**ВВЕДЕНИЕ**

В условиях быстро развивающейся современной высокотехнологичной экономики актуальной задачей является разработка различных автоматизированных систем. Они позволяют наиболее эффективно использовать различные современные средства информационной вычислительной техники и решать прикладные задачи. Для успешного решения таких задач нужно уметь использовать высокоуровневые средства разработки и декомпозировать задачу.

В данной работе решаются задачи высокоуровневого программирования, согласно индивидуальному заданию.

Актуальность курсовой работы связана с наличием большого количества прикладных задач, которые необходимо эффективно решать с помощью инструментов высокоуровневых методов программирования.

**ПОСТАНОВКА ЗАДАЧИ КУРСОВОЙ РАБОТЫ**

Исходные данные к задаче курсовой работы.

1. Формулировка: Создание 2D игры на языке программирования Python:
2. Язык программирования python
3. Основные библиотеки pygame, time, os, numpy, sys, math

Игра сделана в формате 2D. Игра имеет функциональное меню с кнопками 'Start', 'Continue', 'Options', 'About', 'Quit', из которых все функциональны и стабильно работают. Функция 'Options' включает в себя настройку полноэкранного режима и возможность вкл/выкл 'Показать ФПС'. Кнопки Start, Continue, Quit - работают соответственно. Сама игра сделана в стиле известной игры 'Mario' : ГГ(Главный герой) может передвигатьсяь влево/вправо, прыгать/падать. Задача игрока добраться до финиша на определённом уровне. Когда игрок пройдёт все уровни - он выиграет.

1. Анализ исходных данных курсовой работы
   1. Для выполнения задания нужно проделать следующие этапы:
2. Грамотно распределить время работы в команде для более координированной работы с кодом.
3. Для более удобной работы игры необходимо создать главное меню игры: присваиваем кажой кнопке определённый индекс, добавляем обновление кнопки при нажатии(анимация),добавляем проверку события для нажатой и отпущенной кнопки, добавляем задний фон и анимацию фона и прочее.
4. Для удобства пользователя необходимо добавить функцию полного экрана и возможность убрать/показать FPS в меню 'Options'
5. При написании кода понадобится импортировать библиотеки: Pygame, sys, os, time, numpy(Некоторые будут убраны после оптимизации кода)
6. Будет добавлен файл data/engine.py - в котором будет прописаны: базовая физика и механика игры, прорисовка и отображение игрового мира, подгрузка файлов. А так же это файл отвечает за полную настройку игры. - Ядро игры.
7. Необходимо нарисовать все необходимые изображения и включить их в код, а именно: кнопки(в нажатом и отпущенном виде), игрок(в статичном положении и в движении), фон игры(в игре и в состоянии паузы), эффекты частиц и прочего, игровые объекты(платформы, цепи, стены)
8. Выбор и обоснование необходимых библиотек и среды разработки.
9. Поскольку в методических указаниях к выполнению курсовой работы указано использование языка программирования Python, среды разработки PyCharm, а также стандартных библиотек и модулей стандартной библиотеки, то для решения заданий курсовой работы были выбраны следующие средства: Язык программирования Python3 – согласно рекомендациям; Интегрированная среда разработки PyCharm – согласно рекомендациям;
10. Pygame— библиотека Python для упрощения написания игрового кода на движке Python
11. Sys— библиотека Python для прямого взаимодействия с интерпритатором.
12. Os— библиотека Python для получения наглядной информации о работоспособности кода
13. Time— библиотека Python для более удобной работы с секундами игры, FPS, выводом его на экран и прочее.
14. Math— библиотека Python для более удобной работы с математическими функциями кода, и для составления более сложных вычислений.
15. Numpy— библиотека Python необходима для составления высокоуровневых математических функций, составления массивов и пр.

***Ссылка на проект Github:*** [***https://github.com/Shustovson/game-project***](https://github.com/Shustovson/game-project)

**Выводы по 1 главе.**

Изучив полученный материал из главы 1, можно сделать вывод, что задание включает наиболее часто используемые прикладные задачи высокоуровневого программирования – это построение приложений с графическим пользовательским интерфейсом, использование алгоритмов, работа в наиболее популярной среде разработки, ставшей де-факто стандартом для разработчиков на языке программирования Python.

