Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный университет промышленных технологий и дизайна»

|  |
| --- |
| **Институт дополнительного профессионального образования** |

|  |  |
| --- | --- |
|  |  |
| Направление подготовки: |  |
| Профиль подготовки: | Экономика предприятий и организаций |

|  |
| --- |
| **ОТЧЕТ** |
| о прохождении практики |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель  от профильной организации / структурного подразделения СПбГУПТД\*: |  | Никита Александрович Баев | | | | |  |  |
| *(наименование организации)* |  | *(должность, Ф.И.О., печать организации)* | | | | |  | *(подпись, печать)* |
| Руководитель  от СПбГУПТД: |  | Зубов Александр Михайлович | | | | |  |  |
|  |  | *(должность, ученая степень / звание, Ф.И.О.)* | | | | |  | *(подпись)* |
| Обучающийся: |  | Муродзода Фотима Абдухамид | | | | |  |  |
|  |  | *(Ф.И.О.)* | | | | |  | *(подпись)* |
| Курс | 2 |  |  | Учебная группа: |  | 2-ТД-2 | |

Санкт-Петербург

2023

## РЕФЕРАТ

Курсовая работа содержит 18 страниц, 2 таблицы, 0 рисунков и 0 формул. Список использованных источников содержит 9 наименований.

**PYTHON, PYCHARM, PYGAME, ЗМЕЙКА, КОД, ПРОГРАММИРОВАНИЕ.**

Цель работы – научиться создавать программный код для игр на языке python

Задачи исследования:

- понять, как работает библиотека pygame;

- научиться использовать на практике программные комбинации;

-создать логически связанную игру.

Объект исследования – игры на языке python

Предмет исследования – игра «змейка»

СОДЕРЖАНИЕ

[РЕФЕРАТ 2](#_Toc132821659)

[ВВЕДЕНИЕ 4](#_Toc132821660)

[ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ РЕШАЕМОЙ ЗАДАЧИ. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc132821661)

[1.1. Знакомство с Pygame и основными операторами Python 5](#_Toc132821662)

[1.2. Игровой цикл 6](#_Toc132821663)

[ГЛАВА 2. РЕШЕНИЕ ЗАДАЧИ 9](#_Toc132821666)

[2.1. Продумываем и описываем правила игры 9](#_Toc132821667)

[2.2.Создание змейки 10](#_Toc132821668)

[2.3. программный код 11](#_Toc132821673)

[ЗАКЛЮЧЕНИЕ 16](#_Toc132821674)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ: 17](#_Toc132821675)

## ВВЕДЕНИЕ

Все мы знаем игру «Змейка». Раньше в нее играли на кнопочной Нокии. Сегодня существует миллионы вариаций: коты едят мышей, сетевое соревнование по поеданию разноцветными червяками друг друга и так далее. На самом деле, написать классическую и простейшую «змейку» сможет каждый новичок.

Змейка (Snake) — это игра, созданная Танели Арманто в 1997 году специально для телефона Nokia. Змейка стала переломным моментом в развитии мобильного гейминга и завоевала сердца миллионов людей.

Хоть Танели Арманто и считается создателем змейки, идея игры ему не принадлежит. Первый вариант змейки появился аж в 1976 году и назывался Blockade. После были созданы десятки клонов этой игры с разными названиями. Арманто при создании игры вдохновился одним из таких клонов, который был установлен на его Apple Macintosh.

Для создания змейки можно использовать все языки программирования. Я буду использовать Python!  Язык программирования Python универсален: он подходит как для решения повседневных задач, так и для создания простых игр.

Змейку на Python можно написать несколькими способами. Первый — при помощи библиотеки Python PyGame, которую используют для создания игр. Второй — с помощью встроенной библиотеки Turtle. Для себя я выбрала первый способ.

## ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ РЕШАЕМОЙ ЗАДАЧИ. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

# Знакомство с Pygame и основными операторами Python

Для того, чтобы написать игру на языке Python, нужны специальные библиотеки, которые содержат в себе базу для этих игр.

