МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тюменский индустриальный университет»

Институт сервиса и отраслевого управления

Кафедра «Бизнес-информатики и математики»

**КУРСОВОЙ ПРОЕКТ**

**Дисциплина:** «Программирование»

**Тема:** «Разработка игровой программы “Змейка”»

**Выполнили:**студент 1 курса направление 38.03.05 «Бизнес – информатика» группа БИб-19-1

Лисовский А.И.

студент 1 курса направление 38.03.05 «Бизнес – информатика» группа БИб-19-1

Михеенков А.С.

**Проверил:**

к.с.н., доцент Аханова М.А.

Тюмень – 2019

**ЗАДАНИЕ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ**

Разработать игровую программу «Змейка». Программа должна выполнять следующие функции: обрабатывать игровой алгоритм, позволять игроку контролировать змейку, возможность для пользователя изменять некоторые настройки. Программа должна обладать понятным графическим интерфейсом.

**РЕФЕРАТ**

Курсовая работа содержит 40 страницы, 3истчников, 10 рисунка, листинг кода.

**Ключевые слова**: программа, язык программирования, интегрированная среда разработки, код программы, блок-схема, алгоритм, процедура, команда, интерфейс, спрайт.

**Объект исследования** – игровая программа «Змейка»

**Предмет исследованя** – средство разработки Visual Studio, язык программирования python и его модуль Pygame.

**Цель работы** – разработка игровой программы «Змейка» на языке python.

Результатом работы является программа – игровая программа «Змейка», которая показывает возможности языка программирования Python и модуля PyGame, в частности.

Курсовая работа выполнена в текстовом редакторе Microsoft Office Word 2016 и представлена на компакт-диске (в конверте на обороте обложки).

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc27903291)

[1. Постановка задач 6](#_Toc27903292)

[2. Алгоритм 7](#_Toc27903293)

[2.1. Блок-схема 7](#_Toc27903294)

[2.2. Описание команд программы 9](#_Toc27903295)

[3. Описание средств языка программирования 12](#_Toc27903296)

[3.1. Описание языка программирования Python 12](#_Toc27903297)

[3.2. Описания модуля PyGame 13](#_Toc27903298)

[4. Тестирование программы 14](#_Toc27903299)

# Введение

Компьютерные игры становятся частью нашей повседневной жизни. Появляются новые жанры, разработчики, улучшается качество игр как в плане графики, так и в плане сюжета. Эта индустрия насчитывает огромное количество продуктов, но среди них можно выделить классические игры известные повсеместно. Одной из таких игр является «Змейка», программная реализация которой представлена в данной работе.

**Объект исследования** – игровая программа «Змейка».

**Предмет исследованя** – инструментальные средства для разработки игровой программы «Змейка».

**Цель работы** – разработка игровой программы «Змейка» на языке программирования python.

Для реализации курсового проекта необходимо решить следующий ряд задач:

* Изучить интегрированную среду разработки python;
* Изучить компоненты игровой программы «Змейка»;
* Изучить элементы блок-схем и правила их построения;
* Проанализировать алгоритм работы игровой программы «Змейка»;
* Разработать свой алгоритм работы игровой программы «Змейка»;
* Разработать интерфейс для игровой программы «Змейка»;
* Разработать игровую программу «Змейка»;
* Спроектировать блок-схемы алгоритма работы игровой программы «Змейка»;
* Выполнить тестирование готовой программы.

# 1. Постановка задач

Необходимо разработать программу, выполняющую следующие функции:

* Реализовать алгоритм работы программы «Змейка»;
* Переход между разными окнами;
* Настройки некоторых параметров игры;
* Изменять направление змейки;
* Возможность ставить игру на паузу.

# 2. Алгоритм

## 2.1. Блок-схема



Рисунок – Основной цикл программы с циклами вступления, меню и настроек



Рисунок – Цикл настроек сложности с процессом игры



Рисунок – циклы паузы и экрана поражения.

