Муниципальное автономное общеобразовательное учреждение

«Бардымская средняя общеобразовательная школа №2»

**Индивидуальный проект по теме:**

**«Создание компьютерной игры на языке программирования python с помощью библиотеки pygame»**

**Проект выполнил:**

Тагиров Вадим Эльвирович,

обучающийся 11 «А» класса

**Руководитель проекта:**

Ахмарова Гульнур Кадриевна,

учитель информатики

с. Барда, 2025

**ПАСПОРТ ПРОЕКТА**

|  |  |
| --- | --- |
| **Название проекта** | «Создание компьютерной игры на языке программирования python с помощью библиотеки pygame» |
| **Автор проекта, возраст** | Тагиров Вадим Эльвирович, 17 лет. |
| **Учебный предмет** | Информатика |
| **Руководитель проекта** | Ахмарова Гульнур Кадриевна |
| **Тип проекта** | Творческий |
| **Продукт проекта** | Компьютерная игра |
| **Актуальность проекта** | Изучение программирования, в частности написание игр – это актуальная тема, но многие начинающие программисты сталкиваются с трудными языками программирования. Из-за трудностей они перегорают от этого дела и уходят из сферы. На первых этапах на помощь приходит лёгкий для освоения язык python, который позволит делать игры и другие программы даже новичку. |
| **Цель и задачи проекта** | **Цель** – создание функциональной игры на движке pygame.  **Задачи:**   * Познакомиться с историей возникновения игр, узнать какие виды игр существуют. * Познакомиться с этапами разработки компьютерной игры, с библиотекой pygame * Познакомиться с Объектно-ориентированным программированием.. * Создать свою компьютерную игру. |
| **Этапы работы над проектом** | 1) Сбор данных об истории компьютерных игр  2) Изучение некоторых аспектов объектно-ориентированного программирования  3) Изучение возможностей библиотеки pygame  3) Создание компьютерной игры  4) Узнать мнение людей о моей игре |
| **Результат проекта** | Компьютерна игра «Shaurm Cubeater» |
| **Дальнейшее развитие проекта** | Создание игр с бо́льшим функционалом, сюжетом. |
| **Материально-техническое** | Веб-ресурсы |

**Содержание**

Введение 4

1.Теоретическая часть 6

1.1. История развития компьютерных игр 6

1.2. Жанры компьютерных игр 8

1.3. Язык программирования Python 10

1.4. Объектно-ориентированное программирование 12

2. Практическая часть 14

2.1. Выбор темы для игры 14

2.2. Создание игры «Сбор шаурмы» 15

Список литературы 16

Заключение 16

Приложения 18

# Введение

В наше время компьютерные игры – невероятно прибыльная отрасль индустрии развлечений. Эта индустрия привлекает много людей со всего света, позволяя им погрузиться в виртуальные миры, развлечься, забыть на время о проблемах или рутине жизни, посмотреть чью-то историю от первого лица, принимая в этом своё участие, для кого-то компьютерные игры – средство заработка, то есть средство для существования.

Разработка игр требует от команды или одного человека умения программировать, умения рисовать, писать музыку, сочинять сюжет.

Как я уже и говорил, для написания игры я решил использовать библиотеку pygame и язык программирования python. Важнейшими преимуществами pygame являются простота написания кода, лёгкость интеграции нового в игру и скорость разработки. Это позволяет новичками постигать азы программирования в нескучной игровой форме.

**Актуальность:** Компьютерные игры – популярная тема на сегодняшний день, потому что игровая индустрия испытывает большой спрос и по сей день. Они являются не только средством развлечения, но и способом заработка, способом улучшить свои навыки программирования, но самое главное – искусством! Сегодня создание игр облегчают движки, которые позволяют программисту не думать о реализации базовых вещей, они позволяют ему творить, меньше думая о технических мелочах.

Также, компьютерные игры широко используются в области виртуальной реальности, в образовательных целях. Таким образом, создание компьютерных игр – это важная сфера нашей жизни, которая выполняет много важных функций.

**Цель проекта:** создание трудной, но интересной и функциональной игры на pygame

**Задачи проекта:**

**-** Познакомиться с историей возникновения игр, узнать какие виды игр существуют

- Познакомиться с этапами разработки компьютерной игры, с библиотекой pygame

- Познакомиться с ООП.

