**Комментарий:**

Этот документ отражает идеальное состояние проекта по завершении. В будущем, возможно, этот документ будет изменен. Реальные возможности и цели пока ограничены списком, который хранится в документе «Scroller early versions». Там же можно увидеть диаграммы, поясняющие работу программы.

**Scroller**

Язык написания визуальных квестов, работающих по принципу ленты в социальной сети. Название Scroller означает листать картинки.

Основным элементом в Scroller является комната. Это картинка и поля (пустое место вокруг картинки, также можно назвать фоном).   
  
Фон может быть заполнен цветом или изображением и меняется при помощи команды:

**background** цвет в формате RGB | **background** название картинки

(описание игры)

Скорость листания меняется при помощи команды:

**scrollspeed** цифра от 0 до 10

(описание игры)

Если скорость скроллинга равна нулю, то игрок не может листать комнаты, но может взаимодействовать с интерактивными объектами и диалогами.

**Комната**

Комната – это картинка и поля вокруг картинки (пустое место). Внутри комнаты находятся события, предметы, правила.

Описание комнаты:

**room** Название\_комнаты**:**

[Добавление и поведение элементов]

**end**

(блок комнаты)

У комнаты есть свойства:

**height** – высота комнаты в пикселях (по умолчанию имеет значение близкое к значению высоты экрана). Это свойство можно поменять только один раз в начале комнаты.

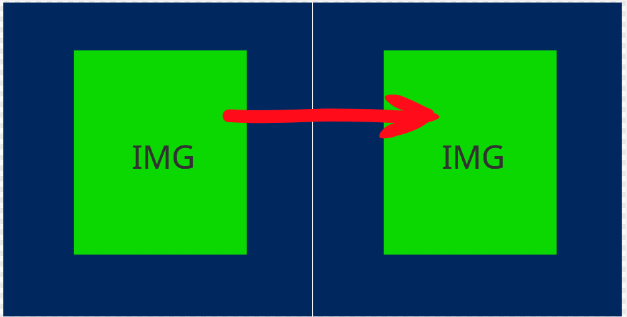
**height** цифра – значение высоты в пикселях

(описание комнаты неизменяемое)

**width** – ширина комнаты в пикселях (по умолчанию имеет значение близкое к значению ширины экрана) Это свойство можно поменять только один раз в начале комнаты.

**width** цифра – значение ширины в пикселях

(описание комнаты неизменяемое)

**Двери**

Комнаты располагаются вплотную друг к другу, так что пользователь может переходить между комнатами в четырех направлениях.

Чтобы обозначить переход между комнатами необходима «дверь». Они бывают двух видов: простые и сложные. Простые двери перелистываются со стандартной скоростью и позволяют задержаться на промежутке между комнатами. Сложная дверь «открывается» с усилием и перемещает пользователя в следующую комнату, не задерживая на промежутке между комнатами. Сложной комнатой можно обозначить необратимый переход или выбор.

**left** Название\_комнаты\_слева | **leftt** Название\_комнаты\_слева

**right** Название\_комнаты\_справа | **rightt** Название\_комнаты\_справа

**up** Название\_комнаты\_сверху | **upp** Название\_комнаты\_справа

**down** Название\_комнаты\_снизу | **downn** Название\_комнаты\_снизу

(описание комнаты изменяемое)

Переход ссылка - пересылает в другую комнату мгновенно, может быть вызван в событии.

**send** Название\_комнаты

(оператор перехода)

**Картинки**

Картинка добавляется в комнату снизу всех объектов и текста. Располагается по центру. Картину можно поменять только внутри самой комнаты.

**img** Название\_картинки

(описание комнаты изменяемое)

**Магнит**

Магнит в скроллере – это перемещение экрана игрока по комнатам и внутри комнаты, без действий самого игрока. То есть, когда человек не листает или не может листать экран (потому что skrollspeed 0) то экран двигается в нужном направлении.

Перемещает игрока к центру комнаты с указанной скоростью:

**magnet center** цифра от 0 до 10

Перемещает игрока в указанном направлении с указанной скоростью:

**magnet right** цифра от 0 до 10

**magnet left** цифра от 0 до 10

**magnet up** цифра от 0 до 10

**magnet down** цифра от 0 до 10

(описание комнаты изменяемое)

Магнит действует только внутри комнаты, где описан. То есть, когда игрока сносит налево и он переходит в левую комнату, то он останавливается.

