МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ  
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
Факультет информационных технологий и робототехники (ФИТР)  
  
Кафедра программного обеспечения информационных систем и технологий  
  
  
  
  
**КУРСОВАЯ РАБОТА**по дисциплине: ”Языки программирования ”  
  
на тему: ***” Разработка игры «Doodle Jump» ”***

Выполнил**:** ст. гр. 10701320 Акимов Е.С.

Приняла**:** ст**.**преподаватель Е.А. Стальцова

Минск 2021

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине «Языки программирования»**

Тема: « Разработка игры «Doodle Jump» »

**Исполнитель**: Акимов Е.С.

(подпись)

**Студент 2 курса 10701320 группы**

**Руководитель** : Стальцова Е.А.

(подпись)

Минск 2021

ОГЛАВЕНИЕ

[ВВЕДЕНИЕ](#_Toc66303656) 4

[1.Сценарий игры 5](#_Toc66303657)

1.1. [Постановка задачи 5](#_Toc66303657)

1.2. [Исходные данные 5](#_Toc66303657)

[2.Обоснование используемых языков и технологий](#_Toc66303658) 6

[3.Моделирование и проектирование программного обеспечения](#_Toc66303659) 7

3.1. [Uml – диаграмма класса SpaceInvaders](#_Toc66303657) 7

3.2. [Uml – диаграмма класса Player1](#_Toc66303657) 8

3.3. [Uml – диаграмма класса Enimes](#_Toc66303657) 9

3.4. [Uml – диаграмма класса Bullet](#_Toc66303657) 10

[4.Реализация программного обеспечения 1](#_Toc66303665)1

[5.Развёртывание и тестирование программного обеспечения 1](#_Toc66303665)7

[ЗАКЛЮЧЕНИЕ](#_Toc66303674) 18

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ](#_Toc66303675) 19

[ПРИЛОЖЕНИЕ](#_Toc66303676) 20

**ВВЕДЕНИЕ**

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Веб-приложения, пользовательские интерфейсы, анализ данных, статистика – для какой бы задачи вам не предстояло найти решение, в Python, скорее всего, найдется подходящий фреймворк.

Это отличный выбор языка как для новичков, так и для опытных программистов.

Python был разработан в 80-ых годах и его автор Guido van Rossum, передал его в некоммерческую организацию Python Software Foundation, которая является администратором данного языка.

Сам язык тоже распространяется с открытым исходным кодом и его можно без ограничений использовать для любых целей. Язык является интерпретируемым объектно-ориентированным, но также и процедурным, функциональным.

Только лишь благодаря его гибкости он является одним из самых широко известных высокоуровневых языков программирования в наше время. За этим языком будущее, и это не преувеличение. Во-первых, он прост в понимании и использовании: Python постепенно вводят в учебную программу, повсеместно вытесняя Паскаль и прочие языки. Во-вторых, это нейронные сети: если разработчик ударяется в машинное обучение, то сразу же обращает свое внимание на Python. Почему? Да просто данный ЯП имеет достаточное количество библиотек, ориентированных на нейронные сети.

Качественные фреймворки, огромное количество учебных материалов, дружелюбый и понятный интерфейс, простота кода: все это делает Python действительно конкурентоспособным языком, который вряд ли сдаст свои позиции в 2019-ом году.

Pygame  — набор модулей (библиотек) языка программирования Python предназначенный для написания компьютерных игр и мультимедиа-приложений.

Что касается игр, то для них Python - это совершенно подходящий язык. Единственная причина, по которой в Python мало игр, заключается в том, что индустрия профессионального развития игр почти исключительно использует С++, что, в свою очередь, связано с комбинацией проблем с производительностью и зависимостью от устаревшего кода.

Следует также отметить, что интеграция Python с другими языками делает его чрезвычайно ценным. Например, движок Panda3D позволяет писать игры (в 3D) в Python, а затем передавать части на С++ как необходимо. Это показывает, что Python достаточно мощный.

Программирование основывается на использовании языков программирования, на которых записываются инструкции для компьютера. Современные приложения содержат множество таких инструкций, связанных между собой.

**1. СЦЕНАРИЙ ИГРЫ**

**1.1 Постановка задачи**

В данной игре реализована стратегия, основанная на перепрыгивании игрока на платформы, передвигаясь вверх и увертываясь от врагов. Также в данной игре реализованы различные предметы на увеличения прыжка, стрельба и убийство врага прыжком.

Целью игры является пройти как можно выше и при этом не проиграв, т. е. нельзя попасть во врага или же упасть вниз.

Данная игра может быть использована для развития детей, а именно таких навыков и умений как: мелкая моторика, так как управление главным героем ведется при помощи стрелок клавиатуры, что способствует развитию этого навыка; реакция – скорость и быстрота ответной реакции на атаку врага; аналитическое мышление и логика – умение анализировать поступившую информацию, мыслить и рассуждать, делая верные и последовательные выводы, например, игрок видит, что около него сейчас будет пролетать враг, но игрок может попробовать перепрыгнуть на платформу выше, тем самым сократить время, но есть шанс погибнуть, или же можно переждать, потратить лишних пару секунд и продолжить перемещение дальше; концентрацию – способность продолжительное время удерживать внимание на решении конкретной задачи. Например, стараться не проиграть и поставить рекордный счет.

Так как переключение на другие виды деятельности благоприятно влияет на производительность человека, его психику и интеллектуальные способности, то эта простая игра может быть использована взрослыми, для того, чтобы немного расслабиться, отвлечься и отдохнуть от своих привычных дел.

Суть игры заключается в прохождении героем через платформы и врагов вверх, пытаясь каждый последующий раз побить предыдущий рекорд.

Наш герой умеет очень хорошо прыгать, и если зажать кнопку, то герой будет двигаться с всё большей и большей скоростью. Преимуществом нашей игры является то, что ее невозможно выиграть, и можно лишь совершенствоваться, улучшая свой результат. Хочу заметит, что платформы бывают разные по дизайну и их характеристиками. Реализовано 3 вида платформ: обычная, надломанная и движущая. «Обычная» всегда будет на одном месте и её невозможно сломать, но будьте осторожны, её охраняют местные жители. «Надломанная» является ловушкой для игрока: при попадании пользователя на эту платформу она сломается, и игрок падает вниз. «Движущая» всегда находится в движении, создавая сложность попадания на неё.

Для всех действий в игре подставленная определённая музыка.

Также еще хочу заметить такую особенность: если герой не попал на платформу, и до конца экрана еще осталась платформа, и герой на нее попал, то он выживает. Это сделано для реалистичности сюжета: если герой упадет с большого расстояния, то он умирает, но если же с небольшого, то есть просто упал на платформу немного ниже, то он выживет.

При попадании игрока на врага, он превращается в «Призрака» и падает вниз сквозь платформы. При этом в игре реализовано 8 видов врагов.

Особую опасность представляет враг «НЛО». Оно перемещается по диагонали тем самым создавая особую сложность. Если наш персонаж попадает в «НЛО», то «НЛО» включает ультразвуковой сигнал, и персонаж оглушается и падает.

Также реализованы различные бустеры: пружины, ракеты, батут при убийстве врага. При попадании на бустер игрок перемещается с огромной скоростью вверх (скорость зависит от типа бустера), при этом игрок невосприимчив к врагам.

Чтобы создать интерес к игре, я создал несколько уровней. При прохождении определённого количества очков меняется уровень вместе с задним фоном игры. Но с определённым количеством набранных очков увеличивается появление врагов, но в скором времени враги смогут заполнить абсолютно всю карту, чтобы этого не произошло, увеличивается появление ракет.

Если герой все же умирает, то высвечивается окно, на котором сообщается о проигрыше, написан ваш счет, рекордный счет и также предложение нажать «Space», чтобы начать играть заново, либо же «Q» для выхода из игры. Окно сопровождается энергичной музыкой, чтобы было желание продолжит игру. Красный цвет шрифта агрессивный, и я выбрал именно его, чтобы показать, что игрок проиграл и что стоит попробовать заново побить свой рекорд.

**1.2 Исходные данные**

* Данные соответствующие сценарию игры;
* Язык программирования Python;
* Архитектура приложения – Model-View-Controller;
* Тип интерфейса приложения – оконный интерфейс;
* Игровой движок (библиотека): pyGame;
* Текстовый редактор для оформления пояснительной записки – MS Word .

**2.** **ОБОСНОВАНИЕ ИСПОЛЬЗУЕМЫХ ЯЗЫКОВ И ТЕХНОЛОГИЙ**

Приложение разрабатывалось в среде IDLE PyCharm Community Edition 2019. IDLE (Integrated DeveLopment Environment) − это [интегрированная среда разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) и обучения на языке [Python](https://ru.wikipedia.org/wiki/Python). С помощью IDLE можно выполнять обычные для интегрированной среды задачи: просматривать, редактировать, запускать, отлаживать программы на Python. [Редактор кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%BA%D0%BE%D0%B4%D0%B0) использует [подсветку синтаксиса](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%81%D0%B2%D0%B5%D1%82%D0%BA%D0%B0_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81%D0%B0). IDLE предлагает дополнительные возможности для опытных пользователей, например, средство просмотра объектов.

Разработка игры на Python очень удобна благодаря читаемости кода и высокому уровню абстракции.

Для создания любых приложений с графическим интерфейсом необходимо использовать событийно-ориентированное программирование **–** это способ построения компьютерной программы, при котором в [коде](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)) (как правило, в головной функции программы) явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события.

