

**КОЛЛЕДЖ КОСМИЧЕСКОГО МАШИНОСТРОЕНИЯ И ТЕХНОЛОГИЙ**

**Курсовой проект**

**По МДК.01.02 Прикладное программирование**

**СОЗДАНИЕ КОМПЬЮТЕРНОЙ ИГРЫ «ЗМЕЙКА»**

Пояснительная записка

КП.09.02.03.22.01П3

Обучающийся группы П1-19 Антипов А. Д.

Руководитель курсового проекта Кочетков С. С.

Результат защиты\_\_\_\_\_\_\_\_\_\_\_

**Королёв, 2022**



**КОЛЛЕДЖ КОСМИЧЕСКОГО МАШИНОСТРОЕНИЯ И ТЕХНОЛОГИЙ**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | «УТВЕРЖДАЮ»  Заместитель директора по учебной работе  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Антропова  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. | |

**ЗАДАНИЕ**

**на курсовой проект**

МДК 01.02 Прикладное программирование

обучающийся специальности 09.02.03 Программирование в компьютерных системах

группы П1-19, 3 курса **Антипов Артём Дмитриевич**

Тема курсового проекта **«Создание компьютерной игры «Змейка»»**

Утверждена приказом от «26» апреля 2022 г. № 01-05/916

**Содержание проекта:**

Введение.

Глава 1. Анализ различных подходов к разработке компьютерных игр.

Глава 2. Создания компьютерной игры “Змейка”.

Глава 3. Анализ работы компьютерной игры “Змейка”.

Заключение.

Список использованной литературы и интернет - ресурсов.

Приложения.

Презентация.

Срок выполнения курсового проекта 23 июня 2022 года

Председатель цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Эшанов

Руководитель курсового проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.С. Кочетков подпись

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Антипов

подпись студента

Дата выдачи задания «26» апреля 2022 года

**Введение.**

Целью проекта является создание компьютерной игры. Данная компьютерная программа служит для организации игрового процесса между компьютером и пользователем, предоставляя пользователю развлечение и положительные эмоции.

Актуальность темы курсовой работы обусловлена тем, что на момент написание этой работы в ходе исследования было обнаружено что огромное количество людей играют в компьютерные игры. Исходя из этого видно, что данная компьютерная игра будет очень актуальной как геймерам, так и простым людям, которые хотят развлечься.

**Глава 1. Анализ различных подходов к разработке компьютерных игр.**

**1.1 Описание предметной области.**

Разработка компьютерных игр — процесс создания компьютерных игр (видеоигр).

Разработкой видеоигр занимается разработчик, который может быть представлен как одним человеком, так и фирмой. Обычно крупномасштабные коммерческие игры разрабатываются командами разработчиков в пределах компании, специализирующейся на играх для персонального компьютера или консолей. Как правило, разработку финансирует другая, более крупная компания-издатель, которая по окончанию разработки занимается изданием игры и связанными с ним тратами. Реже компании-издатели могут содержать внутренние команды разработчиков, или же компания-разработчик может разрабатывать игры за свой счет и распространять их без участия издателей, например, средствами цифровой дистрибуции так называемые инди-игры.

Разработка наиболее крупнобюджетных игр («AAA-игры») может стоить десятки миллионов долларов США, причем в течение последних десятилетий эти бюджеты непрерывно росли, как и численность команд разработчиков и сроки разработки. Средний бюджет ААА-проекта, как правило, выпускаемых крупнейшими компаниями-издателями, продающихся на физических носителях и нередко входящих в состав известной серии из нескольких игр колеблется от 18 до 24 млн. долл.

Крупнобюджетная игра для двух платформ: Xbox 360 и PlayStation 3 обходилась в 2012 году в среднем в 20 миллионов долларов, и для того, чтобы она окупилась, нужно было продать около двух миллионов копий. Благодаря развитию рынка инди-игр, многие разработчики компьютерных игр получили возможность работать над своими игровыми проектами без финансовых и юридических обязательств перед компаниями-издателями.

Инди-игры — это компьютерные игры, созданные отдельными разработчиками или небольшими коллективами без финансовой поддержки издателя компьютерных игр.

Распространение осуществляется посредством каналов цифровой дистрибуции. Масштаб явлений, связанных с инди-играми, ощутимо возрастает со второй половины 2000-х годов, в основном ввиду развития новых способов онлайн-дистрибуции и средств разработки.

Данный проект относится к инди-разработке и развивается только за счет средств его разработчиков.

Благодаря развитию ПО разработки компьютерных игр, команде разработчиков не требуется тратить несколько лет на разработку игрового движка. Это позволяет сразу же приступать к непосредственной работе над игровым проектом и значительно сокращает время его разработки.

