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 учебной дисциплине «Языки программирования»

на тему: **«Разработка игры «Doudle Jump» на языке Python»**

Выполнил: студент группы 10701320 Акимов Е.С.

Приняла: Стальцова Е.А.

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 «Doudle Jump» на языке Python»

**Исполнитель**: Акимов Е.С.

(подпись)

**Студент 2 курса 10701320 группы**

**Руководитель**: Стальцова Е.А.

(подпись)

Минск 2021

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 3](#_Toc88425144)

[ВВЕДЕНИЕ 4](#_Toc88425145)

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА ПРЕДМЕТ РАЗРАБОТКИ И/ИЛИ СЦЕНАРИЙ ИГРЫ «DOODLE JUMP» 6](#_Toc88425146)

[2. ОБОСНОВАНИЕ ИСПОЛЬЗУЕМЫХ ЯЗЫКОВ И ТЕХНОЛОГИЙ 8](#_Toc88425147)

[3. МОДЕЛИРОВАНИЕ И ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 9](#_Toc88425148)

[4. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ООБЕСПЕЧЕНИЯ 10](#_Toc88425149)

[5. РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 13](#_Toc88425150)

[ЗАКЛЮЧЕНИЕ 17](#_Toc88425151)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 18](#_Toc88425152)

[ПРИЛОЖЕНИЕ 1 19](#_Toc88425153)

# 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. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА ПРЕДМЕТ РАЗРАБОТКИ И/ИЛИ СЦЕНАРИЙ ИГРЫ «DOODLE JUMP»

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.

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. «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. 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 и увеличивается появление ракет.

Если герой все же умирает, то высвечивается окно, на котором написано, что вы проиграли, написан ваш счет, рекордный счет и также предложение нажать «Space», чтобы начать играть заново. 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.

# 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, time, 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

Для реализации данного курсового проекта использовались такие программные модули, как pygame, random, time, sys.

Главным образом в курсовом проекте использовалась библиотека pygame, на основе которой и был написан проект.

Библиотека random использовалась также часто, например, для рандомного определения, где построится новая платформа или же в какой части окна появятся новые платформы, враги и бустеры.

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

# 4. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ООБЕСПЕЧЕНИЯ

При запуске нашей игры первым делом открывается окно (см. рисунок 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.