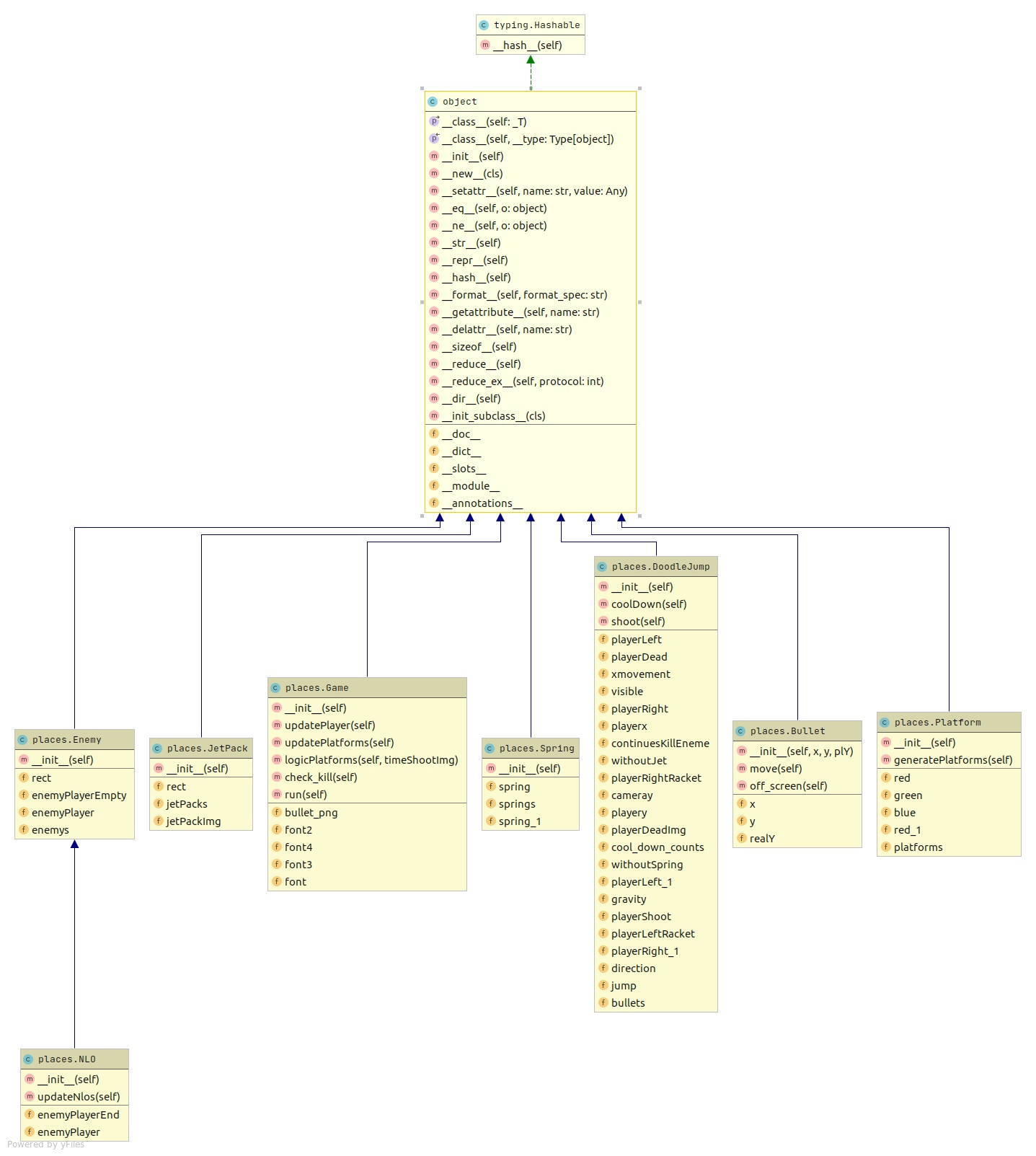
Поэтому для реализации данного курсового проекта использовались такие программные модули, как pygame, random, sys.

Модуль pygame – это фреймворк для разработки игр на языке Python, которые содержит следующие основные методы для разработки графического интерфейса: image (загружает некоторое изображение, и возвращает в виде поверхности, с которой при помощи других функций pygame можно выполнять какие-либо операции), surface (отрисовывает поверхность), event (позволяет взаимодействовать с событиями и запросами), rect (модуль для работы с кортежами типа rect).

**3.МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

**3.1 Uml – диаграмма программы**

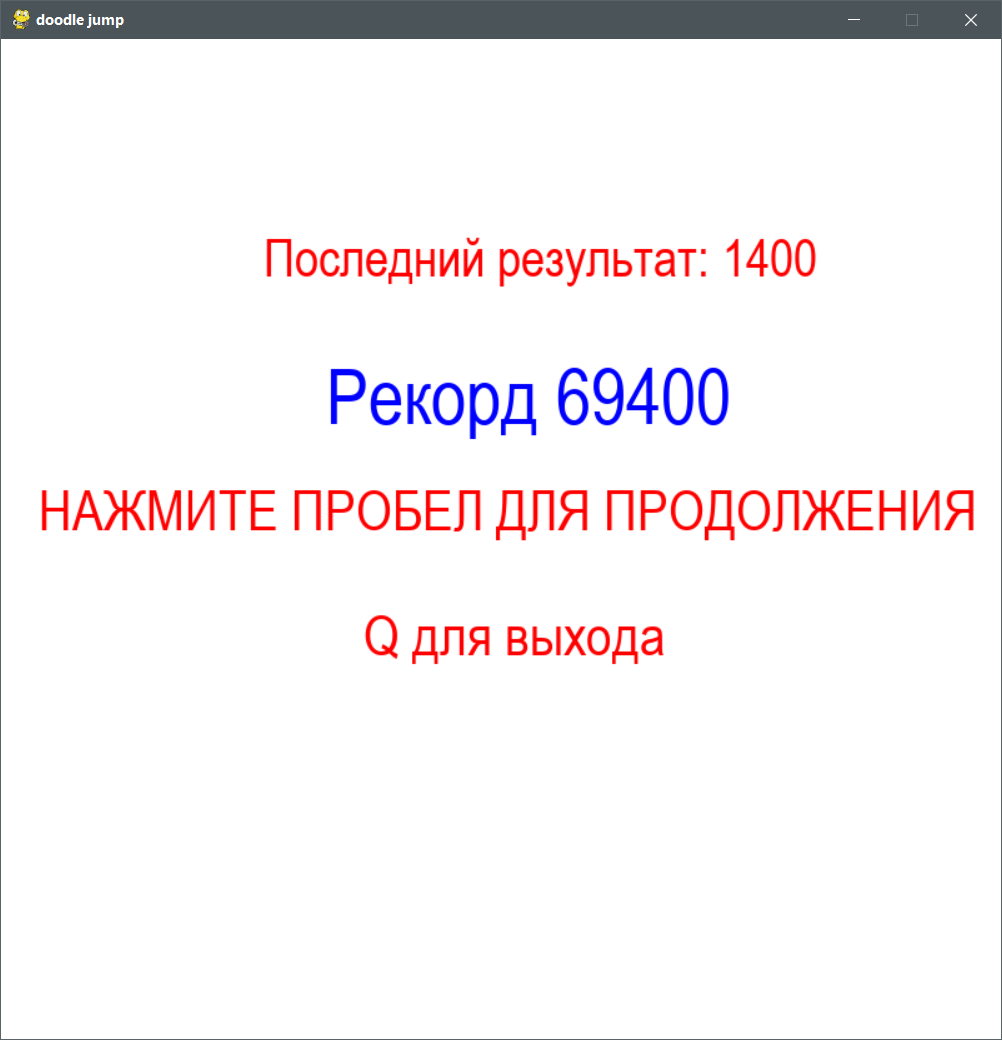
Были созданы следующие объекты: герой (doudleJump), враги обычные (enemy), НЛО (nlo), бустер ракета (jetPack), пружина (springForGreen), plat (платформы), пуля игрока (bullet) и основной класс игры (Game)

**** 

**«**Рисунок 3.1 – Uml – диаграмма программы»

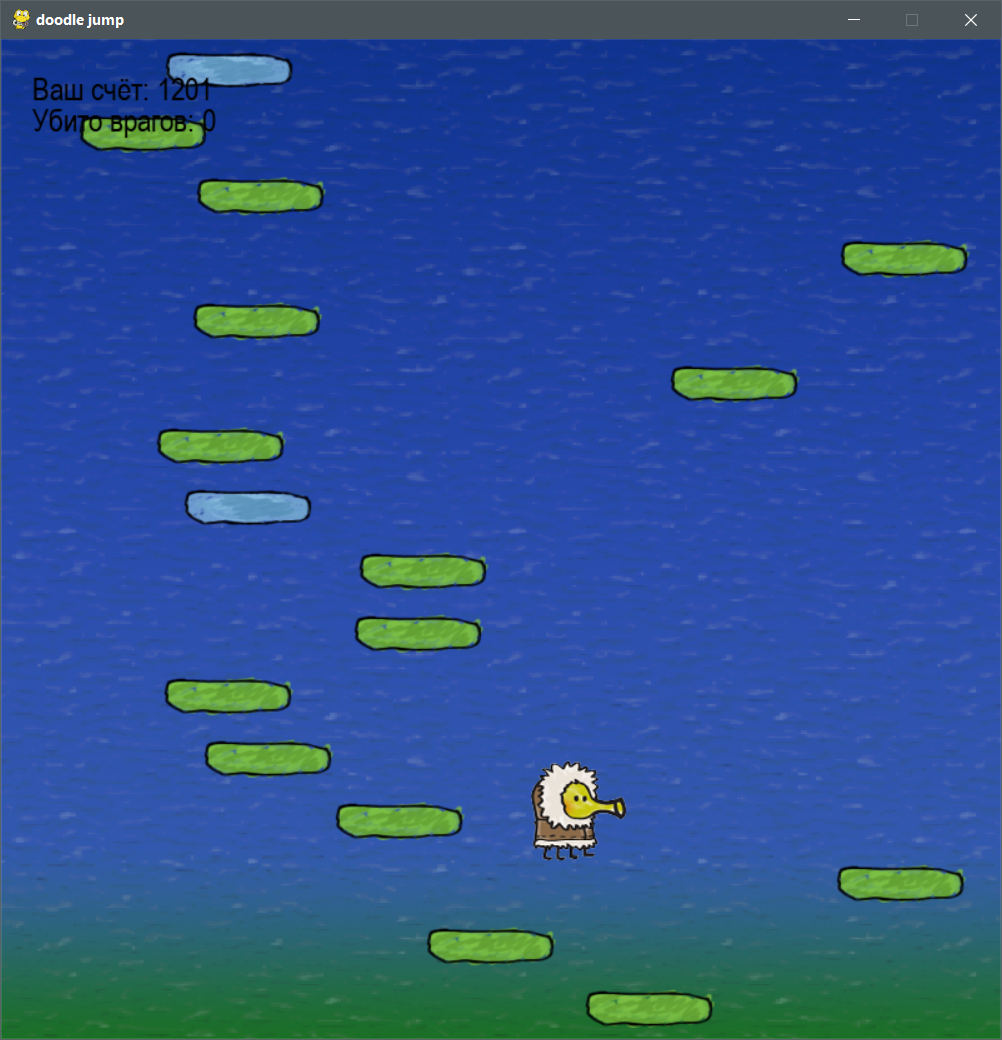
**4.РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

При запуске нашей игры первым делом открывается окно (приводится на рисунке 4.1), на котором посередине написано название игры (проекта), также сверху указан максимальный рекордный счет и еще чуть ниже названия проекта выведено сообщение о том, чтобы мы нажали на любую клавишу клавиатуры, чтобы продолжить.