## 1.2 Описание средств разработки.

Данная компьютерная программа была написана на высокоуровневом языке программирования Python с помощью среды программирования PyCharm.

PyCharm — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django. PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

Python — высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

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

Изначально Pygame был написан Питом Шиннерсом. Начиная примерно с 2004/2005 года поддерживается и развивается сообществом свободного программного обеспечения.

Одна из библиотек, предоставляющих доступ к API SDL (существуют и другие). В тоже время дает возможность написания более высокоуровневого кода.

Pygame-приложения могут работать под Android на телефонах и планшетах с использованием подмножества Pygame для Android (pgs4a). На этой платформе поддерживаются звук, вибрация, клавиатура, акселерометр.

**Глава 2. Создания компьютерной игры “Змейка”.**

## 2.1 Постановка задачи.

В данной курсовой работе были поставлены следующие задачи:

1. В программе должно присутствовать меню, в котором можно перейти к самой игре и выйти из неё.

2. Должно быть определённое количество жизней чтобы игра имела смысл.

3. Игрок должен перепрыгивать препятствие (кактусы которые появляются на экране случайным образом) для набора очков. При этом игрок должен "разбиваться" при столкновении с кактусом на игровом поле и терять здоровье.

4. Должна присутствовать возможность просмотра максимальное набранное количество очков за игровую сессию. И вывод результатов на экран.

**2.2 Составление общего алгоритма.**

После того как программа будет запущена, на экране появится главное меню. С главного меню нажав на кнопку “START GAME” можно перейти непосредственно к самой игре или нажав на кнопку “QUIT” выйти из неё. Управление осуществляется при помощи клавиши пробел. Так же если во время игровой сессии нажать на клавишу Escape игра будет приостановлена и для её продолжения необходимо нажать на клавишу Enter. После проигрыша отобразится максимальное количество очков, которые набрал пользователь за сессию, а также у пользователя будет выбор: Начать игру заново или выйти в главное меню. Схематический пример общего алгоритма продемонстрирован на рисунке 1.

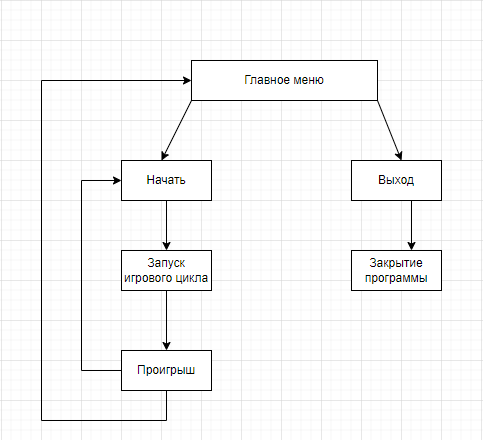


Рисунок 1 — Общий алгоритм программы

## Выделение функциональных частей.

Программа будет состоять из следующих функциональных частей:

* Подпрограмма, которая рисует кнопки и меню.
* Подпрограмма, которая обеспечивает перемещение и создание объектов.
* Подпрограмма, отвечающая за столкновение об объект.
* Подпрограмма, которая отвечает за сохранение результатов текущей сессии.
* Подпрограмма, которая проверяет наличие здоровья и при его отсутствии сообщает об этом.

**2.4 Обоснование алгоритмов отдельных функциональных частей.**

* Функция show\_menu

Цель этой подпрограммы заключается в том, чтобы рисовать кнопки меню и отображение фона меню. Реализацию осуществляется следующим образом: При вызове функции создаётся два объекта при помощи класса Button который принимает параметры отображение по оси XY. Далее запускается цикл, который отображает фон меню и кнопки.

* Функции create\_cactus\_arr, draw\_array и move

Цель этой подпрограммы заключается в том, чтобы создавать объекты и перемещать их. Функция create\_cactus\_arr создаёт список, содержащий в себе один из трёх вариантов изображения кактуса. Функция draw\_array отвечает за отображения списка(кактусов) при помощи функции move которая находится в классе Object и принимает такие параметры как: нахождение объекта по оси XY, его ширина, изображение и скорость перемещение объекта.

* Функция check\_collision

Цель этой подпрограммы заключается в том, чтобы проверять находится ли сейчас игрок по оси X в объекте, а также если игрок делает прыжок, то проверка будет совершаться не только по оси X, но и по оси Y. Если игрок попадёт в кактус, то он сразу же пропадёт, но количество здоровье игрока уменьшится на одну единицу. При попадании в кактус три раза количество здоровье будет равно нулю что приведёт к проигрышу.

* Функция count\_score

Цель этой подпрограммы заключается в том, чтобы проверять перепрыгнул ли игрок кактус и если он это сделал и кактус находится за спиной игрока, то количество очков увеличивается на один.