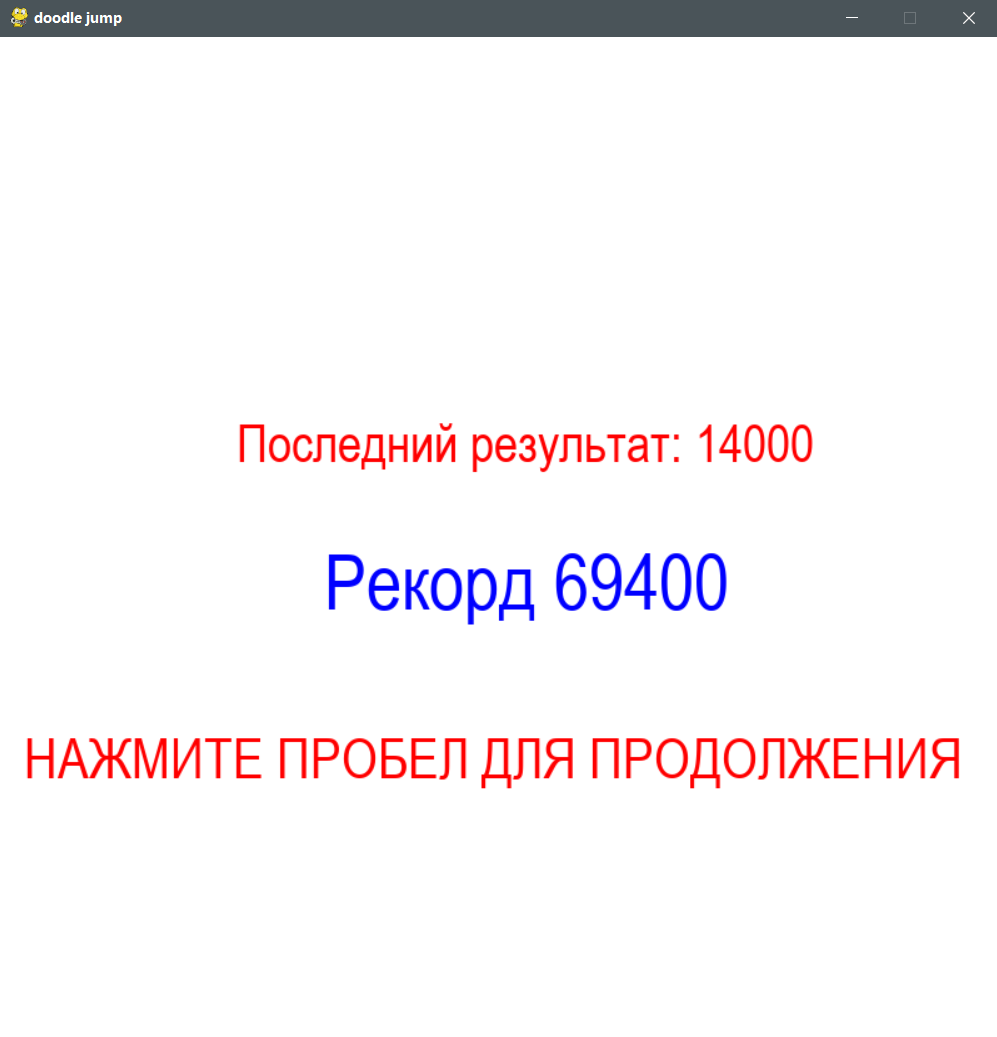


Рисунок 4.1 - стартовое окно игры

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

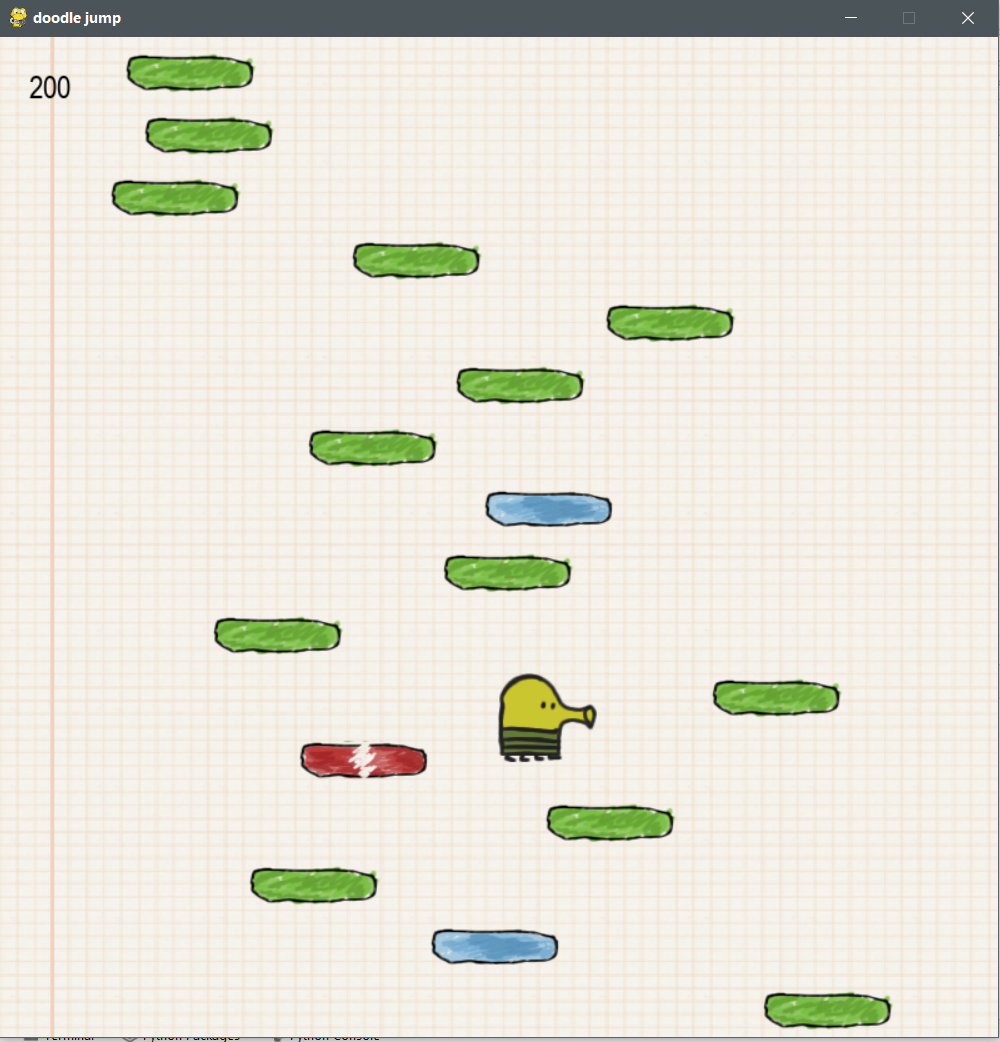


Рисунок 4.2 - Игровое окно

И, конечно же, на нашем экране появляются враги (см. рисунок 4.3), которых он должен аккуратного перепрыгнуть.