Одной из самых популярных библиотек является Pygame. Она представляет собой, как я говорила раннее, «игровую библиотеку», набор инструментов, помогающих программистам создавать игры. К ним относятся:

* Графика и анимация;
* Звук (включая музыку);
* Управление (мышь, клавиатура, геймпад и так далее)[3];

Pygame - эта библиотека поможет нам получить все графические компоненты для сборки игры.

Random - эта библиотека поможет нам создавать еду в случайных местах в игре.

Время - ото помогает нам использовать в игре функции, связанные со временем, например скорость змейки[4].

Главные функци, которые нам пригодятся в работе:

* import pygame отвечает за импорт libraries, включенный to the module;
* init() используется для подготовки модулей к работе;
* screen отвечает за создание графического окна, передав в set\_mode() разрешение в качестве парный цифр;
* while True – основной цикл, перехватывающий событие закрытия основного графического окна посетителем.

После наступления желаемого события происходит завершение работы с библиотекой при помощи функции quit. Exit() отвечает за прекращение использования всего итогового приложения.

Таблица 1 – операторы Python

|  |  |
| --- | --- |
| Функция | Пояснение |
| init() | Инициализирует все модули Pygame (возвращает кортеж в случае успеха или неудачи). |
| isplay.set\_mode() | Для создания поверхности принимает в качестве параметра либо список, либо кортеж (кортеж предпочтительней). |
| update() | Обновляет экран. |
| quit() | Используется для деинициализации всех модулей. |
| set\_caption() | Устанавливает текст заголовка в верхней части экрана |
| event.get() | Возвращает список всех событий. |
| Surface.fill() | Заполняет пространство сплошным цветом. |
| time.Clock() | Отслеживание времени |
| font.SysFont() | Задает шрифт Pygame, используя системные ресурсы. |

# Игровой цикл

В сердце каждой игры лежит цикл, который принято называть «игровым циклом». Он запускается снова и снова, делая все, чтобы работала игра. Каждый цикл в игре называется кадром.

В каждом кадре происходит масса вещей, но их можно разбить на три категории:

Обработка ввода (события)

Речь идет обо всем, что происходит вне игры — тех событиях, на которые она должна реагировать. Это могут быть нажатия клавиш на клавиатуре, клики мышью.

Обновление игры

Изменение всего, что должно измениться в течение одного кадра. Если персонаж в воздухе, гравитация должна потянуть его вниз. Если два объекта встречаются на большой скорости, они должны взорваться.

Рендеринг (прорисовка)

В этом шаге все выводится на экран: фоны, персонажи, меню. Все, что игрок должен видеть, появляется на экране в нужном месте.

Время

Еще один важный аспект игрового цикла — скорость его работы. Многие наверняка знакомы с термином FPS, который расшифровывается как Frames Per Second (или кадры в секунду). Он указывает на то, сколько раз цикл должен повториться за одну секунду. Это важно, чтобы игра не была слишком медленной или быстрой. Важно и то, чтобы игра не работала с разной скоростью на разных ПК. Если персонажу необходимо 10 секунд на то, чтобы пересечь экран, эти 10 секунд должны быть неизменными для всех компьютеров.

Начнем с раздела отрисовки. Персонажей пока нет, поэтому экран можно заполнить сплошным цветом. Чтобы сделать это, нужно разобраться, как компьютер обрабатывает цвета.

