Пояснительная записка.

Traffic Knight

Traffic Knight – простой платформер, написанный на языке программирования Python 3.7 с использованием библиотеки PyGame.

**Содержание**

1. Реализация
   1. Код
      1. Файловая структура
      2. Классовая структура
   2. Анимация
   3. Скриншот
2. Редактор уровней
   1. Инструкция
   2. Реализация
   3. Скриншот
3. Потенциал для развития

**Реализация**

**Код**

**Файловая структура**

Traffic Knight задействует 4 обязательных и один опциональный python-файл. Обязательные файлы:  
- main.py – содержит класс игры и основной цикл  
- classes.py – содержит основные классы объектов  
- assets.py – загружает основные графические и звуковые ресурсы  
- settings.py – содержит основные константы и параметры  
Опциональным является leveleditor.py, который содержит, соответственно, редактор уровней. Ресурсы, такие как уровни и графика, хранятся в папке data. По адресу data/resources хранятся исходные файлы текстур и моделей.

**Классовая структура**

Главным классом является Game, который занимается хранением и обработкой всех данных о текущей сессии игры. Игровыми сущностями управляют классы Player и Enemy, являющиеся наследниками Entity, который, в свою очередь, является наследником pygame.sprite.Sprite. Тайлы создаются через класс Platform, также являющийся подвидом спрайта, а положением объектов в кадре управляет класс Camera.

**Анимация**

Анимация в проекте имеет частоту кадров 60. Для создания такой плавной анимации был нарисован один спрайт персонажа, который затем был разобран на составные части (руки, локти, голова и т.д.), перенесён в Blender как трёхмерная модель из нескольких плоскостей и анимирован уже там. Первая попытка переноса оказалось крайне затратной по времени (около 4 дней), но это вызвано необходимостью полностью освоить и досконально разобрать данный способ анимации. Спрайт противника был нарисован, перенесён в Blender и анимирован целиком за 1 вечер.

**Скриншот**



**Редактор уровней**

**Инструкция**

Для упрощения создания уровней, был создан их редактор. Он не обладает интерфейсом, но зато имеет достаточный функционал. Ниже приводится инструкция по его использованию:

- ЛКМ – вставить текущий блок в сетку

- ПКМ – стереть блок

- Цифры 1-7 – выбрать блок

1 – Трава

2 – Грязь

3 – Камень

4 – Доски

5 – Место спауна игрока (Красный квадрат)

6 – Место спауна противника (Зелёный квадрат)

7 – Финиш

- Стрелки – движение камеры в соответствующем направлении

- Клавиша S – Сохранить уровень (уровни сохраняются в файл data/levels/customlevel.py)

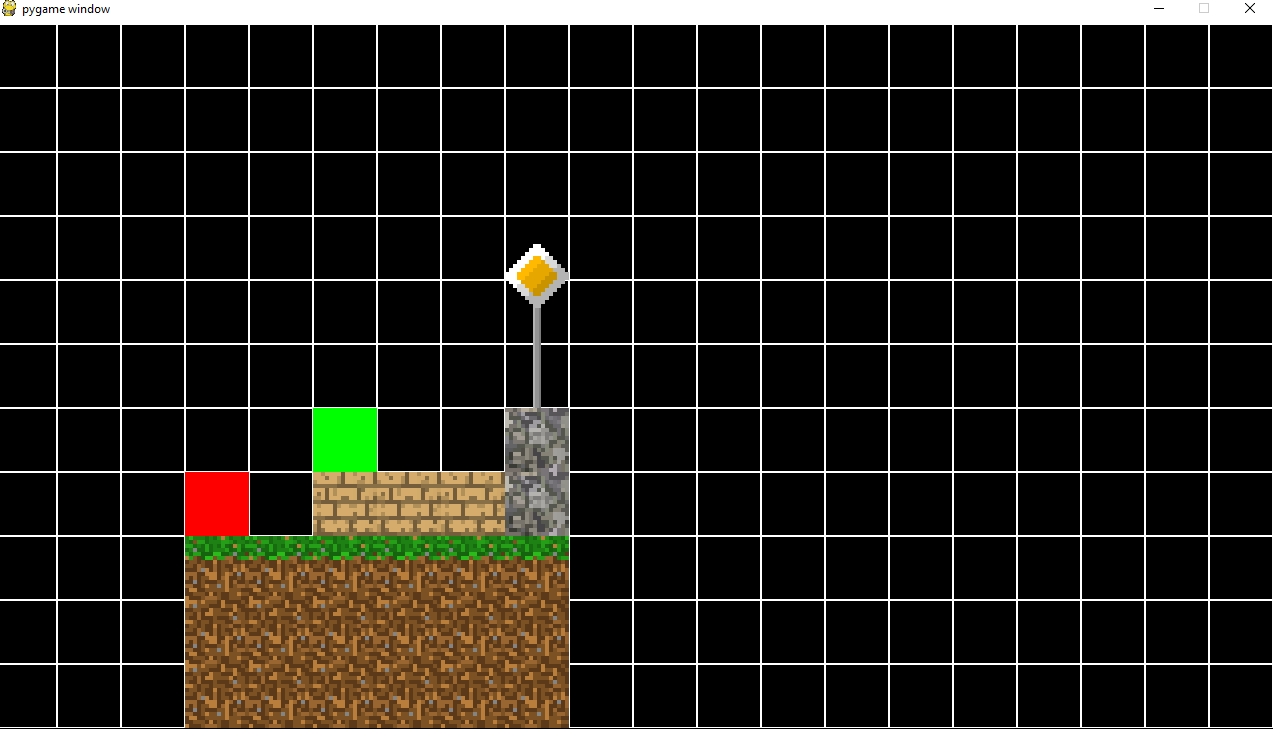
- Клавиша L – Загрузить уровень (уровень загружается из файла data/levels/customlevel.py)

Для загрузки уровня в игру, его нужно переименовать в формат level{}.py, где вместо {} идёт число – номер уровня (отсчёт начинается с единицы). Игра автоматически загрузит этот файл при достижении соответствующего уровня.

**Реализация**

В основе лежит класс Board, обрабатывающий установку и удаление блоков, а также занимающийся отрисовкой сетки выравнивания. Обработка событий происходит непосредственно в самой программе. Файл уровня представляет из себя набор строк для генерации платформ, которые исполняются в игре через функцию eval.

**Скриншот**



**Потенциал для развития**

В данный момент, игра имеет под собой достаточно серьёзный фундамент, который позволяет сравнительно легко расширять доступный контент. В числе возможных нововведений есть новые виды противников, нормальная боевая система с ударами, блоками, экипировкой и т.д., небольшие изменения, вроде заднего фона, систем частиц, системы очков и/или ограничения/счётчика времени.