**Переменные**

Переменные нужны для обозначения какого-то действия и сохранения данных. Их применения в Scroller:

1. Запоминание. Нужно запомнить имя игрока или количество каких-то объектов, для дальнейшего применения и изменения в ходе игры
2. Удобное обращение. Названия шрифтов, картинок, значение цвета в rgb формате, размеры комнат иногда удобнее сохранить в переменных, если планируешь к ним часто обращаться.
3. Обозначение действий. Если игрок уже кликнул по предмету или уже был один раз в какой-то комнате, то данные об его воздействии на мир нужно записать в переменной.

Названные с большой буквы переменные передаются между комнатами, их область видимости – это текстовый файл с комнатами. Они могут быть изменены или вызваны в любой комнате.

**Name =** значение переменой

Переменные, названные с маленькой буквы – локальные, они могут быть изменены и инициализированы внутри комнаты

**name =** значение переменой

(оператор присвоение, инициализация)

Вне комнаты локальная переменная тоже может быть вызвана. Для этого нужно указать название комнаты и через точку – название переменной.

room\_name.var\_name = значение переменой

Предполагается использование двух типов переменных: текстовых и числовых. Числовые переменные при назначении записываются:

**name =** число

а текстовые:

**name =** **«**Текст**»**

При использовании текстовой переменной в диалоге, все специальные символы (@#) – не обрабатываются и записываются как есть. В переменной нельзя перенести строку.

В условии необъявленная переменная имеет значение – ноль (0).

**Условия**

Условия нужны для проверки значений переменных. Условия выглядят так:

**if** условие

**[**

действия

**]**

**else**

**[**

действия

**]**

Или так:

**if** условие

**[**

действия

**]**

(блок условий)

Описание условий:

Условия := название переменной **==** значение |

название переменной **!=** значение |

название переменной **>** значение |

название переменной **<** значение |

название переменной **>=** значение |

название переменной **<=** (значение |

Значение := число | **«**текст**»** | название переменной

Когда игрок попадает в некоторую комнату порядок действий идет сверху вниз

**room** Название\_комнаты**:**

описания комнаты (height, width, img, objects, открытые двери, magnet)

действие 1

действие 2

действие 3

…

**end**

Описания комнаты (такие как: height, width, img, objects) обрабатываются ещё до входа в комнату, другие описания (такие как: открытые двери, magnet) обрабатывается, когда человек наполовину вошел в комнату, то есть центр его экрана оказался на территории комнаты. За тем начинается обработка и выполнение действий, они выполняются один раз при входе в комнату, то есть они не выполняются каждый кадр.

**Объекты**

Объекты в Scroller представлены несколькими видами:

1. Объекты по умолчанию. Они представляют собой самые простые в создании и самые ограниченные по функционалу. Они создаются из класса объектов по умолчанию. Если нужно сделать уникальный объект с несложным функционалом, то объект по умолчанию справится с задачей. (пример: настольная лампочка, имеющая только функцию включения и выключения)
2. Объекты из пользовательских классов. Для этих объектов можно прописать методы. Также удобно их применять, если нужно создать много объектов одного вида.
3. Объекты стандартных классов. Если необходимо справится с сложной задачей, такой, как например вывести текст, дать игроку выбор из двух вариантов, отобразить на интерфейсе количество очков, то пользовательские классы не справятся с этой задачей и необходимо реализовывать класс на уровне языка python и используемого для разработки scroller движка. Планируется в будущем реализовать диалоги, оверлей, эффекты, шейдеры, инвентарь.

Создание объекта по умолчанию:

(Создаются внутри комнаты)

**object** название**(**x **,** y**)**

(х, у – координаты размещения объекта по умолчанию внутри комнаты)

Действия над объектом по умолчанию:

Название объекта**.**действие**(**параметры**)**

Действия:

Поставить картинку в объект –

**.img(**название картинки**)**

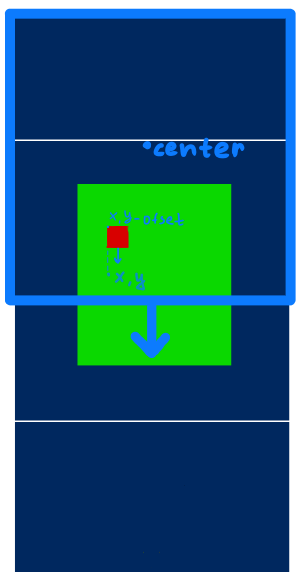
Двигать за определенное время в миллисекундах в определенное место, обозначенное координатами –

