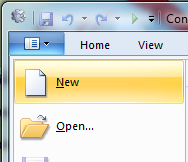
**ЗАДАНИЕ ДЛЯ УЧЕБНОЙ ПРАКТИКИ**

По дисциплине: «Обеспечение качества функционирования компьютерных систем»

**Цель:** Научиться разрабатывать браузерные демо-игры на HTML5, а так же тестирование их и вывод баг-репорт таблицы.

**Задание 1:** Разработка HTML5 демо-игры «Ghost Shooter»

**СОЗДАНИЕ ИГРЫ**  
Кликните в Construct 2 кнопку File и нажмите на New.

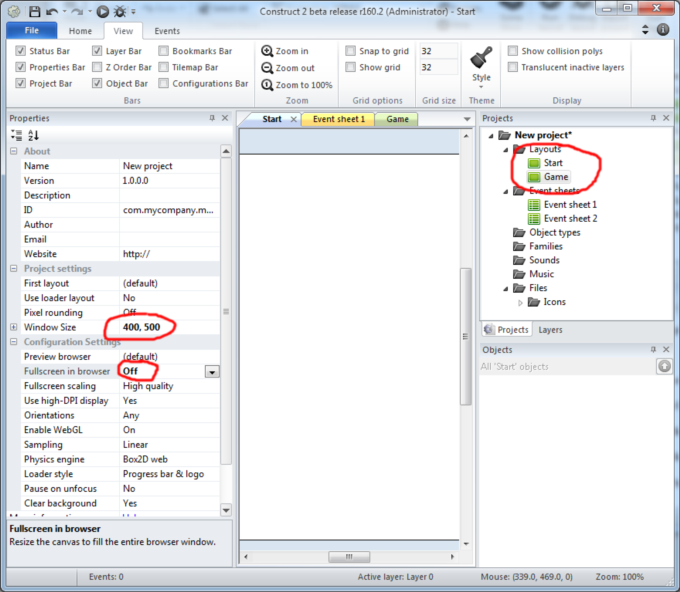
[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19639/filenew.png)

В диалоговом окне *New Project* ничего изменять не надо. Просто кликните *Create project*. Construct 2 хранит проект в *.capx* файле. Сейчас перед Вами пустой *layout* (план) – окно, где вы создаете и располагаете объекты. Layout воспринимайте как уровень игры или экран меню. В других инструментах он мог называться *room*, *scene* или *frame*.

## ШАГ 1 МЕНЮ ИГРЫ

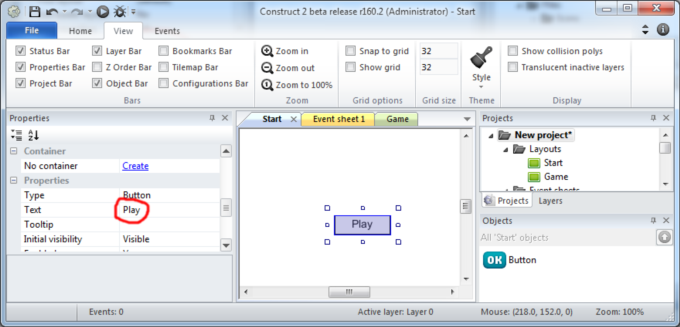
Создайте новый проект и два слоя "Start" и "Game". Установите "Layout Size" так , чтобы он имел размер "400x500".

В окне "Properties" с лева , нажмите "View" у "Project Properties". Установите "Window Size" на "400x500" и в разделе "Configuration Settings" поставьте "Fullscreen in Browser" на "Off".

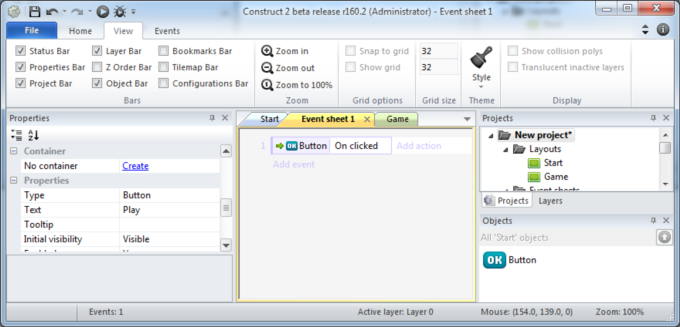
[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19212/1-33.png)

## ШАГ 2

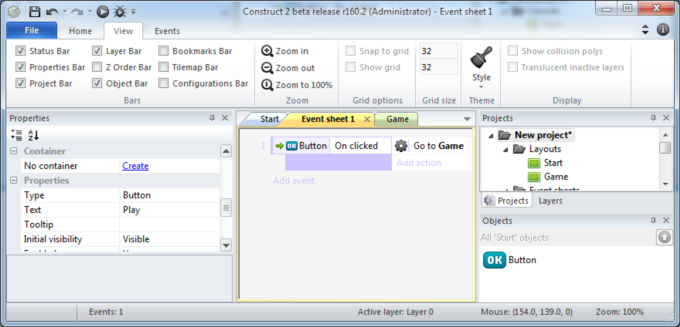
На вашем слое "Start", кликните ПКМ по пространству и выберите "Insert New Object". В разделе "Form Controls" кликните два раза по "Button". Кнопке задайте текст: "Play".

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19213/2-20.png)

Перейди в "Event Sheet" слоя "Start". Нажми "Add event". Двойной клик "Button". Выбери "On clicked". Это должно выглядеть так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19214/3-20.png)

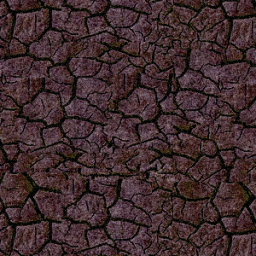
Нажми "Add action" для нового "On clicked" события. Двойной клик по "System" и двойной клик по "Go to layout". Выберите "Game" и нажмите OK. Это должно выглядеть так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19215/4-14.png)