- Создать свою компьютерную игру.

# 1.Теоретическая часть

## 1.1. История развития компьютерных игр

История создания компьютерных игр насчитывает более полувека и охватывает множество этапов, каждый из которых оказал значительное влияние на развитие этой индустрии.

1. **Начало (1950-е - 1970-е годы):**

* **Первые эксперименты:** В конце 1950-х годов учёные начали использовать компьютеры для создания простых игр. Например, 1958 года игра "Tennis for Two", созданная Уильямом Хинтоном, и "Spacewar!", созданная в 1962 году Стэнфордскими студентами.
* **Появление аркадных игр:** В 1971 году вышла первая аркадная игра "Computer Space", а в 1972 году "Pong" от Atari стала одной из первых коммерчески успешных игр.

2. **Эра домашних консолей (1970-е - 1980-е годы):**

* **Первые игровые приставки:** В 1977 году на рынок вышла Atari 2600, одна из первых популярных домашних игровых консолей.
* **Развитие жанров:** В это время стали популярными платформеры, приключенческие игры и текстовые RPG (например, "Zork").
* **Кризис 1983 года:** Перенасыщение рынка и низкое качество игр привели к падению интереса к видеоиграм, особенно в США.

3. **Возрождение и развитие (1985 - 1990-е годы):**

* **Новая волна:** В 1985 году Nintendo выпустила NES (Nintendo Entertainment System), что положило начало новой эре в играх.
* **Развитие жанров:** Появление множества классических игр, таких как "Super Mario Bros.", "The Legend of Zelda" и "Metroid".
* **Первые 3D графики:** В конце 1980-х начались эксперименты с 3D-графикой, отмеченные играми как "Wolfenstein 3D" (1992) и "Doom" (1993).

4. **Эра 3D (1990-е - начало 2000-х годов):**

* **Технологический прогресс:** Появление первых 3D-игр и игровых систем, таких как Sony PlayStation (1994) и Nintendo 64 (1996).
* **Развитие онлайн-игр:** Появление многопользовательских онлайн-игр, таких как "Ultima Online" и "EverQuest".
* **Разнообразие жанров:** В это время становились популярными боевики от первого лица, RPG и стратегии в реальном времени.

5. **Современные игры (2000-е - настоящее время):**

* **Расширение в новые медиа:** Развитие мобильных игр, социальных и казуальных игр (например, "Angry Birds", "Candy Crush").
* **Продолжение развития технологий:** Улучшение графики, реализма и взаимодействия в играх, переход на бесплатную модель с внутриигровыми покупками.
* **Киберспорт и культурное признание:** Игры становятся значимой частью культуры, с ростом киберспорта и признанием игр как вида искусства.

6. **Будущее игр:**

* **Новые платформы:** Развитие технологий, таких как виртуальная и дополненная реальность, искусственный интеллект.

## 1.2. Жанры компьютерных игр

# Компьютерные игры могут быть классифицированы по множеству жанров, каждый из которых имеет свои уникальные характеристики и привлекает определённую аудиторию. Вот краткий обзор наиболее популярных жанров:

# 1. Экшен (Action): Игры, основанные на быстром темпе, акцентирующие внимание на физической активности игрока. К ним относятся платформеры, шутеры и боевые игры.

# 2. Приключение (Adventure): Игры, сосредотачивающиеся на исследовании и решении головоломок. Эти игры часто имеют большую сюжетную составляющую и могут включать элементы ролевых игр.

# 3. Ролевые игры (RPG): Игры, в которых игроки берут на себя роли персонажей в вымышленном мире. RPG могут включать элементы стратегии, исследования, и боевую систему. Поджанры включают JRPG и RPG с открытым миром.

# 4. Стратегия (Strategy): Игры, где игроки должны принимать тактические решения для достижения целей. Это может быть как пошаговая стратегия (Turn-Based), так и реального времени (RTS).

# 5. Симуляторы (Simulation): Игры, имитирующие реальные процессы или действия. Это может быть всё, от управления самолётом до жизни в виртуальных мирах (например, "The Sims").

# 6. Спортивные игры (Sports): Игры, посвященные соревнованиям в различных видах спорта. Это могут быть как реалистичные симуляторы, так и аркадные версии.

# 7. Головоломки (Puzzle): Игры, которые требуют от игроков логического мышления и решения задач. Это могут быть как простые игры на совпадение, так и сложные логические головоломки.