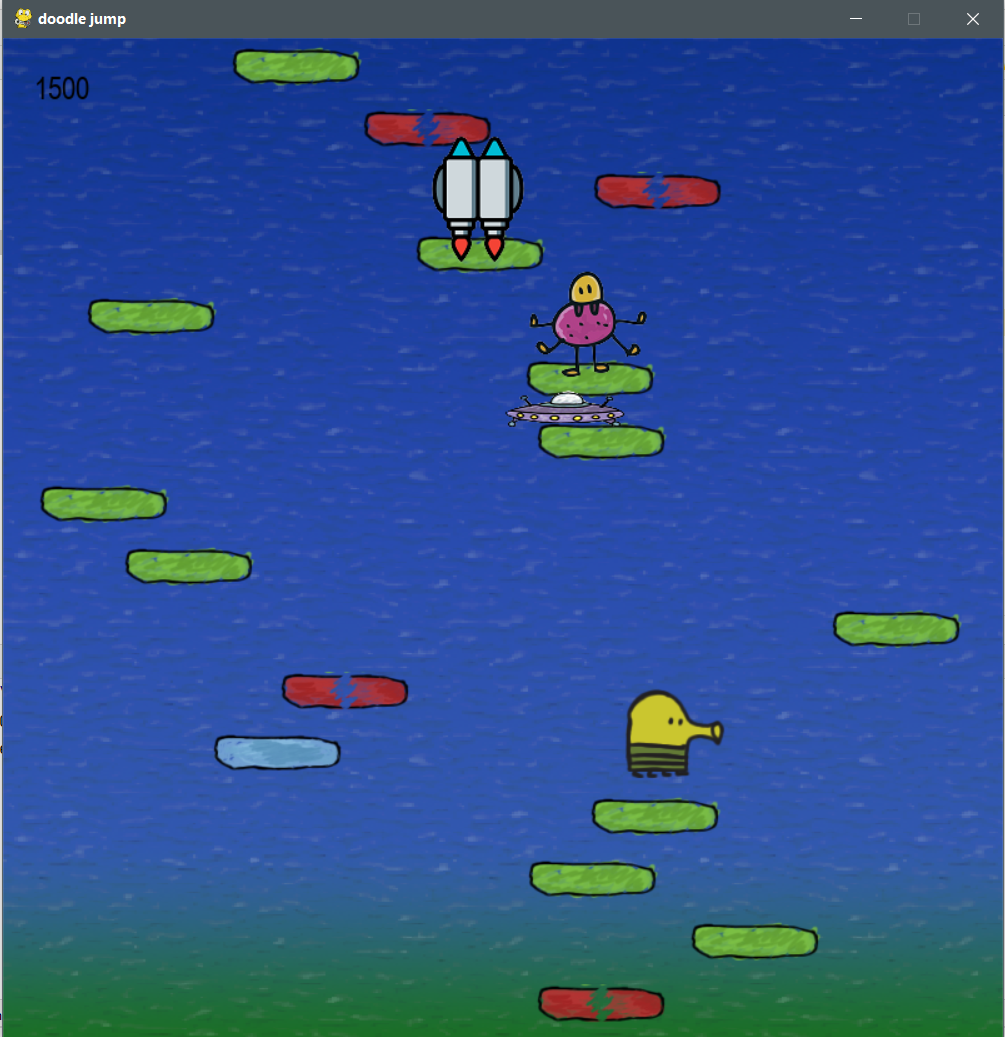


Рисунок 4.3 - Появление врагов

Также мы можем заметить вверху экрана счет игрока (см. рисунок 4.3), при перемещении вверх счет будет увеличиваться. Также созданы бустеры в виде ракеты и пружины (см. рисунок 3).