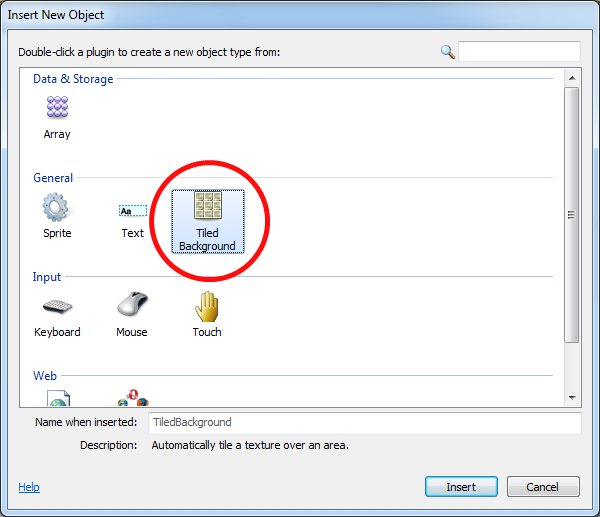
**РАСПОЛОЖЕНИЕ ОБЪЕКТОВ**

**ТАЙЛОВЫЙ ФОН**

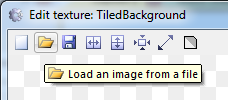
Первое, что мы создадим - повторяющиеся тайлы на заднем фоне. Объект *Tiled Background* может создать такой фон. В первую очередь возьмите текстуру фона – кликните по ней правой кнопкой мыши и сохраните на своем компьютере:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19640/bg.png)

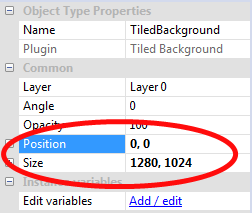
Теперь два раза кликните по плану, чтобы вставить новый объект. (Позже, если он полон, вы также можете кликнуть правой кнопкой мыши и выбрать *Insert new object* (Вставить новый объект). Когда появится диалоговое окно *Insert new object*, кликните два раза на объекте *Tiled Background* в *Insert new object*, чтобы вставить его.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19641/insertobject.png)

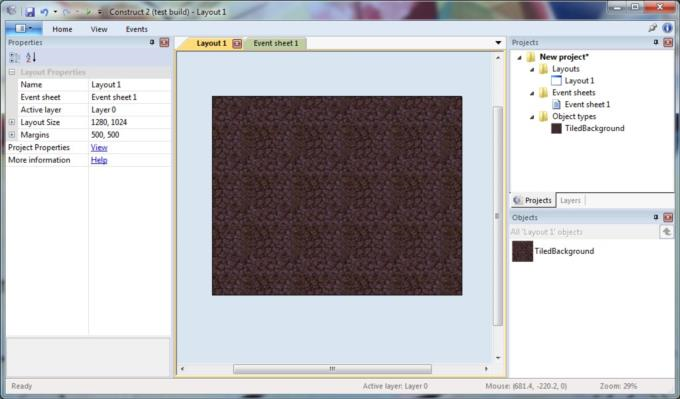
Появится перекрестие, указывающие, куда расположить объект. Кликните примерно посередине плана – откроется *texture editor* (редактор текстуры), который добавит текстуру тайлу. Давайте вставим изображение, которое вы сохрани ранее. Кликните на значок папки, чтобы загрузить текстуру с диска, найдите, куда вы скачали файл и выберите его.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19642/loadtexturefromfile.png)

Закройте редактор текстуры, кликнув *Х* в верхнем правом углу. Если отобразится вопрос - убедитесь, что Вы сохранили объект! Теперь Вы должны увидеть свой объект фона на плане. Давайте изменим размер, чтобы покрыть весь план. Убедитесь, что объект выбран, потом *Properties Bar* (Панель свойств) слева должна показать все настройки для объекта, включая его размер и расположение. Установите расположение на 0, 0 (верхняя левая область плана), а размер на 1280, 1024 (размер плана).

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19643/tiledproperties.png)

Давайте рассмотрим нашу работу. Удерживайте Ctrl и крутите колесо мыши вниз, чтобы уменьшить масштаб либо несколько раз кликните *view - zoom out*. Вы также можете удерживать пробел или среднюю кнопку мыши, для перемещения по плану. Красиво, да? Ваш фон должен покрывать весь план:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19644/tiledui.jpg)

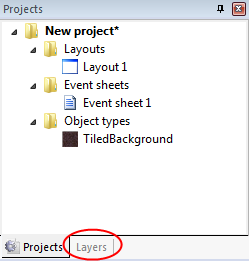
Нажмите Ctr+O или кликните *view - zoom до 100%* , чтобы вернуться к виду 1:1.

### ДОБАВЛЕНИЕ СЛОЯ

Хорошо, теперь мы хотим добавить еще объектов. Однако фон может случайно активироваться, пока мы не заблокируем его, то есть не сделаем его недоступным. Для этого давайте использовать систему слоев (layers).

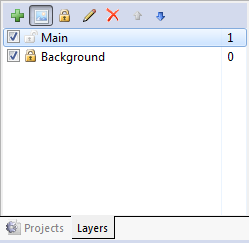
Layouts (планы) могут состоять из множества слоев, которые вы можете использовать для группирования объектов. Представьте слои листами стекла, расположенными один на другом, и на каждом из них нарисован объект. Это позволяет легко организовывать расположение объектов. Слои можно прятать, блокировать, применять эффекты параллакс и т.д. Например, в данной игре мы хотим, чтобы все отображалось поверх фона. Мы можем создать еще один слой, поверх этого, для других объектов.

Для управления слоями кликните **Layers tab**, который обычно находится рядом с Project bar:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19645/layerstab.png)

Вы должны увидеть в списке Layer 0 (Construct 2 начинает отчет с нуля, потому что это удобней для программирования). Кликните иконку карандаш и **назовите** его Background, поскольку этот слой – фон. Теперь кликните зеленую иконку 'add', чтобы добавить новый слой для других объектов. Давайте назовет его Main. Наконец, если вы кликните по маленькой иконке-замку рядом с Background, то слой блокируется, и это означает, что вы не сможете выбрать ничего на этом слое. Это удобно использовать для фона, поскольку его легко случайно выбрать. А нам он уже не нужен. Однако если вам понадобится внести изменения к фону, вы просто можете нажать на иконку-замок снова, чтобы разблокировать слой.

"Галочки" также позволяют прятать слои в редакторе, но нам эта функция пока не нужна. Ваша панель слоев (layers bar) должна выглядит так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19646/layersbar.png)

**Теперь убедимся, что выбран слой 'Main' на панели слоев**. Это важно, поскольку выбранный слой – это активный слой. Все новые объекты размещаются на активном слое, поэтому если он не выделен, мы случайно можем вставить объекты на другой слой. Активный слой показан на панели status bar, а также появляется в контекстном окне при размещении нового объекта. За этим стоит следить.

### ДОБАВЛЕНИЕ ОБЪЕКТОВ

**Обратите внимание на план снова.** Кликните два раза кнопкой мыши, чтобы вставить новый объект. Теперь выберите объект **Mouse**, поскольку нам потребуется размещение объектов с помощью мыши. Сделайте то же и для объекта **Keyboard**.

Прмечание: эти объекты не надо располагать на плане. Они спрятаны и автоматически распространяются на весь проект. Теперь в нашем проекте мы можем размещать объекты, как с помощью мыши, так и клавиатуры.

