МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения вычислительной техники

и автоматизированных систем

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: «Языки программирования»

на тему: «***Разработка игры «Dino» на языке Python»***

Выполнил**:** И.А. Воробей

студент группы 10701118

Руководитель**:** И.О. Лапанович

Старший преподаватель

Кафедры ПОВТ и АС

Минск 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)

Кафедра программного обеспечения вычислительной техники

и автоматизированных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

по дисциплине **«Языки программирования»**

на тему **«*Разработка компьютерной мини-игры «Dino»***

Выполнил:

студент группы 10701118 И.А. Воробей

Руководитель:

Старший преподаватель

Кафедры ПОВТ и АС И.О.Лапанович

Минск 2019

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc27604181)

[1. РАЗРАБОТКА ИГРОВОЙ СТРАТЕГИИ ДЛЯ ИГРЫ 6](#_Toc27604182)

[«Dino» 6](#_Toc27604183)

[2. РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ДЛЯ ИГРОВОГО ПРИЛОЖЕНИЯ 7](#_Toc27604184)

[2.1 Назначение и дизайн 7](#_Toc27604185)

[2.2 Структура игрового приложения 8](#_Toc27604186)

[2.3 Описание игрового приложения 8](#_Toc27604187)

[2.4 Тестирование игрового приложения 12](#_Toc27604188)

[2.5 Руководство пользователя 16](#_Toc27604189)

[ЗАКЛЮЧЕНИЕ 17](#_Toc27604190)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ 19](#_Toc27604191)

[ПРИЛОЖЕНИЕ А 20](#_Toc27604192)

# ВВЕДЕНИЕ

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

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, найдётся подходящий фреймфорк.

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](http://en.wikipedia.org/wiki/Guido_van_Rossum), передал его в некоммерческую организацию Python Software Foundation, которая является администратором этого 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.

Благодаря его гибкости он один из самых широко известных высокоуровневых языков программирования на сегодняшний день.

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, дружелюбный 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](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) (библиотек) языка программирования [Python](https://ru.wikipedia.org/wiki/Python), предназначенный для написания [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0)и [мультимедиа](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D0%BB%D1%8C%D1%82%D0%B8%D0%BC%D0%B5%D0%B4%D0%B8%D0%B0)-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. Кроме того, одна игра, которая была сделана с этим движком, была коммерческой игрой Disney, поэтому есть большие игры, написанные частично с Python.

**Цель курсового проекта:** разработка игры «*Dino*» на Python с использованием модуля PyGame.

**Задачи курсового проекта:**

1. Изучить модуль PyGame.
2. Применить полученные знания на практике при создании курсовой работы.
3. Разработать пользовательский графический интерфейс для игрового приложения.
4. Разработать игровую стратегию.
5. Продумать алгоритмы решения поставленной задачи и программно реализовать их.
6. Протестировать игровое приложение.

# РАЗРАБОТКА ИГРОВОЙ СТРАТЕГИИ ДЛЯ ИГРЫ

# «Dino»

Главная задача игры – перепрыгивать кактусы и убивать птиц.

В этой игре игрок является динозавром, который перепрыгивает кактусы и защищается от птиц.

Существует 3 вида кактусов: низкий продолговатый кактус и два вида высоких. При попадании игроком на кактус активируется характерный звук столкновения и игрок теряет одну из жизней. В случае успешного перепрыгивания кактуса, счет увеличивается на 1.

На игровом поле произвольно генерируются камни и облака, что придает игре реалистичность.

Игрок имеет ограниченное количество жизней (в игре сердечек) – 2. Максимально игрок может собрать только 3 сердечка, чтобы пополнить количество жизней до 5.

О врагах(«Птицах») в данной игре. В игре существуют враги, которые генерируются в произвольном порядке на игровом поле. Враги стреляют в игрока пулями. Если вражеская пуля попадает в игрока, тогда игрок теряет все жизни. Так же врагов можно убивать. Для этого игрок должен стрелять в них пулями.

Также есть уровни сложности. С ростом счета так же возрастает скорость игры. Игрок начинает бежать быстрее и враги начинают появляться чаще.

Игра снабжена веселым музыкальным сопровождением, которое позволит погрузиться в атмосферу приложения.

Данный продукт будет интересен детям и взрослым для веселого времени провождения.

# РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ГРАФИЧЕСКОГО ИНТЕРФЕЙСА ДЛЯ ИГРОВОГО ПРИЛОЖЕНИЯ

Приложение разрабатывалось в среде IDLE. IDLE − 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). Подойдет для новичков и профессионалов. С помощью этой среды можно выполнять обычные для интегрированной среды задачи: просматривать, редактировать, запускать 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). На рисунке 1 продемонстрировано меню игры. Кнопка “ORIGINATOR” открывает информацию о разработчике, кнопка “INSTRUCTION” показывает инструкцию, кнопка “START GAME” запускает игру, кнопка “QUIT” закрывает игру.



Рисунок 1- Запуск игры.

# Назначение и дизайн

Игра предназначена для пользователей, которые любят расслабиться и «бить» свои рекорды!

