconds => 0,7». С помощью этого события, каждые 0,7 секунды будет появляться новая пуля (рис. 16).

Рисунок 16

Теперь нужно запрограммировать выстрел так, чтобы пуля летела в игрока, а не просто прямо. Для этого применяем действие «enemy_bullet => set angle toward position» — выставляем координаты «Player.X» и «Player.Y» (рис. 17).

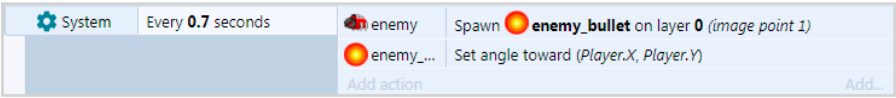

Рисунок 17

Также добавим событие уничтожения пуль при соприкосновении с платформами (рис. 18).

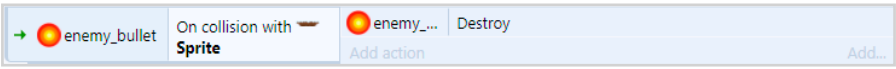

Рисунок 18

Давайте усложним схватку с боссом, сделав его еще и преследующим врагом.

Добавляем событие «System => Compare two values». В появившемся окне выставляем следующие значения:

1. «First value=Player.X».
2. «Comparison=<Less than».
3. «Second value=follow.X» (рис. 19).

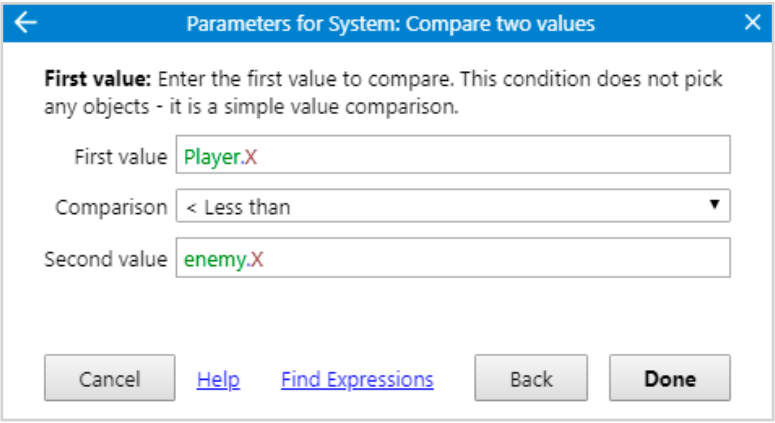

Рисунок 19

Теперь добавим действие. «Follow => Simulate control => Left». То есть, когда позиция врага меньше, чем игрока, будет происходить симуляция нажатия клавиши влево, и первый будет догонять второго (рис. 20).

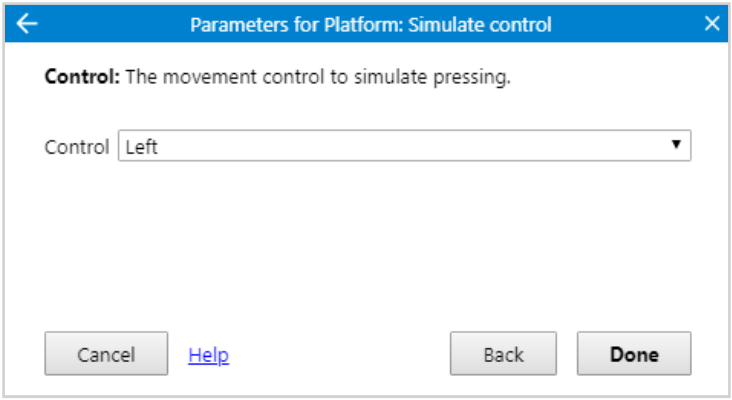

Рисунок 20

Копируем событие и выставляем противоположные значения. Теперь наш враг постоянно преследует игрока (рис. 21).

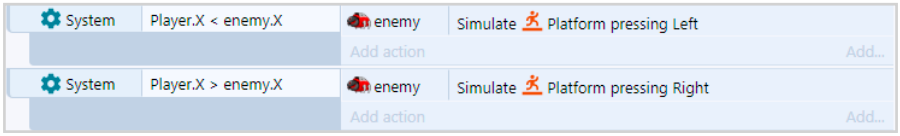

Рисунок 21

Давайте добавим в события разворот врага, чтобы он мог двигаться в обе стороны. Мы уже делали такое ранее. Применяем опцию «Set Mirrored» (рис. 22).

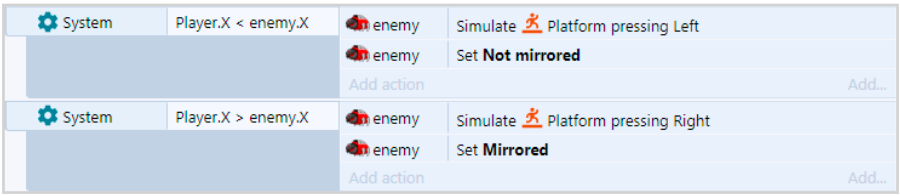

Рисунок 22

Еще можно прописать в цепочке событий ускорение врага в том случае, если у него, например, останется только половина жизни.

Для этого добавляем «Add event => healthbar_enemy => Compare width». Задаем в окне параметров значение «≤ 50» (рис. 23).

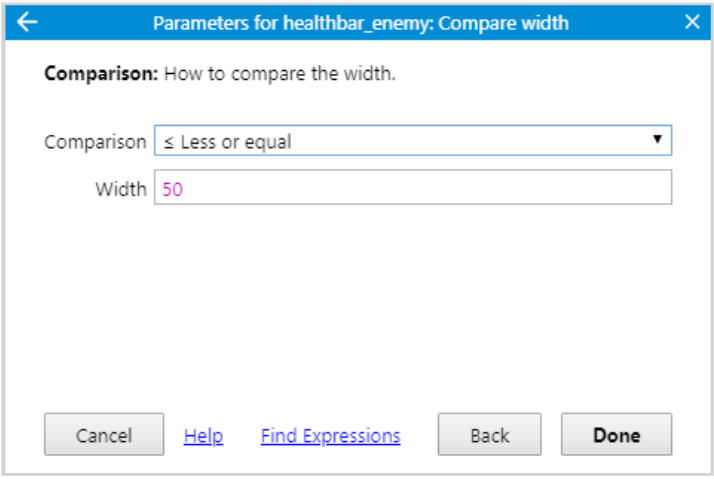

Рисунок 23

Затем выбираем действие увеличения скорости. «Enemy => Set max speed => 250» (рис. 24).

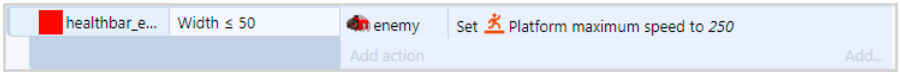

Рисунок 24

Или же, чтобы максимально усложнить этот уровень, можно добавить рандомную скорость. В таком случае, программа будет самостоятельно выбирать случайное значение в пределах указанного диапазона.

Для этого добавим новое событие: «Add event => System => Every X seconds». Выставим 3 секунды. То есть, каждые 3 секунды скорость врага будет меняться (рис. 25).

Рисунок 25

Теперь в действии указываем, что будет происходить при таких условиях. «Enemy => Set max speed». В появившемся окне параметров нужно указать диапазон допустимой скорости. Прописываем «random(150,300)+50». Это значит, что враг будет передвигаться с рандомной скоростью от 150 до 300 (рис. 26).

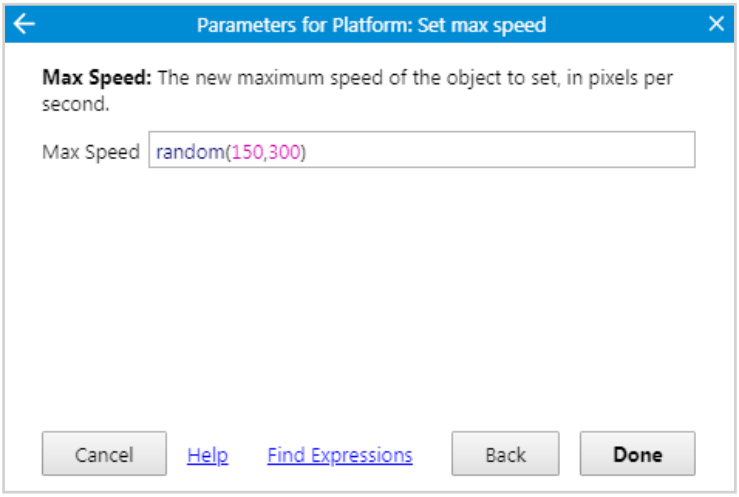

Рисунок 26

Но здесь есть небольшой нюанс: программа может выбирать и дробные числа (165.986542 или 238.458743).

Чтобы программа выбирала только целые числа (155, 240, 187 и т.д), перед указанным параметром «random» необходимо прописать «round». Программа будет округлять значения (рис. 27).

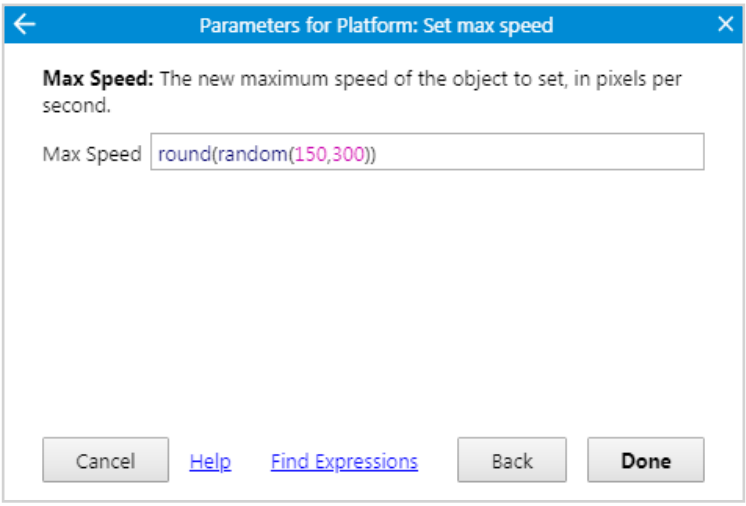

Рисунок 27

В цепочке событий установка рандомной скорости врага занимает всего одно событие (рис. 28).

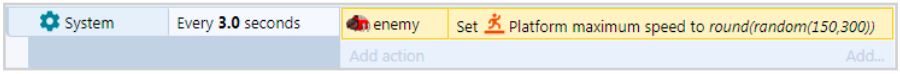

Рисунок 28

Давайте запрограммируем более эффектную стрельбу для игрока и врага. Добавляем элемент «Particles», чтобы создать анимацию взрыва при нанесении урона (рис. 29).

Теперь задаем нужные параметры (рис. 30).

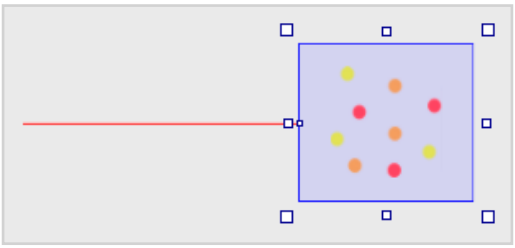

Рисунок 29

Properties	
Rate	10
Spray cone	360
Type	One-shot
Image	Edit
Preview	
Initial particle properties	
Speed	100
Size	32
Opacity	100
Grow rate	0
X randomiser	0
Y randomiser	0
Speed randomiser	0
Size randomiser	0
Grow rate randomiser	0
Particle lifetime properties	
Acceleration	50
Gravity	0
Angle randomiser	0

Рисунок 30

Переходим на страницу событий. Нам нужно задать такое событие, чтобы этот элемент появлялся при попадании пули врага в игрока. Нажимаем «Add event => enemy_bullet => on dectroyed». Сразу добавляем действие «enemy_bullet => spawn another object => Particles_player» (рис. 31).

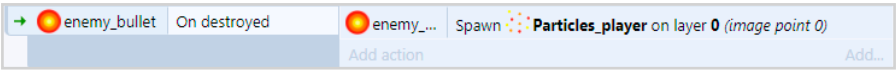

Рисунок 31

Теперь добавим «Particles» и для второго персонажа. Элемент будет воспроизводиться при попадании снежка игрока во врага. Рисуем объект и задаем необходимые параметры (рис. 32).

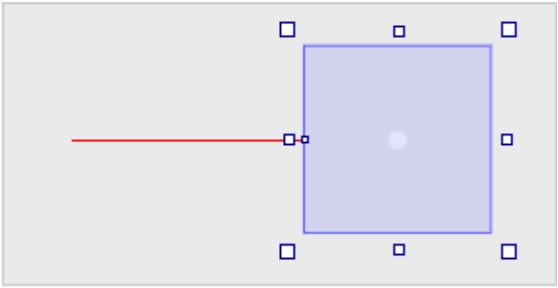

Рисунок 32

Чтобы не добавлять новых повторяющихся событий, следует добавить новые действия к ивентам соприкосновения снежных шаров с врагом.

Добавляем действие «Snowball» (во втором действии «Snowball2) => spawn another object => Particles» (рис. 33).

Рисунок 33

Теперь добавим телепортацию врага относительно игрока. Например, сделаем так, чтобы враг телепортировался и появлялся перед игроком, когда второй проходит определенную координату.

Добавим событие «Player => Compare X». Пускай враг телепортируется, когда игрок переместится на позицию «X ≥ 1797» (рис. 34).

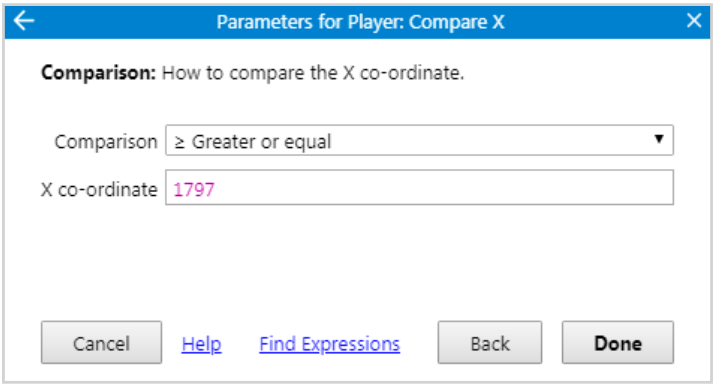

Рисунок 34

Запрограммируем действие: «Enemy => Set X». В окне параметров прописываем позицию игрока «Player.X» и добавляем расстояние, на котором появится враг. Например, +300 (рис. 35).

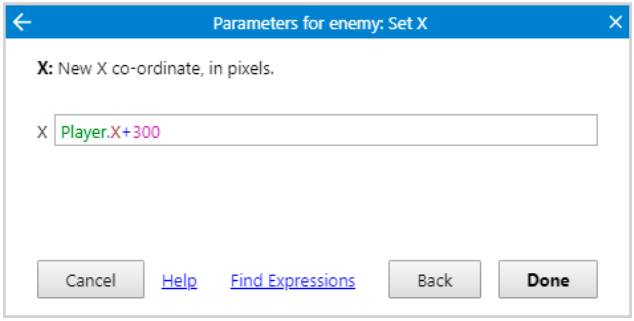

Рисунок 35

При воспроизведении игры, когда игрок пересекает заданную координату, враг появляется в указанном нами месте (рис. 36).

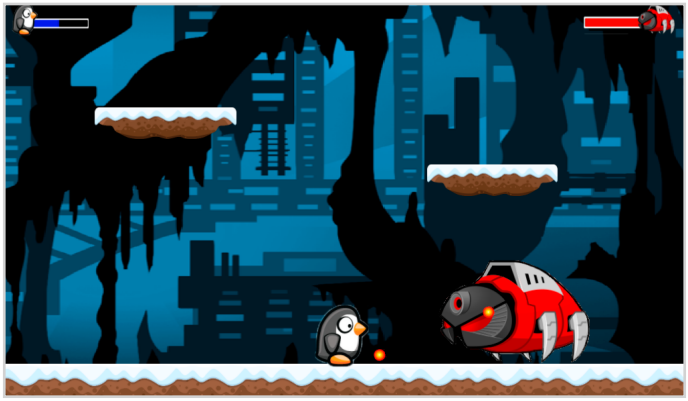

Рисунок 36

Сделаем появление врага эффектнее, добавив небольшую анимацию.

Переходим в окно редактирования врага. Добавляем новую анимацию, называем ее «enemy_appearence» и прорисовываем несколько фреймов. Для того, чтобы сделать его постепенное появление, первый «Image Frame» остав-

ляем пустым, а на последующих постепенно добавляем часть изображения (рис. 37). Можем также увеличить скорость воспроизведения анимации.