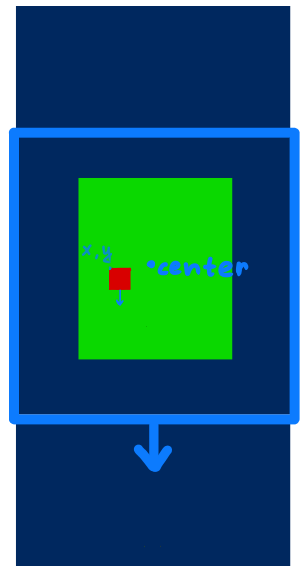
**.move(**х **,** у **,** время**)**

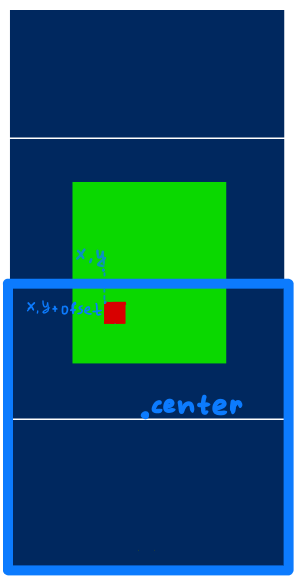
Изменяет скорость объекта относительно картинки (комнаты), при листании (для создания эффекта глубины), при этом объект смещается относительно собственных координат, для того чтобы, при наведении экрана игрока на центр комнаты, этот объект находился на собственных координатах –

**.offset(**значение смещения**)**

Если число отрицательное, объект двигается в сторону противоположную направлению скроллинга (если листаешь вниз, объект двигается вверх, будто ускоряется), при этом объект смещается в сторону от игрока, если число положительное, объект двигается в сторону направления скроллинга (если листаешь вниз, объект двигается вниз, будто замедляется), при этом объект смещается к игроку.

На этой иллюстрации голубым квадратом со стрелкой обозначен экран пользователя с направлением скроллинга (движения). Экран движется вниз, и когда центр экрана оказывается на территории комнаты с объектом (обозначен красным квадратом), объект начинает двигаться вниз с позиции X, Y – значение смещения.

**** На этой иллюстрации центр экрана пользователя оказался в центре комнаты. Объект оказался на своих координатах, указанных при создании. Так как экран продолжает движение вниз, объект тоже движется вниз.

****На этой иллюстрации центр экрана, оказался в низшей точке, комнаты. Объект в ходе движения сместился в точку X, Y + значение смещения. Так как центр переходит в другую комнату, движение объекта остановлено. На иллюстрациях показана картина, при положительном значении смещения, однако смещение может быть отрицательным, тогда мы увидим зеркальную картину.

Выполнить действия при клике на объект:

**click** название объекта**:**

**[**

[Действия]

**]**

Создание класса объектов:

(создаются вне комнат)

**class** название класса**:**

методы

**end**

Методы:

Название метода**(**входные параметры**):**

**[**

[действия]

**]**

Можно изменить инициализацию по умолчанию:

**create(**входные параметры**):**

**[**

**locate(**x**,** y**)**

[Действия]

**]**

(во входных параметрах должны быть координаты, в которые поставят объект, объект ставится при помощи команды **locate**)

**Один стандартный класс – текст.**

Будет реализовано позже

Создание текста

Text название(x , y, width, height)

print название текста:

[текст]

##

**Диалоги**



Диалоги возникают при взаимодействии персонажей, внизу окна появляется панель с текстом. Также, когда начинается диалог, scrollspeed автоматически становится равным нулю, а взаимодействие с объектами прекращается. Однако, всё ещё работает магнит.

**print:**

[текст]

**##**

После двух символов решётка, диалог заканчивается. При клике на экран окно диалога закрывается, и пользователь входит в нормальный режим.

Внутри диалогов могут содержаться определённые действия и изменения, такие как: изменение шрифта, цвета текста, размера текста, ожидание клика игрока, пауза на некоторое время. Также в диалог могут быть вставлены переменные.

Эти действия описываются между двумя символами «@».

Изменить шрифт:

**@font** название шрифта**@**

Изменить цвет шрифта:

**@color** цвет в формате rgb**@**

Изменить размер шрифта:

**@size** размер шрифта**@**

Вставить значение переменной:

**@var** название переменной**@**

Вывести следующий текст после клика:

**@click@**

Сделать паузу перед выводом следующего текста:

**@wait** время в миллисекундах**@**

**Планируется добавить в будущем:**

1. **Оверлеи**
2. **Инвентарь и предметы с возможностью drag and drop**
3. **Меню**
4. **Сохранения**
5. **Файловая система, возможность разделить код игры по разным документам, связывать документы при помощи команды include, создавать переменные и классы общие для всех документов.**
6. **Реализовать больше стандартных классов.**