Экраны компьютеров сделаны из пикселей, каждый из которых содержит 3 элемента: красный, зеленый и синий. Цвет пикселя определяется тем, как горит каждый из элементов:

Таблица 2 – цвета RGB

|  |  |  |  |
| --- | --- | --- | --- |
| Красный | Зеленый | Голубой | Цвет |
| 0 | 0 | 0 | Черный |
| 255 | 0 | 0 | Красный |
| 0 | 255 | 0 | Зеленый |
| 0 | 0 | 255 | Синий |
| 0 | 255 | 255 | Голубой |
| 255 | 0 | 255 | Желтый |
| 255 | 255 | 0 | Фиолетовый |
| 255 | 255 | 255 | Белый |

Раздел ввода (событий)

Игры еще нет, поэтому пока сложно сказать, какие кнопки или другие элементы управления понадобятся. Но нужно настроить одно важное событие. Если попытаться запустить программу сейчас, то станет понятно, что нет возможности закрыть окно. Нажать на крестик в верхнем углу недостаточно. Это тоже событие, и необходимо сообщить программе, чтобы она считала его и, соответственно, закрыла игру.

## Контроль FPS

Пока что нечего поместить в раздел Update (обновление), но нужно убедиться, что настройка FPS контролирует скорость игры.

Команда tick просит pygame определить, сколько занимает цикл, а затем сделать паузу, чтобы цикл (целый кадр) длился нужно время. Если задать значение FPS 30, это значит, что длина одного кадра — 1/30, то есть 0,03 секунды. Если цикл кода (обновление, рендеринг и прочее) занимает 0,01 секунды, тогда pygame сделает паузу на 0,02 секунды.

## Итог

Наконец, нужно убедиться, что, когда игровой цикл завершается, окно игры закрывается. Для этого нужно поместить функцию pygame.quit в конце кода[3].

## 

## 

# ****ГЛАВА 2. РЕШЕНИЕ ЗАДАЧИ****

## ****2.1. Продумываем и описываем правила игры****

Перед тем как начать писать код, необходимо подумать о том, какие правила мы должны учитывать при создании игры. Давайте составим для «Змейки» базовые правила:

* цель игры — получить как змейку как можно длиннее;
* при поглощении специального объекта на игровом экране длина змейки увеличивается на один блок;
* игрок может двигаться влево, вправо, вверх или вниз;
* игрок не может двигаться за пределы экрана, при столкновении с границами игра заканчивается;
* игра заканчивается, когда змейка врезается сама в себя;
* ещё игра заканчивается, когда пользователь закрывает окно [4].

## 2.2. алгоритмизация

Во-первых, для работы с созданием кода на языке Python необходимо установить сам Python или обновить его до версии, как минимум, 3.6. В моем случае я обновила уже имеющийся язык программирования до версии 3.11, так как на сайте представлена самая последняя версия.

Во-вторых, надо установить программу PyCharm. Что я тоже сделала.

Далее, устанавливаем игровую библиотеку Pygame в программе, для того чтобы существовала база для нашей будущей игры.

1.Создание экрана

Для создания экрана при помощи Pygame нужно воспользоваться функцией display.set\_mode. Также необходимо пользоваться методом init для инициализации экрана в начале кода и методом quit для его закрытия в конце. Метод update используется для применения каких-либо изменений на экране. Еще существует метод flip, который работает похожим с update образом. Разница заключается лишь в том, что метод flip переписывает весь экран целиком, а метод update применяет именно изменения (хотя если его использовать без параметров, то он тоже переписывает весь экран). Чтобы экран не исчез через секунду после появления надо добавить цикл while, который будет работать до завершения игры. Теперь чтобы при нажатии на крестик наша игра закрылась, нужно использовать функцию «quit».

## 2.2.Создание змейки

Перед тем как создать змейку, мы инициируем несколько цветовых переменных для раскрашивания самой змейки, еды и экрана. В Pygame используется цветовая схема RGB (RED, GREEN, BLUE). Установка всех цветов в 0 соответствует черному цвету, а в 255 — соответственно, белому.

Фактически, наша змейка является прямоугольником. Чтобы нарисовать прямоугольник в Pygame, можно воспользоваться функцией draw.rect, которая нарисует нам прямоугольник заданного цвета и размера.

