

**Колледж космического машиностроения и технологий**

**КУРСОВОЙ ПРОЕКТ**

**По МДК.01.02 «Прикладное программирование»**

**Тема: «Разработка приложения "Игра платформер"»**

Выполнила студентка

Группы П1-17

Ковалева Р. С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Проверил преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

Королёв 2020 г.

Содержание

[ВВЕДЕНИЕ 3](#_Toc43894624)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc43894625)

[1.1. Изучение предметной области 4](#_Toc43894626)

[1.2. Изучение существующих программ 5](#_Toc43894627)

[2. ПРОЕКТНАЯ ЧАСТЬ  
2.1. Диаграмма прецедентов 8  
2.2. Диаграмма классов 9](#_Toc43894630)

[2.3. Выбор языка программирования 10](#_Toc43894631)  
[2.4. Выбор библиотеки 11](#_Toc43894631)  
2.5. Функция проверки столкновения объектов 13  
2.6. Возможные ошибки и способы их разрешения 14  
2.7. Проектирование сценария 15  
2.8. Проектирование сценария 16

[2.9. Создание проекта 18](#_Toc43894632)

[3. ЭКСПЛУТАЦИОННАЯ ЧАСТЬ 20](#_Toc43894636)

[3.1. Условия выполнения программы 20](#_Toc43894637)

[3.2. Выполнение программы 20](#_Toc43894638)

[3.3. Возможные улучшения 21](#_Toc43894639)

[ЗАКЛЮЧЕНИЕ 22](#_Toc43894643)

[Список использованных источников. 23](#_Toc43894644)  
Приложение 24

# ВВЕДЕНИЕ

В качестве задачи курсового проекта выступает создание игры жанра «платформер». В теоретической части будет описан принцип работы программы. В проектной части описано само создание.

Для создания программы нам потребовалось:

* Изучить игры подобного жанра
* Понять принцип работы подобных игр
* Разработать функцию для «столкновения» объектов
* Написать код

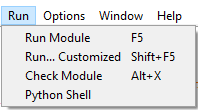
Для разработки данной программы был выбран язык Python. Для выбранной мной игры этот язык имеет ряд плюсов, а именно:

* **Простота**. Его часто советуют в качестве первого «базового» языка, так как он очень прост в изучении и исполнении. В процессе написания программы не требуется использование фигурных скобок, как в других языках, что позволяет не отвлекаться на переключение между клавишами уделять больше внимания разработке программы.
* **Обширность применения**. Питон используется практически повсеместно: для создания сайтов, игр, разработки программных обеспечений, 3D-моделирования и даже для обработки фото.
* **Поддержка**. Python поддерживается разработчиками и сообществом программистов (простыми пользователями), поэтому при возникновении какой-либо проблемы всегда можно узнать способы её решения напрямую.
* **Обширные библиотеки**. Python имеет стандартные библиотеки, в которых можно работать с электронными ресурсами, базами данных, протоколами Интернета и прочими инструментами.
* **Подходит для большинства типов современных операционных систем**. Скрипты, написанные на Питоне, подходят для iOS, Android, Windows и других типов ОС. Это позволяет применять язык программирования в самых разных областях.

**1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

**1.1. Изучение предметной области**

Данная программа разрабатывалась с целью узнать мир разработки игр изнутри. Так же она может иметь практическое применение, а именно – проверка и отработка собственной реакции и точности. Имея исходный код программы, пользователь сможет сам редактировать скорость движения объектов и исходя из этого получать разный результат работы программы.

Для запуска программы достаточно двойного клика по иконке приложения, так же можно запустить игру «внутри» интерпретатора, нажав кнопку F5 или «Run». 

Принцип работы заключается в управлении объектом «летающий корабль» с целью стелять в другой движущийся объект. Корабль управляется с помощью специально назначенных клавиш. В моем случае для управления объектом служат клавиши: W – для движения вверх, A – для движения влево, S – для движения вниз, D – для движения вправо. При нажатии клавиши пробел корабль будет «стрелять» - выпускать новый объект, который будет лететь в сторону цели, в которую нужно попасть. Сама цель – «пришелец» будет двигаться от левого края карты до правого с заданной скоростью, но как писалось выше пользователь сможет сам ее изменить при желании. Так же и все назначенные клавиши могут заменяться.