### ОБЪЕКТЫ ИГРЫ

Время разместить наши объекты! Вот ваши текстуры – сохраните их на диск, как вы делали это ранее.

Игрок:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19647/player.png)

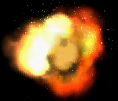
Монстр:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19648/monster.png)

Пуля:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19649/bullet.png)

И Взрыв:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19650/explode.png)

ВЫШЕ УКАЗАНЫ ПРИМЕРЫ ИЗОБРАЖЕНИЕ, ДЛЯ РАЗРАБОТКИ ВОЗЬМИТЕ ИЗОБРАЖЕНИЯ В ИНТЕРНЕТЕ!   
У КАЖДОГО СТУДЕНТА ДОЛЖЕН БЫТЬ СВОЙ СОБСТВЕННЫЙ СТИЛЬ ИГРЫ И ИЗОБРАЖЕНИЯ!

Для каждого из этих объектов мы будем использовать sprite(спрайт) объект. Он просто отображает текстуру, которую можно перемещать, вертеть и менять размер. Игры в основном состоят из спрайтов. Давайте поместим каждый из четырех объектов как спрайт объект. Процесс похож на размещение фона:

1. **Два раза** кликните ЛКМ на insert a new object (вставить новый объект).

2. **Два раза** кликните ЛКМ на 'Sprite' object (спрайт объект).

3. Когда курсор превратиться в перекрестие, кликните на плане. Контекстное окно должно быть 'Main'. (Помните, что это активный план).

4. .Выскакивает редактор изображений. Кликните по открывшейся иконке и **загрузите одну из четырех текстур**.

5. **Закройте** редактор изображений, сохранив изменения.

Внимание: другой способ быстро вставить спрайт объект – это перетащить файл изображения с Windows на область плана. Construct 2 создаст спрайт с этой текстурой. Однако помните, что каждый раз можно перенести лишь один объект. Если вы попробуйте переместить все четыре сразу, то программа создаст один спрайт и четыре кадра анимации.

Переместите спрайты пули и взрыва за край плана – нам незачем видеть их, когда начнётся игра.

Эти объекты называются Sprite, Sprite2, Sprite3 и Sprite4, что неудобно – в названиях легко запутаться. Переименуйте их на Player (Игрок), Monster (Монстр), Bullet (Пуля) и Explosion (Взрыв). Вы можете сделать это, выбрав объект, и изменив **Name** на панели свойств:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19651/objectname.png)

## ДОБАВЛЕНИЕ ПОВЕДЕНИЙ

Поведения – это предварительные функциональные возможности Construct 2. Например, вы можете добавить поведение Platform к объекту и поведение Solid к полу, и вы сразу же можете прыгать вокруг, как игрок в игре-платформе. Вы можете сделать то же в событиях, но это занимает больше времени, и нет смысла заниматься этим, если поведение уже готово! Давайте посмотрим на виды поведений, которые мы можем использовать. Среди всех видов поведений в Construct 2 есть следующие:

- **8 Direction movement** (Движение по 8 направлениям). Позволяет объекту двигаться с помощью клавиш-стрелок. Удобно для движения игрока.

- **Bullet movement** (Движение пули). Это поведение просто двигает объект прямо под готовым углом. Это хорошо применимо для пуль игрока. Несмотря на название, оно также хорошо подходит для движения монстра, поскольку предоставляет возможность двигать объектами прямо на определенной скорости.

- **Scroll to** (Прокручивание). При прокручивании экран следует за объектом, когда он перемещается (также известно как scrolling). Удобная функция для игрока.

- **Bound to layout** (Ограничение по плану). Функция не позволяет объекту выходить за границы плана. Также удобно для игрока – они не могут выходить за территорию игры!

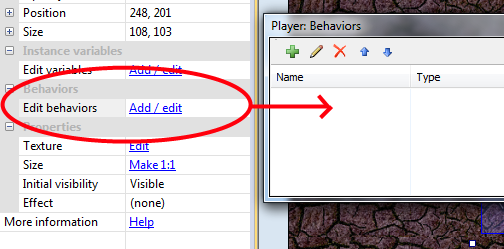
- **Destroy outside layout** (Уничтожение объектов за границами плана). Вместо того чтобы останавливать объект, когда он уходит за границы плана, эта функция уничтожает объект. Этот режим удобен для пули. Без данной функции пули улетали бы с экрана и поглощали бы больше памяти и мощности процессора. Функция уничтожает пули, как только они выходят за границы экрана.

- **Fade** (Затухание). Функция создает эффект постепенного исчезания. Мы будем использовать ее для взрывов.

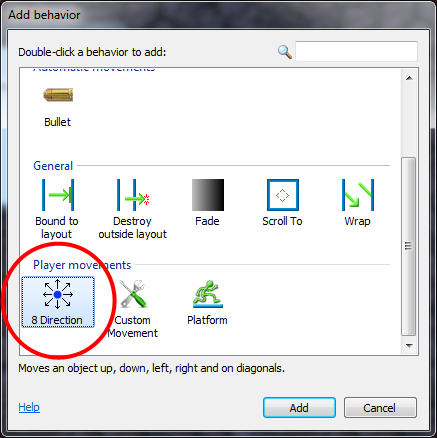
Давайте добавим эти виды режимов необходимым объектам.

### КАК ДОБАВИТЬ ПОВЕДЕНИЕ

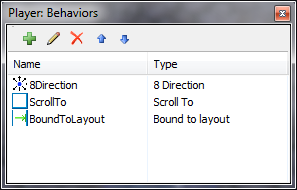
Давайте добавим **8 direction movement** игроку. Кликните по игроку, чтобы выделить его. На панели свойств найдите категорию Behaviors и кликните Add / Edit. Откроется диалоговое окно режимов для игрока.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19652/openbehaviors.png)

Кликните по зеленой иконке 'add behavior' (добавить поведение) в диалоговом окне режимов. Чтобы добавить **8 direction movement** кликните по нему два раза.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19653/add8dir.png)

Сделайте то же, но теперь добавьте **Scroll To**, чтобы экран следовал за игроком, и поведение**Bound to layout**, чтобы игрок оставался внутри экрана. Окно поведений должно выглядеть так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19654/playerbehaviors-2.png)

Закройте диалоговое окно. Нажмите Run, чтобы запустить игру!

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19655/runbutton.png)

Надеюсь, что у Вас установлен браузер, совместимый с HTML5. Если нет, то обязательно установите последнюю версию Firefox или Chrome, или Internet Explorer 9, если у вас установлена Vista или другая из последних версий Windows. Когда игра будет включена, обратите внимание, что Вы уже можете передвигаться, используя клавиши – стрелки, и экран следует за игроком! Вы также не можете перемещаться за пределы плана благодаря режиму Bound to Layout. Вот для чего используются поведения – быстрое добавление основных функций. Скоро мы будем использовать систему событий, чтобы добавить специальные функции.