## 2.2. Описание команд программы

В игровой программе «Змейка» используются 2 типа команд:

* команды для перемещения;
* команды для навигации по программе;

Команды для перемещения позволяют передвигаться змейке при помощи нажатия клавиш: движение вверх (W или UP), движение вниз (S или DOWN), движение влево (A или LEFT), движение вправо (D или RIGHT).

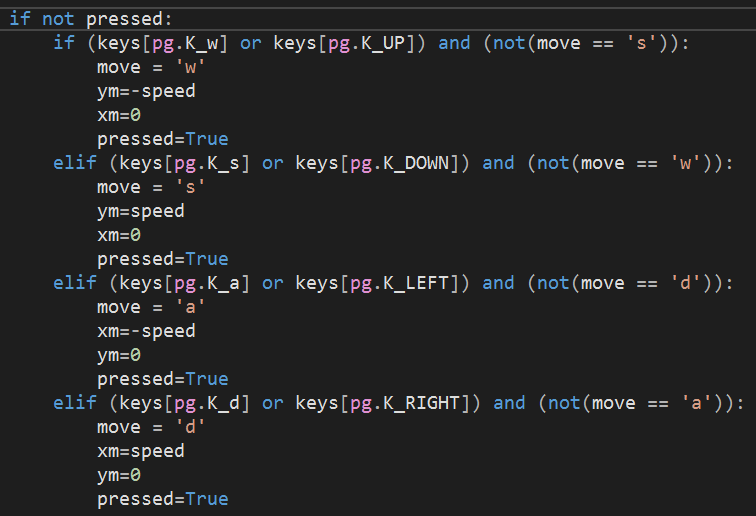


Рисунок – Отвечающий за движение код

Команды для навигации реализуются при помощи левой кнопки мыши, которая отвечает за переход по нажатию созданной кнопки в программе, и клавише Escape, выполняющая функцию перехода назад или выхода из программы.

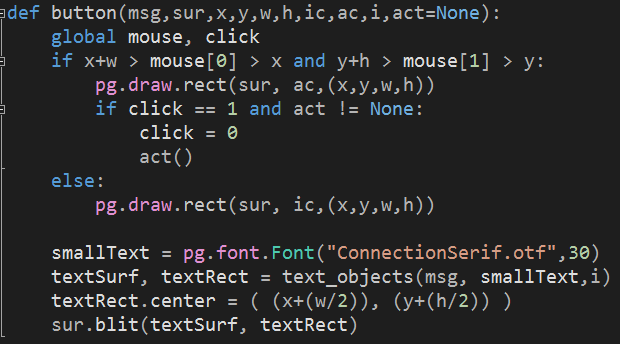


Рисунок – Функция создания кнопок и назначения им действий

# 3. Описание средств языка программирования

## 3.1. Описание языка программирования Python

**Python** — [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%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%D1%8F) общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. [Синтаксис](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_(%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)) ядра Python минималистичен. В то же время [стандартная библиотека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_Python) включает большой объём полезных функций.

Python поддерживает [структурное](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%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), [объектно-ориентированное](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%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), [функциональное](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%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), [императивное](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%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) и [аспектно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%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). Основные архитектурные черты — [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%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)), полная [интроспекция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%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)), механизм [обработки исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), высокоуровневые [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на [модули](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)), которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор [CPython](https://ru.wikipedia.org/wiki/CPython" \o "CPython), поддерживающий большинство активно используемых платформ[[12]](https://ru.wikipedia.org/wiki/Python#cite_note-13). Он распространяется под [свободной лицензией](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая [проприетарные](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E). Есть [реализация интерпретатора для JVM](https://ru.wikipedia.org/wiki/Jython) с возможностью [компиляции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80), [CLR](https://ru.wikipedia.org/wiki/IronPython), [LLVM](https://ru.wikipedia.org/wiki/LLVM), другие независимые реализации. Проект [PyPy](https://ru.wikipedia.org/wiki/PyPy" \o "PyPy) использует [JIT-компиляцию](https://ru.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F), которая значительно увеличивает скорость выполнения Python-программ.