**Глава 3. Анализ работы компьютерной игры “Змейка”.**

**3.1 Тест игрового меню.**

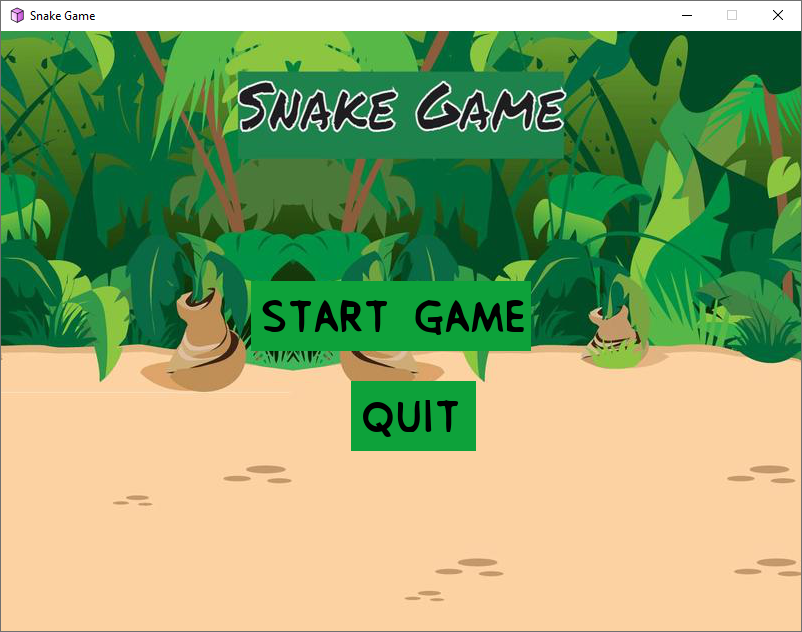
При запуске программы пользователя встречает игровое меню, на котором изображён фон, название игры, кнопка начала игры и кнопка завершения её. Пример игрового меню изображён на рисунке 2. 

Рисунок 2 — Игровое меню

При наведение курсора на одну из двух кнопок цвет фона кнопки поменяется на более светлый цвет и если убрать курсор с кнопки то цвет фона вернётся обратно. Более подробный пример изображён на рисунке 3.

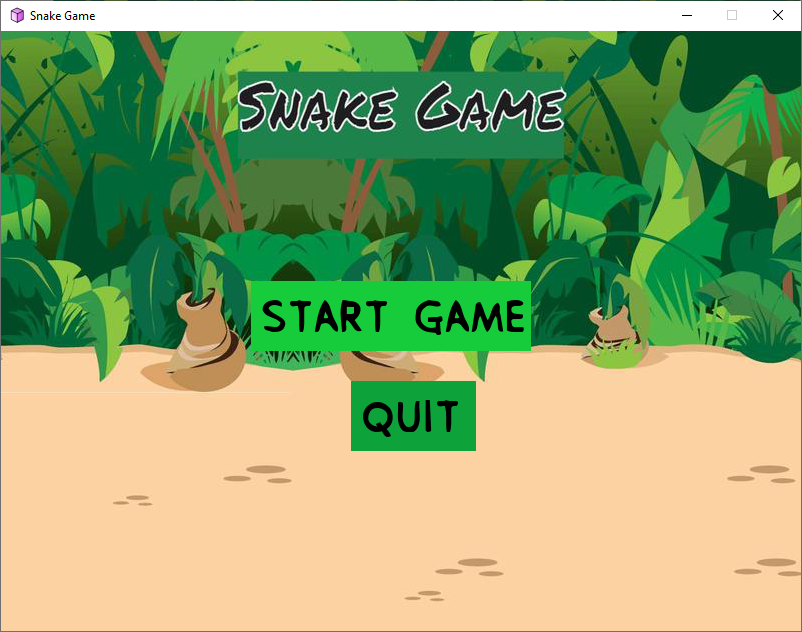


Рисунок 3 — Подстветка кнопки

**3.2 Тест игрового процесса.**

При нажатии на кнопку “START GAME” запуститься игровой процесс на котором будет изображёно: фон игры, персонаж, здоровье персонажа, очки набраные пользователем, облака, камни и кактусы. Более подробный пример изображён на рисунке 4.

