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 4](#_Toc90642986)

[1. СЦЕНАРИЙ ИГРЫ 6](#_Toc90642987)

[1.1. Постановка задачи 6](#_Toc90642988)

[1.2. Исходные данные 8](#_Toc90642989)

[2. ОБОСНОВАНИЕ ИСПОЛЬЗУЕМЫХ ЯЗЫКОВ И ТЕХНОЛОГИЙ 9](#_Toc90642990)

[3.МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 10](#_Toc90642991)

[3.1 Uml – диаграмма программы 10](#_Toc90642992)

[4.РАЗВЁРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 11](#_Toc90642993)

[5. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 12](#_Toc90642994)

[ЗАКЛЮЧЕНИЕ 19](#_Toc90642995)

[СПИСОК ИСПОЛЬЗОВАНННОЙ ЛИТЕРАТУРЫ 20](#_Toc90642996)

[ПРИЛОЖЕНИЕ 1 21](#_Toc90642997)

# 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)

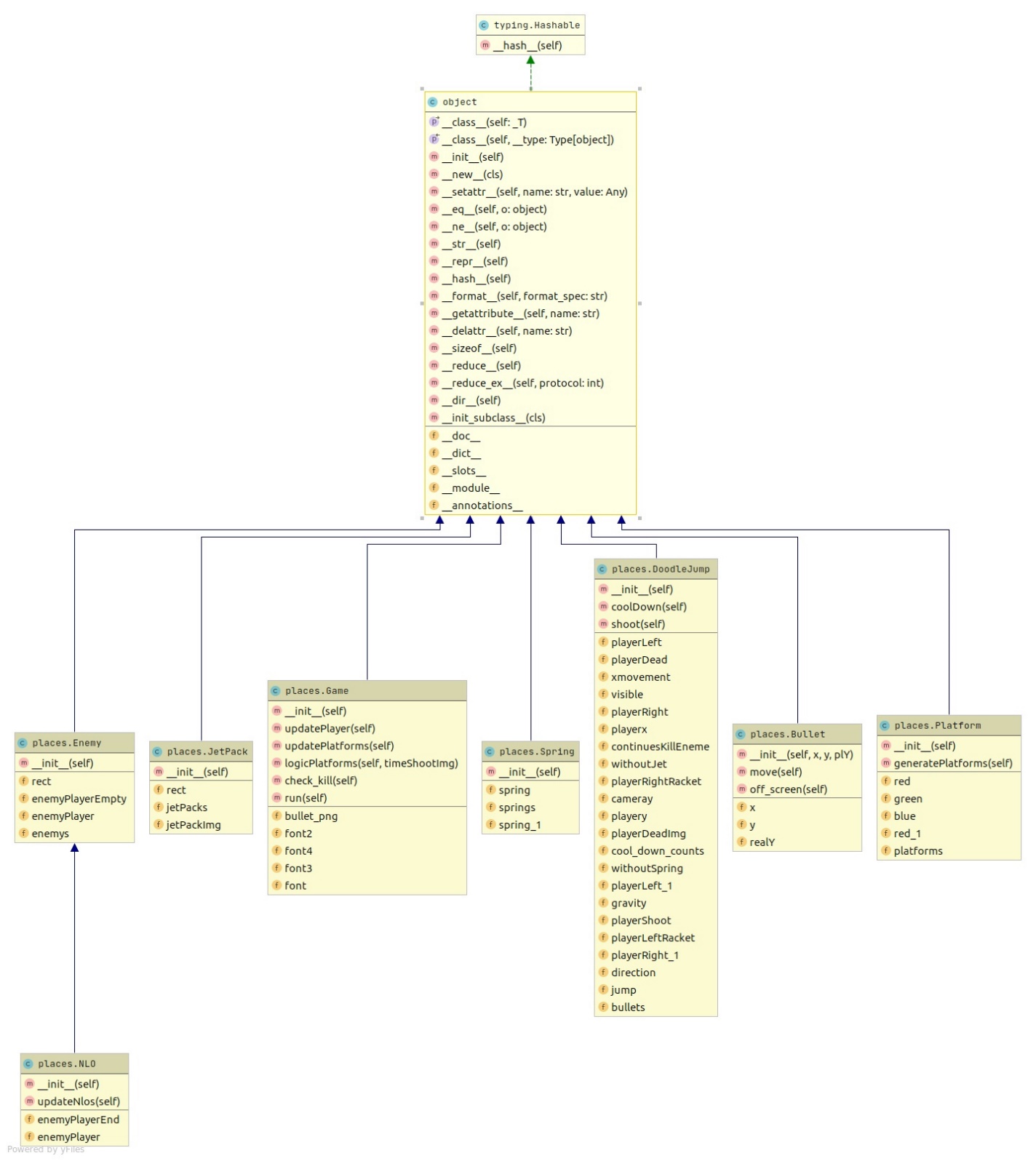
**** 

Рисунок 3.1 – Uml – диаграмма программы

# 4.РАЗВЁРТЫВАНИЕ И ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

При вызове метода run класса Game происходит инициализация окна и создание главного меню.Ппроисходит создание объекта класса DoodleJump, Enemy, NLO, JetPack, Spring, Platform, Bullets

После выбора игрока пункта **«**Продолжить» запускается главный цикл программы.

Далее идут постоянные проверки на выполнение определённых условий, реализуя бизнес-логику: logicPlatforms() отвечает за генерацию последующих платформ, check\_kill() отвечает за проверку условия убийства противника, drawPlatforms() отвечает за прорисовку платформ, updatePlayer() передвижение и управление нашего героя, updatePlatforms() движение и ломание уже созданных платформ, updateNlos() обновление врагов, shoot() – стрельба героя, drawLose() – прорисовка окна проигрыша: срабатывает при условии смерти игрока.

На рисунке 4.1 представлена развёртка приложения.

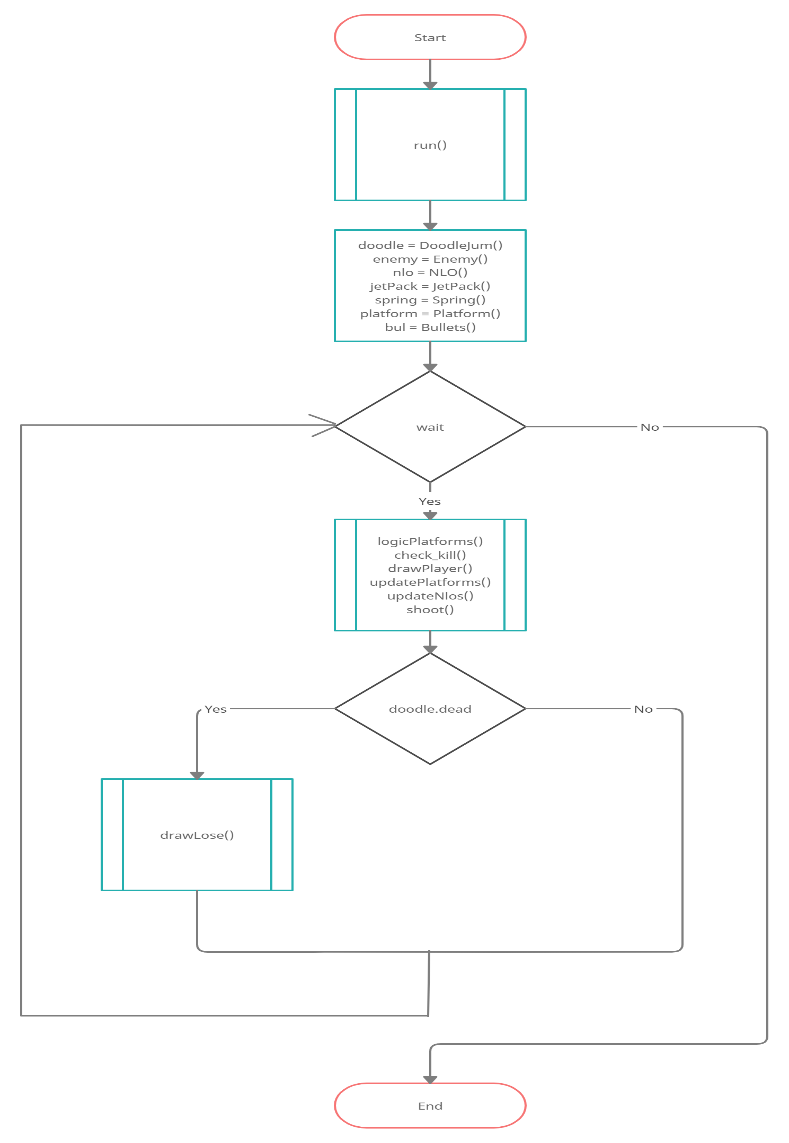


Рисунок 4.1 – Развертка приложения

# 5. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

При запуске нашей игры первым делом открывается окно (приводится на рисунке 5.1), 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.