* Успешное выполнения задания будет свидетельствовать о полноте усвоенного материала в рамках дисциплины «Высокоуровневые методы программирования» и умение решать задачи низкой и средней сложности.

**2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА**

* + 1. Создание движка(ядра) игры. Для этого импортируем библиотеки: pygame, math, os, sys, в файл data/engine а так же из библиотеки pygame импортируем locals.
    2. Добавляем класс Font - Массив, который содержит в себе названия переменных, символов , которыми пользуется игра для корректного чтения кода.

Добавляем и инициализируем(\_\_init\_\_) объекты физики и задаём функцию движения отдельно от игрока(камера центрирована на персонаже)

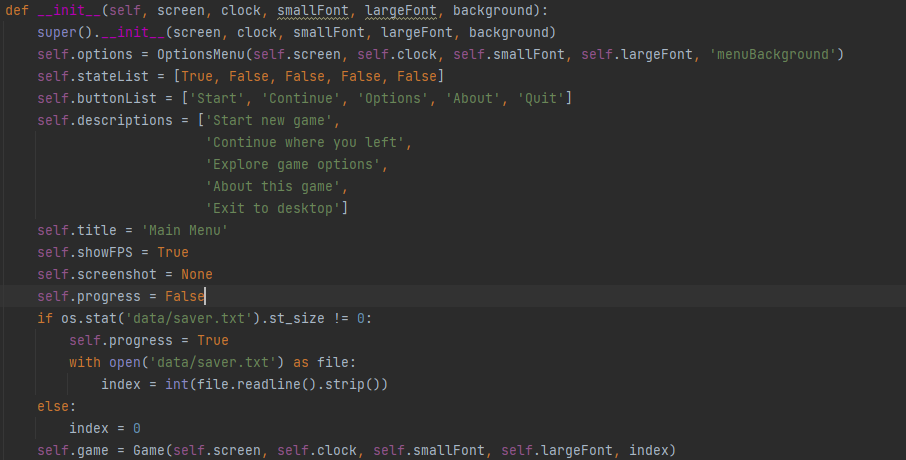
* + 1. Добавляем в код функции для очищения объектов, если персонаж их не видит.
    2. Добавляем отрисовку спрайтов и прочих изображений
  1. Создание главного меню игры. Для этого создаём файл menu.py, в который мы поместили классы: MenuScreen, OptionsMenu, VideoMenu - которые отрисовывают меню и проверяют события стартового экрана игры.(нажатие кнопок) В файл main.py мы поместили класс MainMenu - который является связующим классов выше, который инициализирует, проверяет на события и обнавляет экран главного меню.

2.3 Добавляем файл Settings.py - в который мы добавили такие функции, как длинна и высота окна, размер окна, обозначения цветов, скорость движения игрока - для поддержания работоспособности и читабельности кода.

* 1. Создание игры. Для начала мы добавили файл objects.py, в который поместили класс MapLevel - в котором происходит загрузка изображений для игры (карты, объекты игрового окружения), задаётся размер объектов(изображений), отрисовка игровых плит, стен, шипов и т.д и MapObject в котором будет инициализироваться карта.
     1. Кроме того, в файле objects.py добавляются классы StaticPlatform и MovingPlatform в которых задаются платформы и соответственно прописывается то, где находится платформа, движется она или нет и как движется (Если это предусмотрено)

Отрисовка главного меню как написано выше осуществляется через классы:

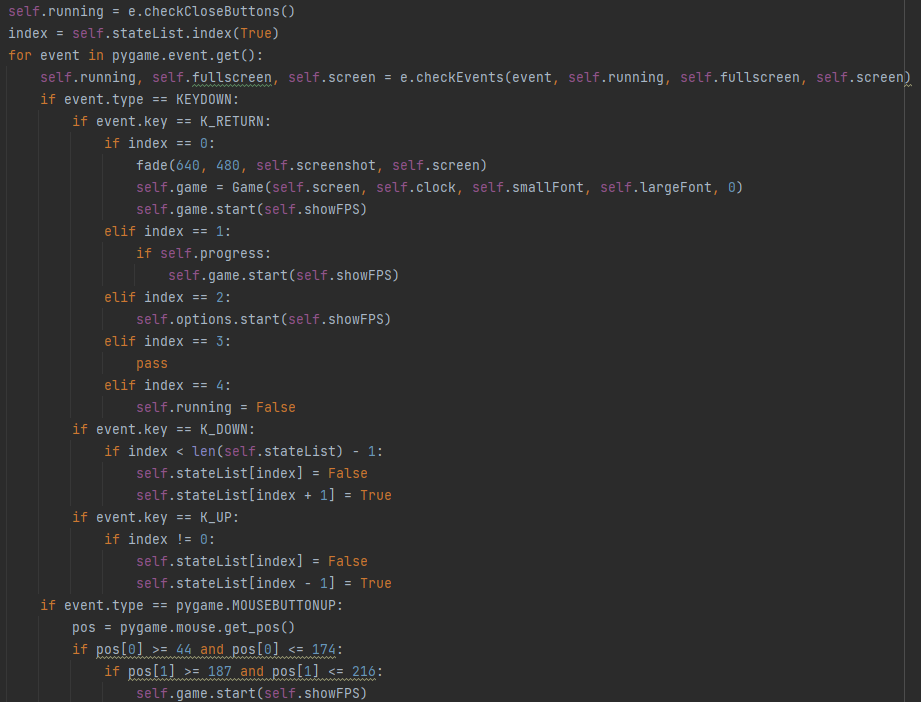
MainMenu:



Где по средствам ‘\_\_init\_\_’ задаются параметры класса(экран, время, и фон)

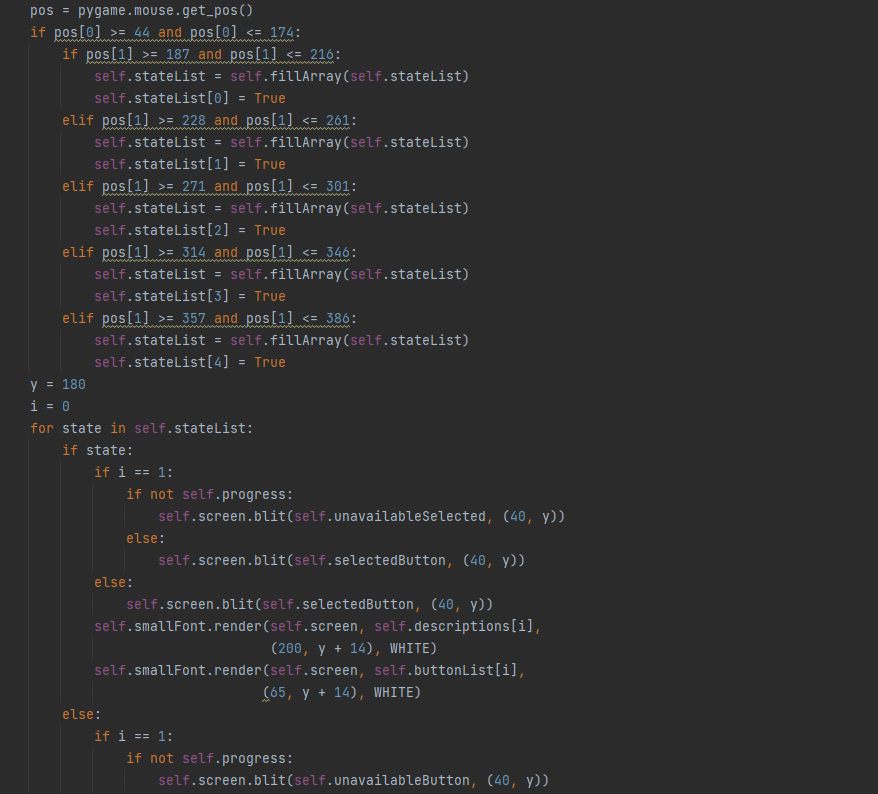
В этой инициализации задаются Наименования кнопок в главном меню, комментарии(описание) кнопок, устанавливается файл для сохранения игры, задаётся заголовок экрана.