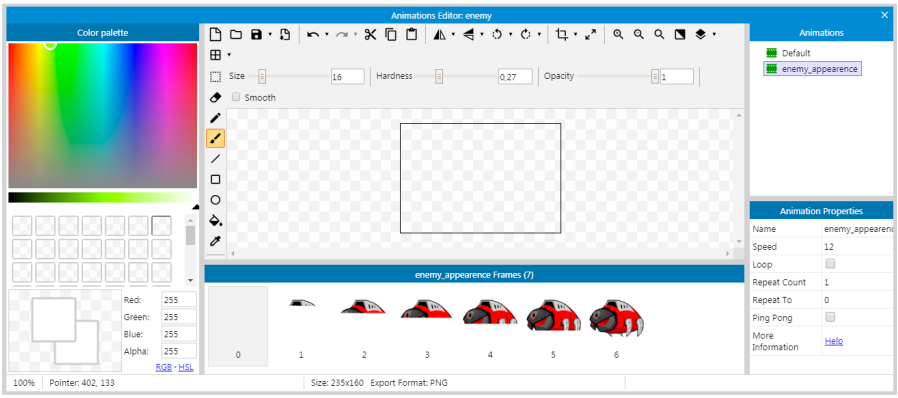

Рисунок 37

После этого добавим в цепочку событий анимацию телепортации врага: «Enemy => Set animation to => enemy appearance». Меняем определение в строке «From на current frame» (рис. 38).

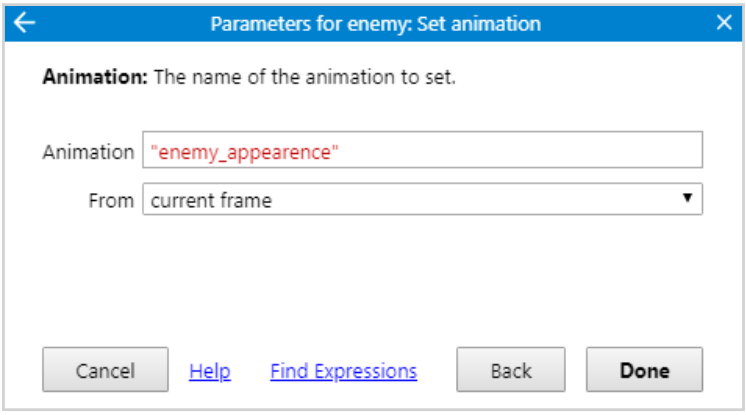

Рисунок 38

В «Event Sheet» эти действия прописываются к одному событию (рис. 39).

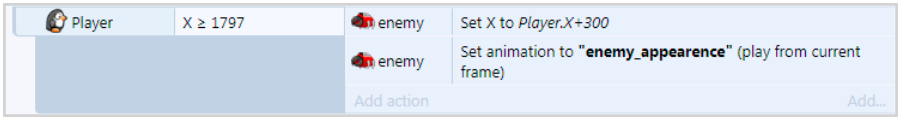

Рисунок 39

Давайте сделаем так, чтобы враг телепортировался только 1 раз, чтобы это действие было внезапным для игрока. Для этого создаем врагу переменную «teleport». Затем, добавляем «add another condition» к предыдущему событию. «Enemy => Compare instance variable => teleport=10» (рис. 40).

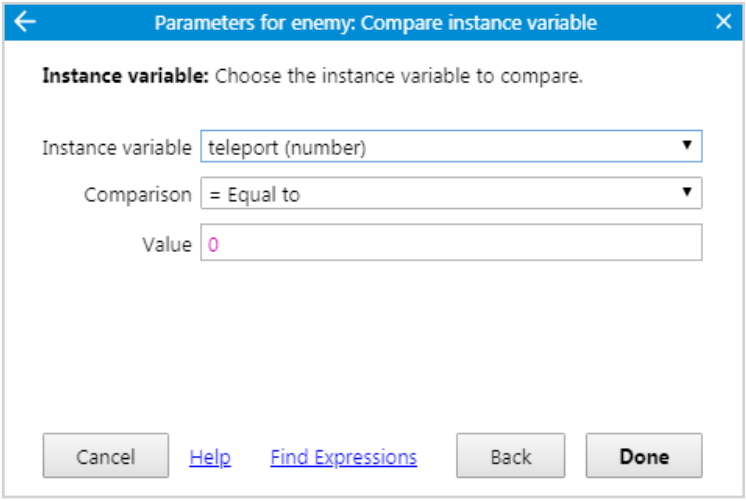

Рисунок 40

Теперь анимация телепортации будет воспроизводиться, когда переменная равна 0. Чтобы действие

не повторялось, достаточно изменить значение переменной после первой телепортации. Например, в действия добавим «Enemy => Add to => teleport (value=1)» (рис.41).

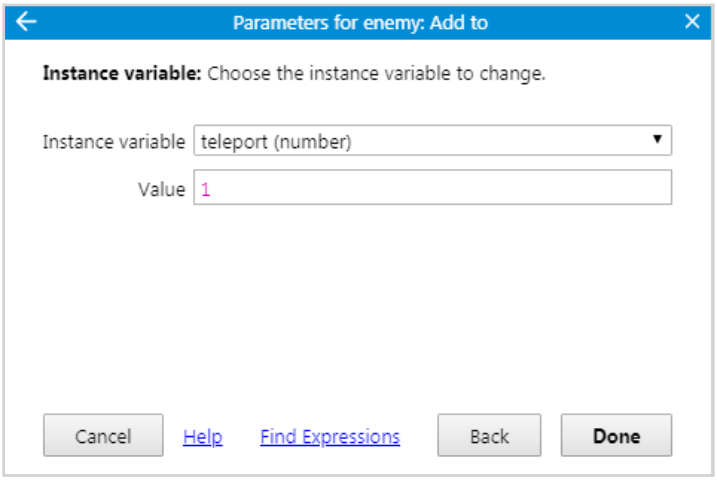

Рисунок 41

Соответственно, после первой телепортации переменная увеличится на 1, а, значит, пока уровень не перезапуститься, телепорт больше не сработает (рис. 42).

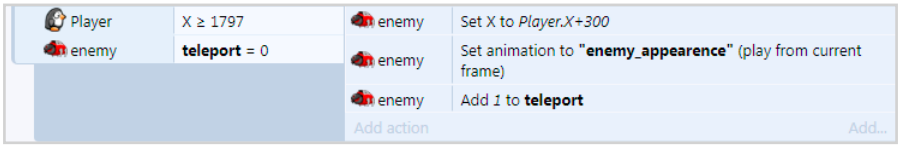

Рисунок 42

Для вдохновения можете рассмотреть, как выглядят финальные битвы с супербоссом в популярных платформерах.

1. «Super Sonic» (рис. 43).

Рисунок 43

2. «Guacamelee!» (рис. 44).

Рисунок 44

3. Мир PIX в: джунгли приключения (рис. 45).

Рисунок 45

Урок 8

Сюжет и кривая обучения в играх. Звуковые эффекты

➢ **ПЛАН РАБОТЫ.**

Рассмотрение разных примеров платформеров, определение основных сюжетных линий и конфликтов, основных притягивающих моментов. Рассмотрение кривых обучения. Создание нестандартного прелоадера и добавление звуковых эффектов.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Сюжет и кривая обучения в играх. Звуковые эффекты

Итак, мы уже ознакомились с созданиями проектов в «Construct 3». Теперь давайте рассмотрим несколько популярных платформеров и их сюжетные линии, чтобы затем создать свою интересную и уникальную игру!

1. «Ori and the Blind Forest» (рис. 1).

Рисунок 1

В основе сюжета этого платформера — персонаж Ори, который узнал о том, что лес, в котором он жил всю свою жизнь, планируют уничтожить темные силы. Герой не может смириться с этим, поэтому во что бы то ни стало он пытается противостоять врагам, спасти лес и всех его жителей (рис. 2).

Рисунок 2

2. «Unravel» (рис. 3).

Рисунок 3

Главный герой этой игры — клубок ниток по имени Ярни. Он потерял своих родных и ничего не помнит из прежней жизни. Тем не менее персонаж верит в то, что он

сможет восстановить воспоминания. Сложность игры в том, что на каждом уровне необходимо тщательно продумывать способы взаимодействия с окружающим миром, а также рассчитывать количество оставшейся пряжи (рис. 4).

Рисунок 4

3. «Oscura: Second Shadow» (рис. 5).

Рисунок 5

Увлекательный платформер с необычным визуальным решением. Главный герой попадает в мир, в котором похитили свет. Его цель — исправить эту ситуацию и вернуть потерю. Сложность прохождения уровней в том, что даже крошечные кристаллы очень сложно достать, и они надежно охраняются (рис. 6).

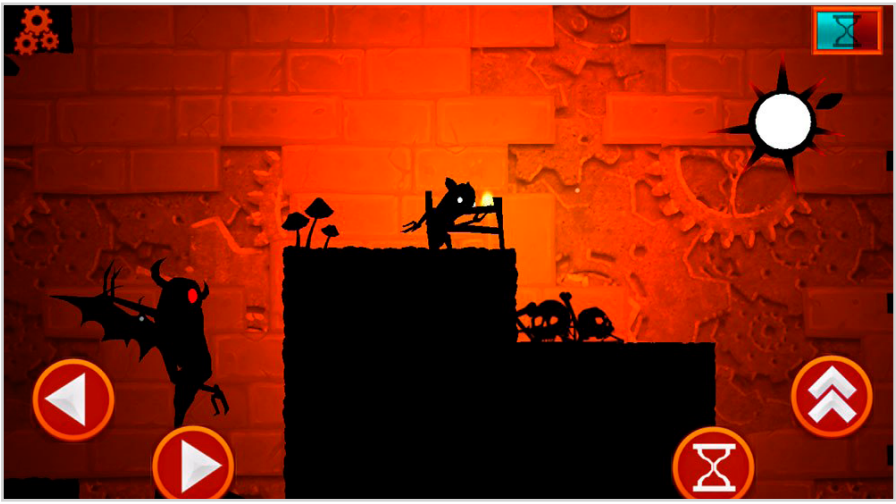

Рисунок 6

Сюжет сделал эти игры увлекательными и популярными. Учтите, уникальность и продуманность вашего сюжета также будут одними из главных моментов при оценке финальной работы. Уделите своей истории как можно больше времени!

Создавая игру, чтобы она была ее прохождение было максимально интересным, следует придерживаться всех особенностей кривой обучения. Мы уже немного вспоминали о ней ранее, но давайте подробнее разберемся, как ее реализовать.

Разработка игры — это создание отдельного мира, где существуют свои правила. Именно поэтому очень важно держать игрока в постоянном восторге от прохождения каждого последующего уровня. В этом нам поможет кривая обучения.

Это график, в котором одна ось отображает время освоения деятельности, а вторая — уровень мастерства игрока (рис. 7).

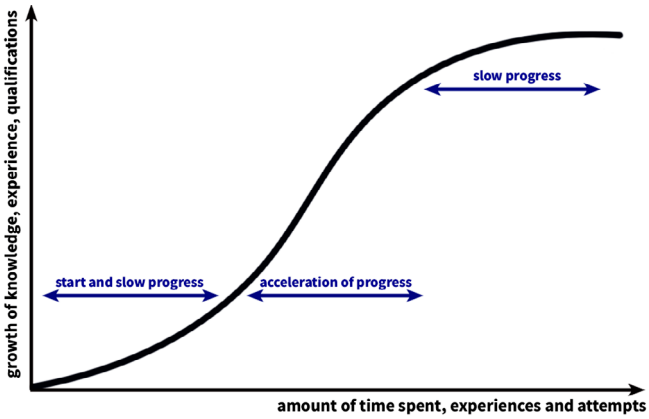

Рисунок 7

Этот график отображает принцип кривой обучения: медленный старт, затем быстрый прогресс. Замедление прогресса происходит тогда, когда обучаемый освоил новое умение и оно стало его навыком.

Но с платформерами дело обстоит немного иначе. Давайте разберем, как выглядят кривая для этих игр. Нарисуем график, где одна ось — уровень сложности «входа» в игру, вторая — насколько сложно стать мастером в этой игре.

Порассуждаем логически. Во всех платформерах достаточно легко научиться передвигаться и применять ос-

новной функционал персонажа. Разработчики обычно не добавляют преграды, которые игроку сложно преодолеть. В противном случае, он просто прекратит играть. В платформере главное не прогресс игрока, а разнообразие. Поэтому, чем больше разных тематических уровней, трюков и прочих элементов вы добавите в игру, тем охотнее люди будут в нее играть.

Разрабатывая игру по такому принципу, кривая обучения будет выглядеть следующим образом (рис. 8):

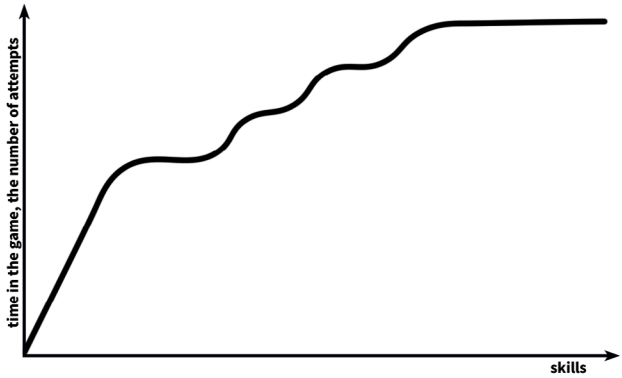

Рисунок 8

В процессе разработки нужно позаботиться о плавности кривой обучения. В первую очередь это касается ввода в игровые карты новых противников/препятствий/ловушек и способностей игрока. Игрок должен успевать обучаться новым трюкам, знакомиться с функционалом и особенностями нововведений. Если же все новые вводные обрушатся на него скопом, это негативно скажется на его игровом опыте и, возможно, отобьет желание взаимодействовать с игрой в дальнейшем.

Еще один способ удержать внимание и интерес игрока, пока происходит загрузка игры или очередного уров-

ня — прелоадер. Давайте рассмотрим способы его создания в «Construct 3».

Для начала создаем новый «Layout» и «Event Sheet». Называем их «Preloader» и «Preloader_Event» соответственно. Задаем фон (рис. 9).

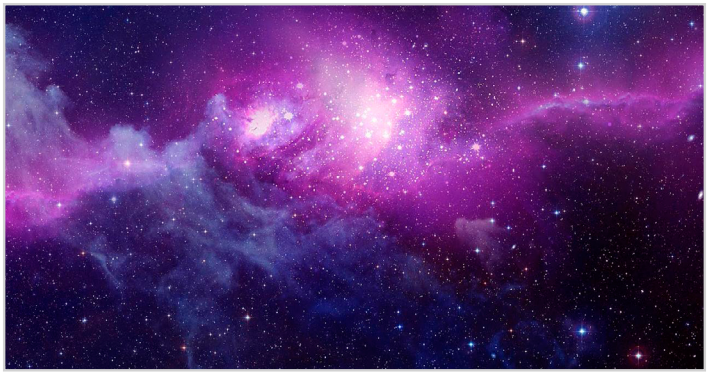

Рисунок 9

Добавляем элемент «Text», где прописываем «Loading» и спрайт с контуром, в котором будет отображаться анимация загрузки (рис. 10).

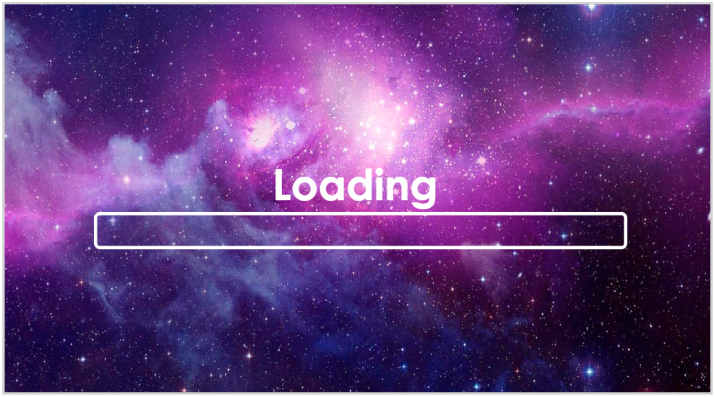

Рисунок 10

Добавляем еще один спрайт (называем его «ufo»), и помещаем внутрь рамки (рис. 11).

Рисунок 11

Задаем «ufo» поведение «Bullet» и переходим в настройки поведения, где снимаем отметку в чекбоксе «Set angle» (рис. 12).