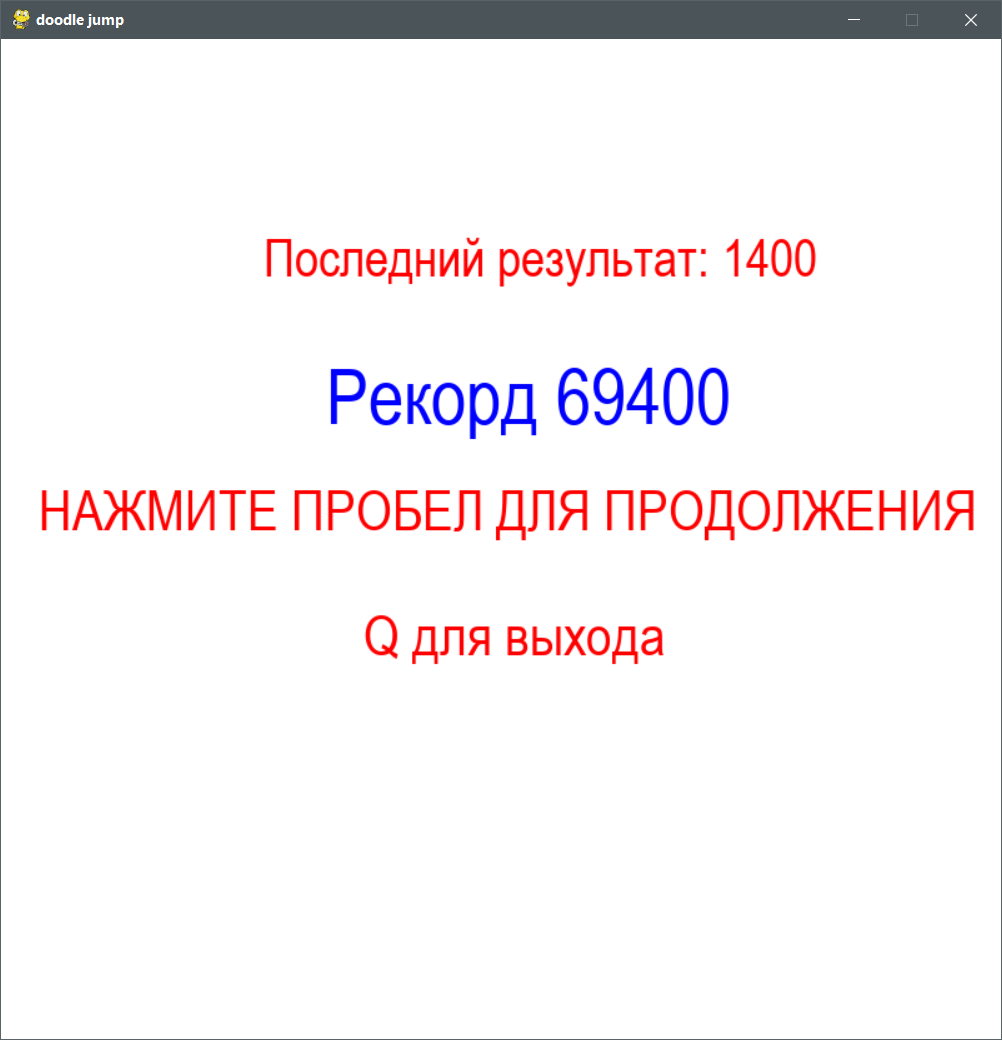


Рисунок 5.1 – Главное меню игры

При нажатии на клавишу «ПРОБЕЛ» у нас запустилось главное окно нашей игры (приводится на рисунке 5.2), на котором прорисован наш игрок.

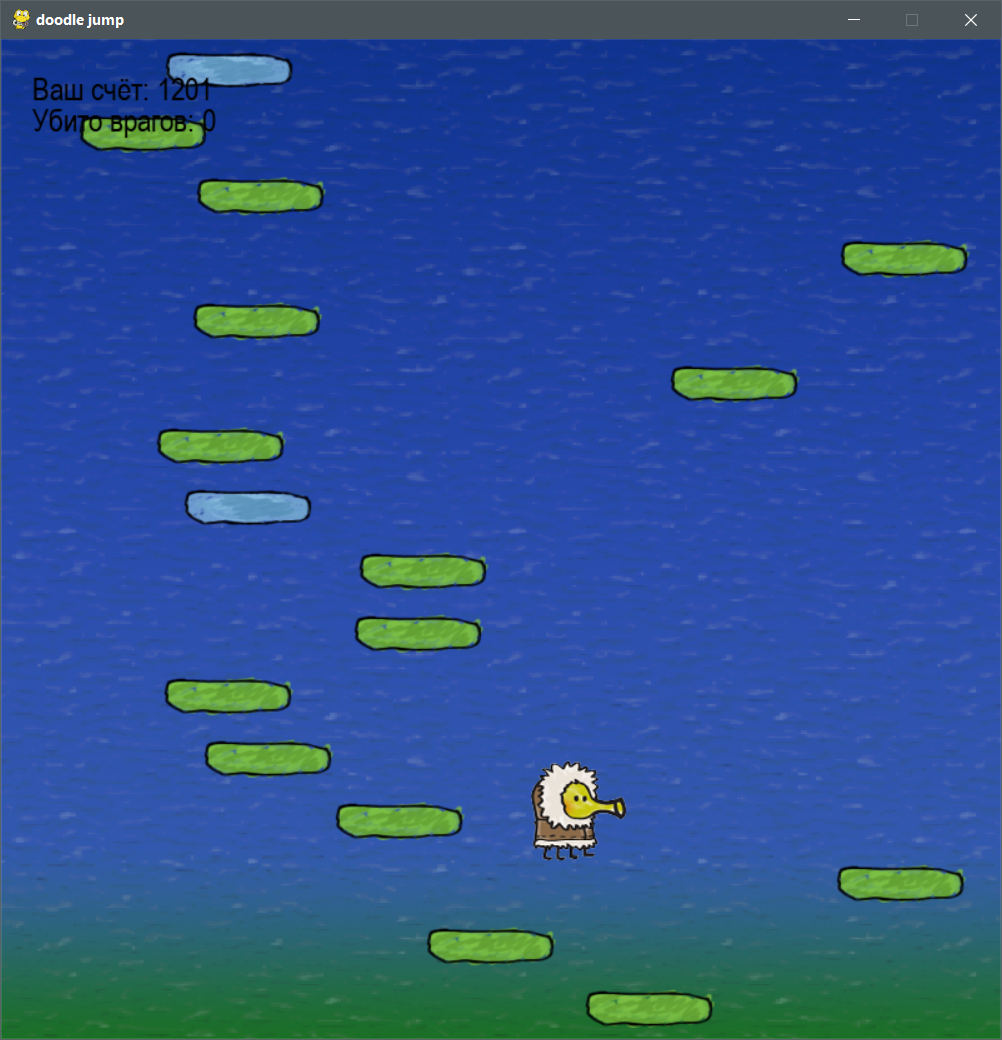


Рисунок 5.2 –Начало игры

После того, как мы пропрыгаем несколько платформ у нас начнут появляться враги (приводится на рисунке 5.3). 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.