### ДОБАВЛЕНИЕ ДРУГИХ ПОВЕДЕНИЙ

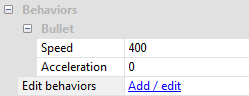
Мы можем добавлять поведения для других объектов таким же методом – выбрать, кликнуть Add / Edit для открытия диалогового окна режимов и добавить режимы. Давайте добавим другие виды поведений:

- К объекту **Bullet** (Пуля) добавьте поведения**Bullet movement** и **Destroy outside layout** (здесь без сюрпризов).

- К объекту **Monster** (Монстр) добавьте поведение**Bullet movement** (Монстр также перемещается по прямой линии).

- К объекту **Explosion** (Взрыв) добавьте поведение**Fade** (чтобы взрыв медленно исчезал). По умолчанию режим Fade также уничтожает объект после его исчезновения, что также предотвращает скопление невидимых объектов в игре.

Если вы запустите игру, вы заметите, что монстры появляются довольно внезапно. Давайте замедлим их движение. Выберите объект **Monster**. Обратите внимание, что после добавления поведения на панели свойств появились дополнительные свойства:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19656/bulletproperties.png)

Это позволяет налаживать работу поведений. Измените скорость с **400** до **80** (в пикселях за секунду). Так же измените скорость объекта **Bullet** до 600, а **Fade out time** (Время исчезновения) в режиме **Fade** объекта **Explosion** до **0.5** (полсекунды).

### СОЗДАНИЯ НОВЫХ МОНСТРОВ

Удерживая клавишу Ctr, кликните и перетащите объект **Monster**. Вы заметите, что это создает другой экземпляр. Это просто другой объект вида Monster.

Виды объектов являются важными «классами» объектов. В системе событий вы в основном работаете с видами объектов. Например, вы можете создать событие под названием «Пуля сталкивается с Монстром». Эта подразумевает «любую ситуацию, когда объект Bullet сталкивается с любой копией объекта Monster» - это позволяет не создавать отдельное событие для каждого монстра. Все ситуации со спрайтами также имеют одну текстуру. Это очень эффективно: когда игроки играют он-лайн, нет необходимости загружать 8 текстур для 8 монстров. Надо лишь загрузить одну текстуру монстра и программа повторит ее 8 раз. Позже мы детальнее рассмотрим виды объектов и ситуаций. Пока что пример для размышления: разные виды врагов являются разными видами объектов, а сами враги (а их может быть несколько) являются экземплярами этих видов объектов.

Удерживая клавишу Ctr + тащить, **создайте 7 или 8 новых монстров**. Не размещайте их слишком близко к игроку – их сразу же убьют! Вы можете увеличить масштаб, нажав клавишу Ctr и используя колесо мыши, если это удобно, и распространить их по всему плану. Должно получиться что-то похожее.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19657/severalghosts.png)

Теперь время добавить специальные функции через визуальный метод программирования Construct 2– event system (система событий).

## СОБЫТИЯ

Вначале кликните по ярлыку Event sheet 1 (Лист событий) наверху, чтобы перейти в Event sheet editorредактор . Список событий называется Event sheet. Вы можете использовать разные листы событий для разных частей игры или для организации игры. В листы событий также можно «включать» другие листы событий, что позволяет нам заново использовать события на разных уровнях. Но сейчас нам это не понадобится.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19658/eventsheettab.png)

### ПРО СОБЫТИЯ

Как указывает текст на пустом листе, Construct 2 обновляет все на листе событий один раз за тик(tick). Большинство мониторов обновляют экран 60 раз в секунду, поэтому Construct 2 старается подстраиваться под это, чтобы обеспечить лучшее отображение. Это значит, что лист событий в основном обновляется 60 раз в секунду, за чем следует повторная прорисовка экрана. Вот, что такое тик – единица «обновления событий и повторной прорисовки экрана».

События происходят сверху вниз, поэтому первыми происходят события наверху.

### УСЛОВИЯ, ДЕЙСТВИЯ И ПОД-СОБЫТИЯ

События состоят из **условий**, которые проверяют соответствие определенным критериям, например «Пробел зажат?». Если все условия соблюдены, происходят все **действия**в событии, например «Создание объекта пули». После всех действий происходят **под-события**, которые могут проверять соответствие другим условиям, создавать больше действий, больше под-событий и так далее. Используя эту систему, мы можем создавать сложные функциональные возможности для наших игр и приложений. Однако в данном руководстве нам не понадобятся под-события.

Давайте повторим. Коротко говоря, событие в основном происходит так:

Соблюдены ли все условия?

---> **Да**: выполнение всех действий события.

---> **Нет**: переход на следующее событие (без под-событий).

Это немного упрощённо. Construct 2 предоставляет много возможностей событий, которые подходят для самых разных действий. Однако пока что не стоит об этом думать.

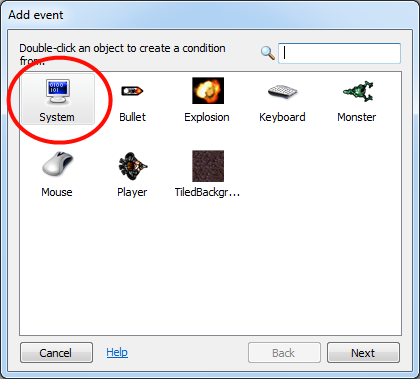
### ВАШЕ ПЕРВОЕ СОБЫТИЕ

Мы хотим, чтобы игрок всегда смотрел по направлению мыши. Когда мы закончим, это будет выглядеть следующим образом:

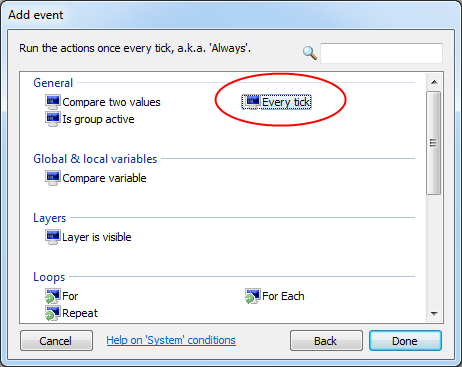
[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19659/alwayslookatmouse.png)

Помните о том, что тик происходит каждый раз, когда заново прорисовывается экран, поэтому если мы сделаем так, чтобы игрок смотрел по направлению мыши при каждом импульсе, он всегда будет по направлению мыши.