# 8. Хоррор (Horror): Жанр, разработанный для создания чувства страха и напряжения у игроков, часто с использованием темных сюжетов и зловещей атмосферы.

# 9. ММО (MMO): Массовые многопользовательские онлайн-игры, позволяющие тысячам игроков взаимодействовать в общем игровом мире.

# 10. Экшен-RPG (Action RPG): Комбинирует элементы ролевых игр и экшена, позволяя игрокам развивать персонажа, управляя боями в реальном времени.

# 11. Тайм-менеджмент (Time Management): Игры, в которых игроки должны эффективно управлять временем и ресурсами, часто в контексте распределения задач, например, в бизнес-симуляторах.

# 12. Квесты (Quest): Игры, главным образом основанные на выполнении различных заданий и квестов, зачастую с акцентом на сюжет и взаимодействие с персонажами.

13. Платформер — это жанр видеоигр, в котором игрок управляет персонажем, перемещающимся по уровню, состоящему из платформ и других объектов. Игрок преодолевает препятствия и собирает награды.

**И многие другие жанры…**

# Каждый жанр имеет свои особенности и обеспечивает уникальный игровой опыт, что позволяет игрокам находить что-то по своему вкусу и интересам.

## 1.3. Язык программирования Python

**Python** — это **высокоуровневый язык программирования(создан в конце 1980х голландским программистом Гвидо Ван Россумом)**, отличающийся эффективностью, простотой и универсальностью использования.

Он широко применяется в разработке веб-приложений и прикладного программного обеспечения, а также в машинном обучении и обработке больших данных.

**Некоторые особенности Python:**

* **Интерпретируемость**.  Код на нём выполняется построчно, в режиме реального времени. Это позволяет быстро исправлять и проверять код без необходимости компиляции.
* **Динамическая типизация**.  Тип переменной определяется автоматически, во время выполнения кода. Это упрощает процесс программирования и делает его гибким при работе с данными различного типа.
* **Объектно-ориентированность**. Написанная на Python программа представляет собой совокупность объектов, каждому из которых присвоены определённый класс и место в иерархии.

**Плюсы**:

1. Легкость обучения: Python является относительно простым языком, который легко освоить, даже для начинающих.

2. Большой сообщество: Python имеет огромное сообщество разработчиков, что означает, что есть много ресурсов для изучения и решения проблем.

3. Библиотеки и модули: Python имеет богатый набор библиотек и модулей, которые обеспечивают поддержку различных функций, таких как работа с файлами, базами данных, веб-разработка и т. д.

4. Высокая скорость разработки: Python позволяет быстро писать и тестировать код, что упрощает процесс разработки.

5. Кроссплатформенность: Python можно использовать на различных платформах, включая Windows, macOS и Linux.

6. Многоязычность: Python можно использовать для работы с различными языками, такими как Java, C++, C# и т. д.

**Минусы**:

1. Время исполнения: Несмотря на то, что Python быстро развивается, он все еще медленнее некоторых других языков, таких как C++ или Java.

2. Скорость выполнения: Python может быть медленнее других языков в некоторых случаях, особенно при работе с большими объемами данных.

3. Низкая безопасность: Python не имеет многих механизмов безопасности, которые присутствуют в других языках, таких как Java или C#.

4. Низкая параллелизация: Python не поддерживает параллелизацию процессов так же хорошо, как некоторые другие языки.

5. Низкая поддержка реального времени: Python не поддерживает работу в режиме реального времени так же хорошо, как некоторые другие языки.

6. Низкая поддержка шифрования: Python не имеет built-in поддержки шифрования данных.

7. Низкая поддержка безопасности данных: Python не имеет built-in поддержки безопасности данных.

В целом, язык Python является мощным и гибким инструментом для разработки программного обеспечения. Его плюсы и минусы зависят от конкретного проекта и требований, поэтому рекомендуется изучить эти факторы перед выбором Python для своих разработок.

# 

# 1.4 Объектно-ориентированное программирование

Объектно-ориентированное программирование (ООП) – это парадигма программирования, основанная на концепции "объектов", которые могут содержать данные и код: данные в виде полей (атрибутов), а код в виде процедур (методов).