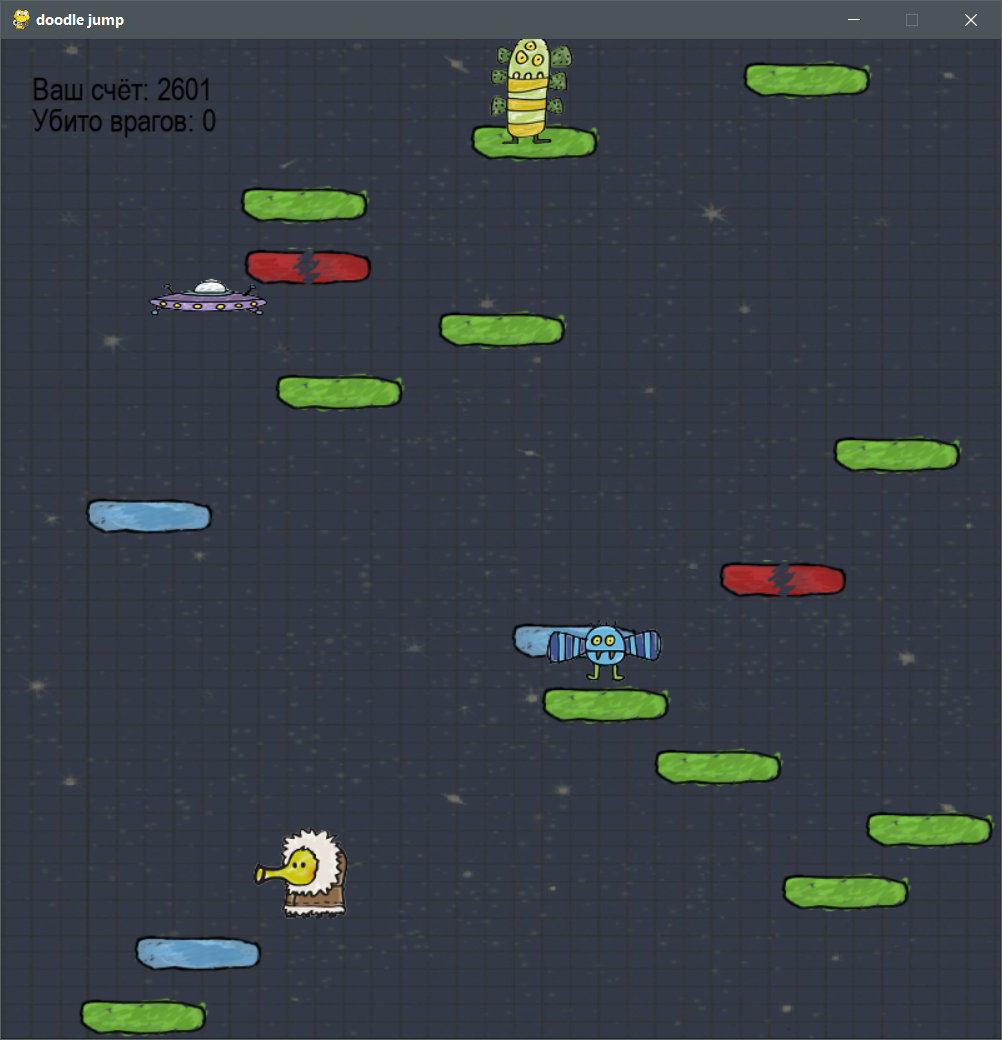


Рисунок 5.3 – Появление врагов

Также мы можем заметить вверху экрана счет игрока (приводится на рисунке 5.4), 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.

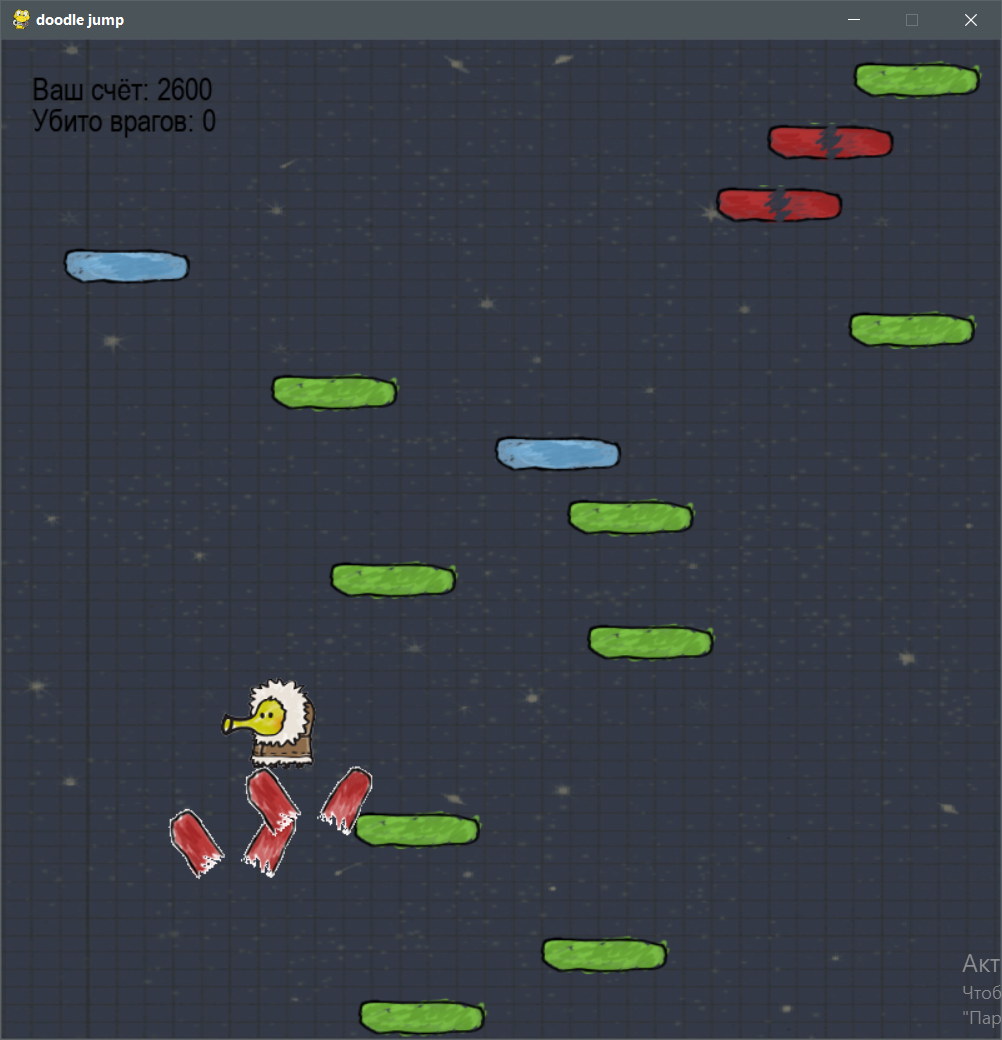


Рисунок 5.4 – Разрушение красных платформ

Для возможности пройти по прямому пути реализована стрельба игрока по врагам. Стрельба реализована только по прямой линии (приводится на рисунке 5.5). 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.