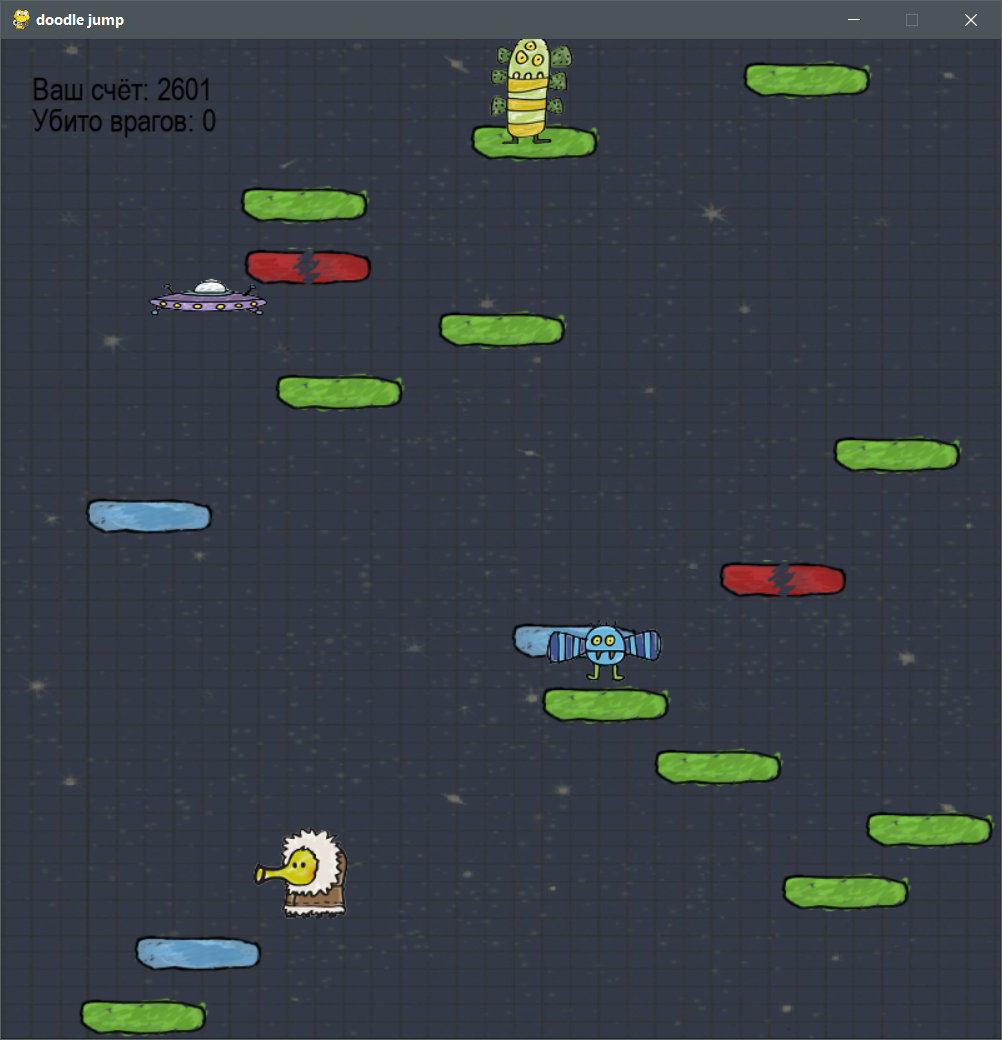
**«**Рисунок 4.1 – Главное меню игры»

При нажатии на клавишу «ПРОБЕЛ» у нас запустилось главное окно нашей игры (приводится на рисунке 4.2), на котором прорисован наш игрок.



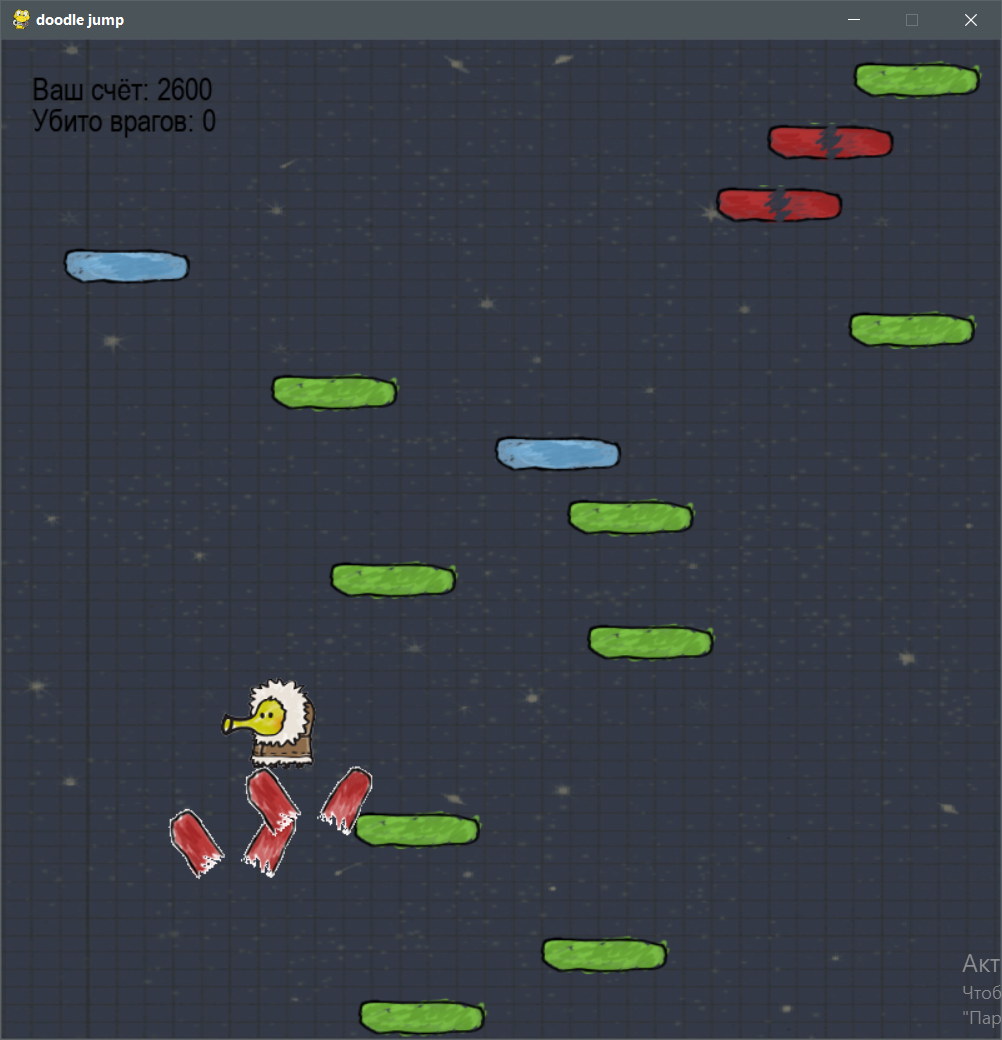
**«**Рисунок 4.2 –Начало игры»

После того, как мы пропрыгаем несколько платформ у нас начнут появляться враги (приводится на рисунке 4.3). Враги появляются в зависимости от набранных нами очков. Количество врагов увеличиваются при набирании каждых 50000 очков, при это количество зелёных платформ уменьшается, а появляется всё больше и больше синих/красных. Таким образом реализовано усложнение, а собственно при увеличении сложности – появляется интерес к моей продукции.



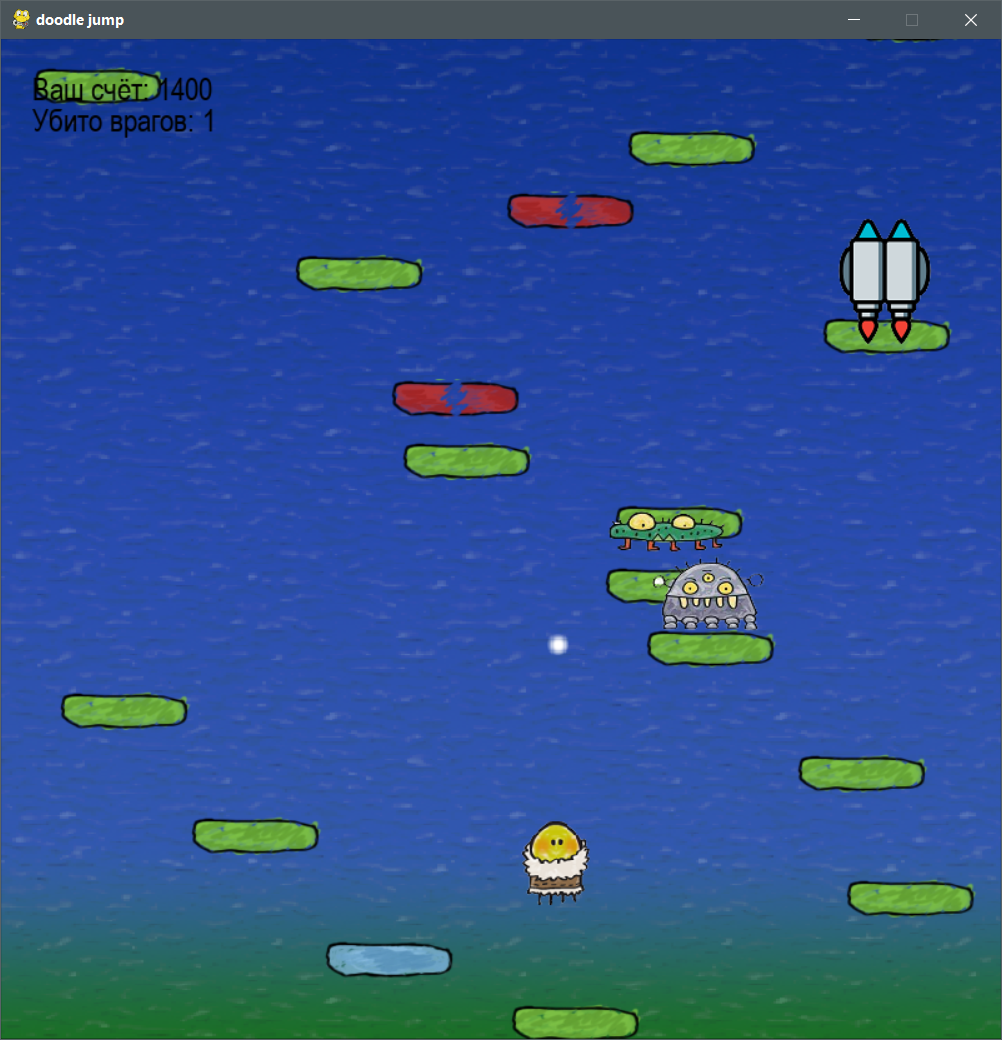
**«**Рисунок 4.3 – Появление врагов»