# 5. РАЗВЕРТЫВАНИЕ И ТЕСТИРОВАНИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Следует рассказать первоначально про руководство об эксплуатации игры, то есть, как играть.

Главный герой перепрыгивает с платформы на платформу в зависимости от нажатой клавиши. Передвижение реализовано через проверку нажатия клавиш вправо-влево. Если клавиша не нажата, то герой стоит на месте. Также хочу отметить, что при зажатии одной из клавиш скорость перемещения героя увеличивается.

При нажатии клавиши «ВПРАВО» персонаж перемещается вправо, соответственно при нажатии клавиши «ВЛЕВО» передвигается влево.

Далее мы начинаем играть, то есть перепрыгивать с платформы на платформу, пытаясь не попасть во врагов. И на пути герой замечает бустер (см. рисунок 5.1)

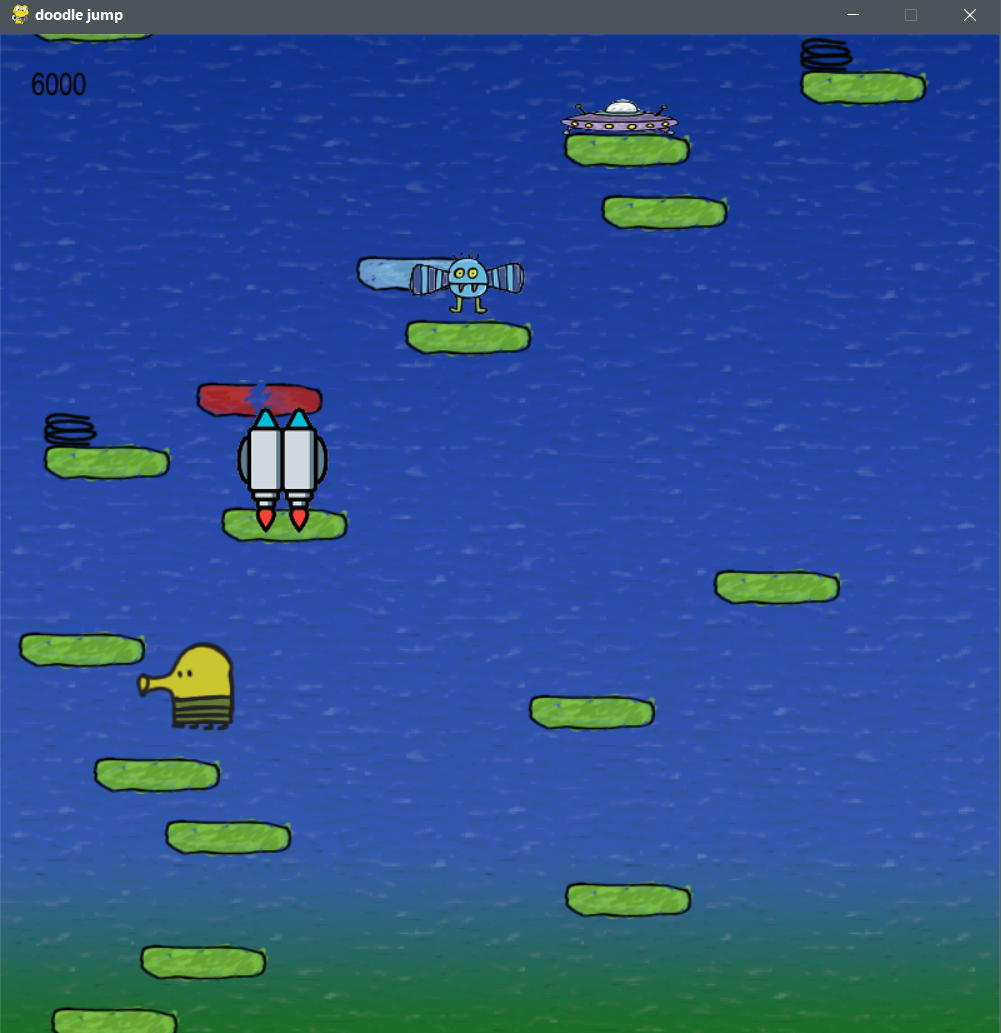


Рисунок 5.1 - нахождение бустера

Далее наш герой прыгает на платформу с бустером и пролетает некоторое расстояния, не обращая внимания даже на врагов.