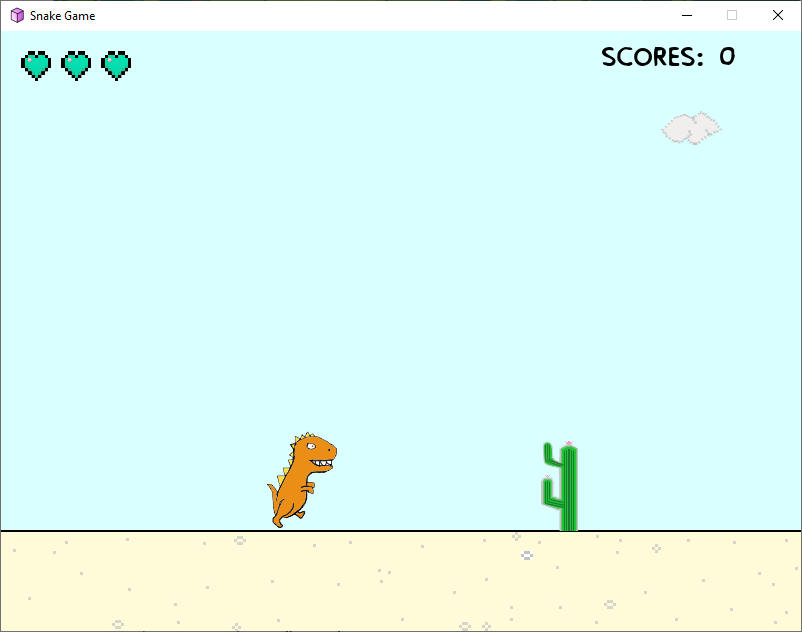


Рисунок 4 — Игровой процесс

При нажатие на пробел кординаты Y персонажа будут менятся имитируя прыжок. Более подробный пример изображён на рисунке 5.

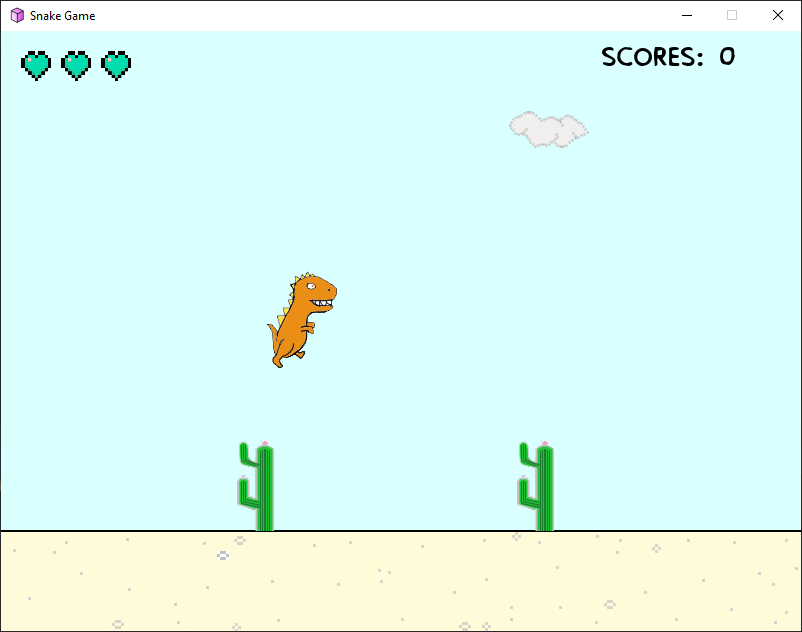


Рисунок 5 — Прыжок

Когда персонаж попадает в кактус его здоровье уменьшается на одно при этом кактуст в который попал персонаж пропадает и количество отображаемых сердец уменьшается на 1. Более подробный пример изображён на рисунке 6 и на рисунке 7.



Рисунок 6 — Перед попадания в кактус



Рисунок 7 — После попадания в кактус

Когда количество здоровья персонажа будет равна нулю игровой процесс закончится и игра предложит возобновить его или же вернуться в главное меню. Более подробный пример изображён на рисунке 8.



Рисунок 8 — Поражение

Для продолжения игры необходимо нажать клавишу Enter. Для выхода в главное меню необходимо нажать клавишу Escape.

Когда персонаж перепрыгивает кактус количество очков пользователя увеличивается на 1. Более подробный пример изображён на рисунке 9.