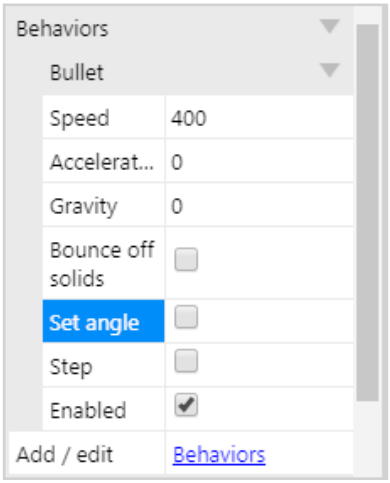

Рисунок 12

Переходим в окно редактирования изображения спрайта и перемещаем вправо «Collision Polygon» (рис. 13).

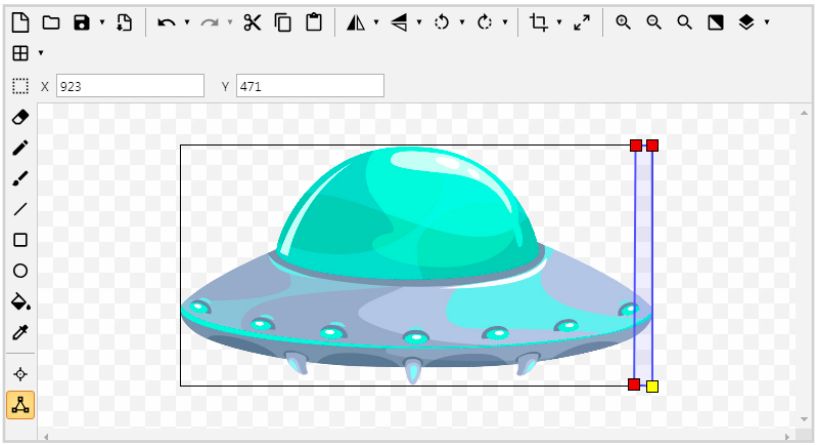

Рисунок 13

Теперь в «Preloader_Event» прописываем событие «Add event => ufo => Is overlapping => loadboard». Затем сразу добавим действие «ufo => Set angle of motion => 0 degrees». Значение угла «=0», поскольку объект должен двигаться только по горизонтали (рис. 14).

Рисунок 14

Копируем событие и применяем опцию «Invert» (рис. 15).

Рисунок 15

Переходим к рабочей области и выбираем элемент «ufo». На панели «Properties» есть данные о размещении объекта на игровой карте (рис. 16).

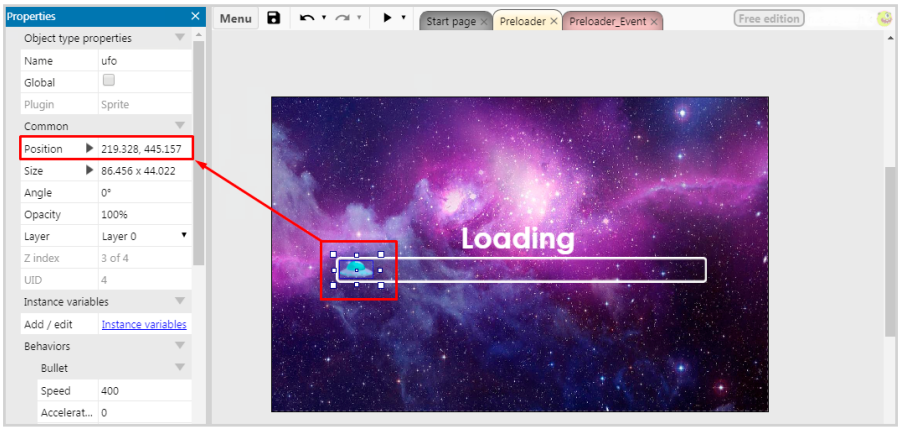

Рисунок 16

В цепочке событий нужно добавить действие, где указываем, что в том случае, когда «ufo» не соприкасается с «loadbord», он появляется на своей первоначальной позиции. «Add event => Set position». Указываем координаты посадочной позиции (рис. 17).

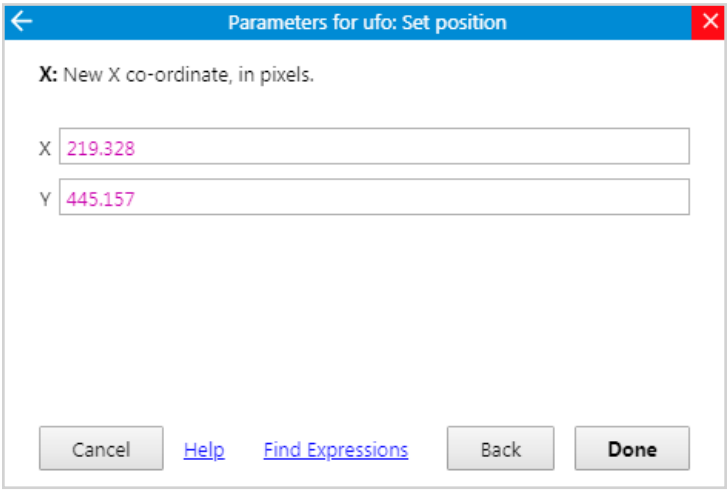

Рисунок 17

В цепочке событий это выглядит следующим образом (рис. 18).

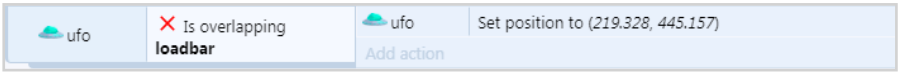

Рисунок 18

Давайте сделаем прелоадер интереснее. Например, добавим ему анимацию с изменением цвета. Для этого переходим в окно редактирования объекта. Нажимаем «Add new animation». Создаем несколько новых фреймов других цветов (рис. 19).

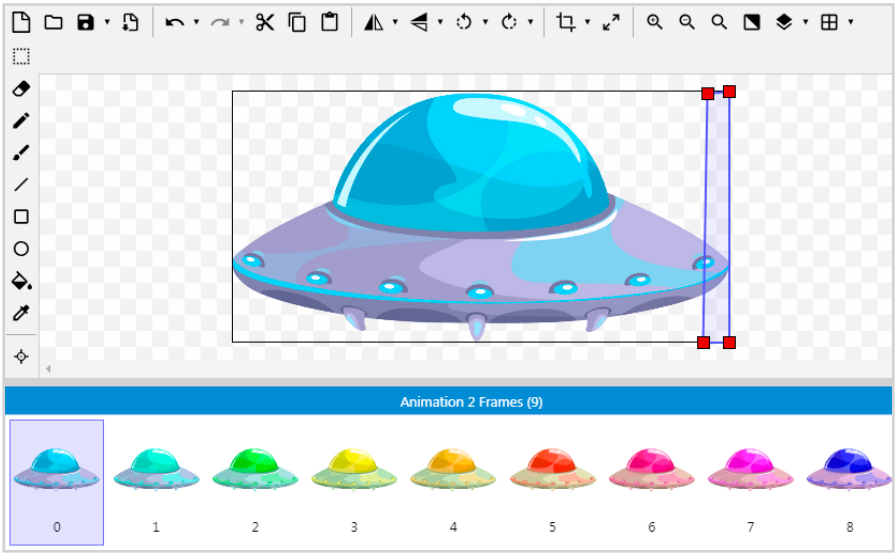

Рисунок 19

Переходим в цепочку событий и добавляем действие в первый «event». «Ufo => Set animation => Animation 2» (рис. 20).

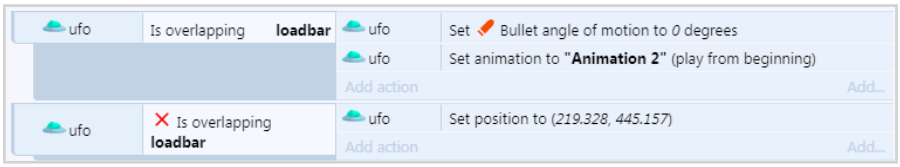

Рисунок 20

Теперь при воспроизведении игры, когда летающая тарелка передвигается, она меняет цвет (рис. 21–22).

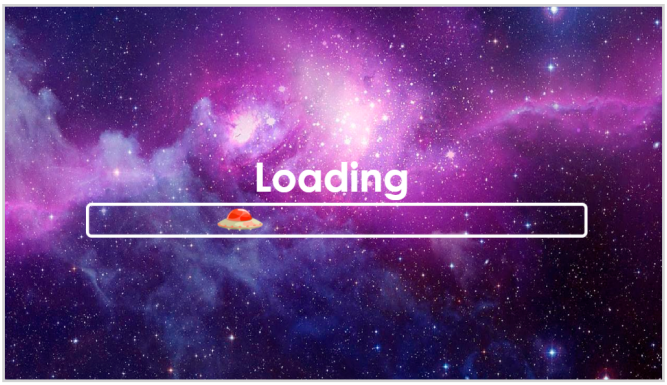

Рисунок 21

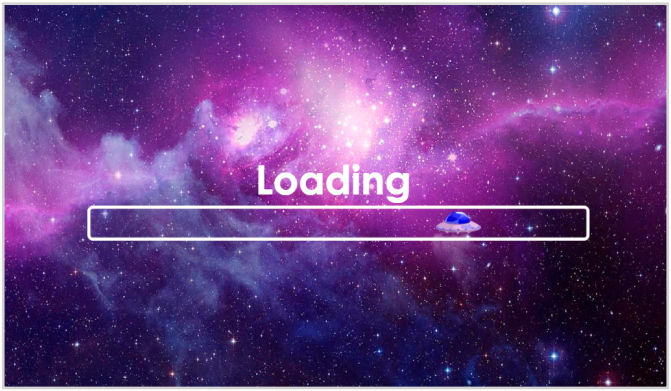

Рисунок 22

Для того, чтобы уровни игры выглядели завершенными, нужно добавить звуковые эффекты. Переходим к игровой карте и добавляем новый объект «Audio» (рис. 23).

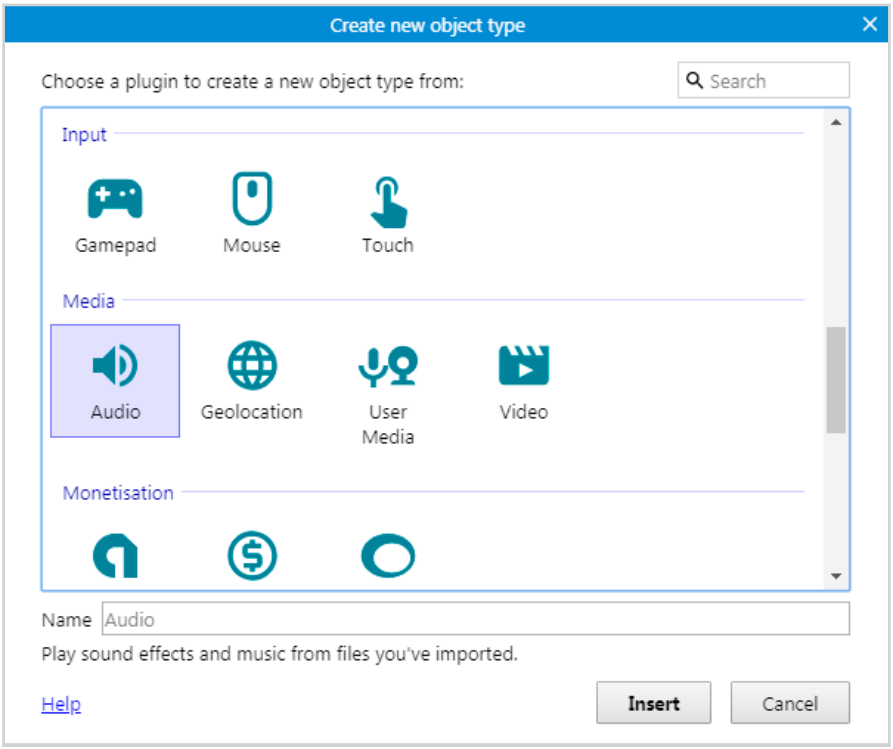

Рисунок 23

Затем переходим на панель «Project» и кликаем правой кнопкой мыши по папке «Sounds». Нам необходимо добавить в проект трек, который, например, будет озвучивать собранные игроком льдинки. Выбираем опцию «Import sounds» (рис. 24).

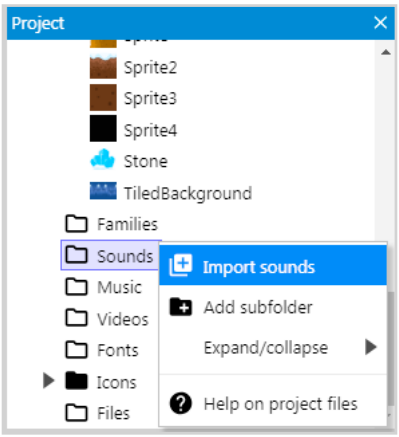

Рисунок 24

Важно помнить, что следует выбирать короткий звуковой эффект, продолжительностью 1–2 секунды.

В появившемся окне либо нажимаем «Import audio», либо перетягиваем из Проводника в выделенную пунктиром область нужный аудиофайл (рис. 25).

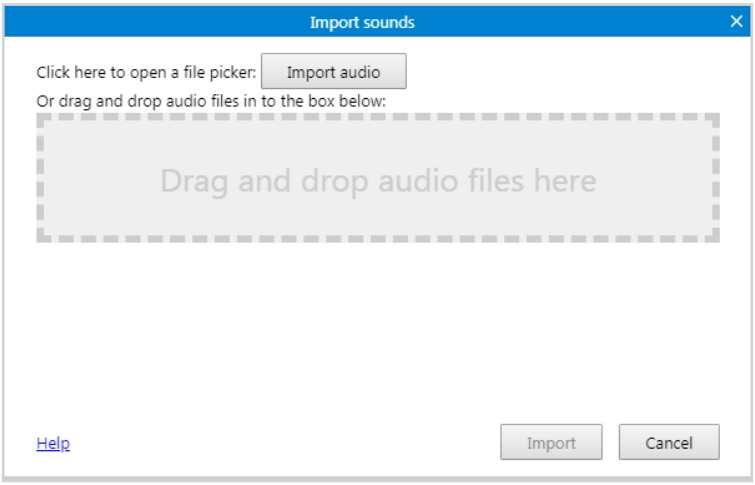

Рисунок 25

После загрузки файл и информация о нем появится в небольшой таблице в этом же окне. Нажимаем «Import» (рис. 26).

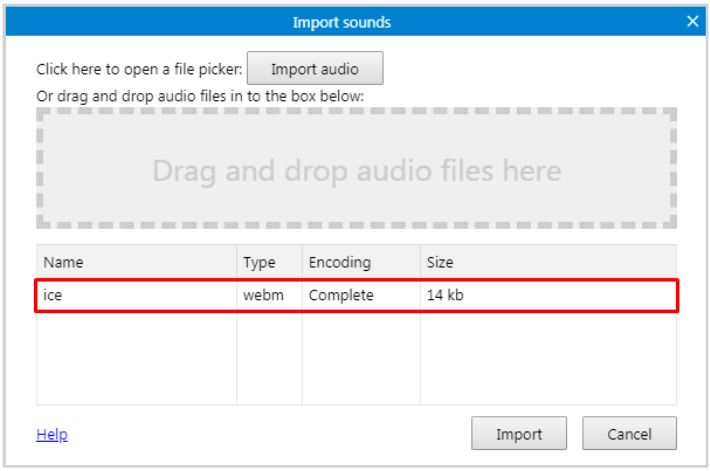

Рисунок 26

Все импортированные в проект звуки будут отображаться на панели «Project» в папке «Sounds» (рис. 27).

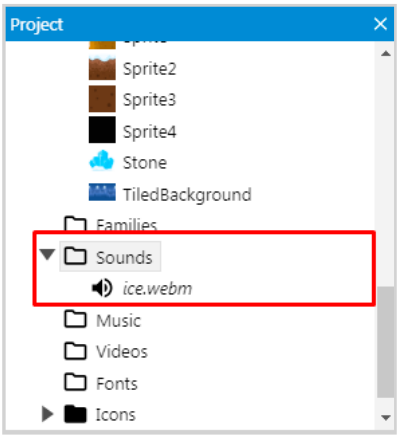

Рисунок 27

Теперь попробуем добавить этот звук в воспроизведение игры. Это можно сделать с помощью цепочки событий. Переходим в «Event Sheet» и находим событие сбора льдинок (рис. 28).

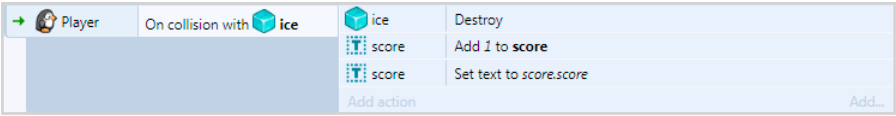

Рисунок 28

К этому ивенту добавим новое действие. «Add action => Audio => Play» (рис. 29).