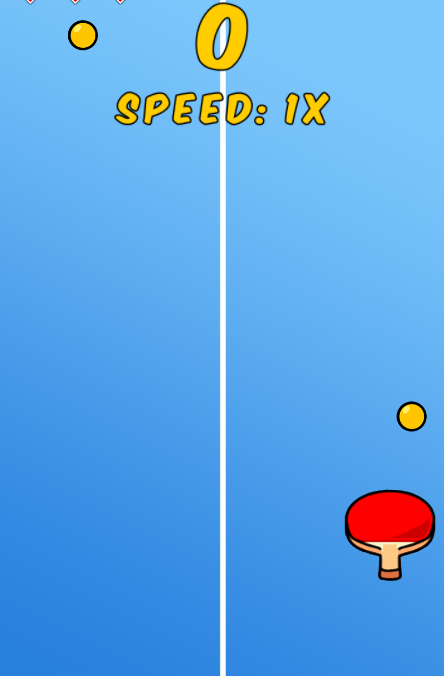
**1.2. Изучение существующих программ**

На данный момент в интернете находится куча подобных приложений. Игры жанра «платформер» являются весьма популярными в наши дни и имеют ряд плюсов. Из них можно выделить такие как:

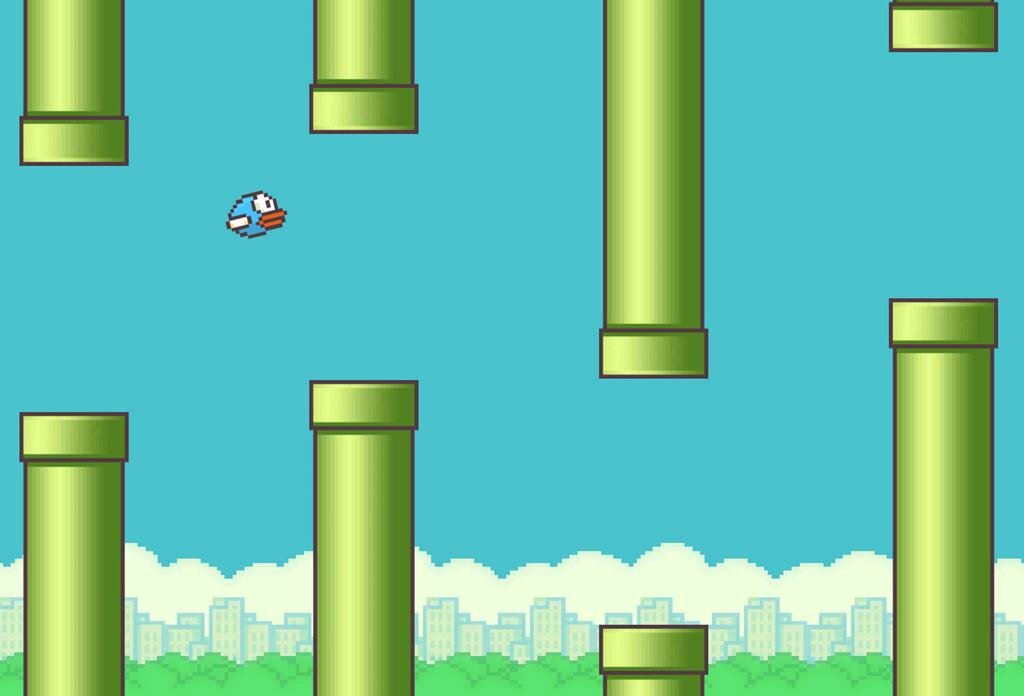
* **Сложность игры.** Платформеры бывают очень разными от совсем «детских» сложностей до безумно сложных. Это может привлечь самую разную аудиторию.
* **Доступность.** Множество игр можно как найти в открытом доступе в интернете без скачивания.
* **Простота использования.** Тут мало что можно добавить – простое и понятное всем управление не могло не подкупить пользователей.
* **Заработок.** Если ориентироваться на создание мобильных приложений, то они легко могут принести доход на рекламе или на покупке пользователем внутри игровой валюты.
* **Создание.** Игры этого жанра очевидно более простые в создании, чем других жанров. Поддерживать игру обновлениями не всегда будет нужно.