Также мы можем заметить вверху экрана счет игрока (приводится на рисунке 4.4), при перемещении вверх счет будет увеличиваться. Также реализована смена заднего фона, который имеет зависимость от набранных нам очков. Фон меняется от земных объектов к космосу. Так же можно увидеть, что происходит при попадании нашего героя на красную платформу: она рушится и падает вниз, что несомненно усложняет игру и повышает нашу внимательность.



**«**Рисунок 4.4 – Разрушение красных платформ»

Для возможности пройти по прямому пути реализована стрельба игрока по врагам. Стрельба реализована только по прямой линии (приводится на рисунке 4.5). На карте может размещаться только один патрон. При попадании патрона в НЛО/врага – объект исчезает навсегда, при этом нам освобождается путь для прыжка на следующую платформу. Также убийство врагов доступно не только стрельбой. При прыжке на голову противника – он исчезнет, а наш герой подпрыгнет вверх как на батуте, тем самым мы реализовываем не простую прыгалку, а уже **«**Бег за убийствами», что не может не вызывать интерес к нашей игре.

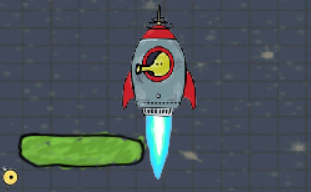


**«**Рисунок 4.5 – Стрельба персонажа»

Чтобы нормализовать соотношение интереса и сложности, были введены в игру бустеры (приводится на рисунке 4.6). Они подкидывают персонажа вверх с огромной скоростью, мину все преграды.



**«**Рисунок 4.6 – Бустер-ракета»



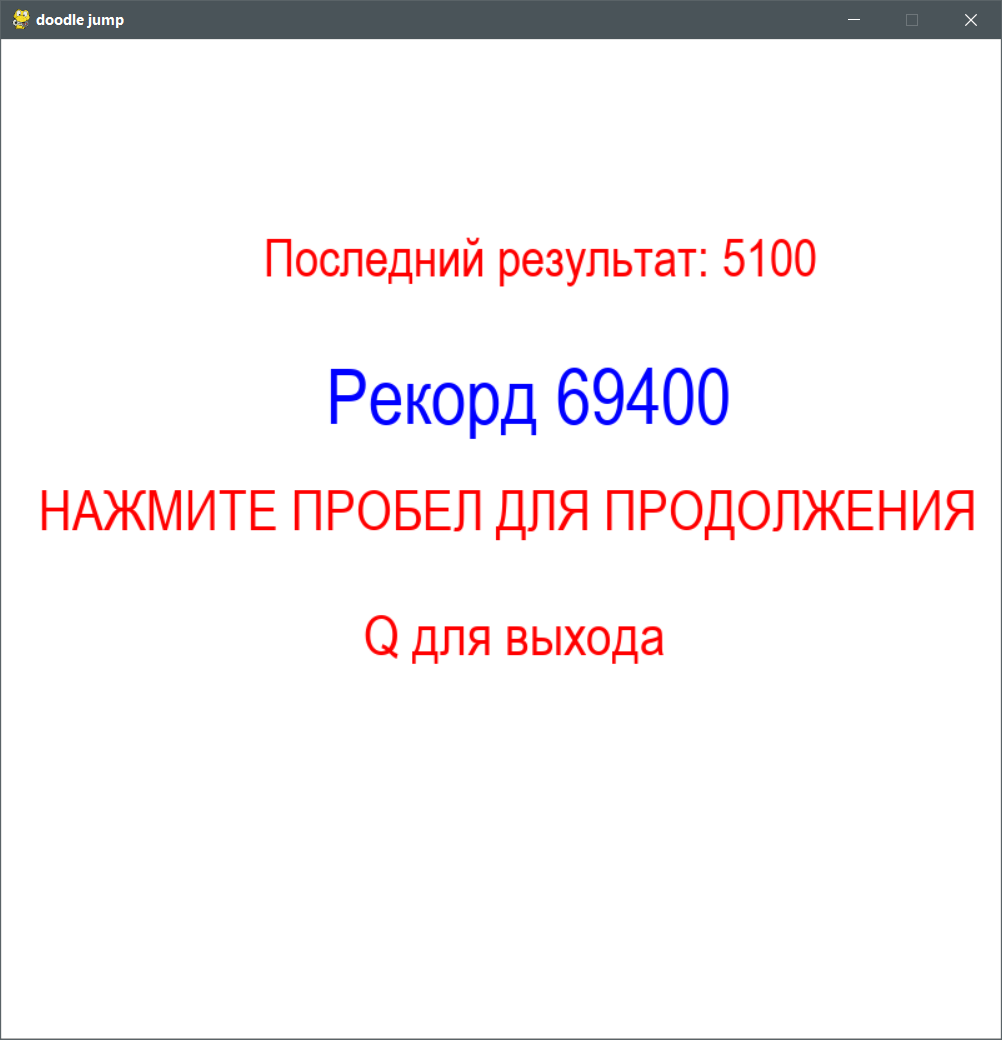
**«**Рисунок 4.7 – Герой в бустер-ракете»

Также реализован мини-бустер в виде простой пружины (приводится на рисунке 4.8). Подкидывает персонажа на маленькое расстояние, минуя все преграды.



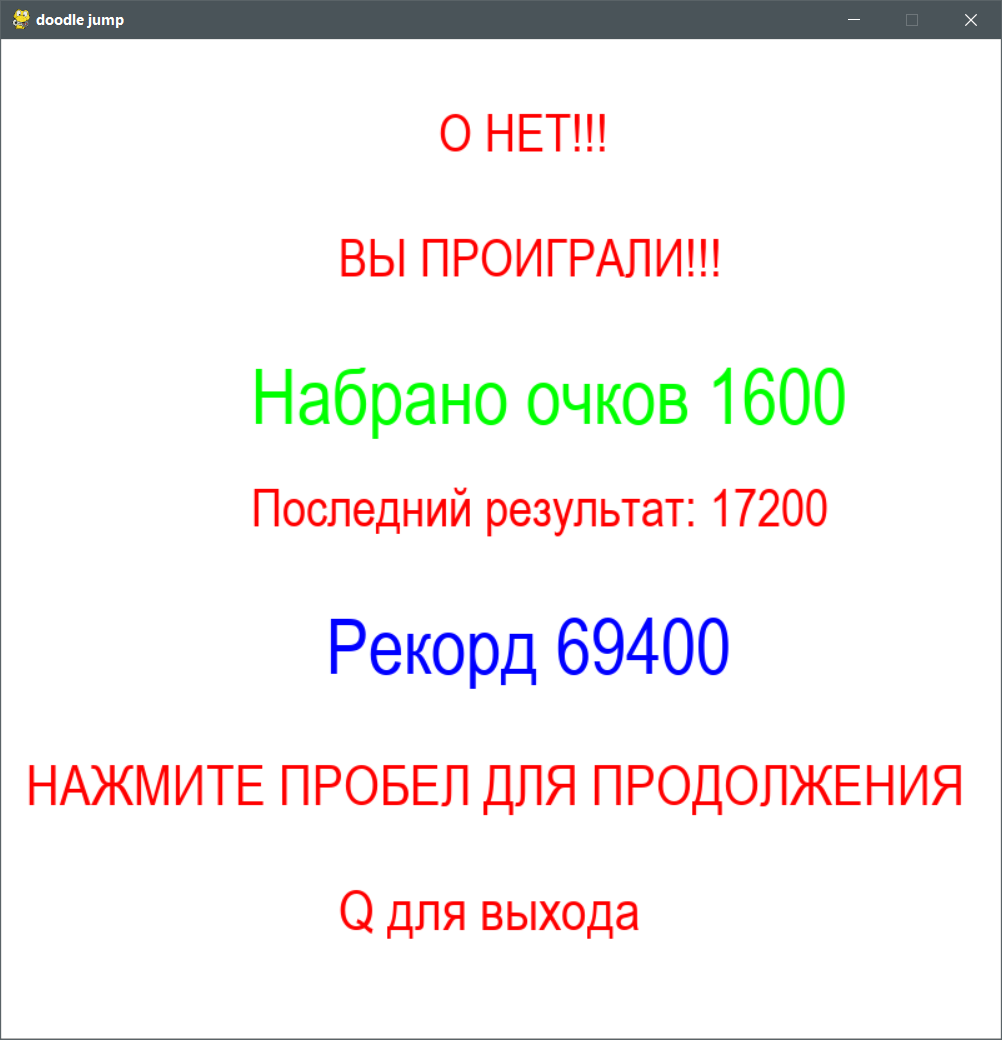
**«**Рисунок 4.8 – Бустер-пружина»

Игру всегда можно поставить на паузу нажав на клавишу **«**ESC» (приводится на рисунке 4.9). Закрытие окна на кнопку **«**Q». Продолжение «SPACE»



**«**Рисунок 4.9 – Пауза»

При проигрыше нас встречает окно поражения (приводится на рисунке 4.10) с предложением начать всё сначала или выйти из игры.