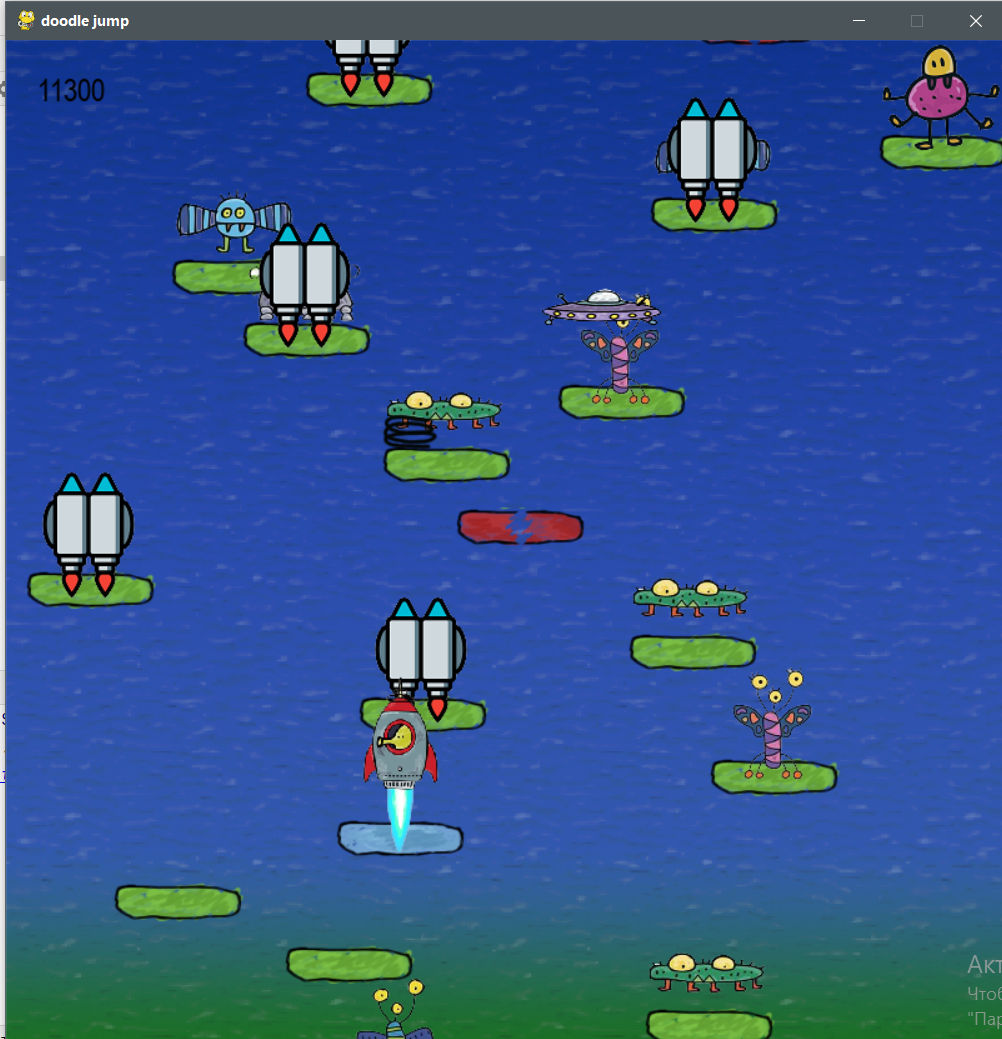


Рисунок 5.2 - Использование бустера

И когда герой все-таки проиграл, то выскакивает окно с результатами (см. рисунок 5.3).