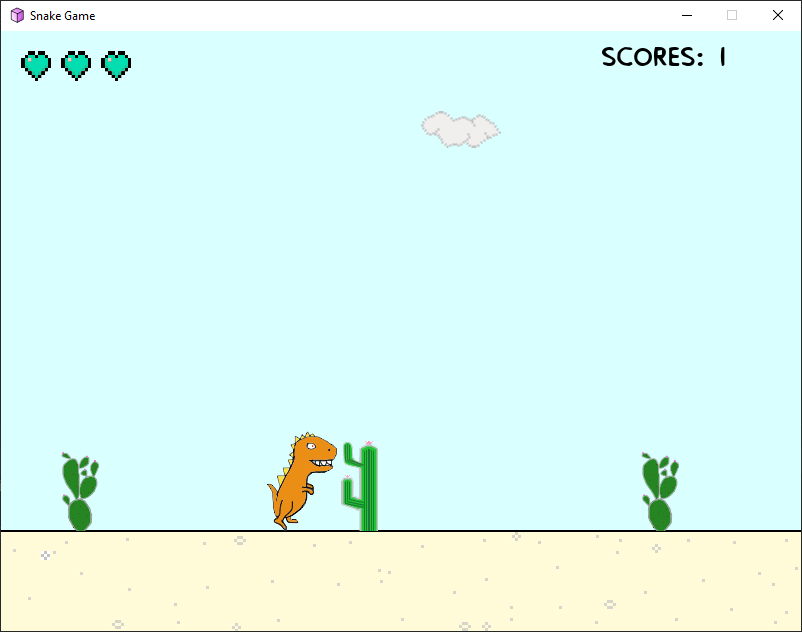


Рисунок 9 — Набор очков

Каждый раз при наборе пяти очков скорость игрового процесса будет увеличиваться. Для демонстрации примера скорость персонажа была выведена в консоли до того как было набрано пять очков и после. Более подробный пример изображён на рисунке 10 и на рисунке 11.



Рисунок 10 — Начальная скорость



Рисунок 11 — Увеличинная скорость

После поражения так же отображается максимальное количество очков набранное за текущую сессию. Более подробный пример изображён на рисунке 12.



Рисунок 12 — Максимальное количество очков

Также в игре присутсвует остановка игрового процесса. Если во время игрового процесса нажать на клавишу Escape игра приостановится. Для возобновления игрового процесса необходимо нажать клавишу Enter. Более подробный пример изображён на рисунке 13.



Рисунок 13 — Пауза

**3.3 Тест анимаций компьютерной игры.**

Для того чтобы компьютерная игра была более реалестичной были добавлены анимации. Первая анимация которая находится в компьютерной игре это изменение отображаемого персонажа. Для создания этой анимации было нарисовано четыре похожих картинок персонажа и загружены в проект. Далее при помощи цикла for отображение персонажа каждый раз меняется по порядку для того чтобы создать иллюзию движения персонажа. Более подробный пример изображён на рисунке 14 и на рисунке 15.

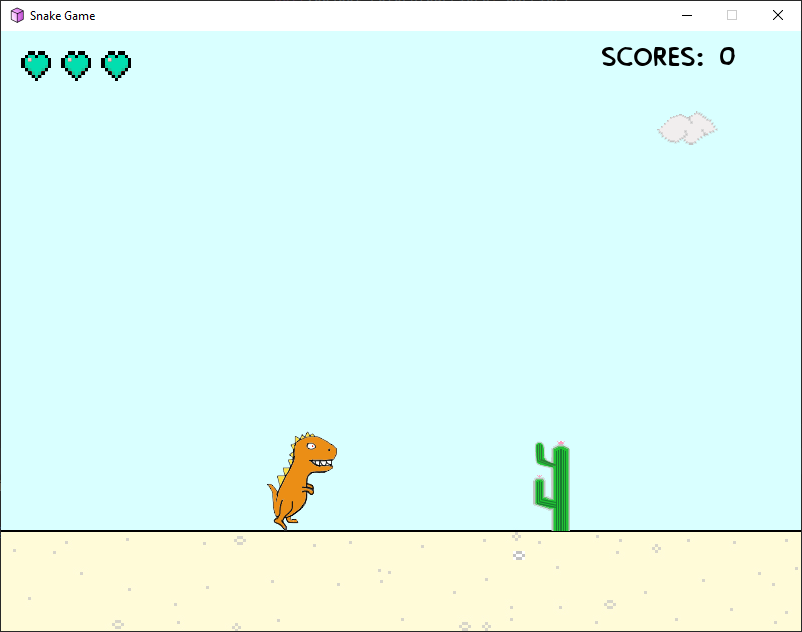


Рисунок 14 — Движение персонажа



Рисунок 15 — Движение персонажа

Так же была добавлена анимация облак. Для того чтобы создать эту анимацию было нарисовано два облака. Далее случайным образом выбирается одно облако и отображается на игровом экране. Движение осуществляется по оси X при помощи функции move. Более подробный пример изображён на рисунке 16 и на рисунке 17.



Рисунок 16 — Движение облака



Рисунок 17 — Движение облака

Последний анимацией является перемещение камней. Для создания этой анимацией было нарисовано два изображения камня. Так же как и обака выбирается случайное изображение камня и отображается на игровом экране. Движение осуществляется по оси X при помощи функции move. Более подробный пример изображён на рисунке 18 и на рисунке 19.