**«**Рисунок 4.10 – Окно проигрыша»

**5.РАЗВЁРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

При вызове метода start класса SpaceInvaders происходит инициализация окна и создание главного меню.

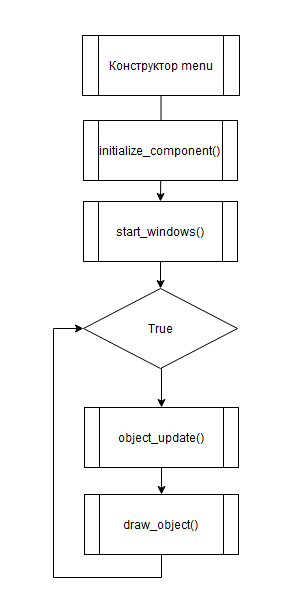
После выбора игрока пункта «Play» происходит создание объекта класса HandlerGameEvent. Он вызывает метод initialize\_component(), который отвечает за инициализацию основных переменных, загрузку изображений, анимаций, музыки, создание событий и т.д.

Далее вызывается метод start\_windows() класса HandlerView, который рисует окно приветствия.

После этого следует главный цикл игры, который состоит из двух методов object\_update() и draw\_object(). Метод object\_update() отвечает за передвижение объектов, проверку столкновений, обновление результатов игры, проверку на завершение игры и т.д. Метод draw\_object() используется для отрисовки объектов, анимации, бонусов и т.д.

Выход из приложения осуществляется по нажатию клавиши Esc.

На рисунке 5.1 представлена развертка приложения.



**«**Рисунок 5.1 – Развертка приложения»

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была создана игра «Space Invaders». Игра проста в использовании, обладает дружественным интерфейсом и интересным геймплеем.

Для создания игры использовались знания ООП, библиотека Pygame, спрайтовая анимация, звуковые эффекты, шрифты для сообщений. При запуске уровня на экране появляются враги. Убивая их, игрок получает очки при промахах утрачивает 100 очков. В конце игры пользователь получает поздравление и конечное число очков. дает возможность пользователю следить за временем прохождения игры. По окончании игры затраченное время выводится на экран вместе с предложением пользователю начать игру заново или выйти. Игра сопровождается фоновой мелодией, которая делает прохождение игры более увлекательной и захватывающей. Игра не занимает много времени для прохождения, но от этого она не становится скучной.

**СПИСОК ИСПОЛЬЗОВАНННОЙ ЛИТЕРАТУРЫ**

1. Гуриков, С.Р. Основы алгоритмизации и программирования на Python / С.Р. Гуриков. - М.: Форум, 2018. - 991 c

2. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.

3. Лутц, М. Программирование на Python т.1 / М. Лутц. - М.: Символ-Плюс, 2011. - 992 c

4. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. - М.: Символ-Плюс, 2011. - 608 c.

5. Эрик, Мэтиз Изучаем Python. Программирование игр, визуализация данных, веб-приложения / Мэтиз Эрик. - М.: Питер, 2017. - 551 c.

ПРИЛОЖЕНИЕ

**Листинг исходных кодов программ**

**Файл Draw.py**

import pygame

import Controller.event\_keybord

from Controller.main\_handler import Handler

from Controller.main\_handler import HandlerGameEvent

from Controller.main\_handler import HandlerView

from Model.AI import Enimes

from Model.Barrier import Barrier

from Model.Bonus import MachineGunBonus

from Model.Bullet import Bullet

from Model.User import Player1

from View.view import Images

class Drawing:

@staticmethod

def draw\_object():

HandlerView.heart\_score\_screen()

for i in Handler.arr\_animation:

i.update()

for i in Handler.arr\_object:

if isinstance(i, Bullet):

Images.draw\_shot(i) # draw bullet

elif isinstance(i, Enimes):

i.update\_image() # draw enimes

elif isinstance(i, Player1):

Images.draw\_plane(i) # draw plane

elif isinstance(i, MachineGunBonus):

Images.draw\_bonus\_machine\_gun(int(i.startTime))

elif isinstance(i, Barrier):

Images.draw\_barriers(i)

HandlerGameEvent.draw\_menu()

pygame.display.update()

**Файл event.py**

class Event:

def \_\_init\_\_(self):

self.\_\_events = []

def start(self):

for i in self.\_\_events:

i()

def add(self, func):

self.\_\_events.append(func)

**Файл event\_keybord.py**

import sys

import pygame

class KeyBoardEvent:

@staticmethod

def event\_to\_level():

for event in pygame.event.get(): # test for exit

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.KEYDOWN: # test for Enter(next level)

if event.key == pygame.K\_RETURN:

return True

if event.key == pygame.K\_ESCAPE:

pygame.quit()

sys.exit()

@staticmethod

def status\_keyboard():

for event in pygame.event.get(): # test for exit

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

if event.type == pygame.KEYDOWN:

if event.key == pygame.K\_ESCAPE:

pygame.quit()

sys.exit()

**Файл global\_data.py**

class GlobalData:

realx\_r=1149

realx\_left=0

kol = 0

kol\_life = 0

kol\_kill = 0

level = 1

kol\_heart = 3

start\_kol\_plane = kol\_plane = 1

shirina\_enimes\_m = minm = 2

dlina\_enimes\_n = minn = 2

maxm = 3

maxn = 3

x\_win = 1200

y\_win = 800

y\_sdvig = 70

speed\_shot = 800

highScore = score = previosScore = 0

width\_plane = 65

dlina\_plane = 89

**Файл loop\_level.py**

import pygame

import pygame\_menu

import Controller.global\_data

from Controller.Draw import Drawing

from Controller.main\_handler import HandlerGameEvent, HandlerView

from Model.Object import GameObject

class SpaceInvaders:

clock = 0

@staticmethod

def set\_plane(smth,value):

Controller.global\_data.GlobalData.start\_kol\_plane=Controller.global\_data.GlobalData.kol\_plane=value

@staticmethod

def start():

pygame.init()

screen = pygame.display.set\_mode((1200, 800))

pygame.display.set\_caption("Space Wars")

menu = pygame\_menu.Menu('Main Menu',400,300,theme=pygame\_menu.themes.THEME\_BLUE)

menu.add.selector('Players :',[('One',1),('Two',2)], onchange = SpaceInvaders.set\_plane)

menu.add.button('Play',SpaceInvaders.start\_game)

menu.add.button('Quit',pygame\_menu.events.EXIT)

menu.mainloop(screen)

@staticmethod

def start\_game():

handler\_event = HandlerGameEvent()

handler\_event.initialize\_component()

HandlerView.start\_windows()

SpaceInvaders.clock = pygame.time.Clock()

pygame.time.delay(1500)

while True:

SpaceInvaders.clock.tick()

GameObject.dt = SpaceInvaders.clock.get\_time() / 1000

handler\_event.object\_update()

Drawing.draw\_object()

**Файл main\_handler.py**

import pygame

import Controller.event\_keybord

import Model.Barrier

import Model.Bonus

import Model.Bullet

from Controller.event import Event

from Controller.global\_data import GlobalData

from Model.AI import Enimes

from Model.User import Player1

from Model.User import Player2

from Model.view\_obj import Heart

from View.Animation import Animation

from View.Music import Music

from View.view import Images

class Handler:

screen = ""

arr\_animation = []

arr\_object = []

arr\_hearts = []

class HandlerView(Handler):

@staticmethod

def start\_windows():

Images.initialize()

Images.draw\_screen() # start's screen

Images.draw\_hello(GlobalData.level)

Images.draw\_heart(Handler.arr\_hearts)

pygame.display.update()

@staticmethod

def heart\_score\_screen():

Images.draw\_screen()

Images.draw\_score()

Images.draw\_high\_score()

Images.draw\_heart(Handler.arr\_hearts)

class HandlerGameEvent(Handler):

\_\_is\_end\_level = False

\_\_temp\_event = None

\_\_is\_end\_game = False

def \_\_init\_\_(self):

self.\_\_event\_win\_game = Event()

self.\_\_event\_lose\_game = Event()

self.\_\_event\_win\_level = Event()

self.\_\_event\_lose\_level = Event()

def initialize\_event(self):

self.\_\_event\_win\_game.add(Images.draw\_end\_game\_win)

self.\_\_event\_lose\_game.add(Images.draw\_end\_game\_lose)

self.\_\_event\_win\_level.add(Images.draw\_win)

self.\_\_event\_lose\_level.add(Images.draw\_lose)

def initialize\_component(self): # первичная инициализация данных перед циклом уровня

Handler.screen = pygame.display.set\_mode((1200, 800))

pygame.display.set\_caption("Space Wars") # заголовок

self.update\_data()

self.update\_data\_game()

Animation.initialize()

self.initialize\_event()

Music.back\_music()

@classmethod

def update\_data(cls):

GlobalData.kol\_plane = GlobalData.start\_kol\_plane

HandlerGameEvent.\_\_is\_end\_level = False

GlobalData.kol = GlobalData.kol\_life = GlobalData.shirina\_enimes\_m \* GlobalData.dlina\_enimes\_n

GlobalData.kol\_kill = 0

Handler.arr\_object = []

cls.creat\_arr\_enimes(Handler.arr\_object, GlobalData.dlina\_enimes\_n, GlobalData.shirina\_enimes\_m)

cls.add\_plane()

cls.creat\_arr\_barriers()

@classmethod

def update\_data\_game(cls):

GlobalData.level = 1

GlobalData.score = GlobalData.previosScore = 0

GlobalData.shirina\_enimes\_m = GlobalData.minm

GlobalData.dlina\_enimes\_n = GlobalData.minn

Handler.arr\_hearts = Heart.creat\_mas\_hearts(GlobalData.kol\_heart)

def object\_update(self): # передвижение врагов

Enimes.find\_move\_key()

for i in Handler.arr\_object:

i.update()

self.test\_game()

self.update\_score()

Controller.event\_keybord.KeyBoardEvent.status\_keyboard()

def test\_game(self): # все враги убиты

if not HandlerGameEvent.\_\_is\_end\_level:

if GlobalData.kol\_kill == GlobalData.kol: # update screen and plain when end game

if GlobalData.shirina\_enimes\_m == GlobalData.maxm and GlobalData.dlina\_enimes\_n == GlobalData.maxn:

HandlerGameEvent.\_\_temp\_event = self.\_\_event\_win\_game

HandlerGameEvent.\_\_is\_end\_game = True

#HandlerGameEvent.update\_data\_game()

else:

HandlerGameEvent.\_\_temp\_event = self.\_\_event\_win\_level

HandlerGameEvent.edit\_kol\_enimes()

GlobalData.previosScore = GlobalData.score

HandlerGameEvent.\_\_is\_end\_level = True

if self.test\_come() or GlobalData.kol\_plane == 0: # враги дошли до границы или попал враг

del Handler.arr\_hearts[-1]

if not Handler.arr\_hearts: # сердца кончились

HandlerGameEvent.\_\_temp\_event = self.\_\_event\_lose\_game

HandlerGameEvent.\_\_is\_end\_game = True

#HandlerGameEvent.update\_data\_game()

else:

HandlerGameEvent.\_\_temp\_event = self.\_\_event\_lose\_level

GlobalData.score = GlobalData.previosScore

HandlerGameEvent.\_\_is\_end\_level = True

@classmethod

def edit\_kol\_enimes(cls): # проверка на изменение количства врагов

GlobalData.level += 1

if GlobalData.dlina\_enimes\_n < GlobalData.maxn:

GlobalData.dlina\_enimes\_n += 1

elif GlobalData.shirina\_enimes\_m < GlobalData.maxm:

GlobalData.shirina\_enimes\_m += 1

@classmethod

def creat\_arr\_enimes(cls, arr, n, m):

index = 0

nomer\_bonus = Model.Bonus.Bonus.get\_numer()

count = 0

for i in range(n):

for j in range(m):

if nomer\_bonus.count(index) and count == 0:

enimes = Enimes(i \* Enimes.width, j \* Enimes.dlina + GlobalData.y\_sdvig, "blue")

count += 1

elif nomer\_bonus.count(index):

enimes = Enimes(i \* Enimes.width, j \* Enimes.dlina + GlobalData.y\_sdvig, "yellow")

else:

enimes = Enimes(i \* Enimes.width, j \* Enimes.dlina + GlobalData.y\_sdvig, "")

index += 1

arr.append(enimes)

@classmethod

def creat\_arr\_barriers(cls):

x = x\_sdvig = ((GlobalData.realx\_r + 1) - 3 \* Model.Barrier.Barrier.width) / 4

y = GlobalData.y\_win - 200

for i in range(3):

barrier = Model.Barrier.Barrier(x, y)

Handler.arr\_object.append(barrier)

x += x\_sdvig + Model.Barrier.Barrier.width

@classmethod

def test\_come(cls):

for plane in Handler.arr\_object:

if isinstance(plane, Player1):

for i in Handler.arr\_object:

if isinstance(i, Enimes):

if i.y > plane.y:

return True

break

return False

@classmethod

def add\_plane(cls):

plane = Player1(GlobalData.x\_win / 2, GlobalData.y\_win - 100, 400)

Handler.arr\_object.append(plane)

if GlobalData.start\_kol\_plane==2:

plane2 = Player2(GlobalData.x\_win / 2 + 1 \* 100, GlobalData.y\_win - 100, 400)

Handler.arr\_object.append(plane2)

@classmethod

def update\_score(cls):

if GlobalData.highScore < GlobalData.score:

GlobalData.highScore = GlobalData.score

@staticmethod

def draw\_menu():

if HandlerGameEvent.\_\_is\_end\_level:

HandlerGameEvent.\_\_temp\_event.start()

if Controller.event\_keybord.KeyBoardEvent.event\_to\_level():

HandlerGameEvent.update\_data()

if HandlerGameEvent.\_\_is\_end\_game:

HandlerGameEvent.update\_data\_game()

HandlerGameEvent.\_\_is\_end\_game = False

**Файл AI.py**

import random

import Controller.main\_handler

import Model.Bonus

import Model.Bullet

import View.Animation

import View.Music

from Controller.global\_data import GlobalData

from Model.Object import GameObject

class Enimes(GameObject):

\_\_key\_move = 1

\_\_past\_key = 1

killer = ""

width = 50

dlina = 42

def \_\_init\_\_(self, x, y, color):

self.x = x

self.y = y

self.\_\_speed = 50

self.\_\_max\_speed = 100

self.\_\_color = color

self.\_\_animation = View.Animation.AnimationEnimesMove.activate(self.x, self.y, self.\_\_color)

Enimes.\_\_key\_move = Enimes.\_\_past\_key = 1

def move(self):

if Enimes.\_\_key\_move == 0:

self.y = self.y + self.dlina

else:

self.x = self.x + self.\_\_speed \* GameObject.dt \* Enimes.\_\_key\_move

def test\_kill(self):

Enimes.killer = ""

kill = False

ind\_shot = 0

ind = 0

for i in Controller.main\_handler.Handler.arr\_object:

if isinstance(i, Model.Bullet.Bullet):

if i.parent != "enimes":

if ((self.x <= i.x <= self.x + Enimes.width or self.x <= i.x + 7 <= self.x + Enimes.width)

and (self.y <= i.y <= self.y + self.dlina or self.y <= i.y + 10 <= self.y + self.dlina)):

ind\_shot = i

Enimes.killer = i.parent

ind = 1

kill = True

GlobalData.score += 100

break

if ind:

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(ind\_shot)]

return kill

def shoot(self):

chance = random.randint(1, 1000)

if chance < 5:

shot = Model.Bullet.Bullet(self.x + Enimes.width // 2, self.y + Enimes.dlina, GlobalData.speed\_shot, 1, "enimes")

Controller.main\_handler.Handler.arr\_object.append(shot)

View.Music.Music.shoot\_enimes()

def cheak\_bonus(self):

if self.\_\_color == "yellow":

Model.Bonus.MachineGunBonus.acticate\_bonus(Enimes.killer)

if self.\_\_color == "blue":

Model.Bonus.ShelderBonus.acticate\_bonus(Enimes.killer)

def update(self):

if self.test\_kill():

GlobalData.kol\_kill += 1

GlobalData.kol\_life -= 1

self.cheak\_bonus()

View.Animation.AnimationKill.activate(self.x, self.y)

View.Music.Music.kill\_enimes()

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(self)]

self.upspeed()

else:

self.move()

self.shoot()

def update\_image(self):

self.\_\_animation.update(self)

@staticmethod

def find\_move\_key(): # 1-right 0-down -1-left

mas = Controller.main\_handler.Handler.arr\_object

if Enimes.\_\_key\_move == 1:

for i in mas:

if isinstance(i, Enimes):

if i.x + i.width <= GlobalData.x\_win - i.width:

Enimes.\_\_key\_move = 1

else:

Enimes.\_\_key\_move = 0

Enimes.\_\_past\_key = -1

break

elif Enimes.\_\_key\_move == -1:

for i in mas:

if isinstance(i, Enimes):

if i.x - i.width >= 0:

Enimes.\_\_key\_move = -1

else:

Enimes.\_\_key\_move = 0

Enimes.\_\_past\_key = 1

break

elif Enimes.\_\_key\_move == 0:

Enimes.\_\_key\_move = Enimes.\_\_past\_key

@staticmethod

def upspeed():

k = GlobalData.kol\_life / GlobalData.kol

for i in Controller.main\_handler.Handler.arr\_object:

if isinstance(i, Enimes):

i.\_\_speed = round(1 / k \* i.\_\_max\_speed)

**Файл Barrier.py**

import Controller.main\_handler

import Model.Bullet

from Model.Object import GameObject

class Barrier(GameObject):

width = 200

dlina = 60

def \_\_init\_\_(self, x, y):

self.\_\_life = 18

self.\_\_startLife = 18

self.x = x

self.y = y

self.stage = 3

self.\_\_startStage = 3

def update(self):

self.test\_hit()

self.test\_kill()

def test\_hit(self):

for i in Controller.main\_handler.Handler.arr\_object:

if isinstance(i, Model.Bullet.Bullet):

if ((self.x <= i.x <= self.x + Barrier.width or self.x <= i.x + 7 <= self.x + Barrier.width) and (

self.y <= i.y <= self.y + Barrier.dlina or self.y <= i.y + 10 <= self.y + Barrier.dlina)):

self.\_\_life -= 1

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(i)]

self.edit\_stage()

break

def test\_kill(self):

if self.\_\_life == 0:

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(self)]

def edit\_stage(self):

if (self.\_\_startLife - self.\_\_life) % (self.\_\_startLife // self.\_\_startStage) == 0:

self.stage -= 1

**Файл Bonus.py**

import random

import Controller.global\_data

import Controller.main\_handler

from Model.Object import GameObject

from Model.User import Player1

class Bonus(GameObject):

def \_\_init\_\_(self, killer):

self.player = killer

self.plane = ""

self.startTime = 15

self.\_\_timer = 15

for i in Controller.main\_handler.Handler.arr\_object:

if isinstance(i, Player1):

if i.name == killer:

self.plane = i

@staticmethod

def acticate\_bonus(killer):

pass

@staticmethod

def get\_numer():

n = Controller.global\_data.GlobalData.kol\_life

while True:

i = random.randint(0, n - 1)

j = random.randint(0, n - 1)

if i != j:

return [i, j]

def update(self):

pass

class ShelderBonus(Bonus):

@staticmethod

def acticate\_bonus(killer):

bonus = ShelderBonus(killer)

if bonus.plane != "":

bonus.plane.life = 2

bonus.plane.skin = "shelder"

Controller.main\_handler.Handler.arr\_object.append(bonus)

def update(self):

if self.plane.life == 1:

self.plane.skin = ""

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(self)]

class MachineGunBonus(Bonus):

@staticmethod

def acticate\_bonus(killer):

bonus = MachineGunBonus(killer)

if bonus.plane != "":

bonus.plane.timeShot = 0.1

bonus.plane.timerShot = 0

bonus.plane.isMachineGun = True

Controller.main\_handler.Handler.arr\_object.append(bonus)

def update(self):

self.startTime -= GameObject.dt

if self.startTime < 0:

self.plane.timeShot = 0.7

self.plane.isMachineGun = False

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(self)]

**Файл Bullet.py**

import Controller.main\_handler

from Controller.global\_data import GlobalData

from Model.Object import GameObject

class Bullet(GameObject):