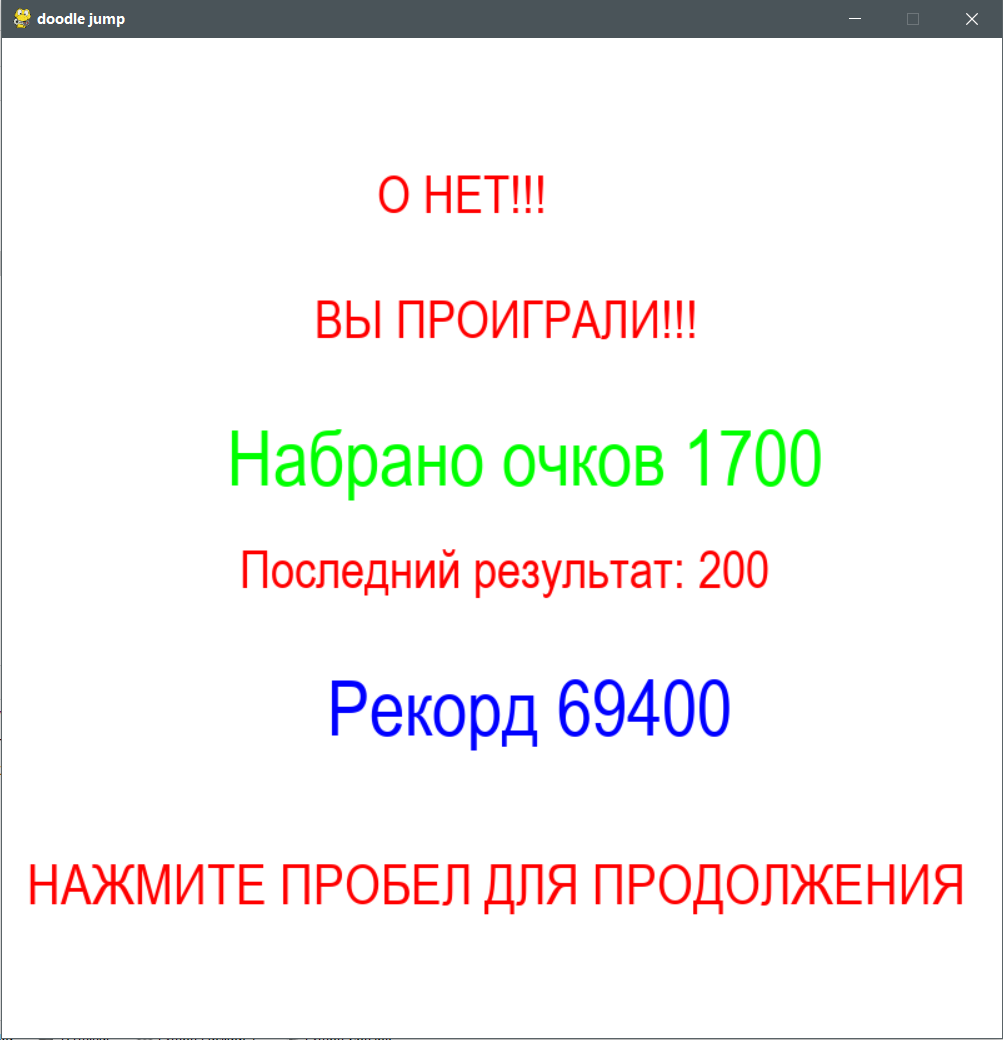


Рисунок 5.3 - окно поражения

Выход из игры происходит через нажатие на крестик в правом верхнем углу экрана.

# ЗАКЛЮЧЕНИЕ

В курсовом проекте написана и отлажена программа, представляющая собой игру, написанную на языке Python с использованием модуля 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 ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
2. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ - Плюс, 2011. – 1280 с.
3. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Сим¬вол-Плюс, 2011. – 992 с.
4. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
5. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.
6. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. – М.: Альт Линукс, 2010. — 126 с. (Библиотека ALT Linux).