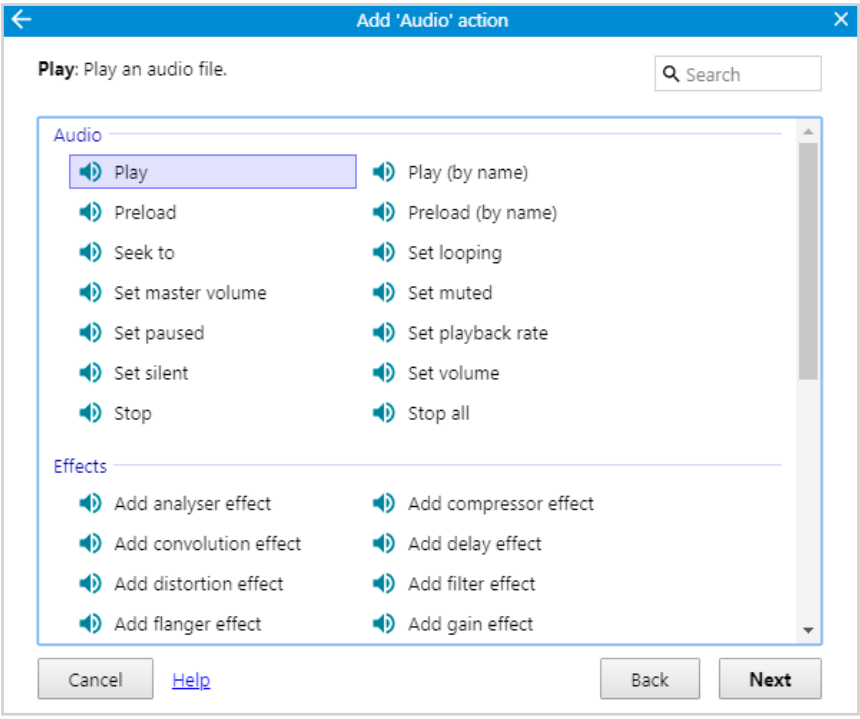

Рисунок 29

В появившемся окне «Parameters» есть несколько строк, которые нужно заполнить:

1. «Audio file» — выбираем нужный аудиофайл из тех, которые импортировали.
2. «Loop» — указываем, нужно ли зациклить аудио.
3. «Volume» — устанавливаем громкость воспроизведения аудио (рис. 30).

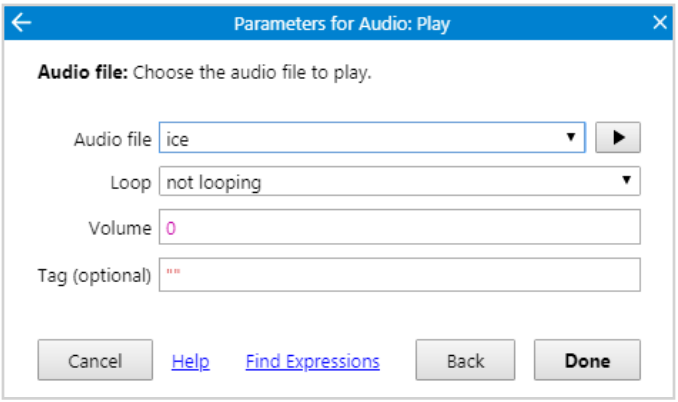

Рисунок 30

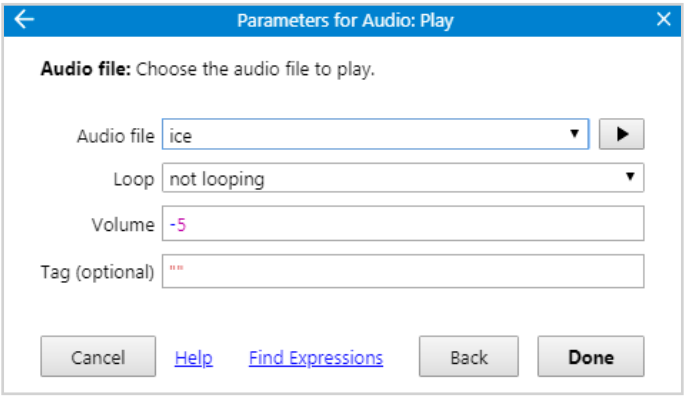

Рисунок 31

Для нашего действия выберем файл «ice», затем «not looping» и зададим громкость –5 (рис. 31).

Теперь при воспроизведении игры, когда персонаж подбирает льдинку, воспроизводится звуковой эффект (рис. 32).

Рисунок 32

В свободное время добавьте к каждому действию в игре звуковой эффект.

Чтобы найти нужные вам эффекты, можете воспользоваться несколькими сайтами со свободным доступом к скачиванию аудиофайлов.

1. Freesound (рис. 33).

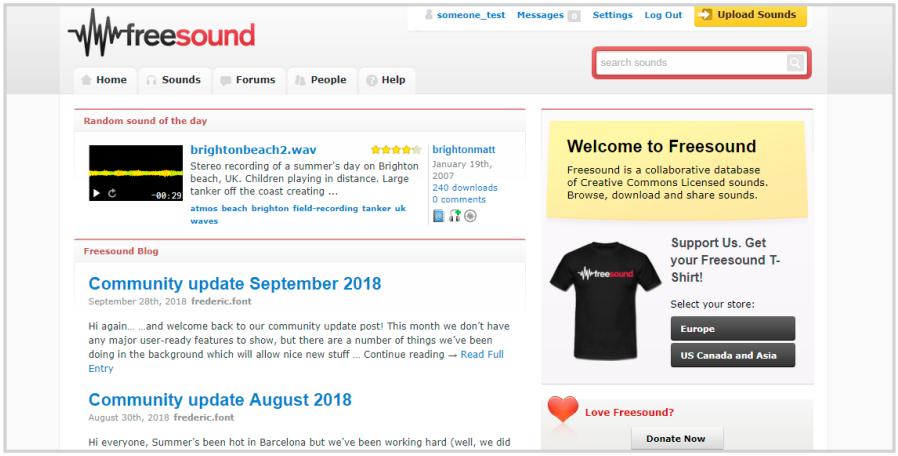

Рисунок 33

2. ZapSplat (рис. 34).

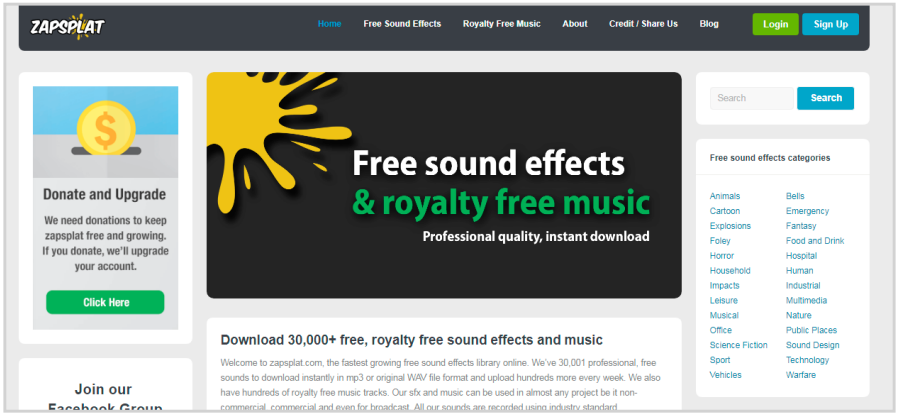

Рисунок 34

3. OpenGameArt (рис. 35).

Урок 9

Создание главного меню. Сохранение игрового процесса

➢ **ПЛАН РАБОТЫ.**

Сохранение игры и главное меню. Создание кнопок Save и Load. Создание нового layout и event sheet для главного меню. Добавление фона и кнопок «Новая игра», «Загрузить игру», «Уровень 1», «Уровень 2», «Об игре». Программирование увеличение размера и изменения прозрачности кнопок при наведении. Создание сцены для кнопки «Об игре».

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Создание главного меню. Сохранение игрового процесса

В *Construct 3* есть очень удобная функция сохранения и загрузки текущего состояния персонажа во время прохождения уровня. Давайте рассмотрим, как это реализуется. Нам нужно добавить кнопки «Save» и «Load».

Переходим к цепочке событий, «Add event => Keyboard => on key pressed => S» (рис. 1).

Рисунок 1

Затем добавляем событие, которое происходит при нажатии во время игры кнопки «S. Add action => System => Save» (рис. 2).

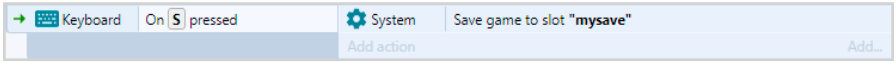

Рисунок 2

По аналогичному принципу задаем событие и действие для загрузки игры. При нажатии кнопки «L», игра загружается с того момента, где была сохранена (рис. 3).

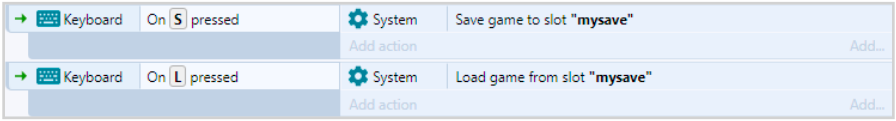

Рисунок 3

Давайте немного визуализируем сохранение и загрузку, и добавим для этих опций кнопки непосредственно на уровне. Создаем два новых спрайта и называем их «Load» и «Save» (рис. 4).

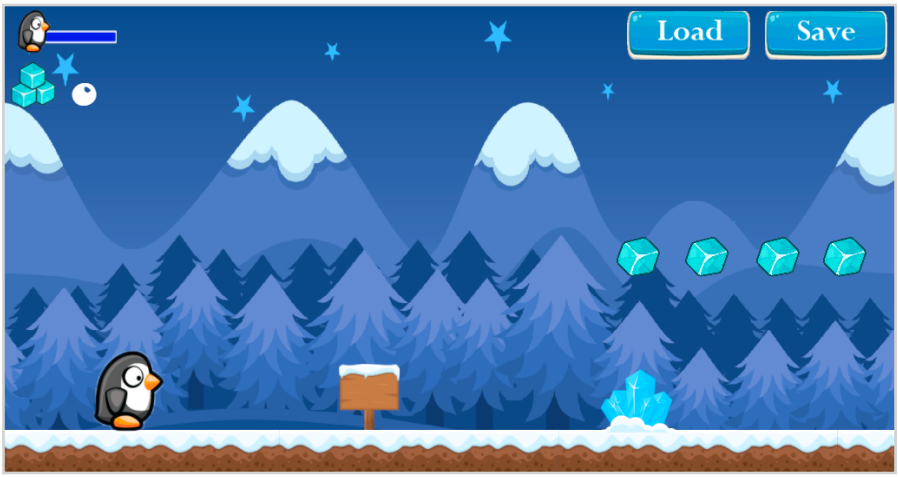

Рисунок 4

Задаем каждой кнопке анимацию с одним фреймом другого цвета (рис. 5).

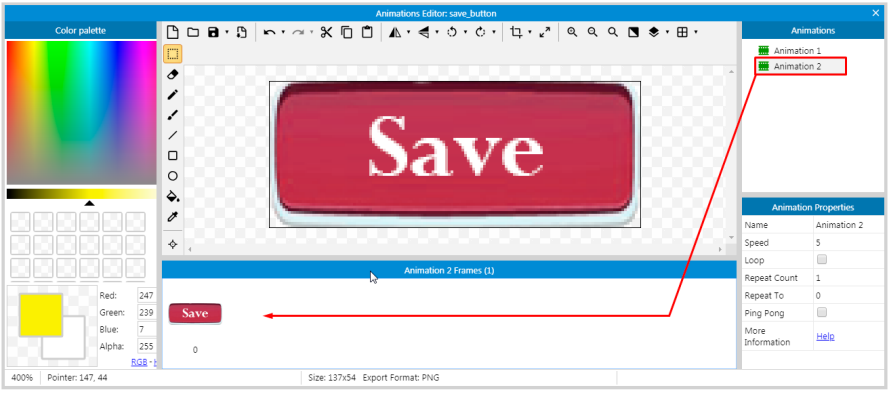

Рисунок 5

Теперь добавим этим кнопкам изменение цвета при нажатии на «keyboard». Переходим в цепочку событий и добавляем новое действие к уже существующим событиям. Там, где было запрограммировано сохранение добавляем «Save_button => Set animation => Animation 2 (from current frame)». По такому же принципу добавляем действие для «Load_button» (рис. 6).

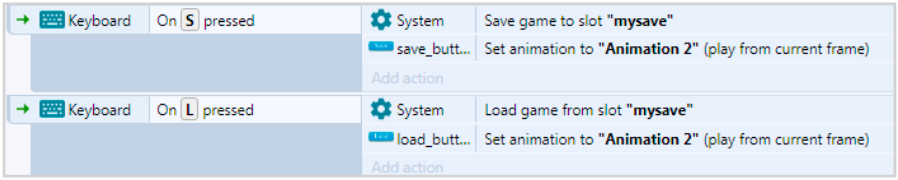

Рисунок 6

Осталось только вернуть кнопки в первоначальный вид, когда загрузка или сохранение будут завершены. Для этого переходим «Add event => On save complete». Затем добавляем «Save_button => Set animation => Animation 1 (from current frame)». Повторяем последовательность и для кнопки «Load» (7).

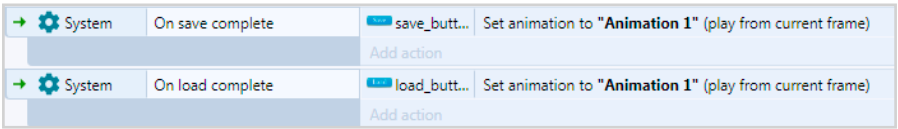

Рисунок 7

Теперь, при прохождении игры, можно нажать на «S», чтобы сохранить персонажа и значения его переменных на этой позиции. В этот момент кнопка поменяет свой цвет на красный. Затем, пройдя еще часть уровня можно нажать «L», чтобы вернуть персонажа к сохраненной

позиции. В этом случает подсвечиваться красным будет кнопка «Load» (рис. 8).

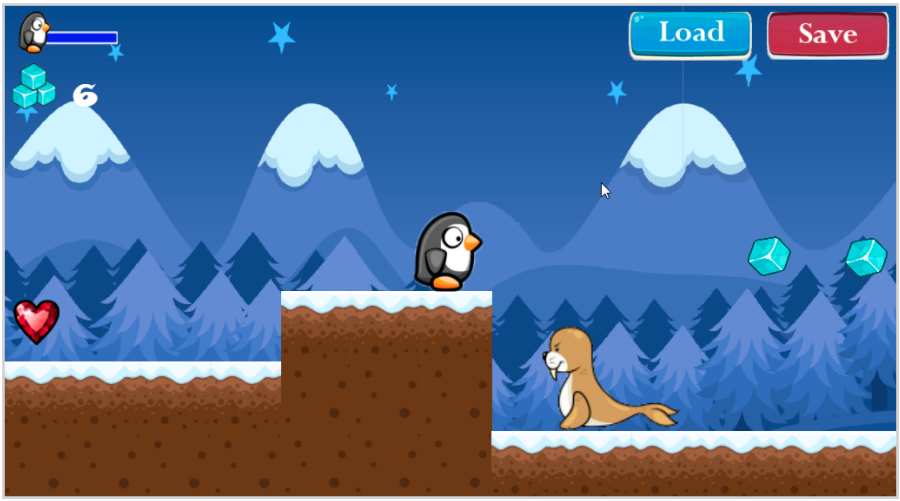

Рисунок 8

С кнопками «Save» и «Load »проходить уровни гораздо легче, потому что игрок в любой момент может вернуться к сохраненной ранее позиции и полученным очкам.

Давайте теперь создадим новый «Layout »и «Event Sheet, »чтобы добавить меню к игре. Только сделаем их уже в другом стиле. Добавляем соответствующий «background» (рис. 9).

Также создаем три спрайта — кнопки «Play», «Load» и «Exit» (рис. 10).

Каждой кнопке следует добавить по одному дополнительному фрейму, который будет немного светлее предыдущего (рис. 11).