Вот некоторые подобные игры, которые можно предоставить в качестве примера:

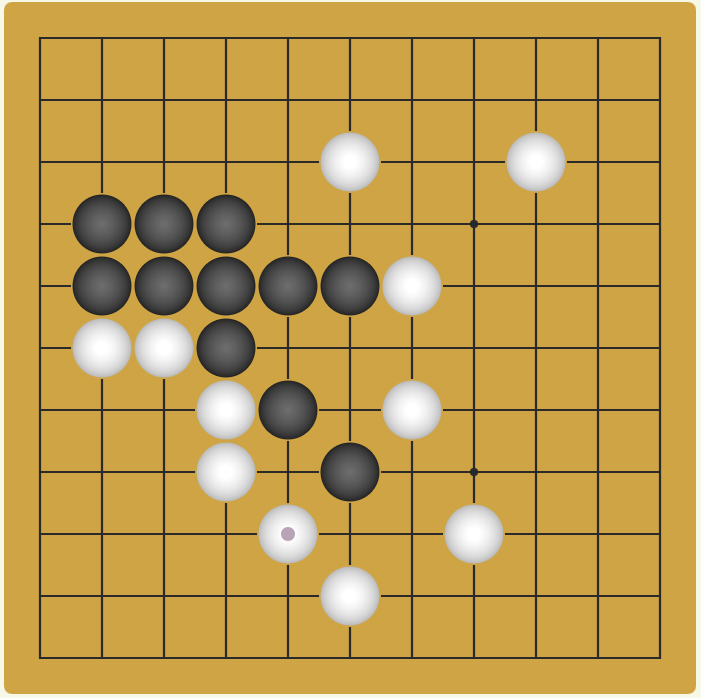
1. **Пинг-понг.** Весьма простая игра, где пользователю нужно отбивать летящие к нему шарики ракеткой. В данной игре управление идет с помощью мышки, но двигать ракетку вверх и вниз – нельзя. Каждый уровень имеет определенное количество шариков, которые нужно отбить и после его прохождения вам откроется следующий уровень. Он будет иметь большее количество шариков и более быструю скорость полета шариков.



2. **Flappy bird.** Игра для мобильных устройств, в которой игрок с помощью касаний экрана должен контролировать полёт птицы между рядами зелёных труб, не задевая их. Была реализована на платформах iOS и Android. Но так же по стечению времени появились ее аналоги для ПК.



3. **Го.** Или как ещё называют эту игру – китайские шашки. Логическая настольная игра с глубоким стратегическим содержанием, возникшая в Древнем Китае. По общему числу игроков — одна из самых распространённых настольных игр в мире. Входит в число пяти базовых дисциплин Всемирных интеллектуальных игр. Естественно данная игра не могла избежать появления на ПК и мобильных устройствах в наши дни. Вы сможете играть как против компьютера с регулируемыми уровнями сложности, так и с другими пользователями по сети. При игре с компьютером пользователь должен выбрать размер доски и уровень сложности противника.

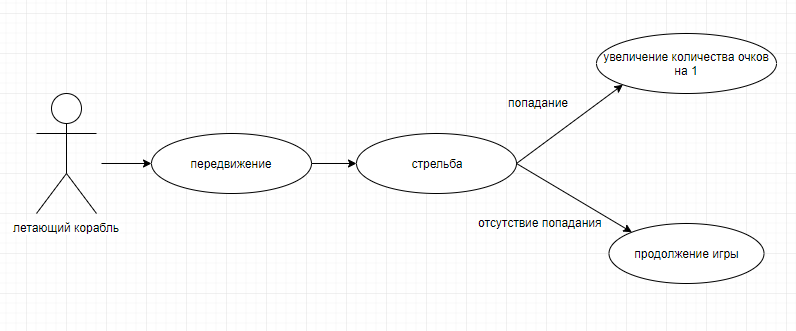


**2. ПРОЕКТНАЯ ЧАСТЬ**

**2.1. Диаграмма прецедентов**

В игре происходит взаимодействие пользователя с управляемым космическим кораблем. Тот в свою очередь может передвигаться и стрелять.

1. Совершается передвижение и стрельба.  
 2. Если выстрел попадает в цель к счету прибавляется 1 очко.  
 3. Если выстрел не попадает, то стрела просто пропадает и очки не прибавляются.



**2.2. Диаграмма классов**

В диаграмме представлены два класса player (игрок) и target(противник).