Давайте начнем создавать это событие. Кликните два раза на листке событий. Это поможет нам добавить **условие** для нового события.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19660/newevent-2.png)

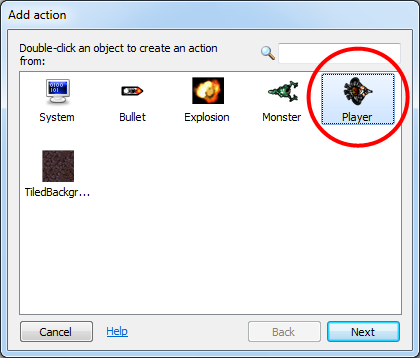
Разные объекты имеют разные условия и действия в зависимости от того, что они могут делать. Существует также **System object** (объект системы), который представляет собой встроенную функцию Construct 2. **Два раза** кликните по System object, как показано на рисунке. В диалоговом окне будут показаны все условия объекта System.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19661/everytickcnd.png)

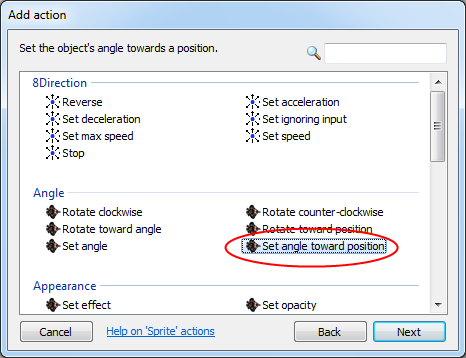
Кликните **два раза** по условию Every tick (всегда), чтобы вставить его. Диалоговое окно закроется, и будет создано событие без действий. Оно должно выглядеть следующим образом:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19662/everytickempty.png)

Теперь мы хотим добавить действие, чтобы игрок смотрел на мышь. Кликните линк Add action справа от события. (Убедитесь, что вы нажимаете линк Add action, а **не** Add event под ним, поскольку он добавит новое событие). Появится диалоговое окно Add Action (Добавить действие):

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19663/addactiondlg.png)

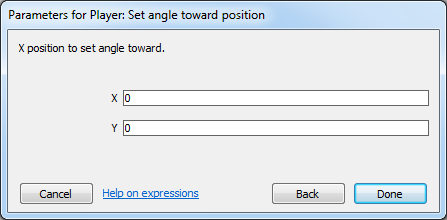
Как и при добавлении события у нас есть тот же список объектов, но в данном случае для добавления действия. Старайтесь не запутаться между добавлением условий и действий! Как показано на рисунке, кликните **два раза** по объекту Player, поскольку нам надо, чтобы на мышку смотрит именно игрок. Появится список действий, доступных для игрока:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19664/playersetanglepos.png)

Обратите внимание, что поведение 8-direction movement игрока имеет собственный список действий. Но пока что нет необходимости об этом думать.

Вместо ручной настройки угла игрока удобно использовать действие **Set angle towards position**, которое автоматически просчитает угол от игрока к текущим координатам X и Y, и установит угол объекта к ним. Кликните **два раза** по действию Set angle towards position.

Теперь Construct'у 2 необходимо знать координаты X и Y, на которые надо направлять игрока:

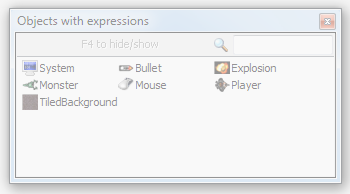
[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19665/setangleposparams.png)

Это **параметры** действия. Условия также могут иметь параметры, но Every tick не нуждается в параметрах.

Мы хотим установить угол по направлению к местоположению мыши. Это может сделать объект Mouse. Введите **Mouse.X** для X, и **Mouse.Y** для Y. Это называется expressions. Они как просчитанные суммы. Например, вы также можете ввести Mouse.X + 100 или sin(Mouse.Y) (хотя именно эти примеры могут не пригодиться!). Таким образом, вы можете использовать любые данные с любого объекта или любые подсчеты для разработки параметров в действиях и условиях. Это очень мощный инструмент и скрытый источник гибкости работы в Construct 2.

Если у Вас появилась ошибка «Mouse is not an object name?». Убедитесь, что вы добавили объект Mouse. Вернитесь к странице 2 и проверьте «Add the input objects».

Возможно, вы думаете, как вы можете запомнить всевозможные выражения, которые надо вводить. К счастью, есть object panel (панель объектов). По умолчанию панель исчезает, чтобы не отвлекать вас.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19666/objectpanel.png)

Подведите курсор к панели или кликните по ней, и она становится видимой. Она служит словарем всех выражений, которые вы можете использовать. Она содержит описание выражений, что помогает запомнить их. Если вы кликните два раза по объекту, вы увидите все выражения. Если вы кликните два раза по выражению, то оно будет вставлено в нужное место, что исключает необходимость печатать выражение.

В любом случае кликните **Done** на окне параметров. Действие добавлено! Как вы видели раньше, оно должно выглядеть так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19659/alwayslookatmouse.png)

Это ваше первое событие! Попробуйте запустить игру. Теперь игрок должен двигаться, как раньше, но теперь он всегда направлен по направлению мыши. Это наша первая специальная функция.

**ДОБАВЛЕНИЕ ФУНКЦИЙ ИГРЕ**

Если мы будем объяснять каждое событие так же детально, как раньше, руководство будет очень большим. Давайте сократим описание следующих событий. Помните следующие шаги для добавления условия или действия:

1. Два клика, чтобы добавить новое событие или один клик для линка *Add action*, чтобы добавить действие.

2. Два клика по объекту, в котором находится условие/действие.

3. Два клика для условия/действия, которое вы хотите добавить.

4. Введите параметры, если есть необходимость.

Теперь события будут описываться как объект, за которым следует условие/действие, а потом параметры. Например, событие, которое мы только что добавили, можно написать так:

**Add condition [i]System[/i] -> [i]Every tick[/i]**

**Add action [i]Player[/i] -> [i]Set angle towards position[/i] -> X: [i]Mouse.X[/i], Y: [i]Mouse.Y[/i]**

**ИГРОК СТРЕЛЯЕТ**

Когда игрок кликает мышкой, должна вылетать пуля. Этого можно достичь действием *Spawn an object* в объекте Player, что создает новую ситуацию объекта при тех же параметрах положения и угла. Добавленная ранее *Bullet movement*, позволит пуле лететь вперёд. Создайте следующее событие:

**Condition: [i]Mouse[/i] -> [i]On click[/i] -> Left clicked (the default)**

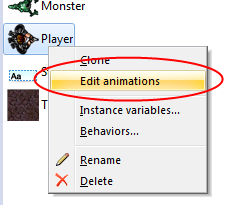
**Action: [i]Player[/i] -> [i]Spawn another object[/i] -> For [i]Object[/i], choose the [i]Bullet[/i] object. For [i]Layer[/i], put [b]1[/b] (the "Main" layer is layer 1 - remember Construct 2 counts from zero). Leave [i]Image point[/i] as 0.**

Ваше событие должно выглядеть так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19668/spawnbullet1.png)

Если вы запустите игру, вы заметите, что пули стреляют из тела игрока, а не из пистолета. Давайте исправим это, поместив image point на конце пистолета. (Image point (точка изображения) – это точка на изображении, из которого вы можете выбрасывать объекты).