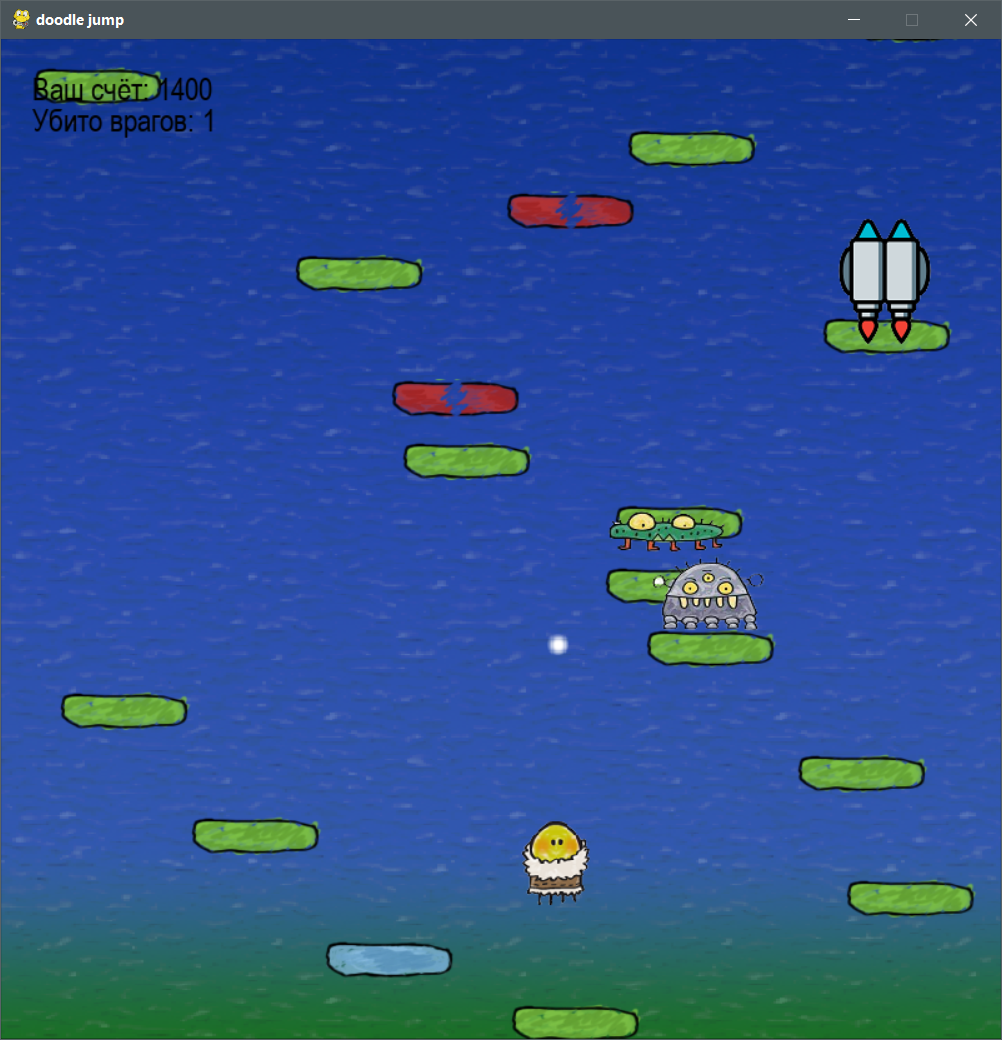


Рисунок 5.5 – Стрельба персонажа

Чтобы нормализовать соотношение интереса и сложности, были введены в игру бустеры (приводится на рисунке 5.6). Они подкидывают персонажа вверх с огромной скоростью, мину все преграды.



Рисунок 5.6 – Бустер-ракета

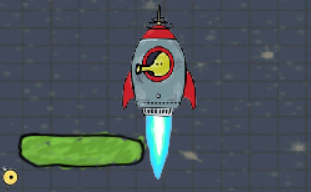


Рисунок 5.7 – Герой в бустер-ракете

Также реализован мини-бустер в виде простой пружины (приводится на рисунке 5.8). Подкидывает персонажа на маленькое расстояние, минуя все преграды.



Рисунок 5.8 – Бустер-пружина

Игру всегда можно поставить на паузу нажав на клавишу **«**ESC» (приводится на рисунке 5.9). Закрытие окна на кнопку **«**Q». Продолжение «SPACE»

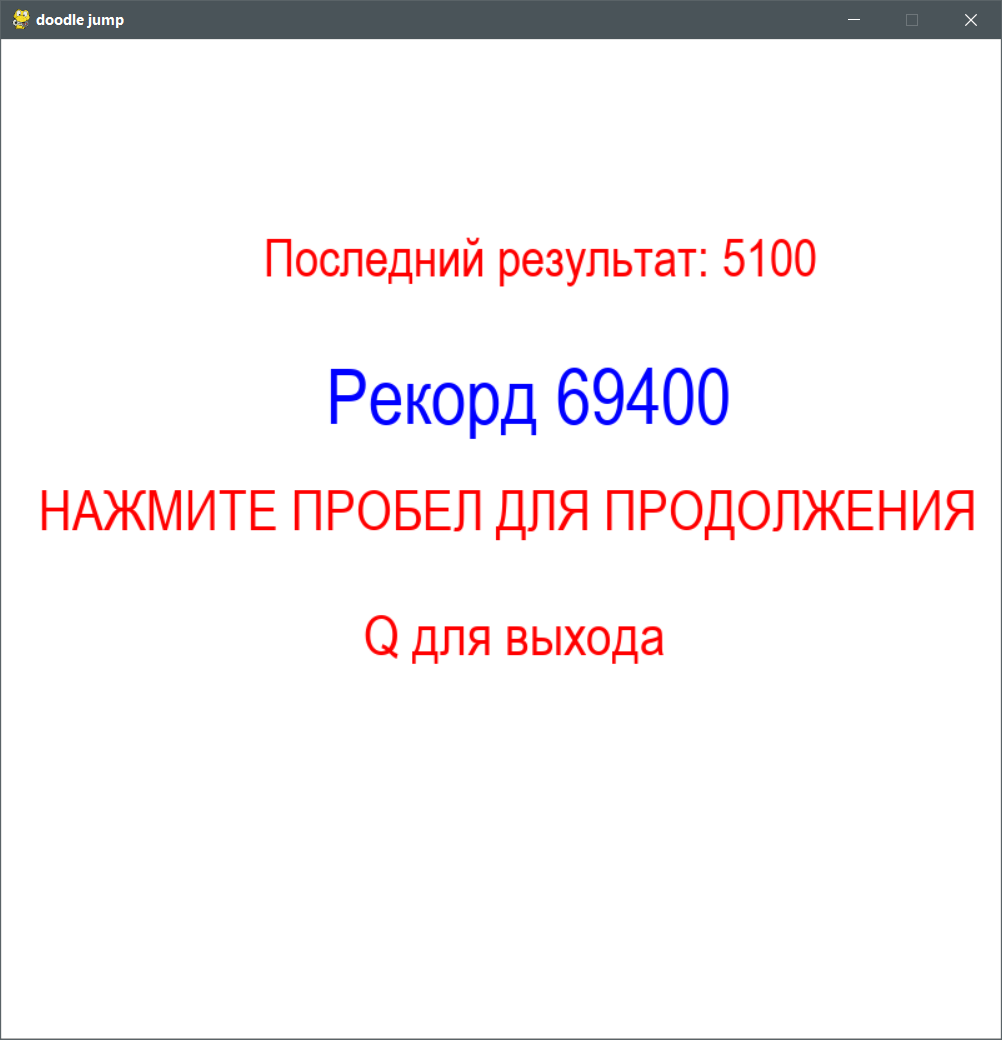


Рисунок 5.9 – Пауза

При проигрыше нас встречает окно поражения (приводится на рисунке 5.10) с предложением начать всё сначала или выйти из игры.

