Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчёт**

по лабораторной работе № 5 (15)

по дисциплине ***«Языки программирования»***

тема: **«Разработка интерактивных приложений с пользовательским графическим интерфейсом на языке Python. Создание игры»**

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | студент группы 10701118  Воробей И.А. |
| Приняла: |  | ст. пр. Лапанович И. О. |

2018-2019 учебный год

**ЛАБОРАТОРНАЯ РАБОТА № 5 (15)**

**РАЗРАБОТКА ИНТЕРАКТИВНЫХ ПРИЛОЖЕНИЙ С ПОЛЬЗОВАТЕЛЬСКИМ ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ НА ЯЗЫКЕ PYTHON. СОЗДАНИЕ ИГРЫ.**

Цель работы:

Изучить механизмы работы и способы создания приложения (игры) с GUI (графический пользовательский интерфейс) на основе библиотеки pygame.

Требования к выполнению:

1. Программа должна обязательно быть снабжена комментариями на английском языке
2. Каждый класс должен иметь адекватное осмысленное имя (обычно это *имя существительное*) и начинаться с заглавной буквы. Имена полей и методов должны начинаться с маленькой буквы и быть также осмысленными (имя метода, который что-то вычисляет, обычно называют *глаголом*, а поле – именем существительным).
3. Каждый класс необходимо разместить в отдельном модуле, который затем подключается в другом модуле, где происходит создание объекта данного класса и его использование.
4. При проектировании классов необходимо придерживаться принципа единственной ответственности (Single Responsibility Principle), т.е. классы должны проектироваться и реализовываться таким образом, чтобы они были слабо завязаны с другими классами при своей работе – они должны быть самодостаточными (стремитесь всегда разрабатывать масштабируемый и универсальный код).
5. Программа для демонстрации работоспособности разработанных классов должна быть снабжена дружелюбным и интуитивно понятным интерфейсом.

ОСНОВНОЕ ЗАДАНИЕ

**Описание предметной области:**

Динозаврик бежит и перепрыгивает колючки. Если он сталкивается с колючкой , то игрок проиграл. Если динозаврик успешно перепрыгивает колючку то счет увеличивается с каждой колючкой.

Руководство пользователя

Цель игры:

набрать как можно больше очков, не сталкиваясь с колючками.

Клавиши управления:

1) прыжок – пробел;

2) остановка игры – эскейп.

Основные элементы:

- главный персонаж.

C:\Users\Acer\Desktop\Языки программирования\я.п.лр4\Cactus11.png- препятствия.

C:\Users\Acer\Desktop\Языки программирования\я.п.лр4\Cloud0.png- облако.

C:\Users\Acer\Desktop\Языки программирования\я.п.лр4\Stone1.png- камень.

Результаты выполнения основного задания:



Рис. 1 – меню



Рис. 2 – игра на паузе



Рис. 3 – Game over

*ПРИЛОЖЕНИЕ А*

Фрагменты кода программы:

lab4.py

import pygame

import random

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.load('background.mp3')

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

cactus\_img=[pygame.image.load('Cactus00.png'),pygame.image.load('Cactus11.png'),pygame.image.load('Cactus22.png')]

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

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')]

img\_counter=0

scores=0

max\_above=0

max\_scores=0

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

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

usr\_width=60

usr\_height=100

usr\_x=display\_width //2.5

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

clock=pygame.time.Clock()

make\_jump=False

jump\_counter=30

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)

show=True

while show:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

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

start\_btn.draw(270,200, 'Start game',start\_game,50)

quit\_btn.draw(358,300, 'quit',quit,50)

pygame.display.update()

clock.tick(30)

def start\_game():

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

while game\_cycle():

scores=0

make\_jump=False

jump\_counter=30

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

def game\_cycle():

global make\_jump

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()

while game:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

quit()

keys=pygame.key.get\_pressed()

if keys[pygame.K\_SPACE]:

make\_jump=True

if keys[pygame.K\_ESCAPE]:

pause()

if make\_jump:

jump()

count\_scores(cactus\_arr)

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

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

draw\_array(cactus\_arr)

move\_objects(stone,cloud)

drow\_dino()

if check\_collision(cactus\_arr):

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

game=False

pygame.display.update()

clock.tick(100)

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 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:

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:

return True

else:

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

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

return True

else:

if not make\_jump:

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

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:

return True

elif jump\_counter<=-1:

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

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

return True

else:

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

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

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 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 draw\_array(array):

for cactus in array:

check=cactus.move()

if not check:

radius=find\_radius(array)

choice=random.randrange(0,3)

img=cactus\_img[choice]

width=cactus\_options[choice\*2]

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

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

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 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 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()

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)

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)

show\_menu()

pygame.quit()

quit()

Что я освоил в процессе выполнения лабораторной работы (выводы):

Разработал игру на языке Python на основе библиотеки pygame.