В эту игру может играть каждый, будь то ребенок, взрослый или же вовсе пожилой человек, однако ограничение в возрасте 3+. Суть этого ограничения в том, что младенцы обычно не играют в игры.

Дизайн выполнен в стиле пустыни: желтый песок, яркое солнце, птицы и кактусы. В главном меню кнопки расположены на фоне приятного пустынного пейзажа. Игрок может перейти в меню “INSTRUCTION” и увидеть инструкцию по игре на фоне голубого неба.

Необходимо отметить, что приложение является оконным среднего размера, в результате чего, негативное воздействие на зрение пользователей будет снижено.

# Структура игрового приложения

Для реализации данного курсового проекта использовались такие программные модули, как PyGame.

Были созданы следующие модули: globalVars,bullet,button,items,bird,lab4,utils, а в них в свою очередь классы: Object, Bullet,Bird,Button.

Модуль globalVars содержит глобальные переменные, используемые в программе.

Модуль items содержит загрузки картинок, звуков и шрифта.

Модуль button содержит реализацию класса Button, который отвечает за пули игрока и врагов.

Модуль bird содержит класс Bird, отвечающий за свойства и поведение врагов.

Модуль lab4 содержит реализацию игрового цикла и класс Object.

Модуль utils содержит вспомогательные методы.

# Описание игрового приложения

При запуске приложения появляется меню, с помощью которого пользователь выбирает посмотреть инструкцию, узнать о разработчике, выйти или начать игру рисунок 2.



Рисунок 2- Меню.

После нажатия на “START GAME” начинается игра рисунок 4.



Рисунок 4 - Начало игры

Управление игроком осуществляется через прыжок нажатием клавиши “SPACE” и стрельба через ЛКМ. Перемещение игрока показано на рисунке 5. Меню проигрыша показано на рисунке 6.



Рисунок 5 - Перемещение игрока.

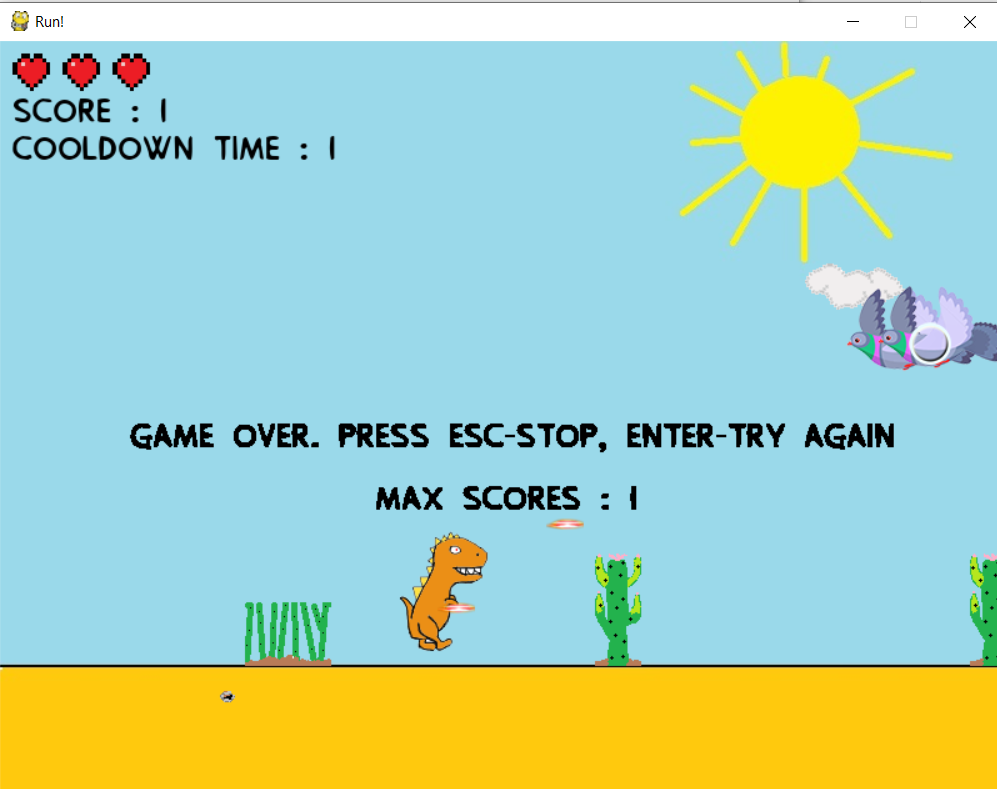


Рисунок 6 - Проигрыш из-за попадания пули и меню проигрыша.

# Тестирование игрового приложения

Приложение было протестировано 28-ю пользователями. В результате чего, общая оценка приложения по их отзывам составила 95 % (из 100% возможных).

Некорректной и нестабильной работы приложения в ходе тестирования не выявлено.

Ниже представлены скриншоты результатов тестирования приложения. Рисунок 7 демонстрирует старт игры с главным меню. Рисунок 8 демонстрирует начало игрового процесса. Рисунок 9 показывает перемещение игрока в игровом мире. На рисунке 10 представлено меню паузы. На рисунке 10 представлена инструкция.