Чтобы передвигать змейку, мы будем использовать ключевые события из класса KEYDOWN библиотеки Pygame. События K\_UP, K\_DOWN, K\_LEFT, и K\_RIGHT заставят змейку двигаться вверх, вниз, влево и вправо соответственно. Также, цвет дисплея меняется от черного (по умолчанию) до белого при помощи метода fill.

Для сохранения изменений координат x и y мы создали две новых переменные: x1\_change и y1\_change.

В игре змейка игрок проигрывает, если касается границы экрана. Чтобы задать такое поведение, мы должны использовать инструкцию if, которая будет следить за тем, чтобы координаты x и y были меньше размеров экрана. Мы будем использовать для этого переменные, чтобы вы могли потом, при случае, легко внести какие-либо изменения в игру.

Теперь мы добавим еды для змейки, и когда она будет ее пересекать, ее размер будет также увеличиваться на один блок. Помимо этого, я внесу небольшие изменения, которые дадут возможность прекращать игру, а также начинать ее заново в случае поражения.

Следующий код будет увеличивать длину змейки после поглощения ею еды. Также, если змейка сталкивается с собственным хвостом, игра заканчивается и выводится сообщение: “ вы проиграли! Нажмите C-начать сначала или Q-выход “. Длина змейки хранится в списке, а базовые значения заданы в следующем коде [6].

## 2.3. программный код

import pygame  
import time  
import random  
  
pygame.init()  
  
white = (50, 153, 213)  
yellow = (255, 255, 102)  
black = (0, 0, 0)  
red = (213, 50, 80)  
green = (0, 255, 0)  
blue = (255, 255, 255)  
  
dis\_width = 600  
dis\_height = 400  
  
dis = pygame.display.set\_mode((dis\_width, dis\_height))  
pygame.display.set\_caption('Змейка')  
  
clock = pygame.time.Clock()  
  
snake\_block = 10  
snake\_speed = 7  
  
font\_style = pygame.font.SysFont("timesnewroman", 15)  
  
  
  
def our\_snake(snake\_block, snake\_list):  
 for x in snake\_list:  
 pygame.draw.rect(dis, black, [x[0], x[1], snake\_block, snake\_block])  
  
  
def message(msg, color):  
 mesg = font\_style.render(msg, True, color)  
 dis.blit(mesg, [dis\_width / 6, dis\_height / 3])  
  
  
def gameLoop():  
 game\_over = False  
 game\_close = False  
  
 x1 = dis\_width / 2  
 y1 = dis\_height / 2  
  
 x1\_change = 0  
 y1\_change = 0  
  
 snake\_List = []  
 Length\_of\_snake = 1  
  
 foodx = round(random.randrange(0, dis\_width - snake\_block) / 10.0) \* 10.0  
 foody = round(random.randrange(0, dis\_height - snake\_block) / 10.0) \* 10.0  
  
 while not game\_over:  
  
 while game\_close == True:  
 dis.fill(blue)  
 message("вы проиграли! Нажмите C-начать сначала или Q-выход", red)  
  
 pygame.display.update()  
  
 for event in pygame.event.get():  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_q:  
 game\_over = True  
 game\_close = False  
 if event.key == pygame.K\_c:  
 gameLoop()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 game\_over = True  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_a:  
 x1\_change = -snake\_block  
 y1\_change = 0  
 elif event.key == pygame.K\_d:  
 x1\_change = snake\_block  
 y1\_change = 0  
 elif event.key == pygame.K\_w:  
 y1\_change = -snake\_block  
 x1\_change = 0  
 elif event.key == pygame.K\_s:  
 y1\_change = snake\_block  
 x1\_change = 0  
  
 if x1 >= dis\_width or x1 < 0 or y1 >= dis\_height or y1 < 0:  
 game\_close = True  
 x1 += x1\_change  
 y1 += y1\_change  
 dis.fill(blue)  
 pygame.draw.rect(dis, green, [foodx, foody, snake\_block, snake\_block])  
 snake\_Head = []  
 snake\_Head.append(x1)  
 snake\_Head.append(y1)  
 snake\_List.append(snake\_Head)  
 if len(snake\_List) > Length\_of\_snake:  
 del snake\_List[0]  
  