Python — активно развивающийся [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%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%D1%8F), новые версии с добавлением/изменением языковых свойств выходят примерно раз в два с половиной года. Язык не подвергался официальной стандартизации, роль стандарта де-факто выполняет [CPython](https://ru.wikipedia.org/wiki/CPython" \o "CPython), разрабатываемый под контролем автора языка. В настоящий момент Python занимает третье место в рейтинге TIOBE с показателем 8,5 %. Аналитики отмечают, что это самый высокий балл Python за все время его присутствия в рейтинге.

## 3.2. Описания модуля PyGame

Pygame — набор модулей языка программирования Python, предназначенный для написания компьютерных игр и мультимедиа-приложений. Pygame базируется на мультимедийной библиотеке SDL.

Изначально Pygame был написан [Питом Шиннерсом](https://ru.wikipedia.org/w/index.php?title=%D0%9F%D0%B8%D1%82%D0%BE%D0%BC_%D0%A8%D0%B8%D0%BD%D0%BD%D0%B5%D1%80%D1%81%D0%BE%D0%BC&action=edit&redlink=1) . Начиная примерно с [2004](https://ru.wikipedia.org/wiki/2004)/[2005 года](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4) поддерживается и развивается сообществом свободного программного обеспечения.

# 4. Тестирование программы

Для тестирования программы был выполнен переход по интерфейсу всей программы, а также основной цикл. Данная программа предоставляет возможность поиграть в «Змейку». Программа позволяет протоколировать все выполняемые операции в отдельном окне. В программе имеется возможность изменять фоновую музыку или отключать её. Программа прошла тестирование и работает корректно.

# 5. Интерфейс программы



Рисунок – Главное меню



Рисунок – Настройки

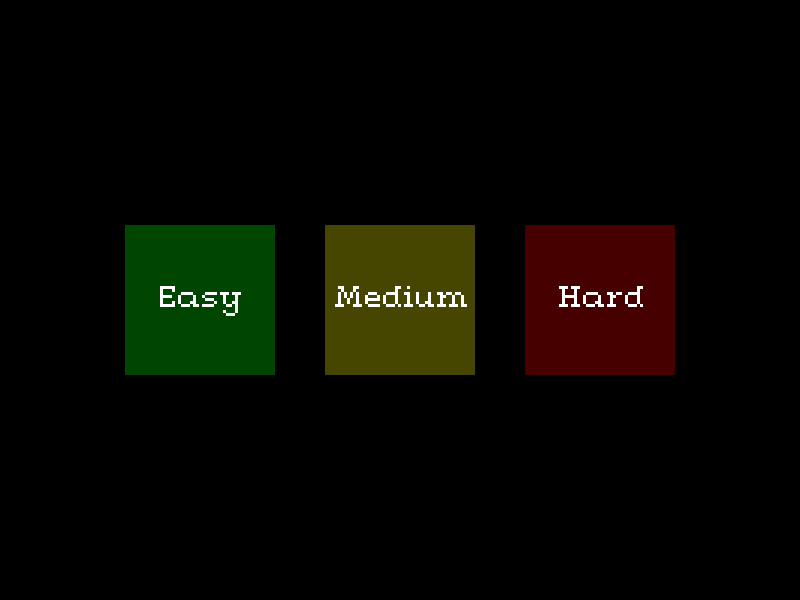


Рисунок – Выбор сложности



Рисунок – Игра «Змейка»



Рисунок – Конец игры

# Заключение

В процессе выполнения курсового проекта была разработана игровая программа «Змейка. Данная программа была написана на высокоуровневом языке программирования Python при помощи модуля PyGame. Программа имеет графический интерфейс, позволяет выбирать уровень сложности, звуковое сопровождение. Перемещение змейки осуществляется при помощи клавиш WASD и стрелок. Программа прошла тестирование, работает корректно.

# Список использованных источников

1. Документация Python 3 [Электронный ресурс] / Python – Электронные данные. Python Software Foundation © 2001-2009. – Режим доступа: <https://www.python.org/doc/> - (Дата обращения 15.10.2019)
2. Документация PyGame [Электронный ресурс] / PyGame – Электронные данные. Режим доступа: <https://www.pygame.org/docs/index.html> - (Дата обращения 20.10.2019)
3. Учим Python, делая крутые игры (2017). Python 3. [Текст] / Эл Свейгарт; 4-е издание. – Москва: Эксмо, 2018. – 416 с.