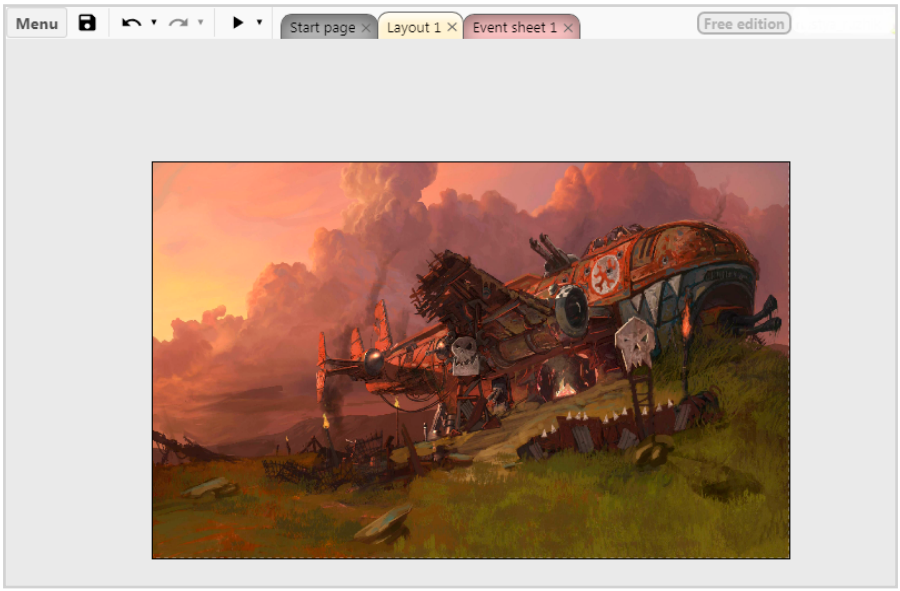

Рисунок 9

Рисунок 10

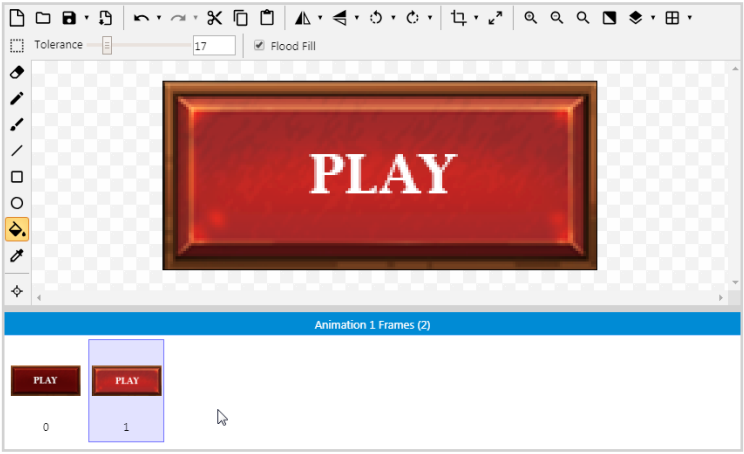

Рисунок 11

Далее добавим в проект еще один элемент — «Mouse» (рис. 12).

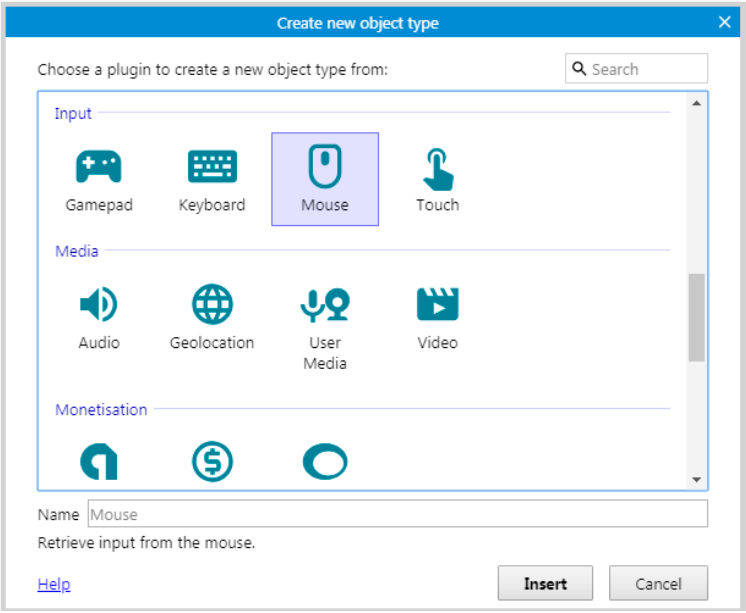

Рисунок 12

Переходим в цепочку событий. «Add event => Mouse => Cursor is over object => Play» (рис. 13).

Рисунок 13

Сразу же добавим действие, чтобы, при наведении курсора на кнопку, появлялся фрейм, который мы добавили. «Add action => Set frame => 1» (рис. 14).

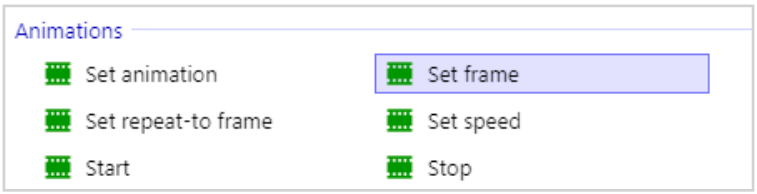

Рисунок 14

В цепочке событий это выглядит следующим образом (рис. 15).

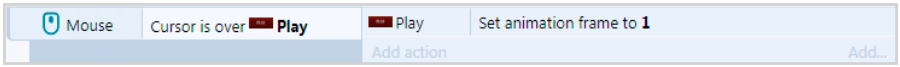

Рисунок 15

Теперь копируем этот ивент, но инвертируем наведение курсора на кнопку. Затем возвращаем фрейм 0 (рис. 16).

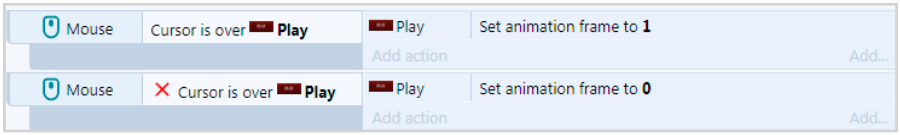

Рисунок 16

Теперь, при наведении курсора, кнопка будет загораться, а когда мы его уберем — гаснуть (рис. 17–18).

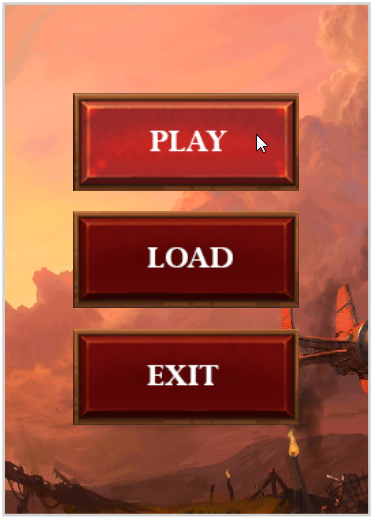

Иллюстрация 17

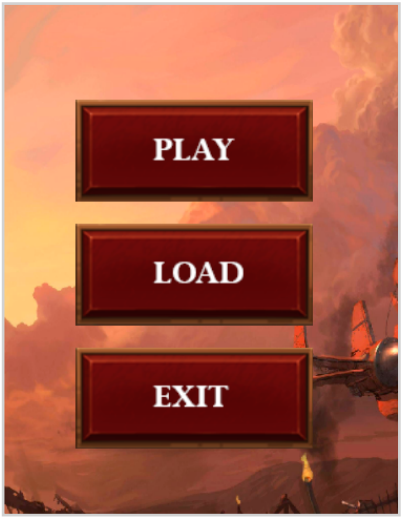

Иллюстрация 18

Теперь подсоединим кнопку «Play» к следующему окну, в котором игрок сможет выбрать уровень для прохождения.

Сначала создадим новый «Layout» и «Event Sheet». На нем разместим текст «Choose Level», а также два спрайта, на которых будут отображаться игровые карты этих уровней. По тому же принципу, что мы делали кнопки, анимируем эти элементы, сделав второй фрейм с синей, а не белой рамкой (рис. 19).

В цепочке событий запрограммируем анимацию при наведении на них мыши (рис. 20).

Чтобы сделать переход из «Menu» на «Choose Layout», нажимаем «Add event => Mouse => On object clicked => Play». И сразу добавим действие, чтобы при нажатии на Play открывался первый уровень игры: «System => Go to Layout => Choose Layout» (рис. 21).

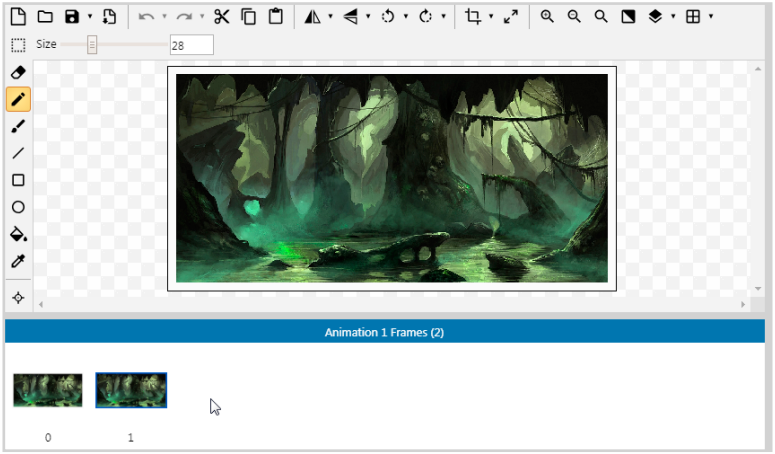

Рисунок 19

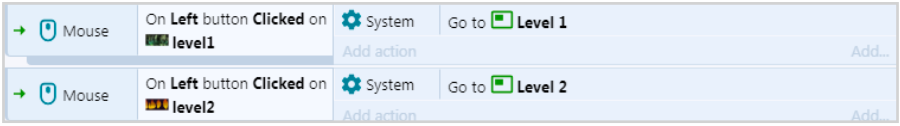

Рисунок 20

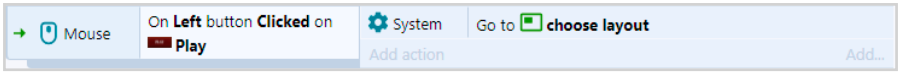

Рисунок 21

После переходим на вкладку с событиями для «Choose layout». Здесь добавляем то же событие для перехода, что и на странице с кнопками. Отдельно прописываем переход на первый и второй уровни (рис. 22).

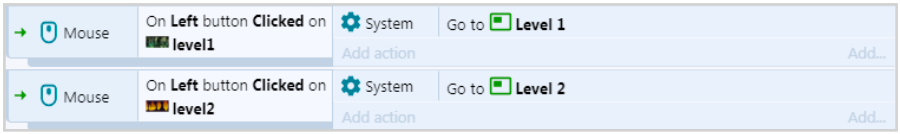

Рисунок 22

Проверяем, работает ли кнопка. При нажатии на «Play», игрок попадает на страницу «Choose Layout», а уже затем на один из двух созданных уровней (рис. 23–25).

Рисунок 23

Рисунок 24

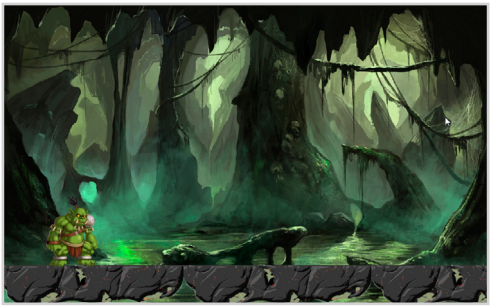

Рисунок 25

По аналогии добавим «Layout», на который игрок будет попадать при нажатии в «Menu» кнопки «About». Создаем фон для новой страницы и добавляем текст об игре (рис. 26).

Рисунок 26

Затем в цепочке событий прописываем событие и действие (рис. 27).

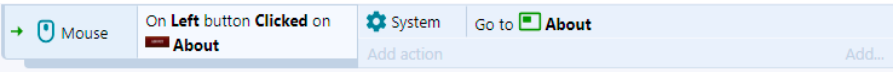

Рисунок 27

Сейчас для меню нам не хватает еще одной кнопки — «Back». Давайте заанимируем ее немного по-другому. Сделаем так, чтобы при наведении курсора она увеличивалась, и менялась ее прозрачность.

Рисунок 28

Начнем со знакомой нам последовательности. Переходим на страницу «About». Добавляем текст на игровую карту, разместив его в левом нижнем углу (рис. 28).

Теперь уменьшим «Opacity» элемента, чтобы, при наведении на него мышкой, он становился ярче. Сделать это можно на панели «Properties» в разделе «Common» (рис. 29).

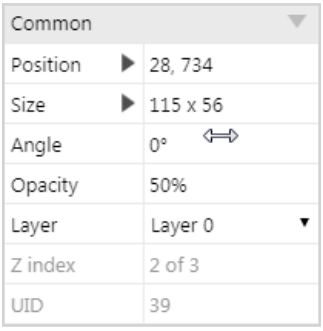

Рисунок 29

Текст стал полупрозрачным (рис. 30).

Рисунок 30

Переходим в цепочку событий. Выбираем «Add event => Mouse => Cursor is over object => Back». Добавляем два действия:

1. Увеличение размера: «Back => Set size => 140×80».
2. Увеличение opacity: «Back => Set opacity => 100» (рис. 31).

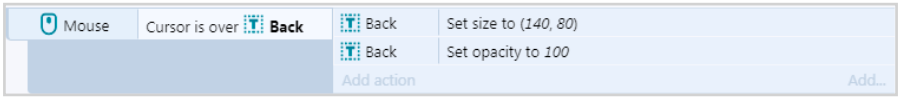

Рисунок 31

Дублируем это событие и инвертируем его. Это значит, что, при отведении курсора от текста, он должен вернуться к первоначальным параметрам. Выставляем размер «115×55», и «opacity=50%» (рис. 32).

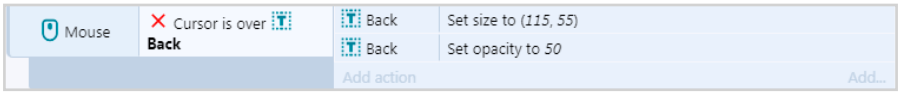

Рисунок 32

И после этого добавляем в цепочку событий переход на главное меню, после нажатия на «Back» (рис. 33).

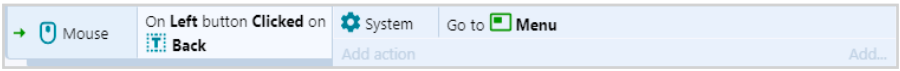

Рисунок 33

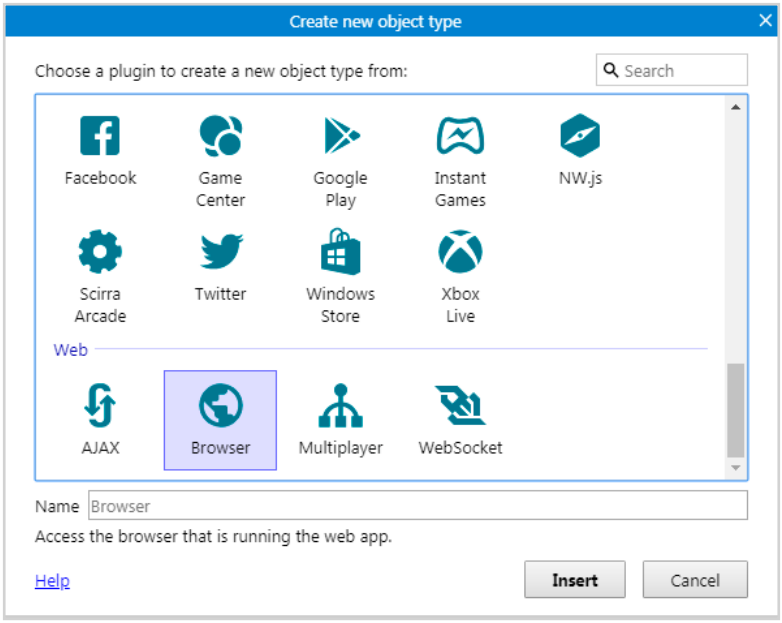

Рисунок 34

Теперь рассмотрим, как добавить действие выхода на кнопку «Exit». Для этого нужно добавить в проект еще один элемент — «Browser» (рис. 34).

Затем переходим к «Menu_Events». Добавляем уже знакомое нам событие «Mouse => On object clicked => Exit» (рис. 35).

Рисунок 35

Затем добавим действие «Add action => Browser». Из предоставленного списка выбираем «Close» (рис. 36).