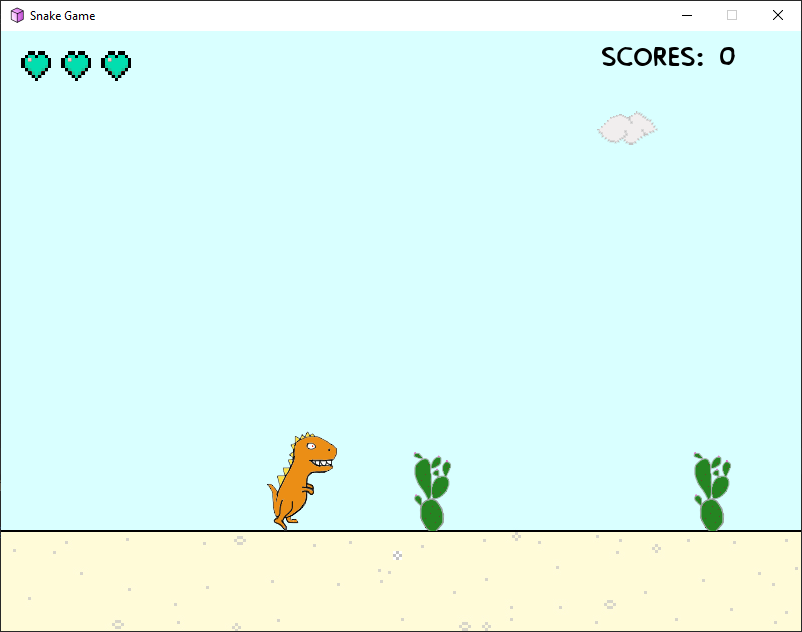


Рисунок 18 — Движение камня

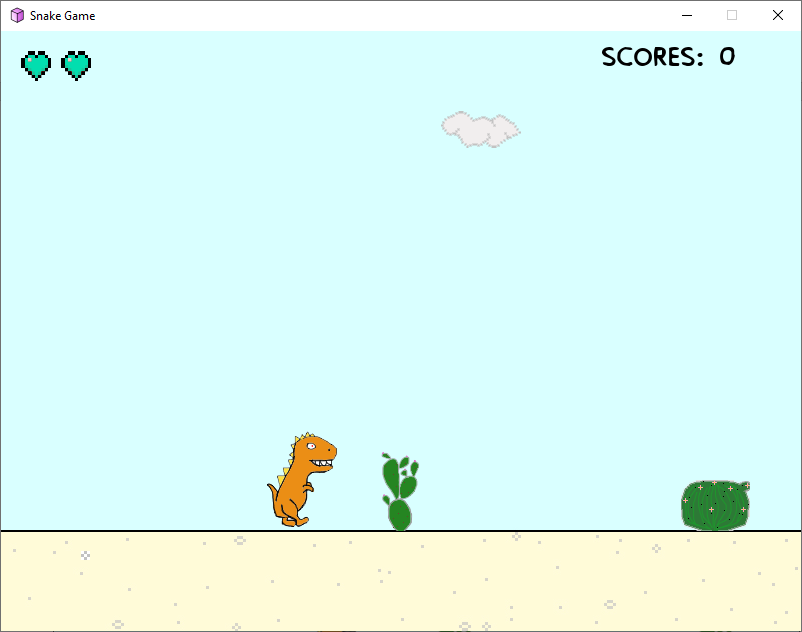


Рисунок 19 — Движение камня

**Заключение**

По итогу разработки курсового проекта было разработано:

* Игровое меню с анимированными кнопками;
* Игровые объекты;
* Игровое поле;
* Количество здоровье;
* Текущие количество набранное очков;
* Максимальное количество набранное очков за текущую сессию;
* Анимация игровых объектов;
* Столкновение с объектами;
* Система поражения;
* Приостановка игрового процесса;

Изначальные цели и задачи проекта выполнены.

**Список использованной литературы и интернет – ресурсов.**

**Законодательные и нормативные акты:**

1. ГОСТ 7.1. – 2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
2. ГОСТ 7.32 – 2017 Система стандартов по информации, библиотечному и издательскому делу (с поправками) – М.: ИПК Издательство стандартов, 2017. – 21 с.
3. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

## Учебная и научная литература:

1. Эрик Мэтиз. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2020. — 512 с.

## Интернет-документы:

1. Stack Overflow – [Электронный ресурс]. – Режим доступа: <https://stackoverflow.com/>
2. PyGame Library – <https://www.pygame.org/>
3. GitHub - <https://github.com>.