# Листинг

import pygame as pg

import random

pg.init()

Sounds={'eat':'sounds\Eatting.ogg','die':'sounds\Dying.ogg'}

Music = []

Snake\_head = [pg.image.load('sprites\Snake\_head\_w.png'),pg.image.load('sprites\Snake\_head\_d.png'),

pg.image.load('sprites\Snake\_head\_s.png'),pg.image.load('sprites\Snake\_head\_a.png')]

Snake\_body = [pg.image.load('sprites\Snake\_body\_v.png'),pg.image.load('sprites\Snake\_body\_h.png'),

pg.image.load('sprites\Snake\_body\_v.png'),pg.image.load('sprites\Snake\_body\_h.png'),

pg.image.load('sprites\Snake\_body\_t\_wd.png'),pg.image.load('sprites\Snake\_body\_t\_wa.png'),

pg.image.load('sprites\Snake\_body\_t\_sd.png'),pg.image.load('sprites\Snake\_body\_t\_sa.png')]

Snake\_tail = [pg.image.load('sprites\Snake\_tail\_w.png'),pg.image.load('sprites\Snake\_tail\_d.png'),

pg.image.load('sprites\Snake\_tail\_s.png'),pg.image.load('sprites\Snake\_tail\_a.png')]

Fruits=[pg.image.load('sprites\Fruit.png'),pg.image.load('sprites\GFruit.png')]

def setlvlez():

global lvl, presettings, play

lvl=80

presettings=False

play=True

def setlvlmed():

global lvl, presettings, play

lvl=60

presettings=False

play=True

def setlvlhard():

global lvl, presettings, play

lvl=40

presettings=False

play=True

def playmusic(vol):

global playedMus,Music,TimesPlayed

if TimesPlayed==len(Music):

random.shuffle(Music)

TimesPlayed=0

return

name=Music[TimesPlayed]

TimesPlayed+=1

pg.mixer.music.load(name)

pg.mixer.music.set\_volume(vol)

pg.mixer.music.play(0)

def testMusic():

if pg.mixer.music.get\_busy()==0:

playmusic(1)

def playsound(name,i,vol):

sound=pg.mixer.Sound(name)

sound.set\_volume(vol)

sound.play(i)

def fillMusic():

pg.mixer.music.stop()

global songtype, Music, TimesPlayed, playmusic

name=songtype

way='music\%s'% name

fileway=way+'\\'

way+='\songs.txt'

Songs=open(way,'r')

Music=[]

TimesPlayed=0

for line in Songs:

if '\n' in line:

line=line[:len(line)-1:1]

Music.append(fileway+line)

random.shuffle(Music)

playmusic(1)

file=open('songtype.txt','w')

file.write(songtype)

file.close()

def text\_objects(text, font,i):

global sur

textSurface = font.render(text, True,(i,i,i))

return textSurface, textSurface.get\_rect()

def button(msg,sur,x,y,w,h,ic,ac,i,act=None):

global mouse, click

if x+w > mouse[0] > x and y+h > mouse[1] > y:

pg.draw.rect(sur, ac,(x,y,w,h))

if click == 1 and act != None:

click = 0

act()

else:

pg.draw.rect(sur, ic,(x,y,w,h))

smallText = pg.font.Font("ConnectionSerif.otf",30)

textSurf, textRect = text\_objects(msg, smallText,i)

textRect.center = ( (x+(w/2)), (y+(h/2)) )

sur.blit(textSurf, textRect)

def Textsome(msg,sur,x,y,w,h,i,Size):

smallText = pg.font.Font("ConnectionSerif.otf",Size)

textSurf, textRect = text\_objects(msg, smallText,i)

textRect.center = ( (x+(w/2)), (y+(h/2)) )

sur.blit(textSurf, textRect)

def polzinormalno():

global dirmove, tailmove, Slen

for i in range(0,Slen-2):