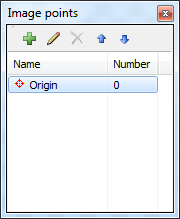
Кликните правой кнопкой мыши по игроку на проекте или панели объектов и выберите Edit animations.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19669/editanimations.png)

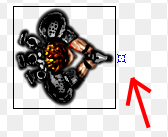
Снова появится редактор изображения. Кликните по инструменту origin and image points (источник и точки изображения):

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19670/imagepointstool.png)

…и откроется диалоговое окно image points:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19671/imagepointsdlg.png)

Обратите внимание, что центр объекта появляется в виде красной точки. Это «горячая точка» или «точка вращения» объекта. Если вы вращаете объект, он крутится вокруг центра. Мы хотим добавить еще одну точку для пистолета, поэтому кликните зеленую кнопку *add*. Появится синяя точка – это наша новая точка изображения. Кликните левой кнопкой мыши на краю пистолета, чтобы разместить там точку изображения:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19672/placingimagepoint.png)

Закройте редактор изображения. Кликните два раза по действию *Spawn an object*, которое мы добавили ранее и измените *Image point* на 1. (Центр всегда является первой точкой изображения. Помните, что Construct 2 начинает отсчет с нуля). Событие должно выглядеть так, как показано на рисунке ниже. Обратите внимание: теперь на нем написано *Image point 1*:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19673/spawnbullet2.png)

Запустите игру. Теперь пули вылетают с пистолета! Однако пока пули ничего не делают. Надеюсь, что, вы понимаете, что когда вы разберетесь с системой событий, вы легко сможете составить функции игры.

Давайте сделаем пули активными – которые могут убивать монстра. Добавьте следующее событие:

**Condition: [i]Bullet[/i] -> [i]On collision with another object[/i] -> pick [i]Monster[/i].**

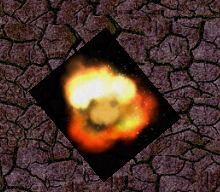
**Action: [i]Monster[/i] -> [i]Destroy[/i]**

**Action: [i]Bullet[/i] -> [i]Spawn another object[/i] -> [i]Explosion[/i], layer [b]1[/b]**

**Action: [i]Bullet[/i] -> [i]Destroy[/i]**

**ЭФФЕКТ ВЗРЫВА**

Запустите игру и попробуйте стрелять в монстра. Ой! Взрыв имеет большую черную границу!

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19674/explosionnoadditive.png)

Возможно, Вы предполагали, что так оно должно выглядеть и задумывались, останется ли этот эффект! Не беспокойтесь – нет. Кликните по Explosion object на Object bar (панели объектов) в нижнем правом углу или по Project bar (панели проектов) (которая была табулирована с layers bar (панель слоев). Слева появятся свойства на properties bar (панели свойств). Внизу установите настройки Blend mode*в режим*Additive\*. Теперь попробуйте запустить игру снова.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19675/explosionadditive.png)

Почему это работает? Если объяснить без деталей – простые изображения *вставляются поверх экрана*. С помощью эффекта Additive каждый пиксель *добавляется* (суммируется) с пикселем фона за ним. Черный – это нулевое значение пикселя, поэтому ничего не добавляется – вы не видите черный фон. Более яркие цвета добавляют больше, поэтому кажутся более сильными. Они отлично подходят для взрыва и световых эффектов.

**ДЕЛАЕМ МОНСТРА НЕМНОГО УМНЕЕ**

На данный момент монстры уходят с плана направо. Давайте сделаем их немного интересней. Вначале давайте запустим их под произвольным углом.

**Condition: [i]System[/i] -> [i]On start of Layout[/i]**

**Action: [i]Monster[/i] -> [i]Set angle[/i] -> random(360)**

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19676/ghostshooterevent4.png)

Они все равно будут исчезать, когда покидают границы плана. Давайте оставим их в границах плана. Мы направим их обратно на игрока, когда они будут подходить к границе. Это имеет два эффекта: они всегда остаются в границе плана и, если игрок остается на месте, монстры идут прямо на него!

**Condition: [i]Monster[/i] -> [i]Is outside layout[/i]**

**Action: [i]Monster[/i] -> [i]Set angle toward position[/i] -> For X, [b]Player.X[/b] - for Y, [b]Player.Y[/b].**

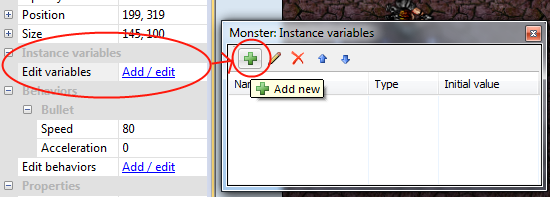
Запустите игру. Если вы поиграете немного, вы заметите, что монстры так же, как игрок, остаются в рамках плана и движутся в разных направлениях. Это не AI, но сойдет и так!

Теперь, я думаю, мы хотим, чтобы монстра можно было убить с пяти выстрелов, а не с одного. Как мы это сделаем? Если у нас есть только один "Health counter" (счетчик здоровья), то когда мы выстрелим в одного монстра пять раз, умрут *все* монстры сразу. Нам необходимо, чтобы *каждый* монстр помнил собственный уровень здоровья. Мы можем сделать это с instance variables (переменные).

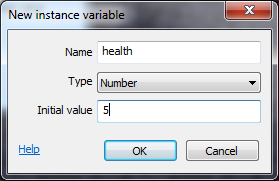
## ПЕРЕМЕННЫЕ

Переменные позволяют каждому монстру иметь собственное значение здоровья. Переменная – это просто значение, которое может меняться. Они хранятся в разных местах для каждой ситуации, поэтому называются переменными.

Давайте добавим переменную health (здоровье) для нашего монстра. Кликните по монстру на панели проектов или панели объектов. Как вариант, вы можете переключиться обратно на план и выбрать объект монстр. Откроются свойства монстра на панели свойств. Кликните **Add/edit** на **Edit variables**.