Кроме того, внутри класса задаются функции:

events:

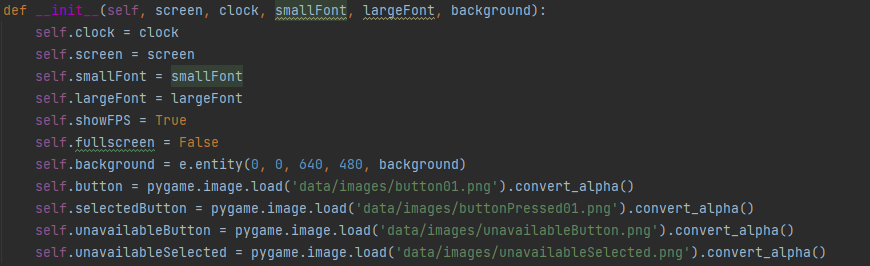
Функция описывает действие программы при нажатии курсора на кнопки в главном меню

update:



Функция обновляет позицию кнопки и меняет её внешний вид (визуальный эффект наведения курсора или использования кнопок key\_arrow\_down, key\_arrow\_up)

MenuScreen:

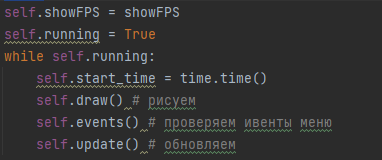


Родительский класс для всех классов связанных с меню  
через \_\_init\_\_ здесь задаются такие базовые параметры как

clock, screen, smallfont, largefont , showFPS , fullscreen а так же изображения кнопок: Активных(без выделения), Активных(с выделением), неактивных(без выделения), неактивных(с выделением)

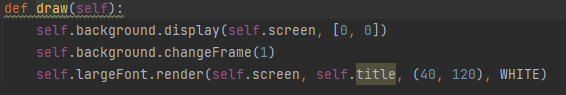
Кроме того, внутри класса задаются функции:

start:



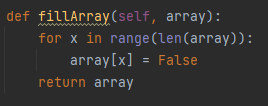
Запускаем отрисовку(draw) меню, показываем его и запускаем цикл running с проверкой на True

draw:



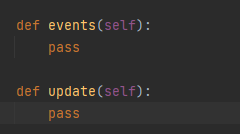
Функция выводит на дисплей задний фон и выводит на экран шрифт(largeFont) с установленными параметрами

fillArray:



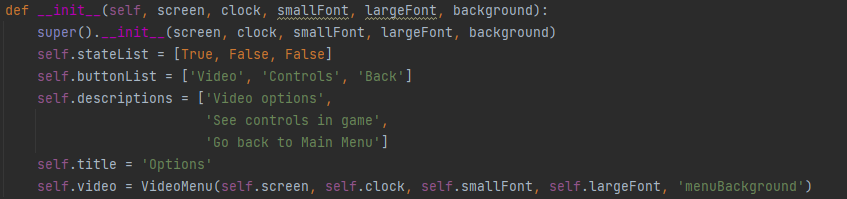
Используется в функции update в классе MainMenu для наполнения массива Активных/Неактивных кнопок

events и update:



Используются в дочерних функциях, в родительской только задаются и пропускаются

OptionsMenu:

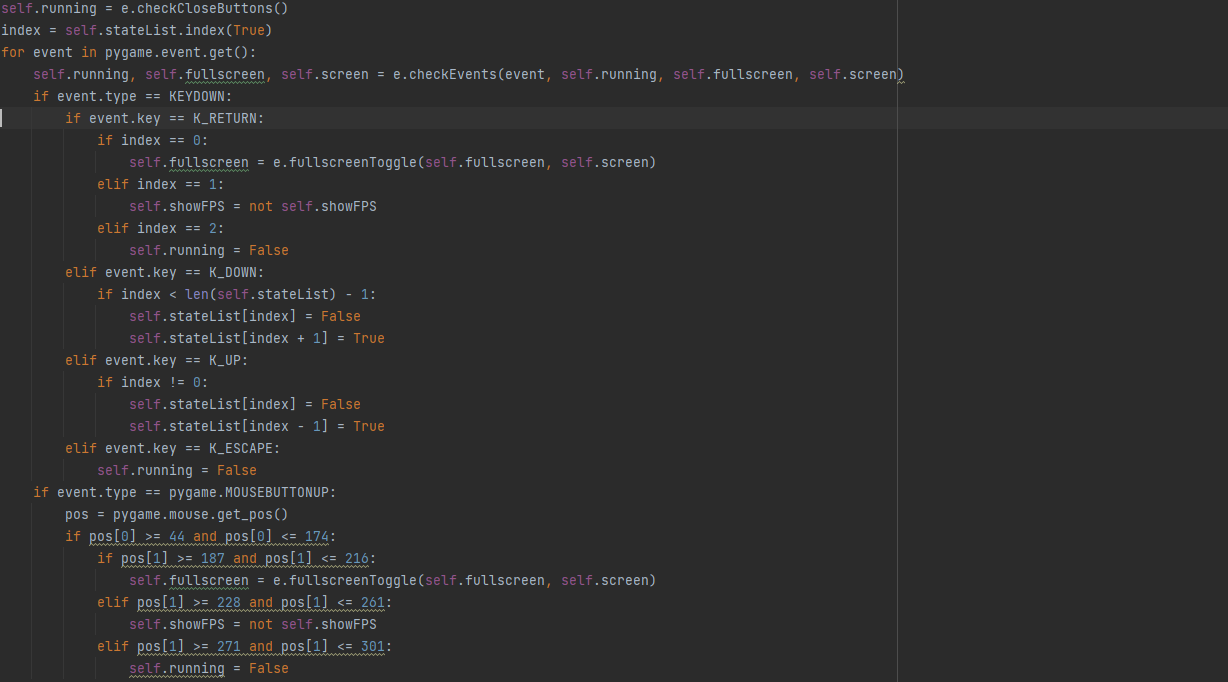


Класс для меню настроек, инициализируется и обращается к материнскому классу через super().\_\_init\_\_

Задаёт лист активных\не активных кнопок, список названия кнопок и их описание, задаётся заголовок экрана и обращается к классу VideoMenu передавая параметры.

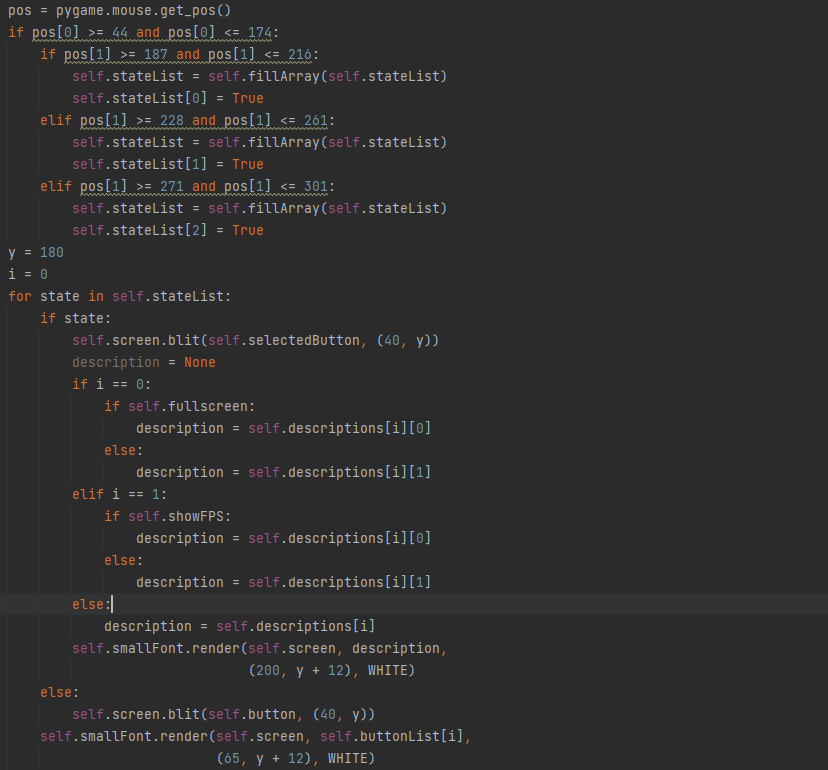
Кроме того, внутри класса задаются функции:

events:



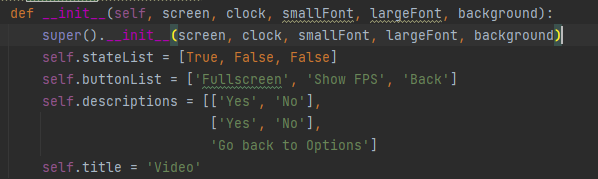
Функция считывает событие нажатия на клавиши и выполняет привязанные к ним команды

update:



Функция считывает курсор и обновляет меню если пользователь куда-то нажал

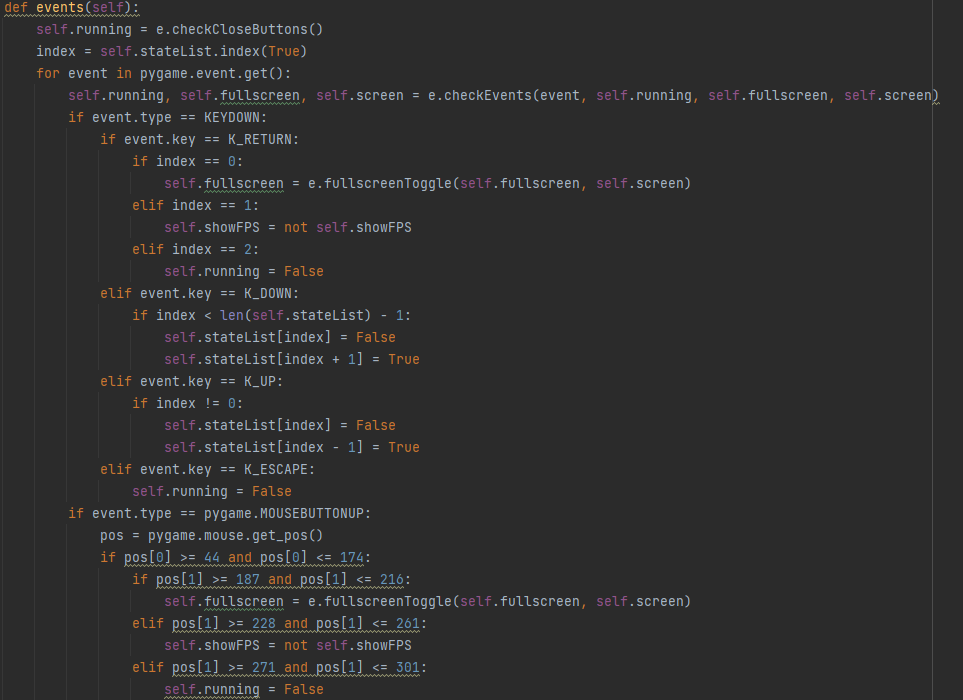
VideoMenu:



Класс Меню видео где можно выбрать режим экрана и отображения FPS в игре, так же идёт с массивами кнопок, их активности/не активности, описанием и заголовком экрана.

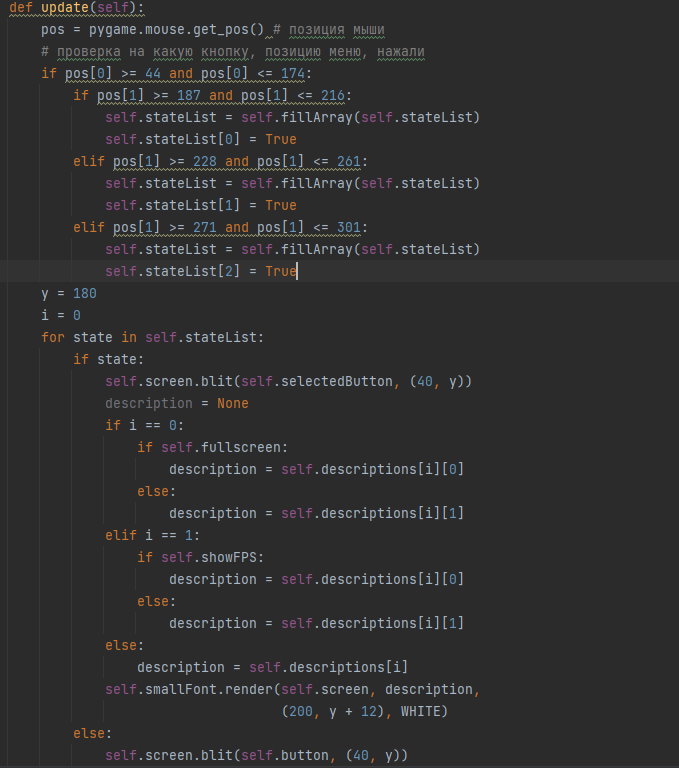
Кроме того, внутри класса задаются функции:

events:



Проверка события нажатия на клавиши

update:



Функция проверки нажатия кнопок.

LevelsMenu:



Класс инициализирующийся во время игры(до конца не реализовано)