if ((tailmove[i+1]==0) and (tailmove[i]==1)) or ((tailmove[i+1]==3) and (tailmove[i]==2)):

dirmove[i]=4

if ((tailmove[i+1]==0) and (tailmove[i]==3)) or ((tailmove[i+1]==1) and (tailmove[i]==2)):

dirmove[i]=5

if ((tailmove[i+1]==2) and (tailmove[i]==1))or((tailmove[i+1]==3)and(tailmove[i]==0)):

dirmove[i]=6

if ((tailmove[i+1]==2) and (tailmove[i]==3))or((tailmove[i+1]==1)and(tailmove[i]==0)):

dirmove[i]=7

def drawobj():

global sur,size, Fruit, GFruit, GFExis, move, dirmove, Slen, Snakehead, Snake, Snake\_head, Snake\_body

dir=0

if GFExis:

sur.blit(Fruits[1],(GFruit[0],GFruit[1]))

sur.blit(Fruits[0],(Fruit[0],Fruit[1]))

if move=='w':

dir=0

if move=='d':

dir=1

if move=='s':

dir=2

if move=='a':

dir=3

sur.blit(Snake\_head[dir],(Snakehead[0],Snakehead[1]))

for i in range(0,Slen-2):

sur.blit(Snake\_body[dirmove[i]],(Snake[i][0],Snake[i][1]))

sur.blit(Snake\_tail[tailmove[Slen-2]],(Snake[Slen-2][0],Snake[Slen-2][1]))

def drawsur():

global sur,W ,H, size

pg.draw.lines(sur,(255,255,255),True,[(0,0),(0,H-1),(W-1,H-1),(W-1,0)])

pg.draw.line(sur,(255,255,255),(0,size\*10),(W-1,size\*10))

msg='Score:'+str(Score)

Textsome(msg,sur,20,70,130,20,255,30)

file=open('record.txt','r')

line=file.read()

msg='Record:'+str(line)

Textsome(msg,sur,620,70,130,20,255,30)

file.close()

if Score > int(line):

file=open('record.txt','w')

file.write(str(Slen))

file.close()

def surupdate():

global sur

pg.display.update()

sur.fill((0,0,0))

def initFruit(W,H,size,Fruit1,Snake,Snakehead,Slen):

global GFMake, GFruit, Fruit

ex=True

while ex:

fx=random.randint(10,((W/size)-1))\*size

fy=random.randint(11,((H/size)-1))\*size

Fruit1=[fx,fy]

if GFMake:

if Fruit1 in Snake or Fruit1==Fruit:

ex = True

else:

ex=False

else:

if Fruit1 in Snake or Fruit1==GFruit:

ex = True

else:

ex=False

return Fruit1

def quit():

global menu, work

menu=False

work=False

def choicelvl():

global presettings, menu

presettings=True

menu=False

def startsettings():

global menu, settings

menu=False

settings=True

def keepplaing():

global Pause

Pause=False

def gotomenu():

global settings, play, menu, Pause

settings=False

play=False

menu=True

Pause=False

presettings=False

work=True

play=False

menu=True

intro=True

settings=False

Pause=False

gameover=False

TimesPlayed=0

file=open('songtype.txt','r')

songtype=file.read()

file.close()

fillMusic()

mouse=pg.mouse.get\_pos()

click=pg.mouse.get\_pressed()

testMusic()

while work :

W=800

H=600

#pg.init()

sur=pg.display.set\_mode(size=(W,H),flags=pg.FULLSCREEN)

i=0

while intro:

i+=1

largeText = pg.font.SysFont('times new roman', 40)

TextSurf, TextRect=text\_objects('The Game by Aytomik and Vaider',largeText,i)