У класса target структура полостью совпадает со структурой класса player, за исключением условий движения объекта.

|  |  |
| --- | --- |
| Класс player | Класс target |
| Включает в себя:   * Координаты переменной player * Изображение для переменной * Движение объекта | Включает в себя:   * Координаты переменной target * Изображение для переменной * Условия движения объекта |

**2.3. Выбор языка программирования**

Для создания приложения мы использовали язык Python. Здесь нам очень помог один из плюсов языка – поддержка множества разных библиотек. В моем проекте использовалась библиотека pygame. Pygame – это библиотека модулей для языка Python, созданная для разработки 2D игр. Также Pygame могут называть фреймворком. В программировании понятия "библиотека" и "фреймворк" несколько разные. Но когда дело касается классификации конкретного инструмента, не все так однозначно.

В любом случае, фреймворк является более мощным по-сравнению с библиотекой, он накладывает свою специфику на особенности программирования и сферу использования продукта. С точки зрения специфики Pygame – это фреймворк. Однако его сложно назвать "мощным инструментом". По своему объему и функционалу это скорее библиотека. Pygame можно сравнить с Tkinter, который через свои функции и классы предоставляет Питону доступ к графической библиотеке Tk.

**2.4. Выбор библиотеки**

Мой основной выбор был между двумя библиотеками, а именно Tkinter и pygame. Рассмотрим обе из них и узнаем какая более подходящая.

[Tkinter Python](https://docs.python.org/3/library/tkinter.html) предлагает простой и быстрый способ создания приложений с графическим интерфейсом. Tkinter - это стандартная библиотека GUI для языка программирования Python. Он предлагает мощный объектно-ориентированный интерфейс для инструментария Tk GUI.

Tkinter предлагает более 15 типов виджетов, включая кнопки, метки и текстовые поля. Каждый из них имеет доступ к некоторым конкретным методам управления геометрией, которые служат для организации виджетов по всей области родительского виджета.

Особенности:

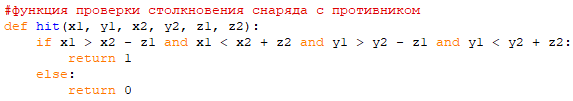
* Поставляется с набором виджетов, которые поддерживают методы управления геометрией
* Облегчает разработку приложений с графическим интерфейсом
* Поддерживает эффективный объектно-ориентированный интерфейс

[Pygame](https://www.pygame.org/news) - это бесплатная библиотека Python с открытым исходным кодом, предназначенная для разработки мультимедийных приложений на Python, особенно для двумерных игровых проектов. Следовательно, он широко используется как начинающими, так и профессиональными разработчиками игр на Python.

* Можно портировать приложения, на платформе Android (смартфоны и планшеты). Для этой цели необходимо использовать расширение [pgs4a](https://www.pygame.org/tags/pgs4a) (подмножество Pygame для Android).
* Особенности:
* Не требует OpenGL
* Облегчает использование многоядерных процессоров
* Для использования всех доступных функций графический интерфейс не требуется
* Обеспечивает поддержку широкого спектра платформ и операционных систем.
* Простой и удобный в использовании
* Использует ассемблерный код и оптимизированный C-код для реализации основных функций

И все же основный задачи у двух этих библиотек немного разные. Tkinter предназначена для разработки графического интерфейса, а Pygame непосредственно для создания 2D игр. Исходя их этих данных и своих личных предпочтений мой выбор пал именно на библиотеку pygame.

**2.5. Функция проверки столкновения объекта**

Рассмотрим уже готовую функцию hit, которая выглядит так:  


Где return 1 вернет истину при столкновении целей, а return 0 соответственно нет.

Возьмем произвольные значения для переменных x1, x2, y1, y2, z1, z2 для того чтобы проверить условия.

|  |  |
| --- | --- |
| X1 | 5 |
| X2 | 3 |
| Y1 | 7 |
| Y2 | 7 |
| Z1 | 2 |
| Z2 | 1 |

Проверим значения для текущей формулы:

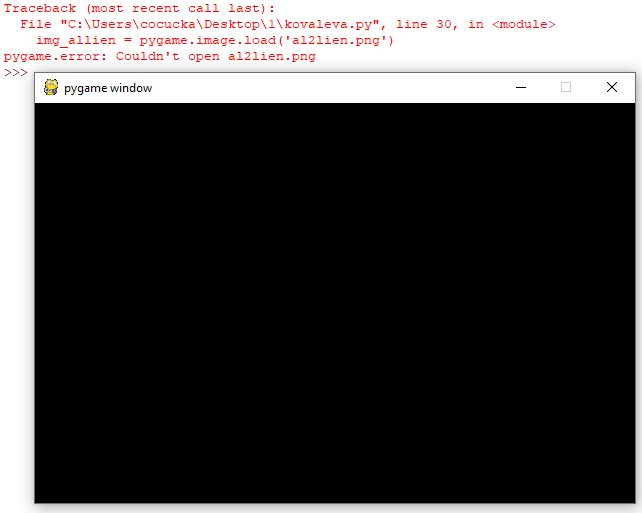
5 > 3 – 1 > 2 – верно

5 < 3 + 1 < 4 – верно   
7 > 7 – 2 > 5 – верно   
7 < 7 + 1 – верно

Все условия сошлись, а значит функция вернет единицу, что обозначает столкновение двух объектов.

**2.6. Возможные ошибки и способы их разрешения**

1. Игра не запускается из-за отсутствия изображений.

Решение: Поместить изображения в одну директорию с файлом игры.  
2. Нельзя открыть файл.  


Решение: Проверить корректность имен всех изображений.

3. Летающий корабль не двигается.  
Решение: Проверка работоспособности клавиатуры и верной привязки клавиш.

**2.7. Проектирование сценария**

Работа сценария строится следующим образом:

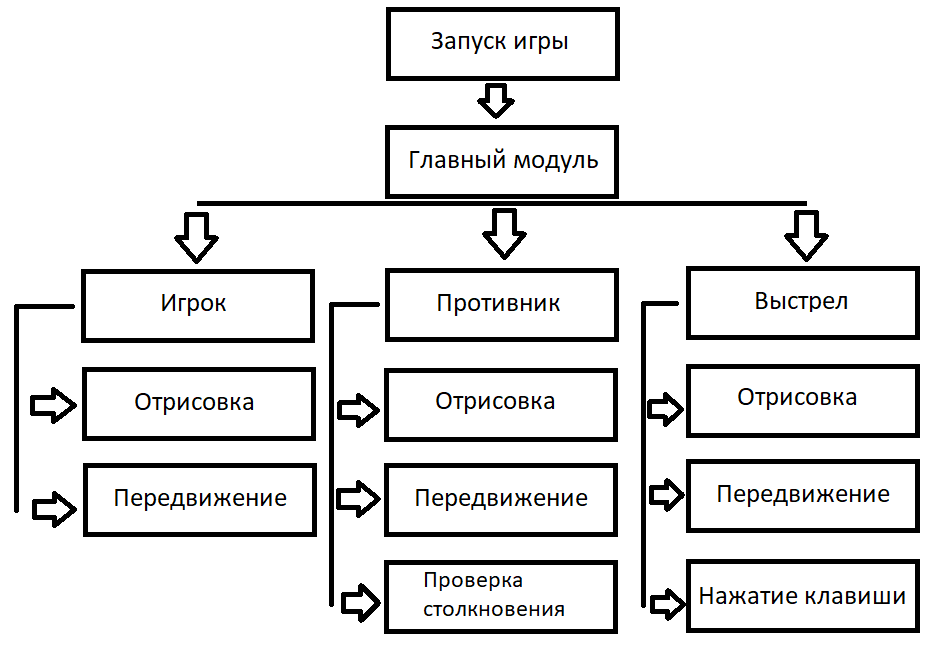
1. Происходит соединение с программой (запуск).

2. Дальше мы попадаем в главный модуль, от куда уже осуществляется взаимодействие с персонажами игры.

3. При работе с персонажем “Игрок” происходит его отрисовка, механика его передвижения.

4. При работе с персонажем “Противник” происходит все тоже самое, что и с персонажем “Игрок”, но с учетом проверки механики столкновения.

5. Работая с объектом “Выстрел” происходит появление его на карте при соблюдении определенных условий (нажатия клавиши пробел).



**2.8. Описание главного модуля**

Главный модуль включает в себя несколько частей:

1. Создание текста и его вывод;
2. Функция проверки столкновения объектов;
3. Подсчет очков;
4. Движение объектов;

1. Создание текста и его вывод.

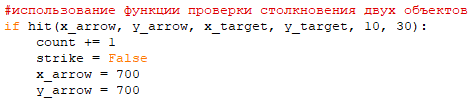




2. Функция проверки столкновения объектов.



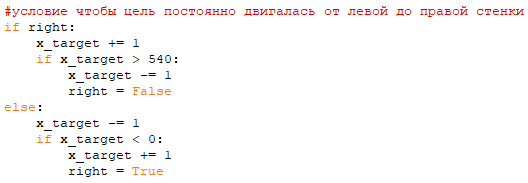
3. Подсчет очков.