**Основные концепции ООП:**

1. **Классы и объекты:**

**Класс** – это шаблон или схема для создания объектов. Он определяет атрибуты и методы, которые будут у объектов этого класса.

**Объект** – это экземпляр класса, который содержит конкретные данные. Например, можно создать класс Car и создать объекты my\_car, your\_car на основе этого класса.

class Car:

def \_\_init\_\_(self, make, model):

self.make = make

self.model = model

my\_car = Car("Toyota", "Corolla")

**Инка**работу с объектами.

1. Инкапсуляция.

Инкапсуляция подразумевает скрытие внутреннего состояния объекта и предоставление доступа к нему только через методы. Это помогает защитить данные и

class BankAccount:

def \_\_init\_\_(self, balance=0):

self.\_\_balance = balance # Приватный атрибут

def deposit(self, amount):

self.\_\_balance += amount

def get\_balance(self):

return self.\_\_balance упрощает

3.Наследование**:**

* + Наследование позволяет создавать новый класс на основе существующего. Новый класс (дочерний) наследует свойства и методы базового класса (родительского).
  + Это позволяет повторно использовать код и создавать иерархии классов.

class Vehicle:

def start\_engine(self):

print("Engine started")

class Car(Vehicle):

def drive(self):

print("Car is driving")

* 1. **Полиморфизм:**
* Полиморфизм позволяет объектам разных классов предоставлять один и тот же интерфейс. Это означает, что можно использовать общий интерфейс для разных типов объектов, что упрощает код.
* class Dog:
* def speak(self):
* return "Woof!"
* class Cat:
* def speak(self):
* return "Meow!"
* def animal\_sound(animal):
* print(animal.speak())
* dog = Dog()
* cat = Cat()
* animal\_sound(dog) # "Woof!"
* animal\_sound(cat) # "Meow!"

**Преимущества ООП:**

* **Организация кода:** ООП помогает структурировать программный код, делая его более понятным и поддерживаемым.
* **Повторное использование:** Классы и наследование позволяют избежать дублирования кода.
* **Упрощение сложных систем:** ООП помогает моделировать и управлять более сложными системами и процессами.

Таким образом, ООП в Python предоставляет мощный механизм для создания и управления сложными программами, упрощая процесс разработки и повышая его эффективность.

# 2. Практическая часть

## 2.1. Выбор темы для игры

Я не долго думал над тем, какой проект мне выбрать и какую тему, потому что с программированием я уже связан последние 2 года. Я решил написать простую в техническом плане, но трудную для прохождения игру. Я захотел создать игру, которую смогу показать друзьям и другим людям, чтобы они смогли посмеяться с некоторых вещей в этой игре. Я создал игру «Shaurm Cubeater», где «Cubeeater» образуется от двух английских слов «Cube» и «eater», что дословно означает «Поедатель шаурмы».

В моей 2д игре игра идёт от лица квадрата, скин(раскрас) которого можно поменять в настройках. В игре всего 3 довольно сложных уровня, в которых нужно будет проходить препятствия в лице шипов, двигающихся врагов, нужно будет преодолевать рвы из воды, прыгать по платформам, взбираться на холмы, покорять пещеры и не забывать собирать шаурму, которая делает игрока сильнее.

Код я писал на языке программирования python на псевдо-движке pygame, скины и текстуры блоков я рисовал в Adobe Photoshop, музыку скачал из интернета из открытых источников.

# Заключение

Когда я закончил проект, я понял, что написать игру – не так уж сложно, даже если ты совсем новичок в программировании. Нужно лишь иметь немного терпения и страсть к этому делу.

Pygame – неплохая библиотека, но в ней нет многого, что есть в других движках по типу Unity, Godot и других, из-за чего мне пришлось многое создавать самому. Разработка компьютерной игры – отличная возможность начать «вливаться» в IT, а python справляется с этой целью как никто лучше. Создание компьютерной игры прокачивает не только навык создавания компьютерных игр, но и многих других качеств: умение творчески размышлять, терпение, усидчивость, стремление учиться новому.

Итак, я потратил не очень много времени, но у меня получилась довольно интересная игра, учитывая, что я не профессиональный gamedev разработчик. Создание игры подарило мне бесценный опыт, который точно пригодится мне не только при создании других игр, но и в других областях.

Я собираюсь дальше улучшать свои навыки создания игр: я планирую создавать игры более сложные технически, более интересные игры.