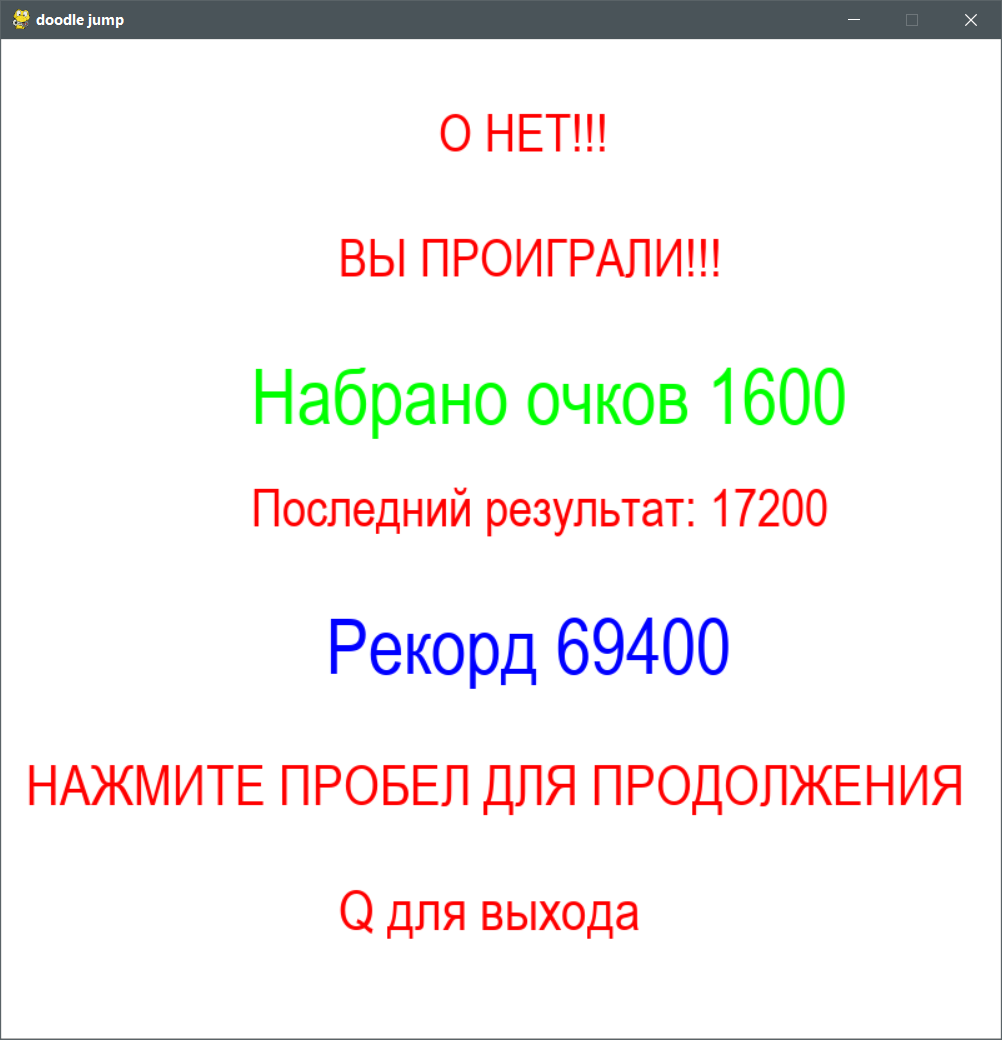


Рисунок 5.10 – Окно проигрыша

# ЗАКЛЮЧЕНИЕ

В курсовом проекте написана и отлажена программа, представляющая собой игру, написанную на языке Python с использованием библиотеки Pygame.

В процессе работы и отладки проекта были изучены новые методы работы, связанные с разработкой окон, загрузкой картинок, шрифтов, звуков. Также изучены прорисовывка объектов, создание игровых окна, подробно рассмотрены способы работы клавиатурой, усвоены навыки работы с фронтэндом и бэкэндом. Изучены методы разработки легко масштабируемого кода.

Данная игра является компьютерным аналогом одной из игр в жанре [платформера](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D1%80), разработанная и выпущенная в 1985 году японской компанией. Её можно модернизировать под Android, а также под кнопочные телефоны и запустить в широкий доступ как отдельный продукт. Поскольку на данный момент популярна разработка приложений для детей, то, возможно, эта игра наберёт большое количество пользователей.

Но, поскольку, игра неидеальна, то можно рассмотреть перспективы дальнейшего усовершенствования программного продукта: сделать возможность выбора персонажей, добавление больше видов бустеров, предусмотреть несколько жизней герою, стрельба врагов.

# 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 1

**Листинг исходного кода**

**Файл main.py**from model.GameModel import Game  
Game().run()

**Файл GameModel.py**

import pygame  
from util.settings import \*  
from util.Doodle import \*  
from util.JetPack import \*  
from util.NLO import \*  
from util.Spring import \*  
from util.Platform import \*  
from util.Bullets import \*  
from view.GameView import doodleDraw  
from view.GameView import drawPlatforms, drawBullet, drawLose, drawStart, drawSpring, fillBackground  
chanceJet = 970  
chanceGreenPlatfrom = 800  
doodle = DoodleJump()  
enemy = Enemy()  
nlo = NLO()  
jetPack = JetPack()  
masEnemy = []  
masEnemy2 = []  
springForGreen = Spring()  
plat = Platform()  
checkForshot = 0  
checkSoungShoot = 0  
chanceEnemy = 940  
class Game:  
 def \_\_init\_\_(self):  
 pygame.font.init()  
 self.font = pygame.font.SysFont("Arial", 25)  
 self.font2 = pygame.font.SysFont("Arial", 42)  
 self.font3 = pygame.font.SysFont("Arial", 62)  
 self.font4 = pygame.font.SysFont("Arial", 45)  
 self.bullet\_png = pygame.image.load("../assets/ice-snow-16.png")  
 self.bullet\_png = pygame.transform.scale(self.bullet\_png, (20,20))  
  
 def updatePlayer(self):  
 if not doodle.jump:  
 doodle.playery += doodle.gravity  
 doodle.gravity += 1  
 elif doodle.jump:  
 doodle.playery -= doodle.jump  
 doodle.jump -= 1  
 if(doodle.jump and doodle.withoutJet and doodle.withoutSpring):  
 pygame.mixer.Sound.play(jump\_sound)  
 key = pygame.key.get\_pressed()  
  
 if key[K\_RIGHT]:  
 if doodle.xmovement < 12: # максимальная скорость  
 doodle.xmovement += 1  
 doodle.direction = 0  
  
 elif key[K\_LEFT]:  
 if doodle.xmovement > -12:  
 doodle.xmovement -= 1  
 doodle.direction = 1  
  
 else:  
 if doodle.xmovement > 0:  
 doodle.xmovement -= 1  
 elif doodle.xmovement < 0:  
 doodle.xmovement += 1  
 if doodle.playerx > 850: #переходы  
 doodle.playerx = -50  
 elif doodle.playerx < -50:  
 doodle.playerx = 850  
 doodle.playerx += doodle.xmovement  
 if (doodle.withoutSpring and doodle.withoutJet and not doodle.playerDead): #перемещение камеры  
 if doodle.playery - doodle.cameray <= 500: #перемщать когда рассттояние 500  
 doodle.cameray -= 7  
 elif (not doodle.withoutJet and not doodle.playerDead):  
 if doodle.playery - doodle.cameray <= 500:  
 doodle.cameray -= 90  
 doodle.visible = False  
 pygame.mixer.Sound.play(pow\_sound)  
 else:  
 if doodle.playery - doodle.cameray <= 500:  
 doodle.cameray -= 50  
 doodle.visible = False  
 pygame.mixer.Sound.play(spring\_sound)  
 # self.visible = False  
 if (doodle.jump < 15 and not doodle.playerDead): #возвращаем к режиму не бога  
 doodle.withoutSpring = True  
 doodle.withoutJet = True  
 doodle.visible = True  
 global checkDirection, checkSoungShoot  
  
 doodleDraw(doodle)  
  
 def updatePlatforms(self):  
 for p in plat.platforms:  
 rect = pygame.Rect(p[0], p[1], plat.green.get\_width() - 10, plat.green.get\_height())  
 player = pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width() - 10,  
 doodle.playerRight.get\_height())  
 if rect.colliderect(player) and doodle.gravity and doodle.playery < (  
 p[1] - doodle.cameray) and doodle.visible:  
 if p[2] != 2:  
 doodle.jump = 15  
 doodle.gravity = 0  
 else:  
 p[-1] = 1  
  
 if(p[2]==2 and p[-1]==1): #0-зелёная, 1-синий , 2 - красный  
 p[1]+=8  
  
 if p[2] == 1: #движение синей  
 if p[-1] == 1:  
 p[0] += 5  
 if p[0] > 550:  
 p[-1] = 0  
 else:  
 p[0] -= 5  
 if p[0] <= 0:  
 p[-1] = 1  
  