Рисунок 7 - Запуск игры

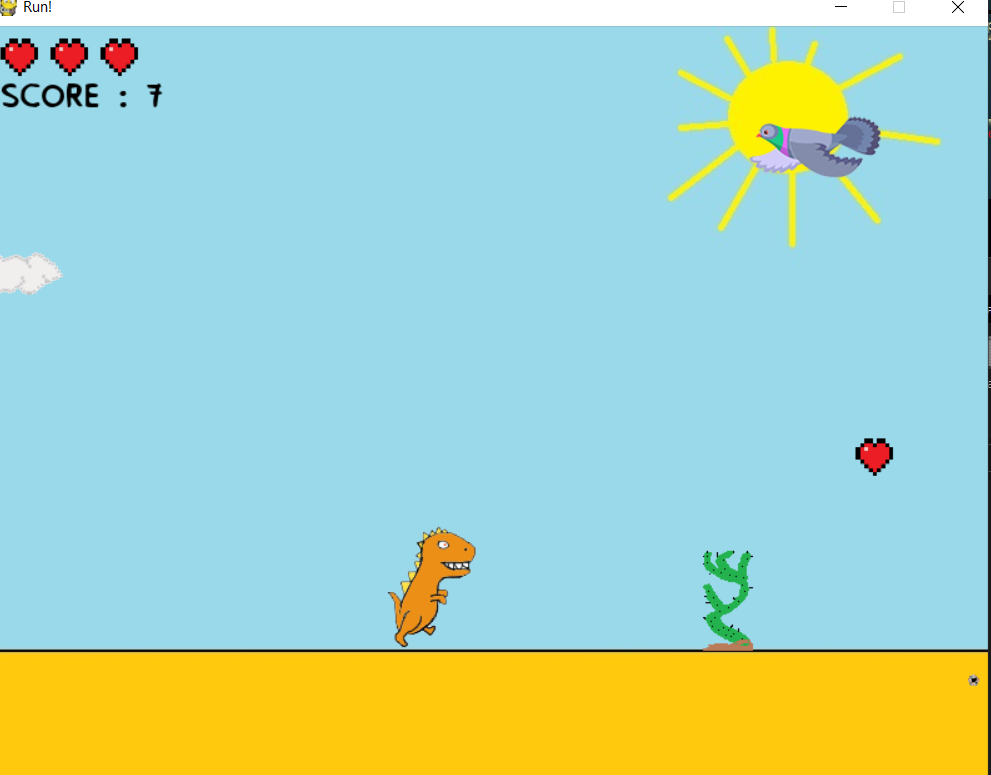


Рисунок 8 - Тестирование начала игры



Рисунок 9 - Тестирование перемещения персонажа

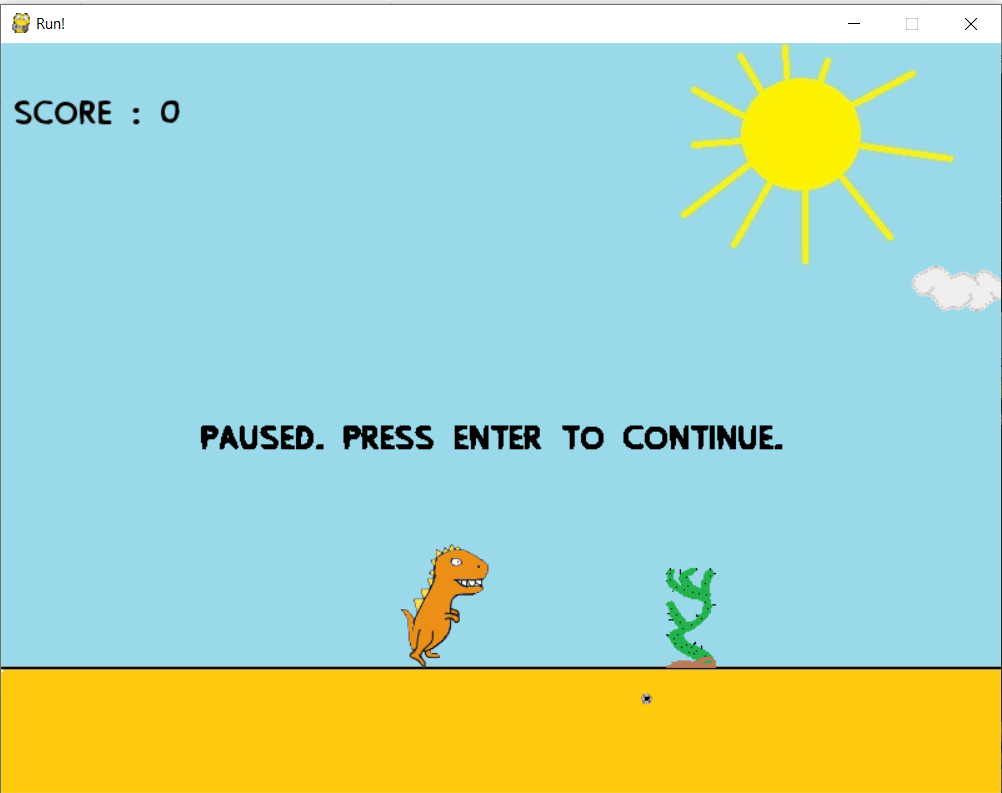


Рисунок 10 - Тестирование паузы

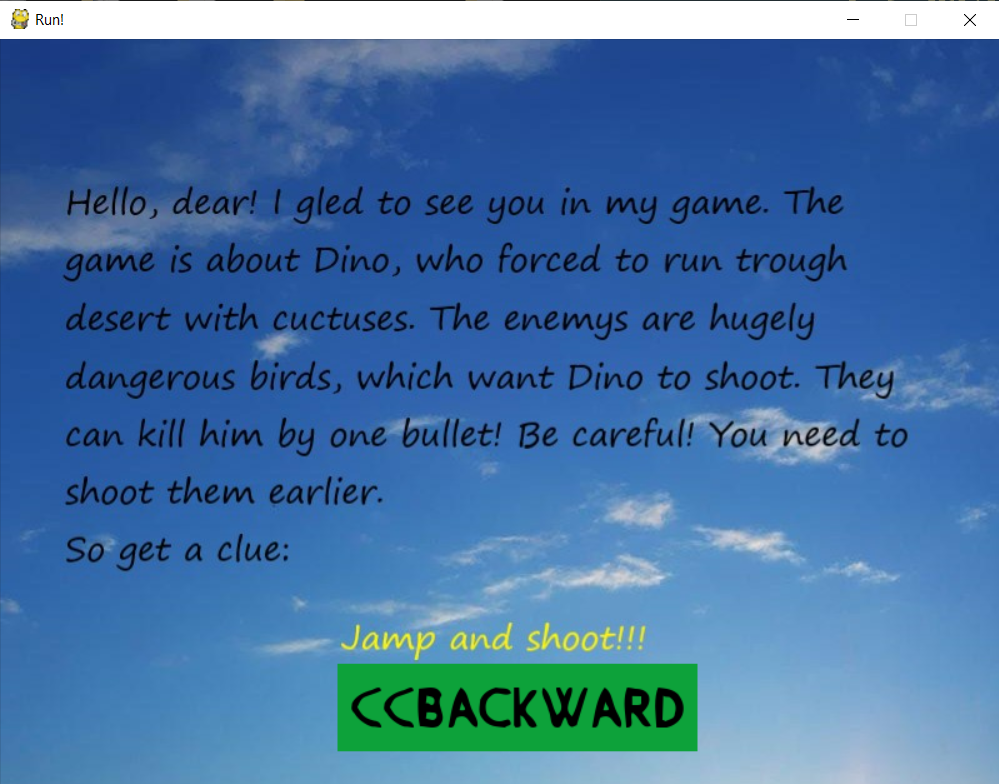


Рисунок 11 - Тестирование просмотра инструкции

# Руководство пользователя

После запуска игры пользователь в главном меню выбирает меню инструкции. Далее он возвращается в главное меню и выбирает кнопу “START GAME”. Игра началась, теперь пользователь должен перепрыгивать кактусы и собирать жизни. А также уворачиваться от пуль врагов и стрелять во врагов, не допуская попадания пуль врагов в себя. С ростом счета, ускоряется игра, соответственно пользователь должен стараться реагировать быстрее. Пользователь может поставить игру на паузу, нажав “ESC”, но при этом расположение врагов и жизней будет скрыто. После проигрыша пользователь может увидеть свой максимальный рекорд за эту игру. В меню проигрыша предоставляется выбор: выйти в главное меню либо начать заново. При выходе в главное меню и нажатии “QUIT” игра закрывается.

# 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 изучили новые методы работы, связанные с разработкой окон, загрузкой картинок, шрифтов, звуков. Также научились прорисовывать объекты, создаватьигровые окна, подробно рассмотрели способы работы с мышью и клавиатурой, усвоили навыки работы с фронтэндом и бэкэндом. Научились делать легко масштабируемый код.

Успешно протестировали полученный продукт.

Игра получилась довольно просто и интересной. Был приобретен бесценный навык программирования игр и приложений.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ

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 А

**lab4.py**

import pygame

import random

from globalVars import \*

from utils import \*

from button import Button

from bullet import Bullet

from bird import Bird

pygame.init()

class Object:

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

self.x=x

self.y=y

self.width=width

self.image=image

self.speed=speed

def move(self):

if self.x >= -self.width:

display.blit(self.image,(self.x,self.y))

self.x-=self.speed

return True

else:

return False

def return\_self(self,radius,y,width,image):

self.x=radius

self.y=y

self.width=width

self.image=image

display.blit(self.image,(self.x,self.y))

def show\_menu():

menu\_bckgr=pygame.image.load('Menu.jpg')

start\_btn=Button(288,70)

quit\_btn=Button(120,70)

ruls\_btn=Button(288,70)

dvlpr\_btn=Button(288,70)

show=True

while show:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

display.blit(menu\_bckgr,(0,0))

dvlpr\_btn.draw(270,100, 'Originator',show\_orgntr,50)

start\_btn.draw(270,300, 'Start game',start\_game,50)

quit\_btn.draw(358,400, 'quit',quit,50)

start\_btn.draw(270,200, 'Instruction',show\_ruls,50)

draw\_mouse()

pygame.display.update()

clock.tick(30)

def start\_game():

global scores,make\_jump,jump\_counter,usr\_y,health,cooldown

while game\_cycle():

scores=0

make\_jump=False

jump\_counter=30

usr\_y=display\_height-usr\_height-100

health=2

cooldown=0

def show\_ruls():

rul\_bckgr=pygame.image.load('rul\_img.jpg')

rulsReturn\_btn=Button(288,70)

stop=1

while stop:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

display.blit(rul\_bckgr,(0,0))

rulsReturn\_btn.draw(270,500, '<<Backward',show\_menu,50)

draw\_mouse()

pygame.display.update()

clock.tick(30)

def show\_orgntr():

rul\_bckgr=pygame.image.load('orgntr\_img.jpg')

rulsReturn\_btn=Button(288,70)

stop=1

while stop:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

display.blit(rul\_bckgr,(0,0))

rulsReturn\_btn.draw(270,500, '<<Backward',show\_menu,50)

draw\_mouse()

pygame.display.update()

clock.tick(30)

def game\_cycle():

global make\_jump,cooldown

pygame.mixer.music.play(-1)

game=True

cactus\_arr=[]

create\_cactus\_arr(cactus\_arr)

land=pygame.image.load(r'Land.jpg')

stone,cloud=open\_random\_objects()

heart= Object(display\_width,330,30,health\_img,4)

all\_btn\_bullets=[]

all\_ms\_bullets=[]

bird1=Bird(-180)

bird2=Bird(-140)

all\_birds=[bird1,bird2]

while game:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

keys=pygame.key.get\_pressed()

mouse=pygame.mouse.get\_pos()

click=pygame.mouse.get\_pressed()

if keys[pygame.K\_SPACE]:

make\_jump=True

count\_scores(cactus\_arr)

display.blit(land,(0,0))

print\_text('Score : '+str(scores),10,40)

draw\_array(cactus\_arr)

move\_objects(stone,cloud)

drow\_dino()

if keys[pygame.K\_ESCAPE]:

pause()

if not cooldown:

if keys[pygame.K\_x]:

pygame.mixer.Sound.play(bullet\_sound)

all\_btn\_bullets.append(Bullet(usr\_x+usr\_width, usr\_y+28))

cooldown=50

elif click[0]:

pygame.mixer.Sound.play(bullet\_sound)

add\_bullet=Bullet(usr\_x+usr\_width, usr\_y+28)

add\_bullet.find\_path(mouse[0],mouse[1])

all\_ms\_bullets.append(add\_bullet)

cooldown=50

else:

print\_text('Cooldown time : ' +str(cooldown//10),10,70)

cooldown-=1

for bullet in all\_btn\_bullets:

if not bullet.move():

all\_btn\_bullets.remove(bullet)

for bullet in all\_ms\_bullets:

if not bullet.move\_to():

all\_ms\_bullets.remove(bullet)

heart.move()

heart\_plus(heart)

if make\_jump:

jump()

if check\_collision(cactus\_arr):

game=False

draw\_birds(all\_birds)

check\_birds\_dmg(all\_ms\_bullets,all\_birds)

#

if check\_dino\_damage(bird1.get\_bullets()) or check\_dino\_damage(bird2.get\_bullets()):

game=False

show\_health()

draw\_mouse()

pygame.display.update()

if scores<5:

clock.tick(30)

elif scores<20:

clock.tick(40)

elif scores<25:

clock.tick(50)

elif scores<30:

clock.tick(60)

elif scores<40:

clock.tick(70)

elif scores<50:

clock.tick(80)

elif scores<60:

clock.tick(90)

elif scores<70:

clock.tick(200)

return game\_over()

def count\_scores(barriers):

global scores,max\_above

above\_cactus=0

if -20<=jump\_counter<20:

for barrier in barriers:

if usr\_y+usr\_height-5<=barrier.y:

if barrier.x<=usr\_x<=barrier.x+barrier.width:

above\_cactus+=1

elif barrier.x<=usr\_x+usr\_width<=barrier.x+barrier.width:

above\_cactus+=1

max\_above=max(max\_above,above\_cactus)

else:

if jump\_counter==-30:

scores+=max\_above

max\_above=0

def object\_return(objects, obj):

radius=find\_radius(objects)

choice=random.randrange(0,3)

img=cactus\_img[choice]

width=cactus\_options[choice\*2]

height=cactus\_options[choice\*2+1]

obj.return\_self(radius,height,width,img)

def check\_collision(barriers):

for barrier in barriers:

if barrier.y==449:

if not make\_jump:

if barrier.x<=usr\_x+usr\_width-20<=barrier.x+barrier.width:

if check\_health():

object\_return(barriers, barrier)

return False

else:

return True

elif jump\_counter>=0:

if usr\_y+usr\_height-5>=barrier.y:

if barrier.x<=usr\_x+usr\_width-70<=barrier.x+barrier.width:

if check\_health():

object\_return(barriers, barrier)

return False

else:

return True

else:

if usr\_y+usr\_height-10>=barrier.y:

if barrier.x<=usr\_x<=barrier.x+barrier.width:

if check\_health():

object\_return(barriers, barrier)

return False

else:

return True

else:

if not make\_jump:

if barrier.x<=usr\_x+usr\_width+0<=barrier.x+barrier.width:

if check\_health():

object\_return(barriers, barrier)

return False

else:

return True

elif jump\_counter==10:

if usr\_y+usr\_height-5>=barrier.y:

if barrier.x<=usr\_x+usr\_width-5<=barrier.x+barrier.width:

if check\_health():

object\_return(barriers, barrier)

return False

else:

return True

elif jump\_counter<=-1:

if usr\_y+usr\_height-3>=barrier.y:

if barrier.x<=usr\_x+34<=barrier.x+barrier.width:

if check\_health():

object\_return(barriers, barrier)

return False

else:

return True

else:

if usr\_y+usr\_height-10>=barrier.y:

if barrier.x<=usr\_x+5<=barrier.x+barrier.width:

if check\_health():

object\_return(barriers, barrier)

return False

else:

return True

return False

def game\_over():

pygame.mixer.music.pause()

global scores,max\_scores

if scores>max\_scores:

max\_scores=scores

stopped=True

while stopped:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

print\_text('Game over. press Esc-stop, enter-try again',104,300)

print\_text('Max scores : '+str(max\_scores),300,350)

keys=pygame.key.get\_pressed()

if keys[pygame.K\_RETURN]:

return True

if keys[pygame.K\_ESCAPE]:

return False

pygame.display.update()

clock.tick(15)

pygame.mixer.music.unpause()

def jump():

global usr\_y,jump\_counter,make\_jump

if jump\_counter>=-30:

if jump\_counter==-25:

pygame.mixer.Sound.play(fall\_sound)

usr\_y-=jump\_counter/4

jump\_counter-=1

else:

jump\_counter=30

make\_jump=False

def create\_cactus\_arr(array):

choice=random.randrange(0,3)

img=cactus\_img[choice]

width=cactus\_options[choice\*2]

height=cactus\_options[choice\*2+1]

array.append(Object(display\_width+20,height,width,img,4))

choice=random.randrange(0,3)

img=cactus\_img[choice]

width=cactus\_options[choice\*2]

height=cactus\_options[choice\*2+1]

array.append(Object(display\_width+300,height,width,img,4))

choice=random.randrange(0,3)

img=cactus\_img[choice]

width=cactus\_options[choice\*2]

height=cactus\_options[choice\*2+1]

array.append(Object(display\_width+600,height,width,img,4))

def draw\_array(array):

for cactus in array:

check=cactus.move()

if not check:

object\_return(array, cactus)

def open\_random\_objects():

choice=random.randrange(0,2)

img\_of\_stone=stone\_img[choice]

choice=random.randrange(0,2)

img\_of\_cloud=cloud\_img[choice]

stone=Object(display\_width,display\_height-80, 10 ,img\_of\_stone,4)

cloud=Object(display\_width,178, 70 ,img\_of\_cloud,1)

return stone,cloud

def move\_objects(stone,cloud):

check=stone.move()

if not check:

choice=random.randrange(0,2)

img\_of\_stone=stone\_img[choice]

stone.return\_self(display\_width,random.randrange(10,80)+500,stone.width,img\_of\_stone)

check=cloud.move()

if not check:

choice=random.randrange(0,2)

img\_of\_cloud=cloud\_img[choice]

cloud.return\_self(display\_width,random.randrange(178,300),cloud.width,img\_of\_cloud)

def drow\_dino():

global img\_counter

if img\_counter==25:

img\_counter=0

display.blit(dino\_img[img\_counter//5], (usr\_x,usr\_y))

img\_counter+=1

def show\_health():

global health

show=0

x=10

while show!= health:

display.blit(health\_img,(x,10))

x+=40

show+=1

def check\_health():

global health

health-=1

if health==0:

pygame.mixer.Sound.play(loss\_sound)

return False

else:

pygame.mixer.Sound.play(fall\_sound)

return True

def heart\_plus(heart):

global health,usr\_x,usr\_y,usr\_width,usr\_height

if heart.x<=-heart.width:

radius=display\_width+random.randrange(500,700)

heart.return\_self(radius,heart.y,heart.width,heart.image)

if usr\_x<=heart.x<=usr\_x+usr\_width:

if usr\_y<= heart.y<=usr\_y+usr\_height:

pygame.mixer.Sound.play(heart\_plus\_sound)

if health <5:

health+=1

radius=display\_width+random.randrange(500,2200)

heart.return\_self(radius,heart.y,heart.width,heart.image)

def check\_dino\_damage(bullets):

global usr\_x,usr\_y,usr\_width,usr\_height,health

for bullet in bullets:

if usr\_x<=bullet.x<=usr\_x+usr\_width-30:

if usr\_y<=bullet.y<=usr\_y+usr\_height:

return True

pass

show\_menu()

pygame.quit()

quit()