**Приложения**

**Приложение 1. Файл main.py**

from Play\_Game.Game import \*  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 new\_game = Play()  
 new\_game.show\_menu()  
  
 pygame.quit()

**Приложение 2. Файл Button.py**

from Play\_Game.effects import \*  
  
class Button:  
 def \_\_init\_\_(self, width, height):  
 self.width = width  
 self.height = height  
 self.inactive\_color = (13, 162, 58)  
 self.active\_color = (23, 204, 58)  
  
 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\_color, (x, y, self.width, self.height))  
  
 if action is not None and click[0] == 1:  
 if action == quit:  
 pygame.quit()  
 quit()  
 else:  
 action()  
  
 else:  
 pygame.draw.rect(display, self.inactive\_color, (x, y, self.width, self.height))  
  
 print\_text(message=message, x=x + 10, y=x + 10, font\_size=font\_size)

**Приложение 3. Файл effects.py**

from Play\_Game.parameters import \*  
  
def print\_text(message, x, y, font\_color=(0, 0, 0), font\_type='Play\_Game/Effects/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))

**Приложение 4. Файл images.py**

import pygame  
  
pygame.init()  
  
icon = pygame.image.load('Play\_Game/Backgrounds/icon.png')  
  
cactus\_img = [pygame.image.load('Play\_Game/Objects/Cactus0.png'), pygame.image.load('Play\_Game/Objects/Cactus1.png'), pygame.image.load(  
 'Play\_Game/Objects/Cactus2.png')]  
  
stone\_img = [pygame.image.load('Play\_Game/Objects/Stone0.png'), pygame.image.load('Play\_Game/Objects/Stone1.png')]  
cloud\_img = [pygame.image.load('Play\_Game/Objects/Cloud0.png'), pygame.image.load('Play\_Game/Objects/Cloud1.png')]  
  
dino\_img = [pygame.image.load('Play\_Game/Dino/Dino0.png'), pygame.image.load('Play\_Game/Dino/Dino1.png'), pygame.image.load(  
 'Play\_Game/Dino/Dino2.png'),  
 pygame.image.load('Play\_Game/Dino/Dino3.png'), pygame.image.load('Play\_Game/Dino/Dino4.png')]  
  
health\_img = pygame.image.load('Play\_Game/Effects/heart.png')  
health\_img = pygame.transform.scale(health\_img, (30, 30))  
  
land = pygame.image.load('Play\_Game/Backgrounds/Land.jpg')  
menu\_bckgr = pygame.image.load('Play\_Game/Backgrounds/Menu.jpg')

**Приложение 5. Файл Object.py**

from Play\_Game.parameters import \*  
import random  
  
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:  
 self.x = display\_width + 100 + random.randrange(-80, 60)  
 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))

**Приложение 6. Файл parameters.py**

import pygame  
  
display\_width = 800  
display\_height = 600  
  
display = pygame.display.set\_mode((display\_width, display\_height))  
  
usr\_width = 60  
usr\_height = 100  
usr\_x = display\_width // 3  
usr\_y = display\_height - usr\_height - 100  
  
clock = pygame.time.Clock()

**Приложение 7. Файл Game.py**

from Play\_Game.Button import \*  
from Play\_Game.Object import \*  
from Play\_Game.effects import \*  
from Play\_Game.images import \*  
import Play\_Game.parameters as p  
import random  
  
class Play:  
 def \_\_init\_\_(self):  
 pygame.display.set\_caption('Snake Game')  
 pygame.display.set\_icon(icon)  
  
 self.cactus\_options = [69, 449, 37, 410, 40, 420]  
 self.img\_counter = 0  
 self.health = 3  
 self.make\_jump = False  
 self.jump\_counter = 30  
 self.scores = 0  
 self.max\_scores = 0  
 self.max\_above = 0  
 self.speed = 70  
 self.raise\_score = 5  
  
 def show\_menu(self):  
 show = True  
  
 start\_btn = Button(280, 70)  
 quit\_btn = Button(125, 70)  
  
 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(250, 250, 'Start game', self.start\_game, 50)  
 quit\_btn.draw(350, 350, 'Quit ', quit, 50)  
  
 pygame.display.update()  
 clock.tick(60)  
  
 def start\_game(self):  
 while self.game\_cycle():  
 self.scores = 0  
 self.make\_jump = False  
 self.jump\_counter = 30  
 p.usr\_y = p.display\_height - p.usr\_height - 100  
 self.health = 3  
  
 def game\_cycle(self):  
 game = True  
 cactus\_arr = []  
 self.create\_cactus\_arr(cactus\_arr)  
  
 stone, cloud = self.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]:  
 self.make\_jump = True  
 if keys[pygame.K\_ESCAPE]:  
 self.pause()  
  
 if self.make\_jump:  
 self.jump()  
  
 self.count\_scores(cactus\_arr)  
  
 display.blit(land, (0, 0))  
 print\_text('Scores: ' + str(self.scores), 600, 10)  
  
 self.draw\_array(cactus\_arr)  
 self.move\_objects(stone, cloud)  
  
 self.draw\_dino()  
  