  
  
 def logicPlatforms(self, timeShootImg):  
 global masEnemy2, masEnemy, chanceGreenPlatfrom, chanceEnemy, score, chanceJet  
 for p in plat.platforms:  
 check = plat.platforms[1][1] - doodle.cameray  
 if check > 800: # на какой высоте появляться  
 platform = random.randint(0, 1000)  
 if platform < chanceGreenPlatfrom:  
 platform = 0  
 elif platform < 900:  
 platform = 1  
 else:  
 platform = 2  
 #  
 plat.platforms.append([random.randint(0, 700), plat.platforms[-1][1] - 50, platform, 0, 0])  
 #  
 coords3 = plat.platforms[-1]  
 checkForJetPack = random.randint(0,1000)  
  
 if(score%50000==1):  
 chanceJet-=5  
 if (score % 50000 == 1):  
 chanceGreenPlatfrom-=100  
 if (checkForJetPack > chanceJet and platform == 0 and plat.platforms[-1][-2]!=1):  
 plat.platforms[-1][-2] = 1  
 jetPack.jetPacks.append([coords3[0], coords3[1] - 25, 0]) #на последнюю вещаем  
 #  
 #  
 coords2 = plat.platforms[-1]  
 checkForEnemy = random.randint(0, 1000)  
  
 if(score%350000==0):  
 chanceEnemy-=5  
 if (checkForEnemy > chanceEnemy and platform == 0 and plat.platforms[-1][-2]!=1):  
 plat.platforms[-1][-2] = 1  
 newEnemy = Enemy()  
 masEnemy.append(newEnemy)  
 enemy.enemys.append([coords2[0], coords2[1] - 25, 0])  
 #  
 coords = plat.platforms[-1]  
 checkForNlo = random.randint(0, 1000)  
 if (checkForNlo > 980 and platform == 0):  
 newEnemy = NLO()  
 masEnemy2.append(newEnemy)  
 nlo.enemys.append([coords2[0], coords2[1] - 25, 0])  
 check = random.randint(0, 1000)  
 if check > 950 and platform == 0 and plat.platforms[-1][-2]!=1: # шанс рандом для пружины  
 plat.platforms[-1][-2] = 1  
 springForGreen.springs.append([coords[0], coords[1] - 25, 0])  
 plat.platforms.pop(0)  
 score += 100  
 drawPlatforms(p,plat, doodle)  
 #  
  
 # drawEnemies(masEnemy,enemy,doodle, deadMostr\_sound)  
 global scoreEnemy  
  
 if (masEnemy):  
 count = 0  
 for enem in enemy.enemys:  
 screen.blit(masEnemy[count].enemyPlayer, (enem[0], enem[1] - doodle.cameray - 53))  
 if(doodle.visible and doodle.gravity and pygame.Rect(enem[0], enem[1],  
 masEnemy[len(masEnemy) - 1].enemyPlayer.get\_width(),  
 masEnemy[  
 len(masEnemy) - 1].enemyPlayer.get\_height() - 53).colliderect(  
 pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(),  
 doodle.playerRight.get\_height()))):  
 doodle.jump = 50  
 # enem[0] = -1000  
 # enem[1] = 1000  
 enemy.enemys.pop(count)  
 masEnemy.pop(count)  
 print(doodle.gravity)  
 print(masEnemy)  
  
 scoreEnemy+=1  
  
 elif (doodle.visible and pygame.Rect(enem[0], enem[1],  
 masEnemy[len(masEnemy) - 1].enemyPlayer.get\_width(),  
 masEnemy[  
 len(masEnemy) - 1].enemyPlayer.get\_height() - 53).colliderect(  
 pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(),  
 doodle.playerRight.get\_height()))):  
 pygame.mixer.Sound.play(deadMostr\_sound)  
 doodle.jump = 20  
 doodle.cameray -= 20  
 doodle.gravity = 10  
 doodle.cameray += 10  
 doodle.visible = False  
 doodle.playerDead = True  
 count += 1  
  
 # drawJet(jetPack,doodle)  
  
 for jet in jetPack.jetPacks:  
 screen.blit(jetPack.jetPackImg, (jet[0], jet[1] - doodle.cameray - 53))  
 if doodle.visible and pygame.Rect(jet[0], jet[1], jetPack.jetPackImg.get\_width(),  
 jetPack.jetPackImg.get\_height()).colliderect(  
 pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(),  
 doodle.playerRight.get\_height())):  
 doodle.jump = 100  
 doodle.withoutJet = False  
  
 if(masEnemy2):  
 for newNlo in nlo.enemys:  
  
 count = 0  
 global checkSoungShoot  
 if timeShootImg%15!=0:  
 # print(timeShootImg)  
 screen.blit(masEnemy2[count].enemyPlayer, (newNlo[0], newNlo[1] - doodle.cameray - 53))  
  
 if (doodle.visible and doodle.gravity and pygame.Rect(newNlo[0], newNlo[1],masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_width(), masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_height() - 53).colliderect(pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(),doodle.playerRight.get\_height()))):  
 doodle.jump = 50  
 newNlo.pop(count)  
 scoreEnemy += 1  
 elif (doodle.visible and pygame.Rect(newNlo[0], newNlo[1],masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_width(), masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_height() - 53).colliderect(pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(),doodle.playerRight.get\_height()))):  
 pygame.mixer.Sound.play(deadNLO\_sound)  
 doodle.jump = 5  
 doodle.cameray -= 1  
 doodle.gravity = 1  
 doodle.cameray += 5  
 doodle.visible = False  
 doodle.playerDead = True  
 masEnemy2[count].enemyPlayer = nlo.enemyPlayerEnd  
 count += 1  
 for spring in springForGreen.springs:  
 drawSpring(spring,springForGreen,doodle)  
 if doodle.visible and pygame.Rect(spring[0], spring[1], springForGreen.spring.get\_width(), springForGreen.spring.get\_height()).colliderect(pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(),doodle.playerRight.get\_height())):  
 doodle.jump = 50  
 doodle.withoutSpring = False  
  
 def check\_kill(self):  
 global scoreEnemy  
 for bul in doodle.bullets:  
  
 countEnemy = 0  
 for enem in enemy.enemys:  
 if (pygame.Rect(bul.x, bul.realY, self.bullet\_png.get\_width(),  
 self.bullet\_png.get\_height()).colliderect(enem[0], enem[1],  
 masEnemy[len(masEnemy) - 1].enemyPlayer.get\_width(),  
 masEnemy[len(masEnemy) - 1].enemyPlayer.get\_height())) or (doodle.visible and doodle.gravity and pygame.Rect(enem[0], enem[1],  
 masEnemy[len(masEnemy) - 1].enemyPlayer.get\_width(),  
 masEnemy[  
 len(masEnemy) - 1].enemyPlayer.get\_height() - 53).colliderect(  
 pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(),  
 doodle.playerRight.get\_height()))):  
 # enem[0] = -1000  
 # enem[1] = 1000  
 enemy.enemys.pop(countEnemy)  
 masEnemy.pop(countEnemy)##  
 print(doodle.gravity)  
  
 doodle.bullets.remove(bul)  
 doodle.bullets = []  
 scoreEnemy+=1  
 break  
 countEnemy+=1  
 countEnemy = 0  
 for newNlo in nlo.enemys:  
 # print("self.bullet\_png.get\_width() "+str(masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_width()) +" "+ str(masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_height()) )  
 if pygame.Rect(bul.x, bul.realY, self.bullet\_png.get\_width(), self.bullet\_png.get\_height()).colliderect(newNlo[0], newNlo[1], masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_width(), masEnemy2[len(masEnemy2) - 1].enemyPlayer.get\_height() - 53):  
 nlo.enemys.pop(countEnemy)  
 doodle.bullets.remove(bul)  
 scoreEnemy+=1  
 break  
 countEnemy += 1  
 def run(self):  
 start = False  
 clock = pygame.time.Clock()  
 plat.generatePlatforms()  
 wait = True  
 global score, scoreEnemy  
 check = 0  
  
 while True:  
 timeShootImg=0  
 key = pygame.key.get\_pressed()  
  
 for ev in pygame.event.get():  
 if ev.type == QUIT:  
 sys.exit()  
  
 if(key[K\_SPACE]):  
 wait = True  
 start = True  
 pygame.mixer.music.pause()  
 if(key[K\_q]):  
 pygame.quit()  
 sys.exit()  
 while wait:  
 pygame.mixer.music.play(-1)  
 clock.tick(60)  
 key = pygame.key.get\_pressed()  
 for ev in pygame.event.get():  
 if ev.type == QUIT:  
 sys.exit()  
 if(key[K\_ESCAPE]):  
 wait = False  
 start = False  
 if (score <= 700):  
 bgX = bg  
 fillBackground(score, bgX)  
 elif (score <= 1000):  
 bgX = bgS  
 fillBackground(score,bgX)  
 elif (score <= 1500):  
 bgX = bg3  
 fillBackground(score,bgX)  
 elif (score <= 2000):  
 bgX = bgF  
 fillBackground(score,bgX)  
  
 elif (score <= 3000):  
 bgX = bg2  
 fillBackground(score,bgX)  
 elif (score <= 5000000):  
 bgX = bgE  
 fillBackground(score, bgX)  
  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 sys.exit()  
  
 if math.fabs(doodle.playery - doodle.cameray > 740):  
 global masEnemy2, masEnemy  
 doodle.cameray = 0  
 f = open("../res.txt", "ab+")  
 f.write((str(score) + '\n').encode())  
 f.close()  
 check = score  
 score = 0  
 scoreEnemy = 0  
 springForGreen.springs = []  
 enemy.enemys =[]  
  
 jetPack.jetPacks = []  
 nlo.enemys = []  
 nlo.enemyPlayer = pygame.image.load("../assets/nloFirst.png").convert\_alpha()  
 masEnemy2 = []  
 masEnemy = []  
 doodle.bullets = []  
 plat.platforms = [[400, 500, 0, 0]]  
 plat.generatePlatforms()  
 doodle.playerx = 400  
 doodle.playery = 400  
 doodle.visible = True  
 doodle.playerDead = False  
 wait = False  
  