# ПРИЛОЖЕНИЕ 1

Листинг исходного кода

**Файл main.py**

import time  
import math  
  
import sys  
import random  
from settings import \*  
from Doodle import \*  
from JetPack import \*  
from Enemy import \*  
from NLO import \*  
from Spring import \*  
from Platform import \*  
import pygame  
chanceJet = 950  
  
  
  
doodle = DoodleJump()  
enemy = Enemy()  
nlo = NLO()  
jetPack = JetPack()  
masEnemy = []  
springForGreen = Spring()  
plat = Platform()  
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)  
  
 def maxResult(self):  
 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(self):  
 f = open(**"res.txt"**, **"r"**)  
 a = []  
 for i in f:  
 a.append(i.strip(**'**\n**'**))  
 f.close()  
 return a[len(a)-2]  
  
 def updatePlayer(self):  
 if not doodle.jump:  
 doodle.playery += doodle.gravity  
 doodle.gravity += 1 # вниз  
 elif doodle.jump:  
 doodle.playery -= doodle.jump # вверх?  
 doodle.jump -= 1  
 if(doodle.jump and doodle.withoutJet and doodle.withoutSpring):  
 pygame.mixer.Sound.play(jump\_sound)  
 key = pygame.key.get\_pressed() # для зажатия клавиши  
 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:  
 doodle.playerx = -50  
 elif doodle.playerx < -50:  
 doodle.playerx = 850  
 doodle.playerx += doodle.xmovement  
 if (doodle.withoutSpring and doodle.withoutJet and not doodle.playerDead):  
 if doodle.playery - doodle.cameray <= 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):  
 doodle.withoutSpring = True  
 doodle.withoutJet = True  
 doodle.visible = True  
 if not doodle.direction and not doodle.playerDead and doodle.withoutJet:  
 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 and not doodle.playerDead and doodle.withoutJet:  
 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 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] == 1:  
 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 drawPlatforms(self):  
 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 < 800:  
 platform = 0  
 elif platform < 900:  
 platform = 1  
 else:  
 platform = 2  
 #  
 plat.platforms.append([random.randint(0, 700), plat.platforms[-1][1] - 50, platform, 0])  
 #  
 coords3 = plat.platforms[-1]  
 checkForJetPack = random.randint(0,1000)  
 global chanceEnemy, score, chanceJet  
 if(score%15000==0):  
 chanceJet-=10  
 if (checkForJetPack > chanceJet and platform == 0):  
 jetPack.jetPacks.append([coords3[0], coords3[1] - 25, 0])  
 #  
 #  
 coords2 = plat.platforms[-1]  
 checkForEnemy = random.randint(0, 1000)  
  
 if(score%15000==0):  
 chanceEnemy-=20  
 if (checkForEnemy > chanceEnemy and platform == 0):  
 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 > 940 and platform == 0):  
 nlo.enemys.append([coords[0], coords2[1] - 25, 0])  
 check = random.randint(0, 1000)  
 if check > 950 and platform == 0: # шанс рандом для пружины  
 springForGreen.springs.append([coords[0], coords[1] - 25, 0])  
 plat.platforms.pop(0)  
 score += 100  
 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[3]:  
 screen.blit(plat.red, (p[0], p[1] - doodle.cameray))  
 else:  
 screen.blit(plat.red\_1, (p[0], p[1] - doodle.cameray))  
 #  
 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 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  
  
  
 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()-53).colliderect(pygame.Rect(doodle.playerx, doodle.playery, doodle.playerRight.get\_width(), doodle.playerRight.get\_height())):  
 doodle.jump = 100  
 doodle.withoutJet = False  
  
 for newNlo in nlo.enemys:  
 screen.blit(nlo.enemyPlayer, (newNlo[0], newNlo[1] - doodle.cameray - 53))  
  
 if (doodle.visible and pygame.Rect(newNlo[0], newNlo[1], nlo.enemyPlayer.get\_width(),nlo.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  
 nlo.enemyPlayer = nlo.enemyPlayerEnd  
  
  
  
 #  
 for spring in springForGreen.springs:  
 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))  
 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 run(self):  
  
 start = False  
 clock = pygame.time.Clock()  
 plat.generatePlatforms()  
 wait = True  
 global score  
 check = 0  
  
 while True:  
  