\_\_y\_sdvig = 70

def \_\_init\_\_(self, x, y, speed, vector, parent):

self.x = x

self.y = y

self.\_\_speed = speed

self.\_\_vector = vector

self.parent = parent

def end\_shot(self):

if self.y < Bullet.\_\_y\_sdvig or self.y > GlobalData.y\_win:

if self.parent[0:-1] == "player" and GlobalData.score > 0:

GlobalData.score -= 100

return True

else:

return False

def update(self):

if self.end\_shot():

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(self)]

else:

self.y += self.\_\_vector \* int(self.\_\_speed \* GameObject.dt) # move

**Файл Object.py**

class GameObject:

dt = 0

def move(self):

pass

def update(self):

pass

**Файл User.py**

import pygame

import Controller.main\_handler

import Model.Bonus

import Model.Bullet

import View.Animation

import View.Music

from Controller.global\_data import GlobalData

from Model.Object import GameObject

class Player1(GameObject):

def \_\_init\_\_(self, x, y, speed):

self.num\_player = 1

self.name = "player1"

self.x = x

self.y = y

self.\_\_speed = speed

self.timerShot = 0.7

self.timeShot = 0.7

self.\_\_event\_key = -1

self.life = 1

self.width = GlobalData.width\_plane

self.dlina = GlobalData.dlina\_plane

self.skin = ""

self.isMachineGun = False

def move(self):

self.\_\_event\_key = -1

keys = pygame.key.get\_pressed()

if keys[pygame.K\_LEFT]:

self.\_\_event\_key = 0

if keys[pygame.K\_RIGHT]:

self.\_\_event\_key = 1

if keys[pygame.K\_SPACE]: # стрельба и обработка массива пуль

if self.timerShot <= 0:

self.sound()

shot = Model.Bullet.Bullet(self.x + self.width // 2 - 3, self.y - 15, GlobalData.speed\_shot, -1, "player1")

Controller.main\_handler.Handler.arr\_object.append(shot)

self.reset\_timer\_shot()

if self.\_\_event\_key == 1 and self.x + \

self.\_\_speed \* GameObject.dt <= GlobalData.realx\_r + self.\_\_speed \* GameObject.dt:

self.x += int(self.\_\_speed \* GameObject.dt)

elif self.\_\_event\_key == 0 and self.x - self.\_\_speed \* GameObject.dt >= 0:

self.x -= int(self.\_\_speed \* GameObject.dt)

def update(self):

self.timerShot -= GameObject.dt

self.move()

self.test\_kill()

def reset\_timer\_shot(self):

self.timerShot = self.timeShot

def test\_kill(self):

ind = ind\_shot = 0

for i in Controller.main\_handler.Handler.arr\_object:

if isinstance(i, Model.Bullet.Bullet):

if ((self.x <= i.x <= self.x + self.width or self.x <= i.x + 7 <= self.x + self.width) and (

self.y <= i.y <= self.y + self.dlina or self.y <= i.y + 10 <= self.y + self.dlina)):

ind\_shot = i

ind = 1

break

if ind:

del Controller.main\_handler.Handler.arr\_object[Controller.main\_handler.Handler.arr\_object.index(ind\_shot)]

self.life -= 1

if self.life == 0:

View.Animation.AnimationDestoyPlane.activate(self.x, self.y)

self.destroy\_bonus()

del Controller.main\_handler.Handler.arr\_object[

Controller.main\_handler.Handler.arr\_object.index(self)]

GlobalData.kol\_plane -= 1

View.Music.Music.destroy\_plane()

def sound(self):

if self.isMachineGun:

View.Music.Music.shoot\_plane\_machinegun()

else:

View.Music.Music.shoot\_plane()

def destroy\_bonus(self):

for i in Controller.main\_handler.Handler.arr\_object:

if isinstance(i, Model.Bonus.Bonus):

if i.player == self.name:

del Controller.main\_handler.Handler.arr\_object[

Controller.main\_handler.Handler.arr\_object.index(i)]

break

class Player2(Player1):

def \_\_init\_\_(self, x, y, speed):

super().\_\_init\_\_(x, y, speed)

self.num\_player = 2

self.name = "player2"

self.x = x

self.y = y

self.\_\_speed = speed

def move(self):

self.\_\_event\_key = -1

keys = pygame.key.get\_pressed()

if keys[pygame.K\_a]:

self.\_\_event\_key = 0

if keys[pygame.K\_d]:

self.\_\_event\_key = 1

if keys[pygame.K\_LCTRL]: # стрельба и обработка массива пуль

if self.timerShot <= 0:

self.sound()

shot = Model.Bullet.Bullet(self.x + self.width // 2 - 3, self.y - 15, GlobalData.speed\_shot, -1, "player2")

Controller.main\_handler.Handler.arr\_object.append(shot)

self.reset\_timer\_shot()

if self.\_\_event\_key == 1 and \

self.x + self.\_\_speed \* GameObject.dt <= GlobalData.realx\_r + self.\_\_speed \* GameObject.dt:

self.x += int(self.\_\_speed \* GameObject.dt)

elif self.\_\_event\_key == 0 and self.x - self.\_\_speed \* GameObject.dt >= 0:

self.x -= int(self.\_\_speed \* GameObject.dt)

**Файл view\_obj.py**

class Heart:

def \_\_init\_\_(self, x):

self.x = x

@staticmethod

def creat\_mas\_hearts(n):

a = []

x\_heart = 60

for i in range(n):

k = Heart(i \* x\_heart)

a.append(k)

return a

**Файл Animation.py**

import random

import pygame

import Controller.global\_data

import Controller.main\_handler

import Model.Object

class Animation:

\_animationKill = []

\_animationEnimesMoveWhite = []

\_animationEnimesMoveBlue = []

\_animationEnimesMoveYellow = []

\_animationDestroyPlane = []

def \_\_init\_\_(self, x, y, sprite):

self.\_\_x = x

self.\_\_y = y

self.\_\_index = 0

self.\_\_timer = 0.15

self.\_\_startTime = 0.15

self.\_\_sprite = sprite

self.\_\_timeDelay = 0

@staticmethod

def initialize():

for i in range(1, 10):

image = pygame.image.load(r'D:\SpaceInvaders\View\explode' + str(i) + '.png')

image.set\_colorkey((0, 0, 0))

Animation.\_animationKill.append(image)

for i in range(1, 5):

image = pygame.image.load(r'D:\SpaceInvaders\View\enimes' + str(i) + '.png')

image.set\_colorkey((0, 0, 0))

Animation.\_animationEnimesMoveWhite.append(image)

image = pygame.image.load(r'D:\SpaceInvaders\View\enimes' + str(i) + 'yellow.png')

image.set\_colorkey((0, 0, 0))

Animation.\_animationEnimesMoveYellow.append(image)

image = pygame.image.load(r'D:\SpaceInvaders\View\enimes' + str(i) + 'blue.png')

image.set\_colorkey((0, 0, 0))

Animation.\_animationEnimesMoveBlue.append(image)

#boom plane

image = pygame.image.load(r'D:\SpaceInvaders\View\boom.png')

left = 0

right = Controller.global\_data.GlobalData.width\_plane

top = 0

bottom = Controller.global\_data.GlobalData.dlina\_plane

for i in range(4):

for j in range(5):

imageDestroy = image.subsurface((left, top, right, bottom))

imageDestroy.set\_colorkey((0, 0, 0))

Animation.\_animationDestroyPlane.append(imageDestroy)

left += Controller.global\_data.GlobalData.width\_plane

left = 0

top += Controller.global\_data.GlobalData.dlina\_plane

def update(self):

pass

@staticmethod

def activate(x, y):

pass

class AnimationKill(Animation):

def \_\_init\_\_(self, x, y, sprite):

self.\_\_x = x

self.\_\_y = y

self.\_\_index = 0

self.\_\_timer = 0.15

self.\_\_startTime = 0.15

self.\_\_sprite = sprite

self.\_\_timeDelay = 0

@staticmethod

def activate(x, y):

animation = AnimationKill(x, y, AnimationKill.\_animationKill)

Controller.main\_handler.Handler.arr\_animation.append(animation)

def update(self):

if self.\_\_index < len(self.\_\_sprite) - 1:

if self.\_\_timer > 0:

self.\_\_timer -= Model.Object.GameObject.dt \* 2

else:

self.\_\_index += 1

self.\_\_timer = self.\_\_startTime

Controller.main\_handler.Handler.screen.blit(self.\_\_sprite[self.\_\_index], (self.\_\_x, self.\_\_y))

else:

del Controller.main\_handler.Handler.arr\_animation[Controller.main\_handler.Handler.arr\_animation.index(self)]

class AnimationDestoyPlane(Animation):

def \_\_init\_\_(self, x, y, sprite):

self.\_\_x = x

self.\_\_y = y

self.\_\_index = 0

self.\_\_timer = 0.15

self.\_\_startTime = 0.15

self.\_\_sprite = sprite

self.\_\_timeDelay = 0

@staticmethod

def activate(x, y):

animation = AnimationKill(x, y, AnimationDestoyPlane.\_animationDestroyPlane)

Controller.main\_handler.Handler.arr\_animation.append(animation)

def update(self):

if self.\_\_index < len(self.\_\_sprite) - 1:

if self.\_\_timer > 0:

self.\_\_timer -= Model.Object.GameObject.dt \* 1000

else:

self.\_\_index += 1

self.\_\_timer = self.\_\_startTime

Controller.main\_handler.Handler.screen.blit(self.\_\_sprite[self.\_\_index], (self.\_\_x, self.\_\_y))

else:

del Controller.main\_handler.Handler.arr\_animation[Controller.main\_handler.Handler.arr\_animation.index(self)]

class AnimationEnimesMove(Animation):

def \_\_init\_\_(self, x, y, sprite):

self.\_\_x = x

self.\_\_y = y

self.\_\_index = 0

self.\_\_timer = 0.15

self.\_\_startTime = 0.15

self.\_\_sprite = sprite

self.\_\_timeDelay = 0

@staticmethod

def activate(x, y, name):

sprite = AnimationEnimesMove.initialize\_name(name)

animation = AnimationEnimesMove(x, y, sprite)

animation.\_\_timeDelay = random.randint(0, 15) / 2

animation.\_\_timer = animation.\_\_startTime = 1

return animation

def update(self, enimes):

if self.\_\_timeDelay > 0:

self.\_\_timeDelay -= Model.Object.GameObject.dt \* 2

else:

if self.\_\_index < len(self.\_\_sprite) - 1:

if self.\_\_timer > 0:

self.\_\_timer -= Model.Object.GameObject.dt \* 2

else:

self.\_\_index += 1

self.\_\_timer = self.\_\_startTime

else:

self.\_\_index = 0

Controller.main\_handler.Handler.screen.blit(self.\_\_sprite[self.\_\_index], (enimes.x, enimes.y))