**bird.py**

import pygame

import random

from utils import \*

from globalVars import \*

from bullet import Bullet

class Bird:

def \_\_init\_\_(self,away\_y):

self.x=random.randrange(550,730)

self.y=away\_y

self.width=105

self.height=55

self.ay=away\_y

self.speed=3

self.dest\_y=self.speed\*random.randrange(20,70)

self.img\_cnt=0

self.cd\_hide=0

self.come=True

self.go\_away=False

self.cd\_shoot=0

self.all\_bullets=[]

def draw(self):

if self.img\_cnt==30:

self.img\_cnt=0

display.blit(bird\_img[self.img\_cnt//5],(self.x,self.y))

self.img\_cnt+=1

if self.come and self.cd\_hide==0:

return 1

elif self.go\_away:

return 2

elif self.cd\_hide>0:

self.cd\_hide-=1

return 0

def show(self):

if self.y<self.dest\_y:

self.y+=self.speed

else:

self.come=False

self.dest\_y=self.ay

def hide(self):

if self.y>self.dest\_y:

self.y-=self.speed

else:

self.come=True

self.go\_away=False

self.x=random.randrange(550,730)

self.dest\_y=self.speed\*random.randrange(20,70)

self.cd\_hide=250 #what time birds go away

def check\_damage(self,bullet):

if self.x<=bullet.x<=self.x+self.width:

if self.y<=bullet.y<=self.y+self.height:

self.go\_away=True

def shoot(self):

if not self.cd\_shoot:

pygame.mixer.Sound.play(bullet\_sound)

new\_bullet= Bullet(self.x,self.y)

new\_bullet.find\_path(usr\_x+usr\_width//2,usr\_y+usr\_height//2)

self.all\_bullets.append(new\_bullet)

self.cd\_shoot=300 #how often they are shooting

else:

self.cd\_shoot-=1

for bullet in self.all\_bullets:

if not bullet.move\_to(reverse=True):

self.all\_bullets.remove(bullet)

def get\_bullets(self):

return self.all\_bullets

**bullet.py**

import pygame

from globalVars import \*

from utils import \*

pygame.init()

class Bullet:

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

self.x=x

self.y=y

self.speed=8

self.speed\_x=8

self.speed\_y=0

self.dest\_x=0

self.dest\_y=0

def move(self):

self.x+=self.speed

if self.x<= display\_width:

display.blit(bullet\_img,(self.x,self.y))

return True

else:

return False

def find\_path(self,dest\_x,dest\_y):

self.dest\_x=dest\_x

self.dest\_y=dest\_y

delta\_x=dest\_x-self.x

count\_up=delta\_x//self.speed\_x

if self.y>= dest\_y:

delta\_y=self.y-dest\_y

self.speed\_y=delta\_y/count\_up

else:

delta\_y=dest\_y-self.y

self.speed\_y=-(delta\_y/count\_up)

def move\_to(self,reverse=False):

if not reverse:

self.x+=self.speed\_x

self.y-=self.speed\_y

else:

self.x-=self.speed\_x

self.y+=self.speed\_y

if self.x<= display\_width and not reverse:

display.blit(bullet\_img,(self.x,self.y))

return True

elif self.x>=0 and reverse:

display.blit(bullet\_img,(self.x,self.y))

return True

else:

return False

**button.py**

import pygame

from globalVars import \*

from utils import \*

pygame.init()

class Button:

def \_\_init\_\_(self,width,height):

self.width=width

self.height=height

self.inactive\_clr=(13,162,58)

self.active\_clr=(23,204,50)

self.draw\_effects=False

self.rect\_h=10

self.clear\_effects=False

self.rect\_w=width

def draw(self,x,y,message,action=None, font\_size=30):

mouse=pygame.mouse.get\_pos()

click=pygame.mouse.get\_pressed()

if x<mouse[0]<x+self.width and y<mouse[1]<y+self.height:

pygame.draw.rect(display,self.active\_clr,(x,y,self.width,self.height))

if click[0]==1:

pygame.mixer.Sound.play(jump\_sound)

pygame.time.delay(300)

if action is not None:

if action==quit:

pygame.quit()

quit()

else:

action()

else:

pygame.draw.rect(display,self.inactive\_clr,(x,y,self.width,self.height))

print\_text(message=message,x=x+10,y=y+10,font\_size=font\_size)

**button.py**

import pygame

from items import \*

pygame.init()

display\_width=800

display\_height=600

display=pygame.display.set\_mode((display\_width,display\_height))

pygame.display.set\_caption("Run!")