# Список литературы

1. <https://www.youtube.com/watch?v=i6xMBig-pP4&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5>

# Приложения

**Приложение 1:**

**from sys import exit as win\_exit**

**from random import randint, choice**

**import pygame**

**#Создание классов и функций**

**class Player: # класс игрока**

**class Monster: # класс монстр**

**class Star: # класс звезды**

**class Background: # класс заднего фона**

**class Snowflake: # класс снежинки**

**class Shaurma: # класс шаурмы**

**class Button: # класс кнопки**

**class Label: #класс текста**

**class MusicPlayer: # класс для фоновой музыки**

**def set\_coords\_condition():**

**def set\_skin(path): # функция, меняющая скин игрока**

**def load\_map(filename): #считывание карты из текстового файла и присваивание в переменную**

**game\_map = []**

**with open(filename, 'r') as file:**

**for line in file:**

**game\_map.append(list(line.strip()))**

**return game\_map**

**#инициализация игры, загрузка игровых карт, первоначальная настройка**

**pygame.init()**

**pygame.mixer.init()**

**pygame.display.set\_caption("Shaurm Cubeater")**

**pygame.display.set\_icon(pygame.image.load('images/skins/eln.png'))**

**music\_player = MusicPlayer()**

**BG = Background(((81, 120, 219))) #создание экземляра класса задний фон, задает изначальный цвет.**

**map1 = load\_map('maps/map1.txt') # загрузка 1, 2, 3 карты**

**map2 = load\_map('maps/map2.txt')**

**map3 = load\_map('maps/map3.txt')**

**win\_size = [1280, 700]; bg\_color = (81, 120, 219)**

**screen = pygame.display.set\_mode((win\_size[0], win\_size[1]))**

**coords\_condition\_label = False**

**current\_window = 'main\_menu'**

**#Загрузка картинок для создания кнопок, блоков, скинов и прочего.**

**skin = 'images/skins/player.png' # начальный скин**

**bg = pygame.image.load('images/other/polskkorova.jpg').convert(); bg\_rect = bg.get\_rect(topleft=(0, 0))**

**button1 = pygame.image.load('images/other/mybutton.png').convert()**

**блоки = pygame.image……………….**

**# Звуки**

**buttonsound = pygame.mixer.Sound("music/button.ogg"); buttonsound.set\_volume(0.5)**

**win\_sound = pygame.mixer.Sound('music/win.ogg')**

**eating\_sound = pygame.mixer.Sound('music/eating\_shaurma.ogg')**

**def play(game\_map, skin, map\_name): # окно игры**

**def option(): # окно настроек**

**def win\_window(): # окно выигрыша**

**def main(): # главное окно, через которое запускаются остальные окна**

**def level\_selection(): # окно выбора уровня**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main() #запуск всей игры**

**Рассмотрим функцию main():**

**В нём и других функциях, которые создают игровые окна, используется игровой цикл, который повторяется сколько-то раз в секунду(можно указать количество кадров в секунду вручную).**

**def main():**

**global current\_window**

**music\_player.play()**

**while True:**

**if current\_window == "main\_menu":**

**main\_menu()**

**elif current\_window == "options":**

**options()**

**elif current\_window == "level\_selection":**

**level\_selection()**

**elif current\_window == 'win\_window':**

**win\_window()**

**if music\_player.is\_playing:**

**music\_player.stop()**

**else:**

**music\_player.play()**

**for event in pygame.event.get():**

**if event.type == pygame.QUIT:**

**pygame.quit()**

**win\_exit()**

**Во всех игровых окнах до запуска игрового цикла while создаются списки, наполненные шаурмой, кнопками, текстами, снежинками, звездами, или монстрами. Всё это экземпляры разных классов. Затем в игровом цикле они отрисовываются с помощью методов классов, проверяется взаимодействие с клавиатурой, мышкой. В функции play(): обрабатываются нажатия клавиш, из-за чего двигается игрок. С помощью методов классов или встроенного функционала pygame проверяются столкновения между различными объектами, такими как блоки, игрок. Монстры.**

**Приложение 2:**

Игра находится в папке Shaurm Cubeater, запускать нужно Shaurm Cubeater.exe.

Посмотреть полный код или скачать игру можно по ссылке

<https://disk.yandex.ru/d/gq-UmZswXAY8gA>