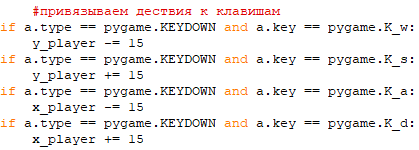
В качестве счетчика выступает переменная count. К ней прибавляет одно очко, когда стрела сталкивается с пришельцем.

4. Движение объектов.

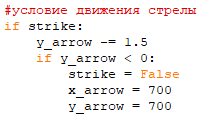
Постоянное движение прищельца:



Движение корабля регулируется самим пользователем с помощью клавиатуры:



Движение выстрела:



**2.9. Создание проекта**

Для начала нужно было определиться с принципом работы приложения – сделать «черновик». После того как я решила, что буду делать нужно было начинать работу с самим кодом. Для начала в новом файле мы должны подключить саму библиотеку – «import pygame». Затем написать ключевые слова pygame.init() и pygame.quit() для запуска и выхода из игры. После идет создание окна и работа с ним. Нужно задать его размер, а позже заняться кастомизацией.

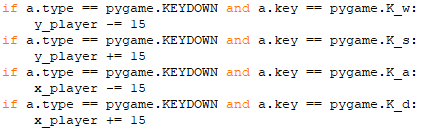




Далее мы должны заняться самим управляемым объектом – задать его размеры, координаты появления. Так же нужно связать объект с клавишами для его движения. И после точно так же – кастомизировать.

Создаем переменную с объектом 

Задаем клавиши для его движения



Можно назначить любые клавиши, в моем случае они стандартные – W A S D.

Число 15 – выступает в качестве шага объекта, его так же можно изменить.

В строках ниже как раз мы связываем нужное изображение с объектом.



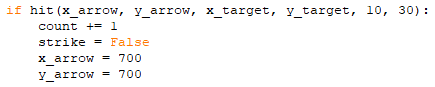
С движущимся объектом для стрельбы и самим «снарядом» мы должны проделать подобные действия.

Так же для ведения счета игры нужно добавить соответствующую надпись и сам подсчет попаданий в цель.



Непосредственно здесь идет сама надпись. В верхней строке задается ее шрифт и размер. А в нижней мы выводим сам текст на экран с функцией счета.

Здесь показано использовании функции hit для подсчета попаданий.



После того как стрела попала в цель мы перемещаем её вне зоны видимости карты.

Заканчиваем весь проект – кастомизацией. Прикрепляем к объектам картинки формата .png и изображение фона. С помощью команды set\_colorkey указываем какой цвет из изображения надо убрать. В моем случае это черный т.к. в png изображении «невидимая» часть отображается именно черным цветом.



Нули здесь – цифры из RGB палитры и три нуля здесь обозначение черного цвета.

**3. ЭКСПЛУАТАЦИОННАЯ ЧАСТЬ.**

**3.1. Условия выполнения программы**

Наличие программы IDLE (Python 3.8 32-bit or 64-bit). Важно иметь именно версию 3.8 т.к. библиотека pygame поддерживается только там.

Операционная система: Windows OS, MAC OS.

Процессор (CPU): Любой.

Оперативная память (RAM): 512 МБ.

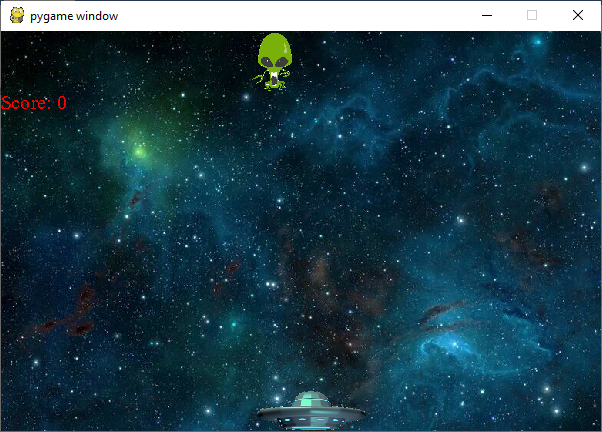
Видеоадаптер: Любой.

**3.2. Выполнение программы**

Программу можно запустить двойным нажатием по файлу или запустить внутри самой программы. Мы получим такое сообщение при удачном запуске.