pygame.mixer.music.set\_volume(0.47)

bullet\_img=pygame.transform.scale(bullet\_img,(30,9))

health\_img=pygame.transform.scale(health\_img,(30,30))

cactus\_options=[69,449,37,410,40,420]

img\_counter=0

scores=0

max\_above=0

max\_scores=0

usr\_width=60

usr\_height=100

usr\_x=display\_width //2.5

usr\_y=display\_height-usr\_height-100

make\_jump=False

jump\_counter=30

health =2

cooldown=0

mouse\_counter=0

need\_draw\_click=False

clock=pygame.time.Clock()

**items.py**

import pygame

pygame.init()

bullet\_sound=pygame.mixer.Sound('shot.wav')

jump\_sound=pygame.mixer.Sound('jump.wav')

fall\_sound=pygame.mixer.Sound('Bdish.wav')

loss\_sound=pygame.mixer.Sound('loss.wav')

heart\_plus\_sound= pygame.mixer.Sound('hp+.wav')

pygame.mixer.music.load('background.mp3')

cactus\_img=[pygame.image.load('Cactus00.png'),

pygame.image.load('Cactus11.png'),pygame.image.load('Cactus22.png')]

stone\_img=[pygame.image.load('Stone0.png'),pygame.image.load('Stone1.png')]

cloud\_img=[pygame.image.load('Cloud0.png'),pygame.image.load('Cloud1.png')]

dino\_img=[pygame.image.load('Dino0.png'),pygame.image.load('Dino1.png'),

pygame.image.load('Dino2.png'),pygame.image.load('Dino3.png'),

pygame.image.load('Dino4.png')]

bird\_img=[pygame.image.load('Bird0.png'),pygame.image.load('Bird1.png'),

pygame.image.load('Bird2.png'),pygame.image.load('Bird3.png'),

pygame.image.load('Bird4.png'),pygame.image.load('Bird5.png')]

light\_img=[pygame.image.load('Light0.png'),pygame.image.load('Light1.png'),

pygame.image.load('Light2.png'),pygame.image.load('Light3.png'),

pygame.image.load('Light4.png'),pygame.image.load('Light5.png'),

pygame.image.load('Light6.png'),pygame.image.load('Light7.png'),

pygame.image.load('Light8.png'),pygame.image.load('Light9.png'),

pygame.image.load('Light10.png')]

health\_img=pygame.image.load('heart.png')

bullet\_img=pygame.image.load('shot.png')

**utils.py**

import pygame

import random

from globalVars import \*

from bird import Bird

pygame.init()

def print\_text(message,x,y,font\_color=(0,0,0), font\_type='PingPong.ttf', font\_size=30):

font\_type=pygame.font.Font(font\_type,font\_size)

text=font\_type.render(message, True,font\_color)

display.blit(text,(x,y))

def find\_radius(array):

maximum=max(array[0].x,array[1].x,array[2].x)

if maximum<display\_width:

radius=display\_width

if radius-maximum<50:

radius+=280

else:

radius=maximum

choice=random.randrange(0,5)

if choice==0:

radius+=random.randrange(15,25)

else:

radius+=random.randrange(250,400)

return radius

def pause():

paused=True

pygame.mixer.music.pause()

while paused:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

print\_text('Paused. Press enter to continue.',160,300)

keys=pygame.key.get\_pressed()

if keys[pygame.K\_RETURN]:

paused=False

pygame.display.update()

clock.tick(60)

pygame.mixer.music.unpause()

def draw\_birds(birds):

for bird in birds:

action=bird.draw()

if action==1:

bird.show()

elif action==2:

bird.hide()

else:

if bird.y>0:

bird.shoot()

def check\_birds\_dmg(bullets,birds):

for bird in birds:

for bullet in bullets:

bird.check\_damage(bullet)

def draw\_mouse():

global mouse\_counter, need\_draw\_click

mouse=pygame.mouse.get\_pos()

click=pygame.mouse.get\_pressed()

mouse\_size=[10,12,16,20,28,34,40,45,54,58]

if click[0] or click[1]:

need\_draw\_click=True

if need\_draw\_click:

draw\_x=mouse[0]-mouse\_size[mouse\_counter]//2

draw\_y=mouse[1]-mouse\_size[mouse\_counter]//2

display.blit(light\_img[mouse\_counter],(draw\_x,draw\_y))

mouse\_counter+=1

if mouse\_counter==10:

mouse\_counter=0

need\_draw\_click=False

def game\_over():

pygame.mixer.music.pause()

global scores,max\_scores

if scores>max\_scores:

max\_scores=scores

stopped=True

while stopped:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

print\_text('Game over. press Esc-stop, enter-try again',104,300)

print\_text('Max scores : '+str(max\_scores),300,350)

keys=pygame.key.get\_pressed()

if keys[pygame.K\_RETURN]:

return True

if keys[pygame.K\_ESCAPE]:

return False

pygame.display.update()

clock.tick(15)

pygame.mixer.music.unpause()

def object\_return(objects, obj):

radius=find\_radius(objects)

choice=random.randrange(0,3)

img=cactus\_img[choice]

width=cactus\_options[choice\*2]

height=cactus\_options[choice\*2+1]

obj.return\_self(radius,height,width,img)