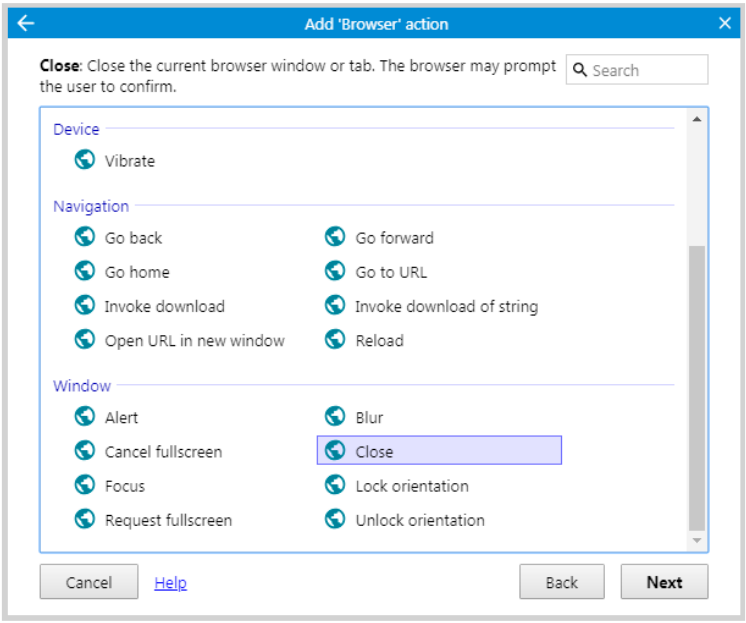

Рисунок 36

В цепочке событий это выглядит следующим образом (рис. 37):

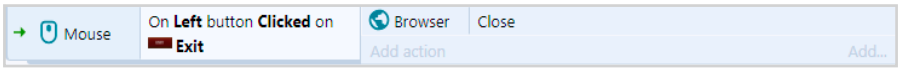

Рисунок 37

По такому принципу можно создавать множество окон и опций в любой игре. Вы можете изменять расположение кнопок, их вид и текст. В свободное время придумайте, как можно красочно оформить главное меню, чтобы привлечь внимание игрока, а также добавить несколько кнопок и переходов на другие «layout».

Посмотрите, как выглядят «Main Menu» в популярных играх:

1. **Roboto** (рис. 38).

Рисунок 38

2. **Staying together** (рис. 39).

Рисунок 39

3. **Rayman fiesta run** (рис. 40).

Рисунок 40

Урок 10

Создание fighting game

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Создание fighting game

Мы уже научились создавать крутые платформеры, которые остается только совершенствовать и усложнять с каждым уровнем. Но, используя рассмотренные ранее принципы, можно разрабатывать и другие крутые игры. Например, «fighting game». Давайте поэтапно рассмотрим, как создать такую игру. Наработки по этой игре можно использовать для создания уровней с финальными боссами (рис. 1).

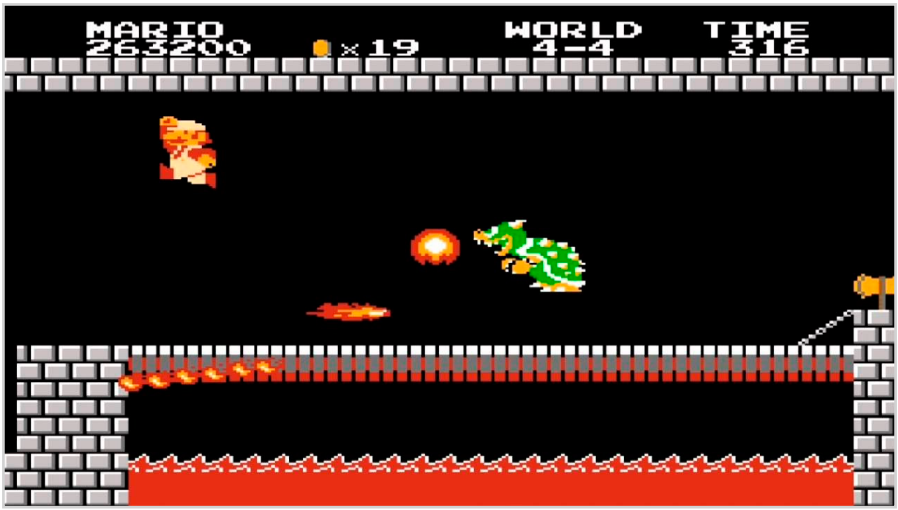

Рисунок 1

Создаем новый проект и задаем параметры «layout». Можно оставить размеры такими же, как они были в предыдущих играх. Выбираем «background» и добавляем его как «Tiled Background» (рис. 2).

Рисунок 2

Добавляем двух игроков. Пусть это будут сражающиеся рыцари. Одного назовем «Player», а другого — «Enemy», чтобы не путать их при построении цепочки событий (рис. 3).

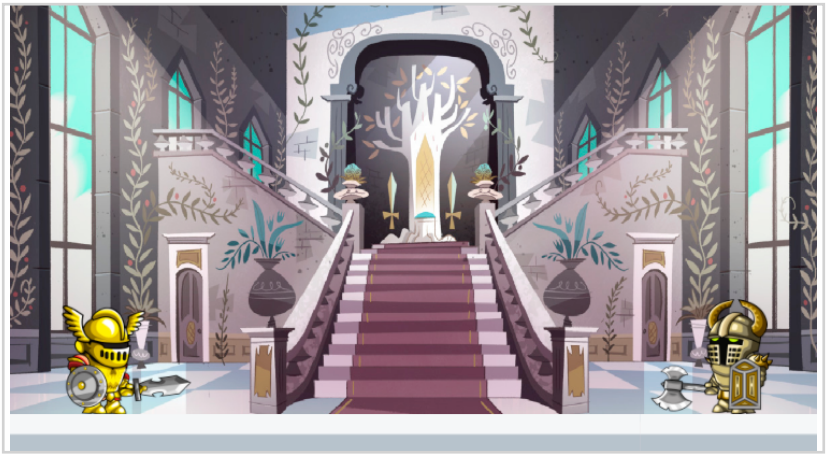

Рисунок 3

Давайте сразу же запрограммируем им анимацию. Добавляем фреймы по уже знакомому нам принципу.

В итоге у каждого персонажа должно быть по 4 базовых анимации: «Stand», «Fight», «Jump», «Run» (рис. 4).

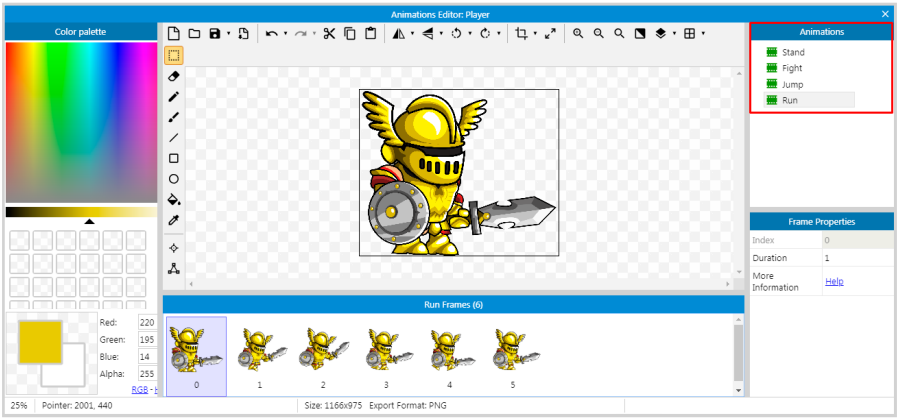

Рисунок 4

В цепочке событий прописываем анимацию для «Player». Когда персонаж двигается, воспроизводится анимация «Run», когда прыгает — «Jump» и когда останавливается или приземляется — «Stand». Принцип анимации игрока в данном случае аналогичен принципу анимирования персонажей в платформерах (рис. 5).

Рисунок 5

У нас осталась еще одна анимация — «Fight». Давайте добавим ее при нажатии на определенную клавишу, это будет анимация атаки. Переходим «Add event => Keyboard => On key pressed». Выбираем пробел. Затем добавляем действие «Player => Set animation => Fight» (рис. 6).

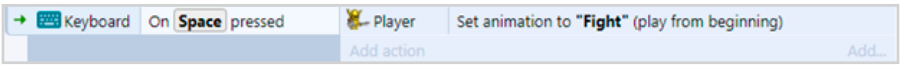

Рисунок 6

Анимация врага будет немного отличаться. Давайте сделаем так, чтобы врагом управляла программа. То есть, «Player» будет играть против компьютера. Для этого нам необходимо запрограммировать врага таким образом, чтобы он преследовал персонажа.

Добавляем« »событие« System => Compare two values. »Задаем условие, где значение координаты «X» врага больше, чем значение координаты «X» игрока (рис. 7).

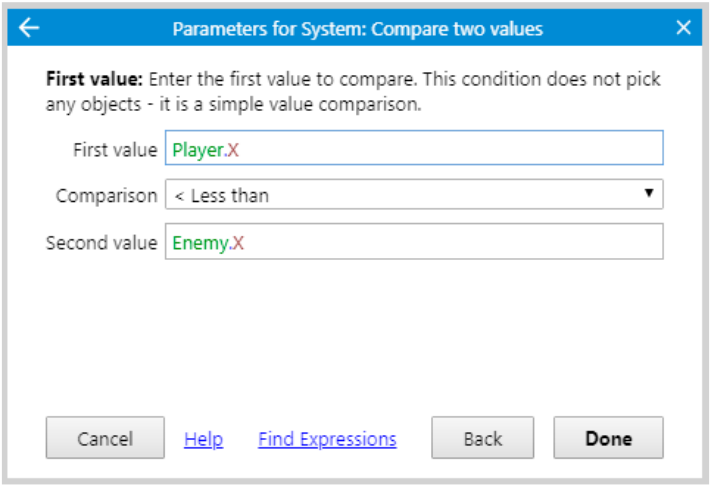

Рисунок 7

В таком случае враг должен двигаться влево. Значит, добавляем действие «Enemy => Simulate control => Left». А также добавляем действие «Set not mirrored», чтобы враг был повернут влево (рис. 8).

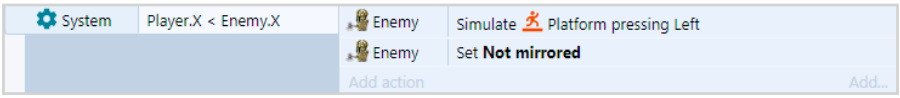

Рисунок 8

Теперь задаем противоположное значение события, когда враг будет находиться слева от игрока и пытаться его догнать. «Add event => Compare two values => Player.X => Enemy.X» (рис. 9).

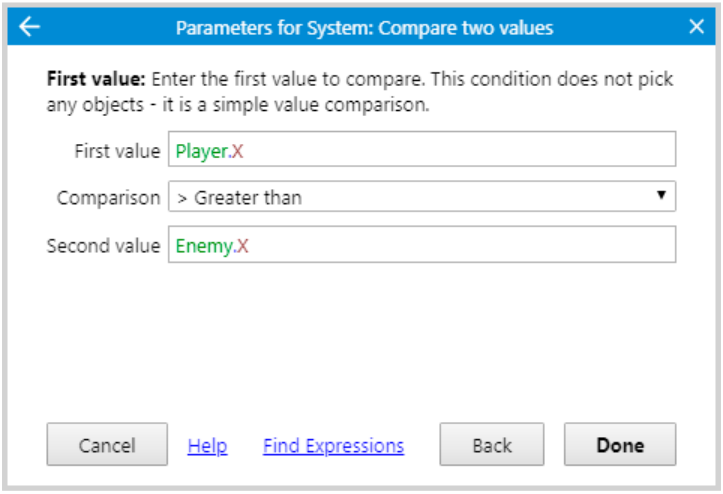

Рисунок 9

Добавляем действие, где «Enemy => Simulate control => Right», а также еще одно действие «Enemy => Set mirrored», чтобы персонаж разворачивался в другую сторону (рис. 10).

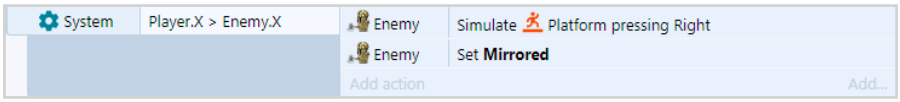

Рисунок 10

Для того, чтобы враг преследовал игрока, все время передвигаясь с анимацией, добавляем «Enemy => On moved», а затем «Enemy => Set animation => Run» (рис. 11).

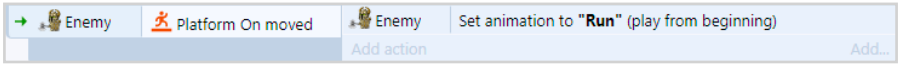

Рисунок 11

Теперь запрограммируем врага так, чтобы, при приближении к игроку на определенное расстояние, он начинал атаку. Сначала добавим врагу поведение «LineOfSight» (рис. 12).

Рисунок 12

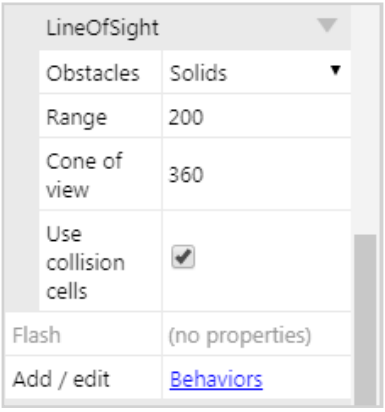

Рисунок 13

Также можно сразу установить расстояние, на котором враг будет замечать игрока и начинать атаковать. Для этого переходим на панель «Properties» и изменяем параметр «Range». Например, выставляем его на 200 (рис. 13).

Переходим к цепочке событий. Добавляем «Enemy => Has LOS to object => Player». Это значит, что действие будет происходить во время приближения к объекту «Player» (рис. 14).

Рисунок 14

Добавляем« »действие« Enemy => Set animation => Fight» (рис. 15).

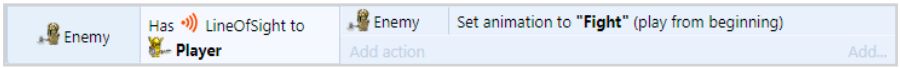

Рисунок 15

Осталось только вернуть врага в анимацию «Run», после того, как игрок выйдет из его поля зрения. Дублируем событие и инвертируем его. Добавляем действие «Enemy => Set animation => Run» (рис. 16).

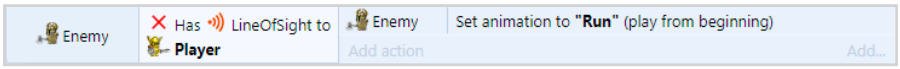

Рисунок 16

Также необходимо создать «healthbar» для персонажей. Создадим его по тому же принципу, как делали жизни для игроков на финальном уровне с супербоссом в предыдущей игре. Добавляем два спрайта. Первый — синий

«healthbar» для «Player», второй — красный «healthbar_enemy» для врага. Можете добавить красивое оформление для спрайтов (рис. 17).

Рисунок 17

Добавляем игроку «Instance variables => lives» со значением 0 (рис. 18).

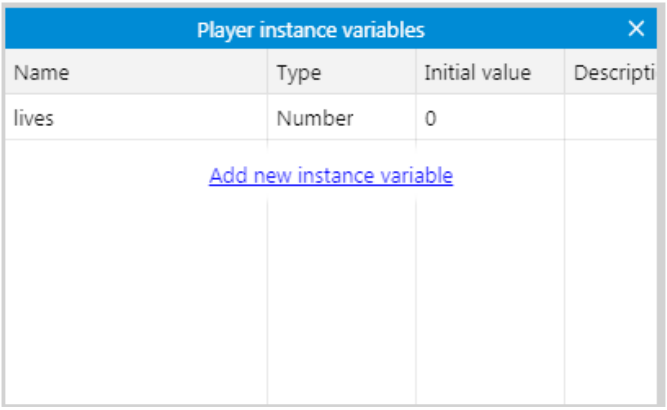

Рисунок 18

Затем добавляем врагу «Instance variables => lives_enemy» со значением 0 (рис. 19).

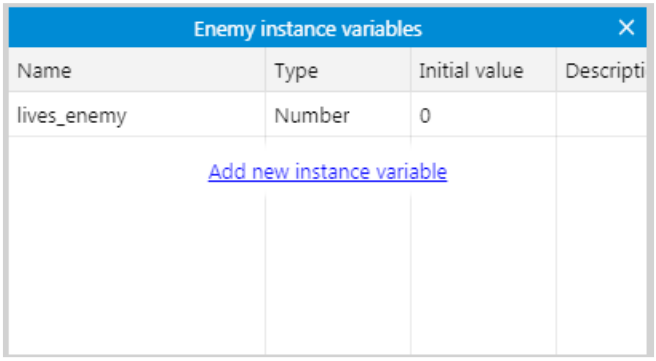

Рисунок 19

Переходим к цепочке событий. Добавим глобальную переменную «health», которой задаем значение, соответствующие длине «healthbar». К примеру, 500 (рис. 20).

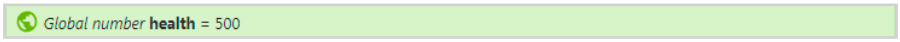

Рисунок 20

Зададим события, которые будут происходить, когда у врага или игрока закончится здоровье:

1. «healthbar => set width => ≤0», действие «System => Restart layer».
2. «healthbar_enemy => set width => ≤0», действие «Enemy => Destroy» (рис. 21).