  
 # platform.drawGrid()  
 self.check\_kill()  
  
 self.logicPlatforms(timeShootImg)  
 self.updatePlayer()  
 self.updatePlatforms()  
 nlo.updateNlos()  
 global checkForshot, checkSoungShoot  
 if (doodle.direction != 2):  
 checkForshot = doodle.direction  
 # print(checkForshot)  
 else:  
 if(timeShootImg%25==0):  
 checkSoungShoot += 1  
 doodle.direction = checkForshot  
 # print("d" + str(doodle.direction))  
 doodle.shoot()  
 # if(doodle.bullets):  
 # screen.blit(doodle.playerShoot, (doodle.playerx, doodle.playery - doodle.cameray))  
 for bullet in doodle.bullets:  
 drawBullet(self.bullet\_png,bullet.x, bullet.y)  
  
 timeShootImg+=1  
  
 screen.blit(self.font.render("Ваш счёт: "+str(score), -1, (0, 0, 0)), (25, 25))  
 screen.blit(self.font.render("Убито врагов: "+str(scoreEnemy), -1, (0, 0, 0)), (25, 50))  
 # print(doodle.direction)  
 lastResult2 = lastResult()  
 maxResult2 = maxResult()  
 if(not wait):  
  
 drawLose(self.font2, self.font3, self.font4,check, str(lastResult2), maxResult())  
 pygame.mixer.music.unpause()  
  
 if(not start):  
 wait = False  
 drawStart(self.font2, self.font3, self.font4, lastResult2, maxResult2)  
 pygame.mixer.music.unpause()  
  
  
 pygame.display.flip()

**Файл Bullet.py**  
class Bullet:  
 def \_\_init\_\_(self, x, y, plY):  
 self.y = y+5  
 self.x = x+30  
 self.realY = plY  
  
  
 def move(self):  
 self.y -= 20  
 self.realY-=20  
  
 def off\_screen(self):  
 return not self.y>0

**Файл Doodle.py**

import math  
  
import pygame  
from pygame.locals import \*  
import sys  
  
from util.Bullets import Bullet  
  
  
class DoodleJump:  
 def \_\_init\_\_(self):  
 self.visible = True  
 self.playerRight = pygame.image.load("../assets/ice-right\_2x.png").convert\_alpha()  
 self.playerRight\_1 = pygame.image.load("../assets/ice-right-odskok\_2x.png").convert\_alpha()  
 self.playerLeft = pygame.image.load("../assets/ice-left\_2x.png").convert\_alpha()  
 self.playerLeft\_1 = pygame.image.load("../assets/ice-left-odskok\_2x.png").convert\_alpha()  
 self.playerDeadImg = pygame.image.load("../assets/deadDoodle.png").convert\_alpha()  
 self.playerShoot = pygame.image.load("../assets/ice-puca\_2x.png").convert\_alpha()  
 self.playerShoot = pygame.transform.scale(self.playerShoot,(90,90))  
 self.playerDead = False  
 self.direction = 0  
 self.playerx = 400  
 self.playery = 400  
 self.cameray = 0  
 self.jump = 0  
 self.gravity = 0  
 self.xmovement = 0  
 self.withoutSpring = True  
 self.withoutJet = True  
 self.playerRightRacket = pygame.image.load("../assets/racketright (2).png").convert\_alpha()  
 self.playerLeftRacket = pygame.image.load("../assets/racket.png").convert\_alpha()  
 self.playerLeftRacket = pygame.transform.scale(self.playerLeftRacket, (150,150))  
 self.playerRightRacket = pygame.transform.scale(self.playerRightRacket, (150,150))  
 self.bullets = []  
 self.cool\_down\_counts = 0  
 self.continuesKillEneme = False  
  
 def coolDown(self):  
 if self.cool\_down\_counts > 50:  
 self.cool\_down\_counts = 0  
 elif self.cool\_down\_counts > 0:  
 self.cool\_down\_counts += 1  
  
 def shoot(self):  
 key = pygame.key.get\_pressed()  
 self.coolDown()  
 for ev in pygame.event.get():  
 if ev.type == QUIT:  
 sys.exit()  
  
 if (key[K\_SPACE] and self.cool\_down\_counts==0 and not self.playerDead and self.withoutJet and self.withoutSpring):  
 bullet = Bullet(self.playerx, self.playery+math.fabs(self.cameray), self.playery)  
 self.bullets.append(bullet)  
 self.cool\_down\_counts = 1  
 self.direction = 2  
  
  
 for bullet in self.bullets:  
 bullet.move()  
 if(bullet.off\_screen()):  
 self.bullets.remove(bullet)

**Файл JetPack.py**

import pygame  
  
class JetPack:  
 def \_\_init\_\_(self):  
 self.jetPackImg = pygame.image.load("../assets/Jet.png").convert\_alpha()  
 self.jetPacks = []  
 self.rect = self.jetPackImg.get\_rect()

**Файл Enemy.py**

import random  
import pygame  
  
class Enemy:  
 def \_\_init\_\_(self):  
 check = random.randint(0,7)  
 self.enemyPlayerEmpty = pygame.image.load("../assets/mnogon.png").convert\_alpha()  
 self.enemyPlayerEmpty = pygame.transform.scale(self.enemyPlayerEmpty,(0,0))  
 self.enemyPlayer = 0  
 if(check==0):  
 self.enemyPlayer = pygame.image.load("../assets/mnogon.png").convert\_alpha()  
 elif(check==1):  
 self.enemyPlayer = pygame.image.load("../assets/mostr1.png").convert\_alpha()  
 elif(check==2):  
 self.enemyPlayer = pygame.image.load("../assets/mostr2.png").convert\_alpha()  
 elif(check==3):  
 self.enemyPlayer = pygame.image.load("../assets/mostr3.png").convert\_alpha()  
 elif(check==4):  
 self.enemyPlayer = pygame.image.load("../assets/mostr4.png").convert\_alpha()  
 elif(check==5):  
 self.enemyPlayer = pygame.image.load("../assets/mostr5.png").convert\_alpha()  
 else:  
 self.enemyPlayer = pygame.image.load("../assets/mosntr6.png").convert\_alpha()  
  
 self.enemys = []  
 self.rect = self.enemyPlayer.get\_rect()

**Файл NLO.py**

import pygame  
from util.Enemy import \*  
  
class NLO(Enemy):  
 def \_\_init\_\_(self):  
 Enemy.\_\_init\_\_(self)  
 self.enemyPlayerEnd = pygame.image.load("../assets/nlo.png").convert\_alpha()  
 self.enemyPlayer = pygame.image.load("../assets/nloFirst.png").convert\_alpha()  
 def updateNlos(self):  
 for p in self.enemys:  
 if(p[0]%3==0):  
 # print(p[0])  
 self.enemyPlayer = pygame.image.load("../assets/bac.png")  
 else:  
 self.enemyPlayer = pygame.image.load("../assets/nloFirst.png").convert\_alpha()  
 if p[-1] == 1:  
 p[0] += 1  
 if p[0] > 550:  
 p[-1] = 0  
 else:  
 p[0] -= 1  
 if p[0] <= 0:  
 p[-1] = 1