 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()  
 while wait:  
 pygame.mixer.music.play(-1)  
 clock.tick(60)  
 if (score <= 700):  
 screen.blit(bg, (0, 0))  
 elif (score <= 1000):  
 screen.blit(bg2, (0, 0))  
 elif (score <= 5000000):  
 screen.blit(bg3, (0, 0))  
  
 for event in pygame.event.get():  
 if event.type == QUIT:  
 sys.exit()  
  
 if math.fabs(doodle.playery - doodle.cameray > 740):  
 doodle.cameray = 0  
 f = open(**"res.txt"**, **"ab+"**)  
 f.write((str(score) + **'**\n**'**).encode())  
 f.close()  
 check = score  
 score = 0  
 springForGreen.springs = []  
 enemy.enemys =[]  
 jetPack.jetPacks = []  
 nlo.enemys = []  
 nlo.enemyPlayer = pygame.image.load(**"assets/nloFirst.png"**).convert\_alpha()  
 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.drawPlatforms()  
 self.updatePlayer()  
 self.updatePlatforms()  
 nlo.updateNlos()  
  
 screen.blit(self.font.render(str(score), -1, (0, 0, 0)), (25, 25))  
 if(not wait):  
 screen.fill((255, 255, 255))  
 screen.blit(self.font2.render(str(**"О НЕТ!!!"**), -1, (255, 0, 0)), (300, 100))  
 screen.blit(self.font2.render(str(**"ВЫ ПРОИГРАЛИ!!!"**), -1, (255, 0, 0)), (250, 200))  
 screen.blit(self.font3.render(str(**"Набрано очков "** + str(check)), -1, (0, 255, 0)), (180, 300))  
 screen.blit(self.font2.render(str(**"Последний результат: "** + self.lastResult()), -1, (255, 0, 0)), (190, 400))  
 screen.blit(self.font3.render(str(**"Рекорд "** + str(self.maxResult())), -1, (0, 0, 255)), (260, 500))  
 screen.blit(self.font4.render(str(**"НАЖМИТЕ ПРОБЕЛ ДЛЯ ПРОДОЛЖЕНИЯ"**), -1, (255, 0, 0)), (20, 650))  
 pygame.mixer.music.unpause()  
  
 if(not start):  
 wait = False  
 screen.fill((255, 255, 255))  
 screen.blit(self.font2.render(str(**"Последний результат: "** + self.lastResult()), -1, (255, 0, 0)), (190, 300))  
 screen.blit(self.font3.render(str(**"Рекорд "** + str(self.maxResult())), -1, (0, 0, 255)), (260, 400))  
 screen.blit(self.font4.render(str(**"НАЖМИТЕ ПРОБЕЛ ДЛЯ ПРОДОЛЖЕНИЯ"**), -1, (255, 0, 0)), (20, 550))  
 pygame.mixer.music.unpause()  
  
  
 pygame.display.flip()  
  
  
  
Game().run()

**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]] # ширина где появ,  
  
 self.blue = pygame.image.load(  
 **"assets/blue.png"**).convert\_alpha() # platform move #используются для преобразования поверхностей в тот же формат пикселей, что и на экране  
 self.red = pygame.image.load(**"assets/red.png"**).convert\_alpha() # platform hurt  
 self.red\_1 = pygame.image.load(**"assets/red\_1.png"**).convert\_alpha() # platform hurt2  
  
  
  
 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 Spring.py**

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

**Файл NLO.py**

import pygame  
from 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[-1] == 1:  
 p[0] += 1  
 if p[0] > 550:  
 p[-1] = 0  
 else:  
 p[0] -= 1  
 if p[0] <= 0:  
 p[-1] = 1

**Файл Enemy.py**

import random  
import pygame  
  
class Enemy:  
 def \_\_init\_\_(self):  
 check = random.randint(0,7)  
 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 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 lowPlatform.py**

import pygame  
from pygame.locals import \*  
  
  
class DoodleJump:  
 def \_\_init\_\_(self):  
 self.visible = True  
 self.playerRight = pygame.image.load(**"assets/right.png"**).convert\_alpha() # player  
 self.playerRight\_1 = pygame.image.load(**"assets/right\_1.png"**).convert\_alpha() # player  
 self.playerLeft = pygame.image.load(**"assets/left.png"**).convert\_alpha() # player  
 self.playerLeft\_1 = pygame.image.load(**"assets/left\_1.png"**).convert\_alpha() # player  
 self.playerDeadImg = pygame.image.load(**"assets/deadDoodle.png"**).convert\_alpha()  
 self.playerDead = False  
 self.direction = 0 # player  
 self.playerx = 400 # player  
 self.playery = 400 # player  
 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))

**settings.py**

import pygame  
  
screen = pygame.display.set\_mode((800, 800))  
pygame.display.set\_caption(**'doodle jump'**)  
score = 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)  
  
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))  
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))