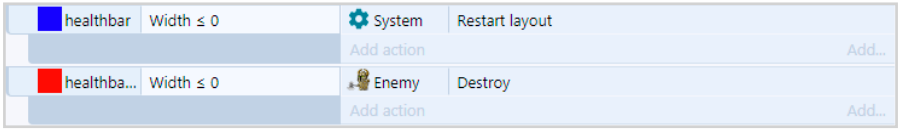

Рисунок 21

Добавим еще один способ атаки врага. Когда игрок будет соприкасаться с врагом, атакуя его, враг будет терять здоровье. «Add event => Player => on collision with => Enemy». Затем необходимо добавить еще одно условие: «Add another condition => Player => Is playing => Fight» (рис. 22).

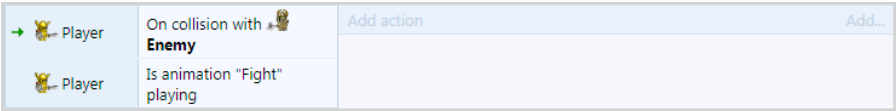

Рисунок 22

Добавляем результат этих событий: «healthbar_enemy => Set width => healthbar_enemy.Width-10» (рис. 23).

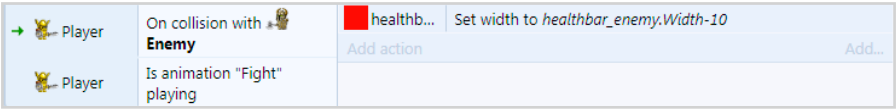

Рисунок 23

Создадим новый элемент — «bomb», еще одно оружие игрока (рис. 24).

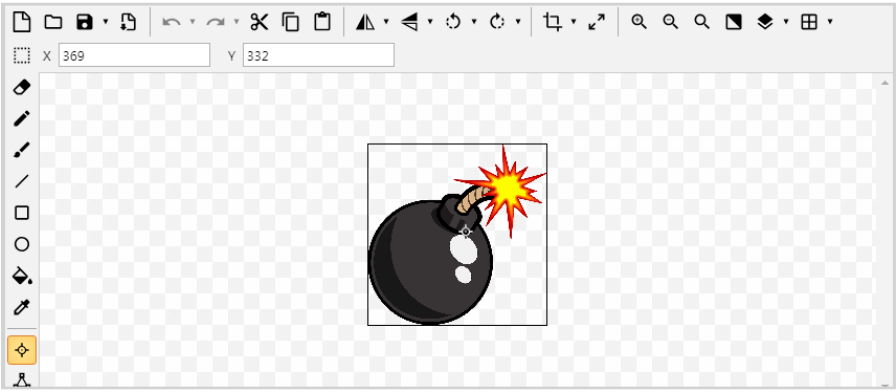

Рисунок 24

Задаем ей поведение «Bullet» (рис. 25).

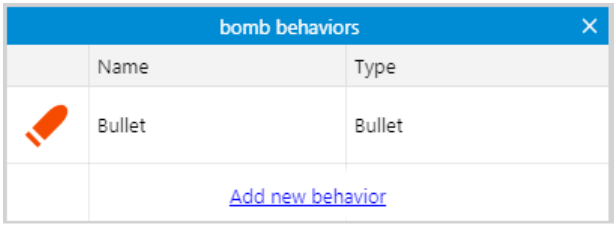

Рисунок 25

Затем переходим в цепочку событий. Атаковать врага с помощью бомбы игрок сможет при нажатии на определенную клавишу, например, «Z». Создадим новое событие: «Add event => Key is down => Z» (рис. 26).

Рисунок 26

Прописываем действие «System => Create object => bomb» (рис. 27).

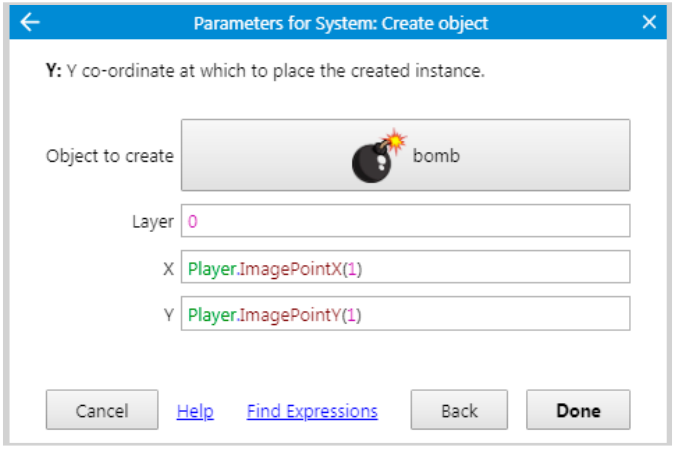

Рисунок 27

В цепочке событий это выглядит следующим образом (рис. 28).

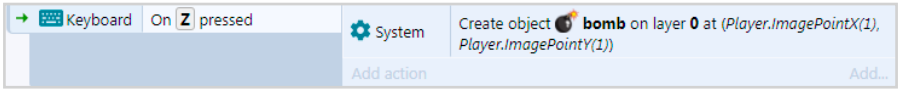

Рисунок 28

Дублируем бомбу на игровой карте, называем ее «bomb2». Затем в цепочку событий добавляем ивент выстрела в противоположную сторону. «Add event => Key is down => Z». Нажимаем «Add another condition => Player => Set mirrored» (рис. 29).

Рисунок 29

Добавляем к этому событию «sub-event. System => Trigger once while true». Затем задаем действия:

1. «Player => spawn new object => bomb2».
2. «bomb2 => Set angle => 180°».
3. «bomb => Destroy» (рис. 30).

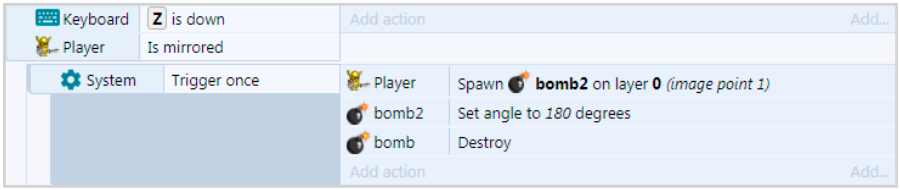

Рисунок 30

Пропишем события о том, что происходит при соприкосновении бомбы и врага. «Bomb => is overlapping => Enemy». И к этому ивенту добавим событие, которое

нанесет урон врагу «healthbar_enemy => Set width => healthbar_enemy.Width-10» (рис. 31).

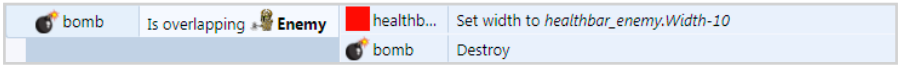

Рисунок 31

Давайте добавим еще эффект взрыва при попадании бомбы во врага. Для этого переходим на игровую карту. «Insert new object => Particles». Добавляем изображение взрыва (рис. 32).

Рисунок 32

На панели параметров справа устанавливаем необходимые значения скорости, угла распространения и размера частиц (рис. 33).

Properties	
Rate	50
Spray cone	360
Type	One-shot
Image	Edit
Preview	
Initial particle properties	
Speed	200
Size	20
Opacity	100
Grow rate	0
X randomiser	0
Y randomiser	0
Speed randomiser	0
Size randomiser	0
Grow rate randomiser	0

Рисунок 33

Возвращаемся к цепочке событий. Добавляем еще одно действие в событие перекрытия бомбы и врага. «Bomb => Spawn another object => Particles» (рис. 34).

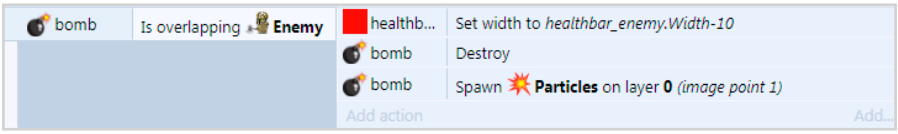

Рисунок 34

Не забывайте о том, что есть еще и второй элемент, которым игрок наносит урок врагу — «bomb2». Дублируем предыдущий ивент с действиями, только вместо «bomb» указываем «bomb2» (рис. 35).

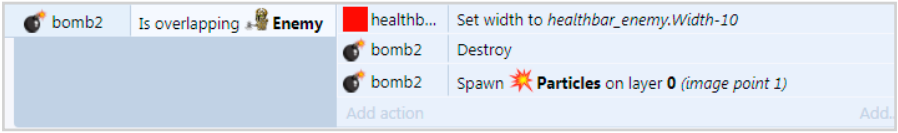

Рисунок 35

Мы уже создали оружие для нашего игрока, пришло время создать оружие и для врага. Добавляем еще один спрайт в виде булавы и задаем ему поведение «Bullet».

Поскольку мы не можем руководить врагом, давайте его запрограммируем. Например, сделаем так, чтобы он атаковал игрока каждые 3 секунды. «Add even => Every X seconds — 3». Затем зададим действие: «Enemy => spawn another object => enemy_bullet» (рис. 36).

Также добавим действие, чтобы булава летела в ту точку, где находится игрок в момент атаки. «Enemy_bullet => Set angle toward position => Player.X, Player.Y» (рис. 37).

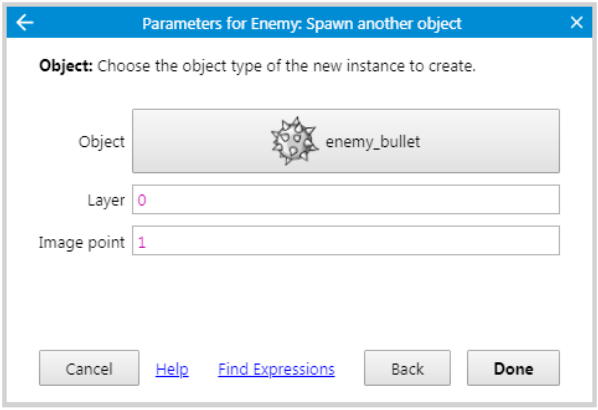

Рисунок 36

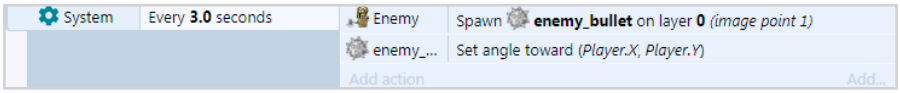

Рисунок 37

Добавляем события, чтобы булава, при соприкосновении с игроком, забирала у него здоровье и исчезала (рис. 38).

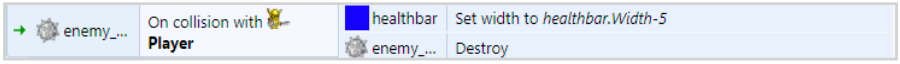

Рисунок 38

Давайте создадим еще и контратаку! Для этого добавим «Particles». Сделаем их в виде шипов и запрограммируем так, чтобы они появлялись при соприкосновении игрока с булавой (рис. 39).

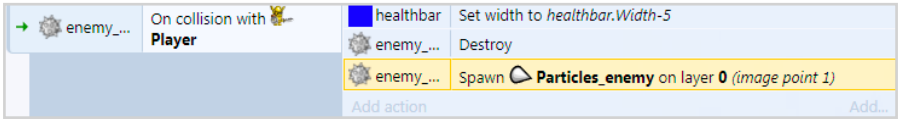

Рисунок 39

Создадим новый «layout», на который будет попадать игрок в случае победы (рис. 40).

Рисунок 40

Теперь добавим переход на эту страницу. Находим событие, где игрок побеждает врага, после чего тот исчезает. Добавим новые действия.

1. «System => Wait => 2 seconds».
2. «System => Go to layout => Win» (рис. 41).

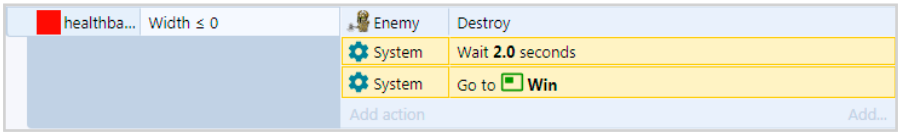

Рисунок 41

Создаем еще один «layout», на который пользователь попадем в случае проигрыша. Добавим его в событие, где ранее было указан рестарт уровня (рис. 42).

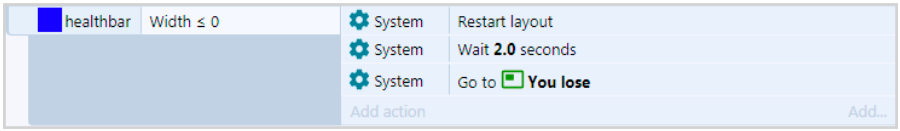

Рисунок 42

Теперь, при потере всего здоровья, мы попадаем на страницу «You lose» (рис. 43).

Рисунок 43

Внизу экрана можно добавить небольшую подсказку, как перезапустить игру. Например, напишем «Press “R” to restart game» (рис. 44).

Рисунок 44

В цепочке событий к «layout You Lose» прописываем событие, где указываем, что при нажатии «R», игра возобновится (рис. 45).

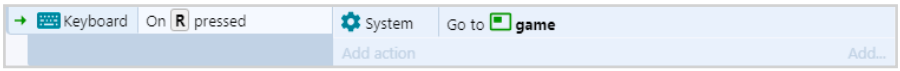

Рисунок 45

В свободное время можете запрограммировать больше вариантов атаки и нанесения урона как врагу, так и игроку. Также можно добавить другие виды оружия и суператаки, которые активируются при определенных условиях.

Для вдохновения посмотрите, какие крутые «Fighting games» можно создавать с помощью «Construct 3»!

1. «Blade Strangers» (рис. 46).

Рисунок 46

2. «Them's Fightin' Herds» (рис. 47).

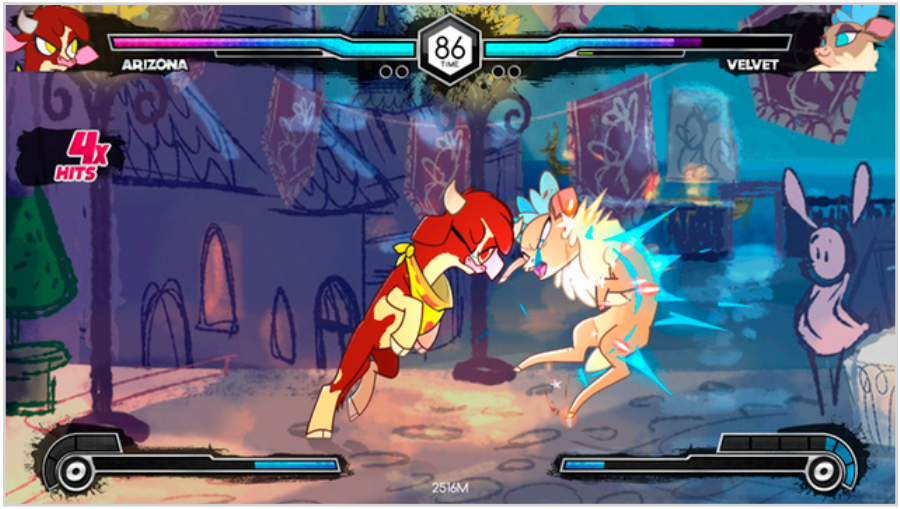

Рисунок 47

3. «Game of Thrones» (рис. 48).

Рисунок 48

Урок 11

Подготовка презентации, обратная связь от одногруппников

➢ **ПЛАН РАБОТЫ.**

Добавление уникального прелоадера, рассмотрение стандартных вариантов, создание специальной сцены во время загрузки с расчётом процента загрузки и сообщением для игроков.

Материалы урока прикреплены к данному PDF-файлу. Для доступа к материалам, урок необходимо открыть в программе Adobe Acrobat Reader.

Подготовка презентации, обратная связь от одногруппников

На сегодняшнем занятии нам предстоит завершить работу над финальным проектом, а затем кратко презентовать одногруппникам наработки по создаваемой игре, ее общую концепцию.

Если же проект готов и не требует дополнительных доработок, давайте сделаем его более красочным, например, добавим к нему кастомный прелоадер. Это может быть сцена с расчетом процента загрузки, а также информационным сообщением для игроков.

Сначала создаем новый «layout» и «event_sheet». На панели «Properties» нажимаем «View» в пункте «Project» «Properties» (рис. 1).

Здесь представлен раздел «Start-up». В нем отображаются настройки старта игры и самого прелоадера. Нам нужно выставить значение для «First Layout», то есть, указать сцену, которой будет начинаться игра. Это и есть наш «Preloader» (рис. 2).