После чего откроется само окно с игрой.



Для выхода из приложения нужно нажать на крестик.

Примечание: для успешного запуска программы изображения должны находиться в одной директории с файлом программы.

**3.3. Возможные улучшения**

1. Добавление нескольких целей для стрельбы.

2. Добавление другого режима стрельбы.

3. Создание системы уровней.

4. Создание «жизней», которые будут убавляться при промахе по цели.

**ЗАКЛЮЧЕНИЕ**

По окончанию написания курсового проекта было создана компьютерная игра жанра платформер на языке Python. Для разработки я ознакомилась с подобными приложениями и библиотекой для написания 2D игр – pygame.

**Список использованных источников**

1. Pythonru. Библиотека Pygame / Часть 1. Введение. <https://pythonru.com/uroki/biblioteka-pygame-chast-1-vvedenie>

2. PyGame — шпаргалка для использования. <https://waksoft.susu.ru/2019/04/24/pygame-shpargalka-dlja-ispolzovanija/>

3. PyGame: учебник по программированию игр на Python. <https://cyberguru.tech/программирование/pygame-учебник-по-программированию-игр-на-python>

**ЛИСТИНГ**

#импорт библиотеки pygame

import pygame

#функция проверки столкновения снаряда с противником

def hit(x1, y1, x2, y2, z1, z2):

if x1 > x2 - z1 and x1 < x2 + z2 and y1 > y2 - z1 and y1 < y2 + z2:

return 1

else:

return 0

pygame.init()

#создаем окно для игры

window = pygame. display.set\_mode((600, 400))

#создаем экран такого же размера

screen = pygame.Surface((600, 400))

#создаем модель персонажа

player = pygame.Surface((110, 40))

#создаем модель снаряда

target = pygame.Surface((40, 60))

#создаем модель противника

arrow = pygame.Surface((10, 30))

count = 0

#связываем необходимые фото с переменными

img\_allien = pygame.image.load('allien.png')

img\_ship = pygame.image.load('ship.png')

img\_space = pygame.image.load('space.jpg')

img\_hit = pygame.image.load('hit.png')

#создаем координаты для игрока, цели и стрелы

x\_player = 255

y\_player = 360

x\_target = 0

y\_target = 0

x\_arrow = 700

y\_arrow = 700

#создаем шрифт и выбираем его размер

font = pygame.font.SysFont('serif', 20)

#убираем черный цвет из изображений

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

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

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

done = False

right = True

strike = False

while done == False:

for a in pygame.event.get():

if a.type == pygame.QUIT:

done == True

#привязываем дествия к клавишам

if a.type == pygame.KEYDOWN and a.key == pygame.K\_w:

y\_player -= 15

if a.type == pygame.KEYDOWN and a.key == pygame.K\_s:

y\_player += 15

if a.type == pygame.KEYDOWN and a.key == pygame.K\_a:

x\_player -= 15

if a.type == pygame.KEYDOWN and a.key == pygame.K\_d:

x\_player += 15

if a.type == pygame.KEYDOWN and a.key == pygame.K\_SPACE:

if strike == False:

strike = True

x\_arrow = x\_player + 55

y\_arrow = y\_player - 30

#условие чтобы цель постоянно двигалась от левой до правой стенки

if right:

x\_target += 1

if x\_target > 540:

x\_target -= 1

right = False

else:

x\_target -= 1

if x\_target < 0:

x\_target += 1

right = True

#условие движения стрелы

if strike:

y\_arrow -= 1.5

if y\_arrow < 0:

strike = False

x\_arrow = 700

y\_arrow = 700

#использование функции проверки столкновения двух объектов

if hit(x\_arrow, y\_arrow, x\_target, y\_target, 10, 30):

count += 1

strike = False

x\_arrow = 700

y\_arrow = 700

#выводим надпись и подсчет очков

string = font.render('Score: '+str(count), 0, (250, 0, 0))

#связываем объекты с изображениями

target.blit(img\_allien, (0, 0))

player.blit(img\_ship, (0, 0))

arrow.blit(img\_hit, (0, 0))

screen.blit(img\_space, (0, 0))

#связываем объекты с координатами

screen.blit(player, (x\_player, y\_player))

screen.blit(target, (x\_target, y\_target))

screen.blit(arrow, (x\_arrow, y\_arrow))

screen.blit(string, (0, 60))

window.blit(screen, (0, 0))

pygame.display.update()

pygame.quit()