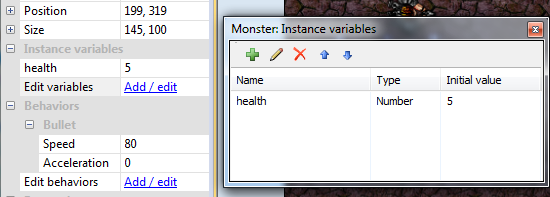
[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19677/instvars.png)

Появится окно Instance Variables. Оно похоже на диалоговое окно Behaviors (Режимов), которое мы видели ранее, но в данном окне вы можете изменять переменные ситуации для объекта. Кликните по зеленой кнопке **Add**, чтобы добавить новую переменную.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19678/healthinstvar.png)

Во всплывающем окне напечатайте название **health**, оставьте Type в виде **Number**, а для Initial value введите **5**. Так мы создали изначальный уровень здоровья для каждого монстра. Когда по монстру стреляют, мы вычисляем 1 со здоровья, а когда здоровье достигнет нуля, мы уничтожим объект.

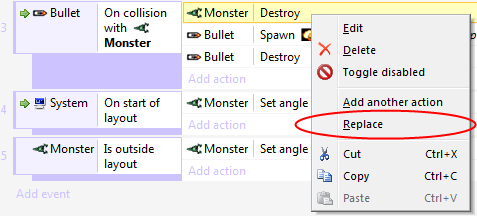
Когда вы закончите, кликните OK. Обратите внимание, что теперь переменная появляется в диалоговом окне переменных ситуаций, а также в свойствах монстра. (Вы можете быстро изменить первоначальные значения на панели свойств, но для того, чтобы добавить или убрать переменные, надо кликнуть по линку Add / Edit).

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19679/healthadded.png)

### ИЗМЕНЕНИЕ СОБЫТИЙ

Переключитесь на event sheet (лист событий). Сейчас монстр умирает, как только пуля попадает в него. Давайте изменим это на subtract 1 (вычитать 1) из его здоровья.

Найдите событие, которое называется Bullet - on collision with Monster. Обратите внимание, что у вас есть действие "destroy monster" (уничтожить монстра). Давайте заменим его на "subtract 1 from health" (вычитать 1 из здоровья). Кликните правой кнопкой мыши по действию "destroy monster" и кликните **Replace**.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19680/replaceaction.png)

Появится то же диалоговое окно, как при создании нового действия, но теперь вместо создания действия, будет произведена замена действия, на которое мы нажали. Выберите Monster -> Subtract from (в категории Instance variables ) -> Instance variable "health", и введите **1** для Value. Кликните **Done**. Теперь действие должно выглядеть так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19681/sub1fromhealth.png)

Теперь, когда мы стреляем в монстра, он теряет 1 здоровье, пуля взрывается. Но мы еще не создали событие, при котором монстр будем умирать, когда уровень здоровья достигает нуля. Добавьте еще одно событие:

**Condition: [i]Monster -> Compare instance variable ->[/i] Health, [i]меньше или равно нулю[/i], 0**

**Action: [i]Monster -> Spawn another object ->[/i] Explosion, layer [b]1[/b]**

**Action: [i]Monster -> Destroy[/i]**

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19682/monsternohealth.png)

Почему меньше или равно нулю, а не просто равно нулю? Предположим, мы добавили еще одно более мощное оружие, которое уменьшило уровень здоровья монстра на **2**. Когда вы стреляете в монстра, его здоровье может быть **5**, **3**, **1**, **-1**, **-3**.... Обратите внимание, что при любом из этих показателей уровень здоровья не был равен нулю. Поэтому при значении равно нулю монстр бы не умер! Поэтому полезно попрактиковаться со значением «меньше или равно» для определения уровня здоровья объекта.

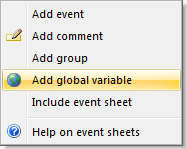
Запустите игру. Теперь, чтобы убить монстра, необходимо выстрелить в него пять раз.

## СЧЕТ

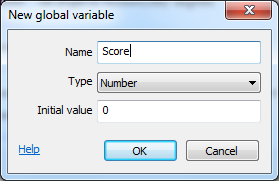
Давайте создадим счет, чтобы игрок знал свою продуктивность. Для этого нам понадобится еще одна переменная. Вы можете подумать «давайте поставим счет, как одну из переменных ситуаций игрока!». Вначале это кажется неплохой идеей, но помните, что значение хранится в объекте. Если нет ситуаций, не будет и переменных! Поэтому если мы уничтожим игрока, мы не будем знать счет, потому что он тоже будет уничтожен.

Мы может использовать **global variable** (глобальная переменная). Как и переменная ситуации, глобальная переменная (или просто "global") может хранить текст или цифру. Каждая переменная может хранить одну цифру или отдельный кусочек текста. Глобальные переменные также доступны для всей игры во всех планах, что удобно при добавлении других слоев.

Кликните **правой кнопкой** мыши на нижней части листка событий и выберите Add global variable (Добавить глобальную переменную).

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19683/addglobal.png)

Введите имя **Score** (Счет). Другие значения по умолчанию подходят нам. Они начинают отсчет с нуля.

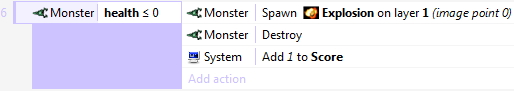
[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19684/addglobalscore.png)

Глобальная переменная появится в виде линии на листке событий. Она находится на этом листке событий, но доступна на любом листке событий и на любом плане.

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19685/globalscorevar.png)

Внимание: существуют также локальные переменные, которые доступны только для некоторых событий, но мы их не рассматриваем.

Давайте дадим игроку очко за убийство монстра. В нашем событии "Monster: health less or equal 0" (когда монстр умирает), кликните **Add action**, и выберите System-> Add to (under Global & local variables) (под глобальными и локальными переменными)-> **Score**, value **1**. Теперь событие должно выглядеть так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19686/scoreeevent.png)

Теперь у игрока есть счет, который увеличивается на 1 за каждого убитого монстра. Но игрок не видит счет! Давайте покажем его с помощью объекта текст.

## СОЗДАНИЕ HEADS-UP DISPLAY (HUD)

HUD – это интерфейс, который показывает уровень здоровья игрока, счет и другую информацию в игре. Давайте создадим простой HUD из объекта текст.

HUD всегда остается в одном месте на экране. Если у нас есть объекты интерфейса, нам не надо, чтобы они исчезали, когда игрок перемещается – они должны оставаться на одном месте. По умолчанию, слои перемещаются. Чтобы удержать их на экране, мы можем использовать установку слоя **Parallax**. Parallax отображает перемещение разных слоев на разных уровнях при эффекте полу-3D. Однако если мы установим parallax на ноль, слой вовсе не будет перемещаться – идеально для HUD.