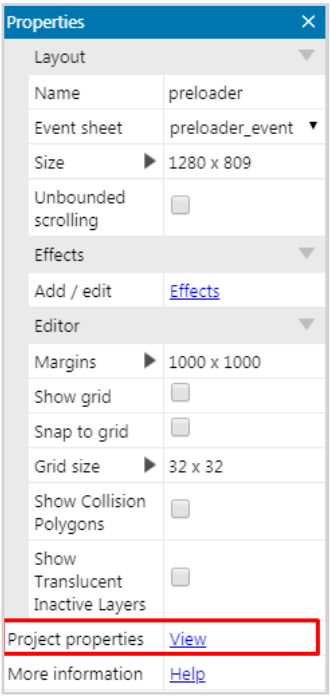

Рисунок 1

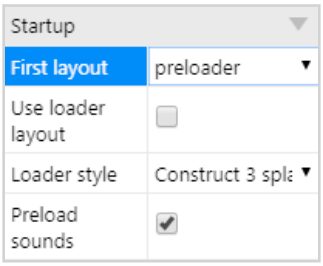

Рисунок 2

Затем нужно поставить отметку «Use loader layout», иначе эта сцена не будет воспроизводиться при загрузке игры (рис. 3).

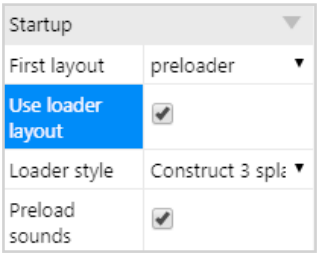

Рисунок 3

Теперь добавим прелоадер на игровую карту. Переходим к рабочей области и добавляем объект «Text» (рис. 4).

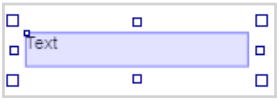

Рисунок 4

Прелоадер должен динамично изменяться, поэтому переходим в «Event Sheet» и добавляем новое событие «System => Every tick». Затем прописываем действие

«Loading => Set text». В появившемся окне «Properties» нажимаем «Find Expression», находим значение «round» и дважды кликаем по нему (рис. 5).

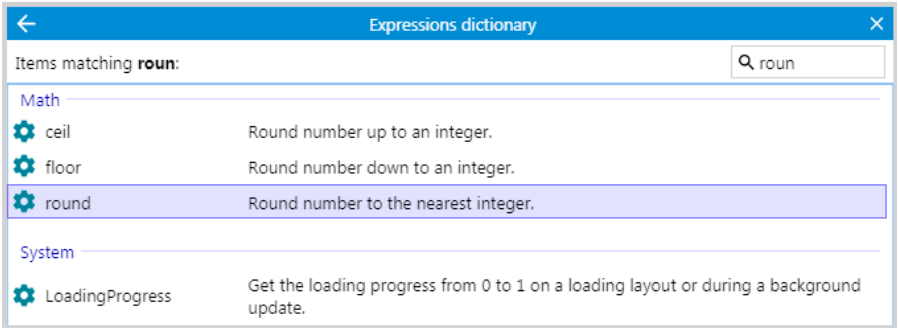

Рисунок 5

В «Text» прописываем «loadingprogress» и умножаем значение на 100 (рис. 6).

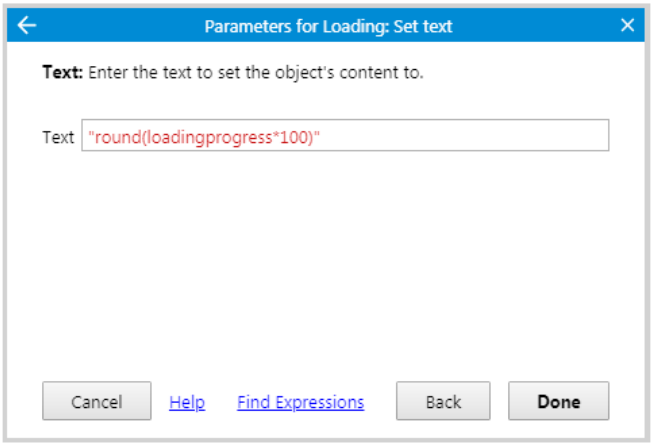

Рисунок 6

Здесь же можно добавить текст. Текст необходимо прописать в кавычках, а также использовать символ & как соединитель введенных данных (рис. 7).

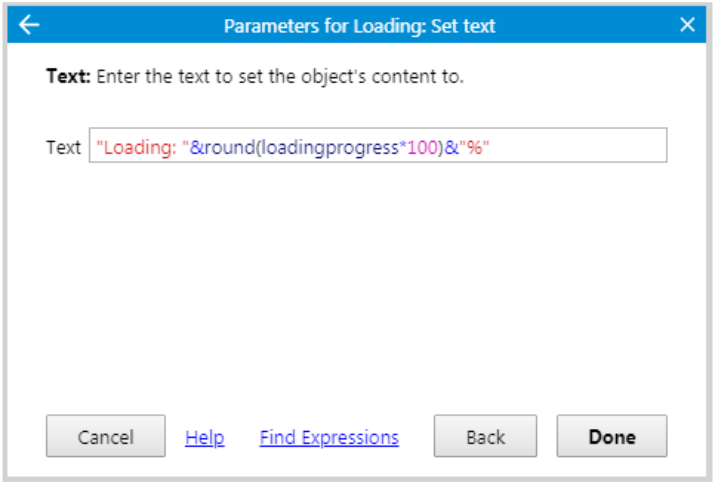

Рисунок 7

В цепочке событий это выглядит следующим образом (рис. 8).

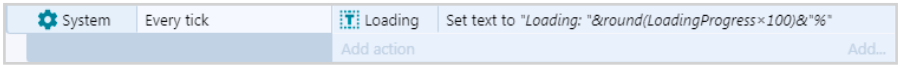

Рисунок 8

После этого необходимо закрыть прелоадер. Для этого добавляем новое событие: «System => On loader layout complete» (рис. 9).

Добавляем действие «System => Go to layout => game», чтобы сразу перейти к игре (рис. 10).

Сейчас прелоадер выглядит очень просто: белая сцена и процентное значение загрузки (рис. 11).

Конечно, такая сцена не подходит для игры. Ее необходимо доработать: изменить шрифт и размер текста. Также следует изменить размещение текста в сцене. Например, перенесем его в нижнюю область экрана (рис. 12).

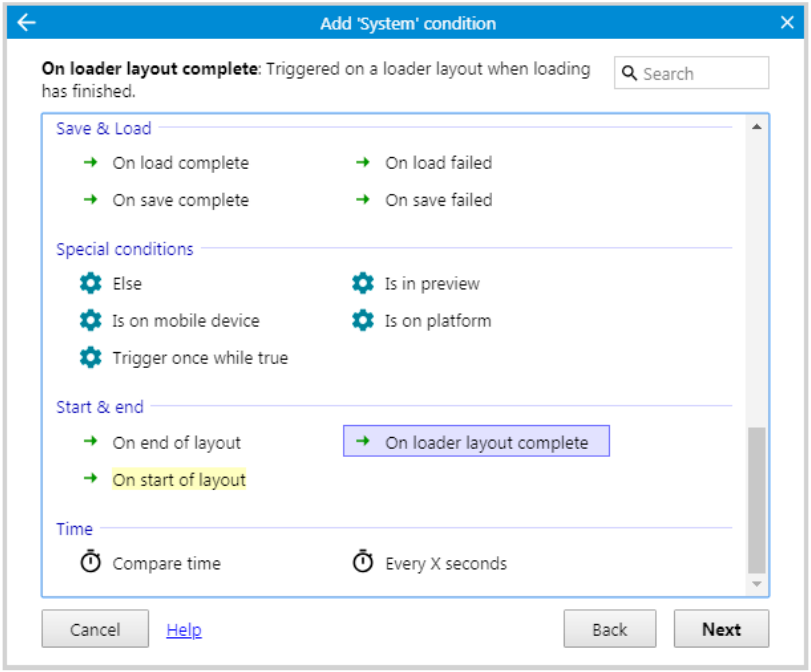

Рисунок 9

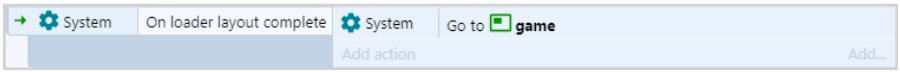

Рисунок 10

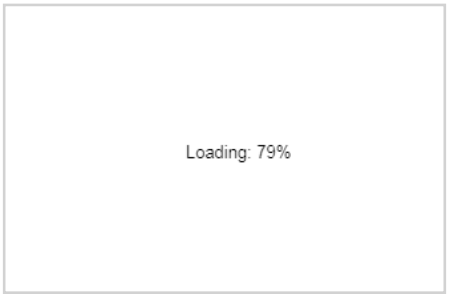

Рисунок 11

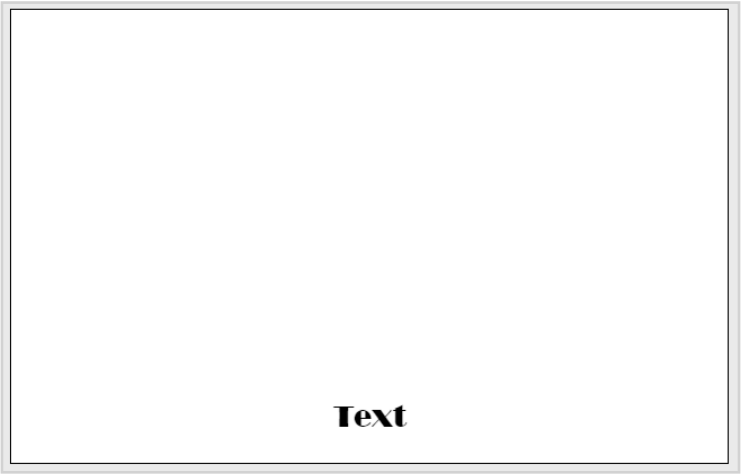

Рисунок 12

Теперь добавим «background» и два спрайта с персонажами (рис. 13).

Рисунок 13

Чтобы загрузка выглядела интересней, добавим игроку и врагу анимацию атаки. Также следует запрограммировать персонажей. Для этого переходим в цепочку событий и добавляем «event». В котором указываем, что, при старте сцены, у персонажей включится анимация (рис. 14).

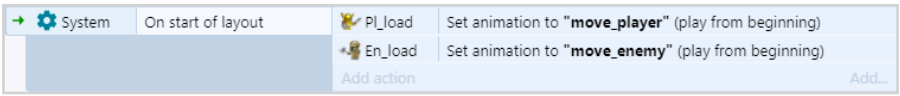

Рисунок 14

Теперь, пока загружается игра, мы видим, как противники замахиваются друг на друга (рис. 15).

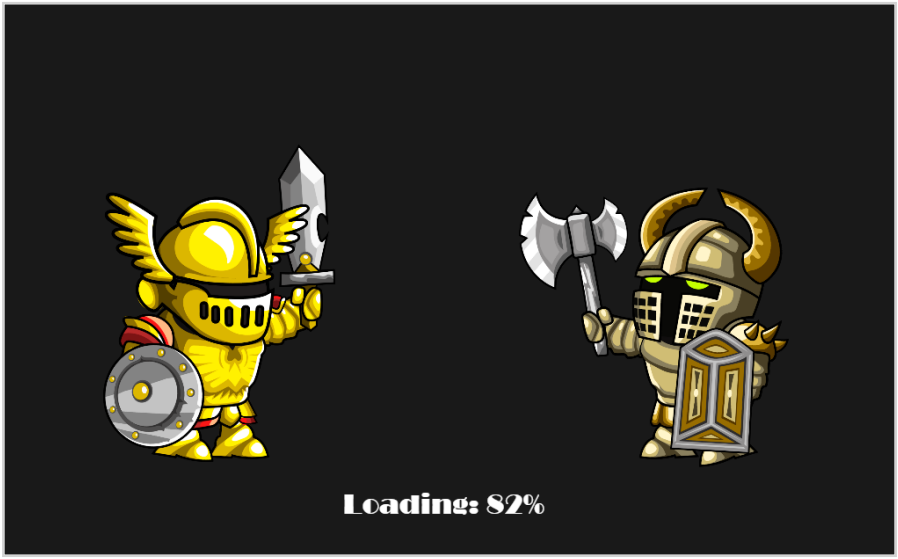

Рисунок 15

Осталось добавить небольшую подсказку игрокам. Подсказку следует оформить как элемент текста, где указана информация о том, что нужно игроку для победы.

Добавляем «Text» и задаем ему нужные параметры (рис. 16).

Рисунок 16

Рисунок 17

Созданный нами прелоадер выглядит таким образом (рис. 17):

Давайте создадим еще один вариант прелоадера, только с добавлением большего количества анимации. Это сделает прелоадер более интересным. Оставим на игровой карте только рыцаря, а также усложним анимацию: запрограммируем его так, чтобы он перемещался из стороны в сторону. В списке событий при старте сцены задаем персонажу анимацию «Run» (рис. 18).

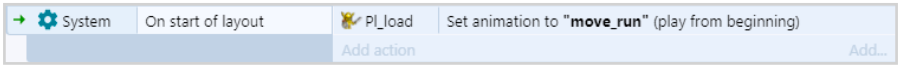

Рисунок 18

Затем переходим в игровую карту. Добавим рыцарю поведение «Sine» и зададим новые параметры (рис. 19).

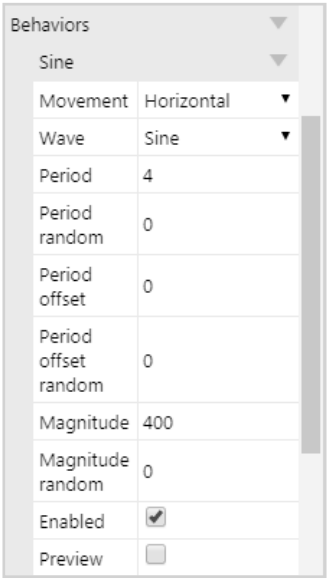

Рисунок 19

Самое сложное в новой анимации — отзеркалить персонажа при передвижении в другую сторону. Для этого воспользуемся событием «System => Is between values».

В появившемся окне прописываем следующее:

1. «Value: Pl_load.Sine.CyclePosition.»
2. «Lover bound: 0,25».
3. «Upper bound: 0,75» (рис. 20).

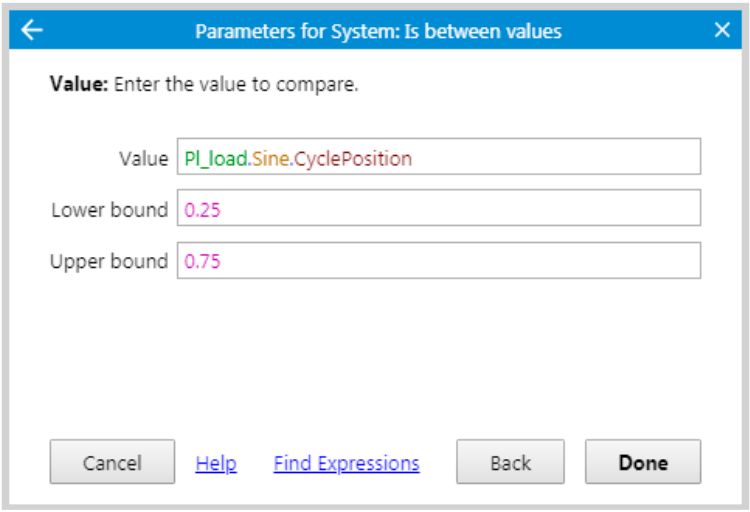

Рисунок 20

При этих условиях добавляем действие «Pl_load => Set Mirrored» (рис. 21).

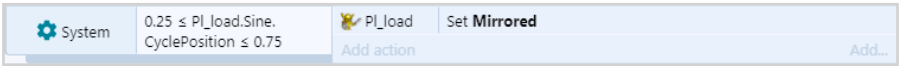

Рисунок 21

Во всех остальных случаях персонаж не должен отзеркаливаться. Поэтому прописываем «System => Else», и добавляем действие «Pl_load => Set Not mirrored» (рис. 22).

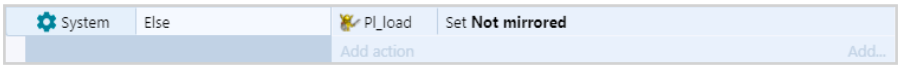

Рисунок 22

Немного изменим фон, добавив, к примеру, замок. Также следует уменьшить персонажа. В итоге получаем красивый анимированный прелоадер (рис. 23).

Рисунок 23

Посмотрите, какие красочные прелоадеры можно создавать для загрузки игры!

1. «Bastion» (рис. 24).

Рисунок 24

2. «Dungeon Boss» (рис. 25).

Рисунок 25

3. «House of Fun» (рис. 26).

Рисунок 26