@classmethod

def initialize\_name(cls, name):

if name == "":

return AnimationEnimesMove.\_animationEnimesMoveWhite

elif name == "yellow":

return AnimationEnimesMove.\_animationEnimesMoveYellow

else:

return AnimationEnimesMove.\_animationEnimesMoveBlue

**Файл Music.py**

import pygame

class Music:

@staticmethod

def back\_music():

pygame.mixer.music.load(r'D:\SpaceInvaders\View\backmusic.mp3')

pygame.mixer.music.play(-1)

pygame.mixer.music.set\_volume(0.1)

@staticmethod

def shoot\_plane\_machinegun():

shoot\_sound = pygame.mixer.Sound(r'D:\SpaceInvaders\View\machinegun.mp3')

shoot\_sound.set\_volume(0.2)

shoot\_sound.play()

@staticmethod

def shoot\_plane():

shoot\_sound = pygame.mixer.Sound(r'D:\SpaceInvaders\View\planeshoot.mp3')

shoot\_sound.set\_volume(0.2)

shoot\_sound.play()

@staticmethod

def shoot\_enimes():

shoot\_sound = pygame.mixer.Sound(r'D:\SpaceInvaders\View\enimesshoot.mp3')

shoot\_sound.set\_volume(1)

shoot\_sound.play()

@staticmethod

def destroy\_plane():

shoot\_sound = pygame.mixer.Sound(r'D:\SpaceInvaders\View\boom.mp3')

shoot\_sound.set\_volume(0.5)

shoot\_sound.play()

@staticmethod

def kill\_enimes():

shoot\_sound = pygame.mixer.Sound(r'D:\SpaceInvaders\View\killEnimes.mp3')

shoot\_sound.set\_volume(0.2)

shoot\_sound.play()

**Файл view.py**

import time

import pygame

import Controller.main\_handler as Hand

from Controller.global\_data import GlobalData

class Images:

\_\_backImage = ""

\_\_imageHeart = ""

\_\_imagePlane1 = ""

\_\_imagePlaneShelder1 = ""

\_\_imagePlane2 = ""

\_\_imagePlaneShelder2 = ""

\_\_imageMachineGun = ""

\_\_imageBarrier1 = ""

\_\_imageBarrier2 = ""

\_\_imageBarrier3 = ""

@staticmethod

def initialize():

Images.\_\_backImage = pygame.image.load(r'D:\SpaceInvaders\View\backImage.png')

Images.\_\_imageHeart = pygame.image.load(r'D:\SpaceInvaders\View\heart.png')

Images.\_\_imagePlane1 = pygame.image.load(r'D:\SpaceInvaders\View\player1.png')

Images.\_\_imagePlaneShelder1 = pygame.image.load(r'D:\SpaceInvaders\View\player1shelder.png')

Images.\_\_imagePlane2 = pygame.image.load(r'D:\SpaceInvaders\View\player2.png')

Images.\_\_imagePlaneShelder2 = pygame.image.load(r'D:\SpaceInvaders\View\player2shelder.png')

Images.\_\_imageMachineGun = pygame.image.load(r'D:\SpaceInvaders\View\machinegun.png')

Images.\_\_imageBarrier1 = pygame.image.load(r'D:\SpaceInvaders\View\barrier1.png')

Images.\_\_imageBarrier2 = pygame.image.load(r'D:\SpaceInvaders\View\barrier2.png')

Images.\_\_imageBarrier3 = pygame.image.load(r'D:\SpaceInvaders\View\barrier3.png')

Images.\_\_imageHeart.set\_colorkey((255, 255, 255))

Images.\_\_imagePlane1.set\_colorkey((0, 0, 0))

Images.\_\_imagePlaneShelder1.set\_colorkey((0, 0, 0))

Images.\_\_imagePlane2.set\_colorkey((0, 0, 0))

Images.\_\_imagePlaneShelder2.set\_colorkey((0, 0, 0))

Images.\_\_imageMachineGun.set\_colorkey((255, 255, 255))

Images.\_\_imageBarrier1.set\_colorkey((0, 0, 0))

Images.\_\_imageBarrier2.set\_colorkey((0, 0, 0))

Images.\_\_imageBarrier3.set\_colorkey((0, 0, 0))

@staticmethod

def draw\_plane(plane):

if plane.skin == "": # prepare to images for plane

if plane.num\_player == 1:

Hand.Handler.screen.blit(Images.\_\_imagePlane1, (int(plane.x), int(plane.y))) # draw plane

else:

Hand.Handler.screen.blit(Images.\_\_imagePlane2, (int(plane.x), int(plane.y))) # draw plane

else:

if plane.num\_player == 1:

Hand.Handler.screen.blit(Images.\_\_imagePlaneShelder1, (int(plane.x), int(plane.y))) # draw plane

else:

Hand.Handler.screen.blit(Images.\_\_imagePlaneShelder2, (int(plane.x), int(plane.y))) # draw plane

@staticmethod

def draw\_screen():

Hand.Handler.screen.fill((0, 0, 0))

# Hand.Handler.screen.blit(Images.backImage, (0,0)) # draw plane

pygame.draw.rect(Hand.Handler.screen, (255, 255, 255), (0, 60, 1980, 5))

@staticmethod

def draw\_heart(mas\_heart):

for i in mas\_heart:

Hand.Handler.screen.blit(Images.\_\_imageHeart, (i.x, 5))

@staticmethod

def draw\_bonus\_machine\_gun(time\_shot):

Hand.Handler.screen.blit(Images.\_\_imageMachineGun, (200, 5))

string\_sec = "X " + str(time\_shot) + " sec"

font1 = pygame.font.Font(None, 60) # print lose

text1 = font1.render(string\_sec, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (260, 11))

@staticmethod

def draw\_lose():

string1 = "Game Over!!!"

string2 = "Plaing again? (Press Enter)"

font1 = pygame.font.Font(None, 200) # print lose

text1 = font1.render(string1, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (170, 100))

font2 = pygame.font.Font(None, 70) # print play again

text2 = font2.render(string2, True, (225, 225, 225))

Hand.Handler.screen.blit(text2, (260, 330))

string2 = " Exit (press Escape)"

text2 = font2.render(string2, True, (225, 225, 225))

Hand.Handler.screen.blit(text2, (310, 460))

@staticmethod

def draw\_win():

string1 = "Victory!!!"

font1 = pygame.font.Font(None, 200) # print win

text1 = font1.render(string1, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (300, 130))

ans = "Next level? (Press Enter)"

font2 = pygame.font.Font(None, 100) #

text2 = font2.render(ans, True, (225, 225, 225))

Hand.Handler.screen.blit(text2, (180, 370))

string2 = " Exit (press Escape)"

text3 = font2.render(string2, True, (225, 225, 225))

Hand.Handler.screen.blit(text3, (150, 460))

@staticmethod

def draw\_shot(shot):

if shot.parent != "enimes":

pygame.draw.rect(Hand.Handler.screen, (21, 255, 2), (int(shot.x), int(shot.y), 7, 10))

else:

pygame.draw.rect(Hand.Handler.screen, (253, 21, 9), (int(shot.x), int(shot.y), 7, 10))

@staticmethod

def draw\_barriers(barrier):

if barrier.stage == 3:

Hand.Handler.screen.blit(Images.\_\_imageBarrier3, (barrier.x, barrier.y))

elif barrier.stage == 2:

Hand.Handler.screen.blit(Images.\_\_imageBarrier2, (barrier.x, barrier.y))

else:

Hand.Handler.screen.blit(Images.\_\_imageBarrier1, (barrier.x, barrier.y))

@staticmethod

def draw\_hello(level):

string2 = "Let's start"

string1 = "Level " + str(level)

font1 = pygame.font.Font(None, 150)

text1 = font1.render(string1, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (430, 100))

font2 = pygame.font.Font(None, 200)

text2 = font2.render(string2, True, (225, 225, 225))

Hand.Handler.screen.blit(text2, (270, 350))

@staticmethod

def draw\_end\_game\_win():

string\_win = "Congratulations!!!"

string1 = "New game (press Enter)"

string2 = " Exit (press Escape)"

font1 = pygame.font.Font(None, 150)

font2 = pygame.font.Font(None, 70)

text1 = font1.render(string\_win, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (140, 200))

text2 = font2.render(string1, True, (225, 225, 225))

Hand.Handler.screen.blit(text2, (350, 350))

text3 = font2.render(string2, True, (225, 225, 225))

Hand.Handler.screen.blit(text3, (350, 460))

@staticmethod

def draw\_end\_game\_lose():

string\_lose = "Game Over!!!"

string1 = "New game (press Enter)"

string2 = " Exit (press Escape)"

font1 = pygame.font.Font(None, 150)

font2 = pygame.font.Font(None, 70)

text1 = font1.render(string\_lose, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (300, 210))

text2 = font2.render(string1, True, (225, 225, 225))

Hand.Handler.screen.blit(text2, (350, 370))

text3 = font2.render(string2, True, (225, 225, 225))

Hand.Handler.screen.blit(text3, (310, 480))

@staticmethod

def draw\_score():

if GlobalData.score > 10000:

string\_score = "Score: " + str(GlobalData.score // 10000) + "kk"

else:

string\_score = "Score: " + str(GlobalData.score)

font1 = pygame.font.Font(None, 58)

text1 = font1.render(string\_score, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (900, 10))

@staticmethod

def draw\_high\_score():

if GlobalData.highScore > 10000:

string\_high\_score = "High Score: " + str(GlobalData.highScore // 10000) + "kk"

else:

string\_high\_score = "High Score: " + str(GlobalData.highScore)

font1 = pygame.font.Font(None, 58)

text1 = font1.render(string\_high\_score, True, (225, 225, 225))

Hand.Handler.screen.blit(text1, (530, 10))

**Файл main.py**

from Controller.loop\_level import SpaceInvaders

SpaceInvaders.start()