Перейдите назад к панели слоев, которую мы использовали раньше. Добавьте новый слой с названием HUD. Убедитесь, что он находится поверх остальных слоев и что вы его выбрали (помните, что при этом этот слой активный). Свойства теперь должны отображаться на панели свойств. Установите свойства **Parallax** на значение **0, 0** (ноль на обеих осях - X и Y).

Кликните **два раза** по области, чтобы вставить еще один объект. Теперь выберите объект **Text**. Поместите его в верхнем левом углу плана. Его сложно будет увидеть, если он будет черным, поэтому на панели свойств установите шрифт на значения жирный, курсив, желтый и выберите больший размер. Он должен выглядеть примерно так:

[](https://construct-static.com/images/v1228/uploads/articleuploadobject/0/images/19687/textinlayout.png)

Перейдите назад к листу событий. Давайте установим обновление текста со счетом игрока. В событии **Every tick**, которое мы добавили ранее, добавьте действие Text -> Set text.

Используя оператор **&**, мы можем преобразовать цифру в текст и присоединить его к другой цепочке текста. Для текста введите следующее:

**"Score: " & Score**

Первая часть ("Score: ") означает, что текст будет всегда начинаться фразой Score:.Вторая часть (Score) – настоящее значение глобальной переменной счета. Значок **&** объединяет их в один текст.

Запустите игру и стреляйте по монстрам. Счет отображается и всегда остается на одном месте на экране.

## ПОСЛЕДНИЕ ШТРИХИ

Мы почти закончили. Давайте внесем последние штрихи.

Вначале давайте создадим регулярное появление монстров. Иначе, когда вы застрелите всех монстров, больше ничего не останется делать. Новый монстр будет появляться каждые 3 секунды. Добавьте новое событие:

**Condition: [i]System[/i] -> [i]Every X seconds[/i] -> [b]3[/b]**

**Action: [i]System[/i] -> [i]Create object[/i] -> [b]Monster[/b], layer [b]1[/b], [b]1400[/b] (for X), [b]random(1024)[/b] (for Y)**

1400 – это координата Х с правого края плана, а random(1024) - это произвольная координата Y высоты плана.

Наконец, давайте создадим призрака, который убивает игрока.

Condition: Monster -> On collision with another object -> **Player**

Action: Player -> Destroy

## РАСШИРЕНИЕ

1. Добавьте ВТОРОЙ и ТРЕТИЙ уровень игры через Layout. ВАЖНО, чтобы уровни шли по увеличению сложности, с помощью перехода из одного уровня в другой (Это может быть портал, дверь, тунель - на выбор, при условии, что перейти на другой уровень нельзя, не пройдя задание на предыдущем)
2. Добавить объекты для взаимодействия с персонажем: наносят урон, ничего не делают, мешают, такие объекты которые можно разрушить атакой игрока. (например: стена которая разрушается атакой персонажа)
3. Добавьте еще функции, такие как текст "Game over" в конце игры и «Level Passed» - при удачном прохождении, монстры, которые будут постепенно ускоряются.
4. Сделайте «Пасхалку» - отдельный раунд-бонус, в который можно попасть только при определенных (нестандартных) условиях, условия придумайте сами, а также раунд-бонус, там может быть развлекательное изображение, либо какой-то объект.

Кликните **Export** в File menu (файловое меню). Сохранить все файлы проекта в папку на компьютере, которую вы можете загрузить или интегрировать на веб страницу.

**Задание 2**: Тестирование игрового приложения и составление таблицы баг-репорта

Необходимо взять работу у одногруппника, создать баг-репорт таблицу с минимум 10 багами разного уровня критичности, а также написать 10 рекомендаций к игре.

**Оформление баг-репорта**

1. Для начала нужно убедиться, что найденный баг ещё не был оформлен. Следует провести поиск его в соответствующем проекте по всем подходящим ключевым словам и\или полям. Если баг уже есть, следует обновить его описание.
2. Если баг не найден – фиксируем в таблице багов. Не стоит забывать важное правило: один дефект - один баг в трекере.
3. Далее нужно постараться описать, что не работает - ЭТО И БУДЕТ ЗАГОЛОВОК БАГ-РЕПОРТА!!!!!!
4. После этого перейти к подробному описанию бага: указать шаги к воспроизведению.
5. Указать ожидаемый результат.
6. Указать полученный результат.
7. Указать версию ПО (по умолчанию у всех – 1.0.0), также указать версию окружения.
8. Необходимо, приложить соответствующие артефакты: скриншоты, дампы.
9. Описать рекомендации к продукту

**Состав 1-го баг репорта приведен в таблице:**

|  |  |
| --- | --- |
| Заголовок (Summary) | Короткое описание проблемы, явно указывающее на причину и тип ошибочной ситуации. |
| Проект (Project) | Название тестируемого проекта |
| Компонент приложения (Component) | Название части или функции тестируемого продукта |
| Номер версии (Version) | Версия, на которой была найдена ошибка |
| Окружение (Environment) | Модель устройства, версия операциорнной системы, характеристики на котором происходит тест |
| Критичность  (Severity) | Наиболее распространена пятиуровневая система критичности:  S1 Блокирующий (Blocker)  S2 Критический (Critical)  S3 Значительный (Major)  S4 Незначительный (Minor)  S5 Тривиальный (Trivial) |
| Приоритет (Priority) | Приоритет дефекта:  P1 Высокий (High)  P2 Средний (Medium)  P3 Низкий (Low) |
| Статус (Status) | Статус бага. Зависит от используемой процедуры и жизненного цикла бага. Например:   * Новый * Открыт * Закрыт |
| Автор (Author) | Создатель баг репорта |
| Назначен на (Assigned To) | Имя сотрудника, назначенного на решение проблемы |
| Описание (Description) | Описание бага |
| Шаги, по которым можно легко воспроизвести ситуацию, приведшую к ошибке. |
| Полученный результат |
| Ожидаемый результат |
| Прикрепленный файл (Attachment) | Файл с логами, скриншот или любой другой документ, который может помочь прояснить причину ошибки или указать на способ решения проблемы |
| Рекомендации (Recommendations) | Описать рекомендации по игре, важные элементы, моменты, которые нужно добавить или доработать с точки зрения тестрировщика. |

**Задание 3:** Создать 5 текст-кейсов, каждый тест кейс должен содержать не менее 3 пунктов с положительным, негативным и деструктивным тестом.

**Задание 4:** Предоставить все отчеты выполнения задания преподавателю.