TextRect.center = ((400),(300))

sur.blit(TextSurf, TextRect)

surupdate()

pg.time.delay(15)

if i == 225:

menu=True

intro=False

for event in pg.event.get():

if event.type==pg.KEYUP or event.type==pg.MOUSEBUTTONDOWN:

menu=True

intro=False

break

while menu:

for event in pg.event.get():

keys=pg.key.get\_pressed()

if event.type == pg.MOUSEMOTION:

mouse=event.pos

if event.type == pg.MOUSEBUTTONUP:

mouse=event.pos

click=event.button

if event.type == pg.QUIT or keys[pg.K\_ESCAPE]:

play = False

pg.quit()

break

n=255

button('Play',sur,W/2-75,H/2-25,150,40,(0,50,0),(100,200,100),n,choicelvl)

button('Quit',sur,W/2-75,H/2+20,150,40,(50,0,0),(200,100,100),n,quit)

button('Settings',sur,W/2-75,H/2+65,150,40,(30,30,30),(50,50,50),n,startsettings)

surupdate()

songtype= open('songtype.txt','r')

testMusic()

while settings:

for event in pg.event.get():

keys=pg.key.get\_pressed()

if event.type == pg.MOUSEMOTION:

mouse=event.pos

if event.type == pg.MOUSEBUTTONUP:

mouse=event.pos

click=event.button

if event.type == pg.QUIT or keys[pg.K\_ESCAPE]:

menu = True

settings = False

break

heights=60

first=150

songtype='Retrowave'

button('Retrowave',sur,W/2-95,first,190,50,(30,30,30),(60,60,60),n,fillMusic)

songtype='Fun'

button('Fun',sur,W/2-95,first+heights,190,50,(30,30,30),(60,60,60),n,fillMusic)

songtype='Pop'

button('Pop',sur,W/2-95,first+heights\*2,190,50,(30,30,30),(60,60,60),n,fillMusic)

songtype='8-bit'

button('8-bit',sur,W/2-95,first+heights\*3,190,50,(30,30,30),(60,60,60),n,fillMusic)

songtype = 'OffMusic'

button('Off Music',sur,W/2-95,first+heights\*4,190,50,(30,30,30),(60,60,60),n,fillMusic)

surupdate()

testMusic()

while presettings:

for event in pg.event.get():

keys=pg.key.get\_pressed()

if event.type == pg.MOUSEMOTION:

mouse=event.pos

if event.type == pg.MOUSEBUTTONUP:

mouse=event.pos

click=event.button

if keys[pg.K\_ESCAPE] :

menu = True

presettings = False

break

button('Easy',sur,W-675,H/2-75,150,150,(0,70,0),(0,100,0),n,setlvlez)

button('Medium',sur,W/2-75,H/2-75,150,150,(70,70,0),(100,100,0),n,setlvlmed)

button('Hard',sur,W-275,H/2-75,150,150,(70,0,0),(100,0,0),n,setlvlhard)

surupdate()

if play==True:

size=10

speed=1\*size

FPS = 10

Score=0

Snakehead=[None,None]

Snake=[]

dirmove=[]

tailmove=[]

Slen=10

mult=3

Counter2=Slen-2

x = random.randint(Slen,int(W/size))\*size

y = random.randint(10,int(H/size))\*size

sur=pg.display.set\_mode(size=(W,H),flags = pg.FULLSCREEN)

pg.display.set\_caption('Snake')

play=True

xm = 0

ym = 0

move='d'

if move=='w':

ym=-speed

if move=='s':

ym=speed

if move=='a':

xm=-speed

if move=='d':

xm=speed

for i in range(0,Slen-1):

Snake.append([None]\*2)

dirmove.append(1)

tailmove.append(1)

for i in range(0,Slen-1):

Snake[i]=[(x-(1+i)\*size),y]

Fruit=[None,None]

GFruit=[None,None]

GFExis=False

GFMake=False

bool=True

counter=0

index=0

Fruit=initFruit(W,H,size,Fruit,Snake,Snakehead,Slen)

GFruit=initFruit(W,H,size,Fruit,Snake,Snakehead,Slen)

while play:

pressed = False

pg.time.delay(lvl)

for event in pg.event.get():

keys=pg.key.get\_pressed()

if event.type == pg.QUIT:

play = False

pg.quit()

break

if event.type == pg.KEYUP:

if event.key == pg.K\_ESCAPE:

Pause = True

Snakehead=[x,y]

if Slen > Score:

Score+=1

if counter>=5:

GFExis=True

if index>(5000/lvl):

counter=0

index=0

GFExis=False

GFMake=True

GFruit=initFruit(W,H,size,Fruit,Snake,Snakehead,Slen)

GFMake=False

index+=1

if Snakehead==GFruit:

playsound(Sounds['eat'],0,1)

GFMake=True

GFruit=initFruit(W,H,size,Fruit,Snake,Snakehead,Slen)

GFMake=False

mult=random.randint(1,4)

for i in range(0,3\*mult):

dirmove.append(int(dirmove[Slen-2]))

tailmove.append(int(tailmove[Slen-2]))

Snake.append([Snake[Slen-2][0],Snake[Slen-2][1]])

Slen+=3\*mult

counter=0

index=0

GFExis=False

if Snakehead in Snake:

playsound(Sounds['die'],0,0.7)

pg.time.delay(3500)

play = False

gameover=True

if Snakehead==Fruit:

playsound(Sounds['eat'],0,1)

Fruit=initFruit(W,H,size,Fruit,Snake,Snakehead,Slen)

mult=random.randint(1,mult)

for i in range(0,mult):

Snake.append([Snake[Slen-2][0],Snake[Slen-2][1]])

dirmove.append(int(dirmove[Slen-2]))

tailmove.append(int(tailmove[Slen-2]))

Slen+=mult

counter+=1

if not pressed:

if (keys[pg.K\_w] or keys[pg.K\_UP]) and (not(move == 's')):

move = 'w'

ym=-speed

xm=0

pressed=True

elif (keys[pg.K\_s] or keys[pg.K\_DOWN]) and (not(move == 'w')):

move = 's'

ym=speed

xm=0

pressed=True

elif (keys[pg.K\_a] or keys[pg.K\_LEFT]) and (not(move == 'd')):

move = 'a'

xm=-speed

ym=0

pressed=True

elif (keys[pg.K\_d] or keys[pg.K\_RIGHT]) and (not(move == 'a')):

move = 'd'

xm=speed

ym=0

pressed=True

x+=xm

y+=ym

if x<0:

x=W-size

if x>(W-size):

x=0

if y<size\*10:

y=H-size

if y>(H-size):

y=size\*10

drawobj()

drawsur()

for i in range((Slen-2),(-1),-1):

Snake[i]=[(Snake[i-1][0]),(Snake[i-1][1])]

Snake[0]=[Snakehead[0],Snakehead[1]]

for i in range((Slen-2),(-1),-1):

dirmove[i]=int(dirmove[i-1])

if move=='w':

dirmove[0]=0

if move=='d':

dirmove[0]=1

if move=='s':

dirmove[0]=2

if move=='a':

dirmove[0]=3

for i in range((Slen-2),(-1),-1):

tailmove[i]=int(tailmove[i-1])

if move=='w':

tailmove[0]=0

if move=='d':

tailmove[0]=1

if move=='s':

tailmove[0]=2

if move=='a':

tailmove[0]=3

surupdate()

polzinormalno()

testMusic()

while Pause:

surupdate()

for event in pg.event.get():

if event.type == pg.MOUSEMOTION:

mouse=event.pos

if event.type == pg.MOUSEBUTTONUP:

click=event.button

button('Resume',sur,W/2-75,H/2-20,150,40,(0,70,0),(0,100,0),n,keepplaing)

button('Menu',sur,W/2-75,H/2+25,150,40,(70,0,0),(100,0,0),n,gotomenu)

testMusic()

blink=0

blink\_phase=0

while gameover:

gameover = True

menu = False

Textsome('Game over',sur, W/2-50, H/2-150,100,300,255,40)

Textsome('Press any key to continue',sur, W/2-50, H/2+50,100,300,blink,22)

Textsome('Your score:'+str(Score),sur, W/2-50, H/2-75,100,300,255,26)

surupdate()

if blink\_phase == 0:

blink+=1

if blink==255:

blink\_phase=1

else:

blink-=1

if blink==0:

blink\_phase=0

for event in pg.event.get():

if event.type == pg.KEYDOWN or event.type == pg.MOUSEBUTTONUP:

gameover=False

menu=True