 for x in snake\_List[:-1]:  
 if x == snake\_Head:  
 game\_close = True  
  
 our\_snake(snake\_block, snake\_List)  
  
 pygame.display.update()  
  
 if x1 == foodx and y1 == foody:  
 foodx = round(random.randrange(0, dis\_width - snake\_block) / 10.0) \* 10.0  
 foody = round(random.randrange(0, dis\_height - snake\_block) / 10.0) \* 10.0  
 Length\_of\_snake += 1  
  
 clock.tick(snake\_speed)  
  
 pygame.quit()  
 quit()  
  
  
gameLoop()

## ЗАКЛЮЧЕНИЕ

В данной работе мы рассмотрели создание игры с помощью языка Python. Для того, чтобы мы смогли это сделать, для начала надо познакомиться с основными понятиями и операторами данного языка программирования. После этого следует подготовить компьютер для работы: обновить Python, PyCharm и так далее. Затем мы определяем правила, которым будем следовать при написании самого кода.

Также мы выяснили ряд ошибок, при наличии которых невозможен запуск кода. В основном сложности возникают на начальном этапе создания игры, а именно при написании кода для экрана. Необходимо учитывать все нюансы. Если писать код по принципу пошаговой инструкции, то вначале окно выскочит на пару секунд и исчезнет. Тогда нужно добавить цикл while. После этого окно перестанет пропадать, при чем совсем: при нажатии на крестик, оно не будет закрываться. Для устранения этой неполадки нам понадобится оператор quit(). Теперь наше поле для игры появляется и закрывается, как надо.

В дальнейшем для написания кода нужно, как я написала выше, понимание основных принципов языка программирования и внимательность.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ:

1. Библиотека Pygame. Часть 1. Введение // PythonRU 2018-2021гг // (электронный ресурс). Режим доступа: https://pythonru.com/uroki/biblioteka-pygame-chast-1-vvedenie?ysclid=lgmaiwi75p731564441
2. Инструкция и выражения для начинающих // PythonRU 2018-2021гг // (электронный ресурс). Режим доступа : https://pythonru.com/uroki/instrukcii-i-vyrazhenija-dlja-nachinajushhih?ysclid=lgntpj92r864883064
3. Как писать игры на Python с Pygame // timeweb cloud 2023 // электронный ресурс). Режим доступа: https://timeweb.cloud
4. Краткий курс ООП на Python: как избежать путаницы в коде // skillbox media 2023г // (электронный ресурс). Режим доступа: https://skillbox.ru/media/code/kak-izbezhat-putanitsy-v-kode-ili-kratkiy-kurs-oop-na-python/?ysclid=lgnu06ihh6659998908
5. Метод Time clock() в Python – подробно на примерах // Помощник Python 2022г // (электронный ресурс). Режим доступа: https://pythonpip.ru/osnovy/metod-time-clock-v-python-podrobno-na-primerah
6. Руководство по программированию игр на Pygame в Python – установка и простые примеры // // Помощник Python 2022г // (электронный ресурс). Режим доступа: https://pythonpip.ru/osnovy/rukovodstvo-po-programmirovaniyu-igr-na-pygame-v-python-ustanovka-i-prostye-primery
7. How to set up the Game Loop in PygGame? // GeeksforGeeks 2021г // (электронный ресурс). Режим доступа: https://www.geeksforgeeks.org/how-to-set-up-the-game-loop-in-pyggame/
8. PyGame: учебник по программированию // DMCA: dmca 2023г // (электронный ресурс). Режим доступа: https://www.codeflow.site/ru/
9. Pygame. Основы работы // OTUS JOURNAL 2023г // (электронный ресурс). Режим доступа: https://otus.ru/journal/pygame-osnovy-raboty/?ysclid=lgns4e8659578299498