**Файл Platform.py**

import pygame  
import random  
  
class Platform:  
  
 def \_\_init\_\_(self):  
 self.green = pygame.image.load("../assets/green.png").convert\_alpha()  
 self.platforms = [[400, 500, 0, 0, 0]] #x,y, type, close, action  
  
 self.blue = pygame.image.load(  
 "../assets/blue.png").convert\_alpha() #ипикселей, что и на экране  
 self.red = pygame.image.load("../assets/red.png").convert\_alpha()  
 self.red\_1 = pygame.image.load("../assets/red\_1.png").convert\_alpha()  
 def generatePlatforms(self): # генерация первых платформ  
 on = 800  
 while on > -100:  
 x = random.randint(0, 700)  
 platform = random.randint(0, 1000)  
 if platform < 800:  
 platform = 0 # зеленые  
 elif platform < 900:  
 platform = 1 # синие  
 else:  
 platform = 2 # красные  
 self.platforms.append([x, on, platform, 0])  
 on -= 50

**Файл Settings.py**

import pygame  
  
screen = pygame.display.set\_mode((800, 800))  
pygame.display.set\_caption('doodle jump')  
score = 1  
scoreEnemy = 0  
pygame.init()  
pygame.mixer.music.load("../sounds/intro.mp3")  
pygame.mixer.music.set\_volume(1)  
jump\_sound = pygame.mixer.Sound('../sounds/jump.wav')  
jump\_sound.set\_volume(0.2)  
deadMostr\_sound = pygame.mixer.Sound('../sounds/deadMostr.wav')  
deadMostr\_sound.set\_volume(0.2)  
deadNLO\_sound = pygame.mixer.Sound('../sounds/deadNLO.wav')  
deadNLO\_sound.set\_volume(0.2)  
soundshoot = pygame.mixer.Sound('../sounds/shot.wav')  
pow\_sound = pygame.mixer.Sound("../sounds/pow.wav")  
pow\_sound.set\_volume(0.2)  
spring\_sound = pygame.mixer.Sound("../sounds/spring.wav")  
spring\_sound.set\_volume(0.2)  
bg = pygame.image.load("../assets/background.png")  
bg = pygame.transform.scale(bg, (800, 800))  
bgS = pygame.image.load("../assets/soccer-bck@2x.png")  
bgS = pygame.transform.scale(bgS, (800, 800))  
bgF = pygame.image.load("../assets/underwater-bck@2x.png")  
bgF = pygame.transform.scale(bgF, (800, 800))  
bg2 = pygame.image.load("../assets/secondLocation.png")  
bg2 = pygame.transform.scale(bg2, (800, 800))  
bg3 = pygame.image.load("../assets/bk3.png")  
bg3 = pygame.transform.scale(bg3, (800, 800))  
bgE = pygame.image.load("../assets/space-bck@2x.png")  
bgE = pygame.transform.scale(bgE, (800, 800))  
def maxResult():  
 f = open("../res.txt", "r")  
 max = 0  
 for i in f:  
 i = i.strip("\n")  
 max = int(max)  
 if max < int(i):  
 max = int(i)  
 f.close()  
 return str(max)  
  
def lastResult():  
 f = open("../res.txt", "r")  
 a = []  
 for i in f:  
 a.append(i.strip('\n'))  
 f.close()  
 return a[len(a)-2]

**Файл Spring.py**

import pygame  
class Spring:  
 def \_\_init\_\_(self):  
 self.spring = pygame.image.load("../assets/spring.png").convert\_alpha()  
 self.spring\_1 = pygame.image.load("../assets/spring\_1.png").convert\_alpha()  
 self.springs = []

**Файл GameView.py**

from util.settings import screen  
from util.settings import soundshoot  
import pygame  
  
def doodleDraw(doodle, ):  
 if (doodle.direction == 2 and not doodle.playerDead and doodle.withoutJet):  
 screen.blit(doodle.playerShoot, (doodle.playerx, doodle.playery - doodle.cameray))  
 # if checkSoungShoot == 0:  
 # pygame.mixer.Sound.play(soundshoot)  
 elif doodle.direction == 0 and not doodle.playerDead and doodle.withoutJet:  
 # checkSoungShoot = 0  
 if doodle.jump:  
 screen.blit(doodle.playerRight\_1, (doodle.playerx, doodle.playery - doodle.cameray))  
 else:  
 screen.blit(doodle.playerRight, (doodle.playerx, doodle.playery - doodle.cameray))  
 elif (doodle.direction == 1 and not doodle.playerDead and doodle.withoutJet):  
 checkSoungShoot = 0  
 if doodle.jump:  
 screen.blit(doodle.playerLeft\_1, (doodle.playerx, doodle.playery - doodle.cameray))  
 else:  
 screen.blit(doodle.playerLeft, (doodle.playerx, doodle.playery - doodle.cameray))  
  
 if (doodle.playerDead):  
 screen.blit(doodle.playerDeadImg, (doodle.playerx, doodle.playery - doodle.cameray))  
 if not doodle.withoutJet:  
 if not doodle.direction:  
 screen.blit(doodle.playerRightRacket, (doodle.playerx, doodle.playery - doodle.cameray))  
  
 else:  
 screen.blit(doodle.playerLeftRacket, (doodle.playerx, doodle.playery - doodle.cameray))  
  
  
  
def drawPlatforms(p, plat, doodle):  
 if p[2] == 0: # прорисовка зелёных  
 screen.blit(plat.green, (p[0], p[1] - doodle.cameray))  
 elif p[2] == 1: # прорисовка синих  
 screen.blit(plat.blue, (p[0], p[1] - doodle.cameray))  
 elif p[2] == 2:  
 if not p[-1]:  
 screen.blit(plat.red, (p[0], p[1] - doodle.cameray))  
 else:  
 screen.blit(plat.red\_1, (p[0], p[1] - doodle.cameray))  
  
def drawBullet(bullet\_png, x, y):  
 screen.blit(bullet\_png, (x+5, y))  
def drawLose(font2, font3, font4, check, lastResult, maxResult):  
 screen.fill((255, 255, 255))  
 screen.blit(font2.render(str("О НЕТ!!!"), -1, (255, 0, 0)), (350, 50))  
 screen.blit(font2.render(str("ВЫ ПРОИГРАЛИ!!!"), -1, (255, 0, 0)), (270, 150))  
 screen.blit(font3.render(str("Набрано очков " + str(check)), -1, (0, 255, 0)), (200, 250))  
 screen.blit(font2.render(str("Последний результат: " + lastResult), -1, (255, 0, 0)), (200, 350))  
 screen.blit(font3.render(str("Рекорд " + str(maxResult)), -1, (0, 0, 255)), (260, 450))  
 screen.blit(font4.render(str("НАЖМИТЕ ПРОБЕЛ ДЛЯ ПРОДОЛЖЕНИЯ"), -1, (255, 0, 0)), (20, 570))  
 screen.blit(font4.render(str("Q для выхода"), -1, (255, 0, 0)), (270, 670))  
def drawStart(font2, font3, font4, lastResult, maxResult2):  
 screen.fill((255, 255, 255))  
 screen.blit(font2.render(str("Последний результат: " + lastResult), -1, (255, 0, 0)), (210, 150))  
 screen.blit(font3.render(str("Рекорд " + str(maxResult2)), -1, (0, 0, 255)), (260, 250))  
 screen.blit(font4.render(str("НАЖМИТЕ ПРОБЕЛ ДЛЯ ПРОДОЛЖЕНИЯ"), -1, (255, 0, 0)), (30, 350))  
 screen.blit(font4.render(str("Q для выхода"), -1, (255, 0, 0)), (290, 450))  
def drawSpring(spring, springForGreen, doodle):  
 if spring[-1]:  
 screen.blit(springForGreen.spring\_1, (spring[0], spring[1] - doodle.cameray))  
 else:  
 screen.blit(springForGreen.spring, (spring[0], spring[1] - doodle.cameray))  
def fillBackground(score, bg):  
 if (score <= 700):  
 screen.blit(bg, (0, 0))  
 elif (score <= 1000):  
 screen.blit(bg, (0, 0))  
 elif (score <= 5000000):  
 screen.blit(bg, (0, 0))