 if self.check\_collision(cactus\_arr):  
 game = False  
  
 self.show\_health()  
  
 if self.scores == self.raise\_score:  
 self.speed += 10  
 self.raise\_score += 5  
  
 pygame.display.update()  
 clock.tick(self.speed)  
  
 return self.game\_over()  
  
 def jump(self):  
 if self.jump\_counter >= -30:  
 p.usr\_y -= self.jump\_counter / 2.5  
 self.jump\_counter -= 1  
 else:  
 self.jump\_counter = 30  
 self.make\_jump = False  
  
 def create\_cactus\_arr(self, array):  
 choice = random.randrange(0, 3)  
 img = cactus\_img[choice]  
 width = self.cactus\_options[choice \* 2]  
 height = self.cactus\_options[choice \* 2 + 1]  
 array.append(Object(p.display\_width + 20, height, width, img, 4))  
  
 choice = random.randrange(0, 3)  
 img = cactus\_img[choice]  
 width = self.cactus\_options[choice \* 2]  
 height = self.cactus\_options[choice \* 2 + 1]  
 array.append(Object(p.display\_width + 300, height, width, img, 4))  
  
 choice = random.randrange(0, 3)  
 img = cactus\_img[choice]  
 width = self.cactus\_options[choice \* 2]  
 height = self.cactus\_options[choice \* 2 + 1]  
 array.append(Object(p.display\_width + 600, height, width, img, 4))  
  
 @staticmethod  
 def find\_radius(array):  
 maximum = max(array[0].x, array[1].x, array[2].x)  
  
 if maximum < p.display\_width:  
 radius = p.display\_width  
 if radius - maximum < 50:  
 radius += 280  
 else:  
 radius = maximum  
  
 choice = random.randrange(0, 5)  
 if choice == 0:  
 radius += random.randrange(10, 15)  
 else:  
 radius += random.randrange(250, 400)  
  
 return radius  
  
 def draw\_array(self, array):  
 for cactus in array:  
 check = cactus.move()  
 if not check:  
 self.object\_return(array, cactus)  
  
 @staticmethod  
 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(p.display\_width, p.display\_height - 80, 10, img\_of\_stone, 4)  
 cloud = Object(p.display\_width, 80, 70, img\_of\_cloud, 2)  
  
 return stone, cloud  
  
 @staticmethod  
 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(p.display\_width, 500 + random.randrange(10, 80), 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(p.display\_width, random.randrange(10, 200), stone.width, img\_of\_cloud)  
  
 def draw\_dino(self):  
 if self.img\_counter == 25:  
 self.img\_counter = 0  
  
 display.blit(dino\_img[self.img\_counter // 5], (p.usr\_x, p.usr\_y))  
 self.img\_counter += 1  
  
 @staticmethod  
 def pause():  
 paused = True  
 while paused:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 quit()  
  
 print\_text('Game paused, press Enter to continue', 160, 300)  
  
 keys = pygame.key.get\_pressed()  
 if keys[pygame.K\_RETURN]:  
 paused = False  
  
 pygame.display.update()  
 clock.tick(15)  
  
 def object\_return(self, objects, obj):  
 radius = self.find\_radius(objects)  
  
 choice = random.randrange(0, 3)  
 img = cactus\_img[choice]  
 width = self.cactus\_options[choice \* 2]  
 height = self.cactus\_options[choice \* 2 + 1]  
  
 obj.return\_self(radius, height, width, img)  
  
 def check\_collision(self, barriers):  
 for barrier in barriers:  
 if barrier.y == 449:  
 if not self.make\_jump:  
 if barrier.x <= p.usr\_x + p.usr\_width - 35 <= barrier.x + barrier.width:  
 if self.check\_health():  
 self.object\_return(barriers, barrier)  
 return False  
 else:  
 return True  
  
 elif self.jump\_counter >= 0:  
 if p.usr\_y + p.usr\_height - 5 >= barrier.y:  
 if barrier.x <= p.usr\_x + p.usr\_width - 35 <= barrier.x + barrier.width:  
 if self.check\_health():  
 self.object\_return(barriers, barrier)  
 return False  
 else:  
 return True  
 else:  
 if p.usr\_y + p.usr\_height - 10 >= barrier.y:  
 if barrier.x <= p.usr\_x <= barrier.x + barrier.width:  
 if self.check\_health():  
 self.object\_return(barriers, barrier)  
 return False  
 else:  
 return True  
 else:  
 if not self.make\_jump:  
 if barrier.x <= p.usr\_x + p.usr\_width - 5 <= barrier.x + barrier.width:  
 if self.check\_health():  
 self.object\_return(barriers, barrier)  
 return False  
 else:  
 return True  
 elif self.jump\_counter == 10:  
 if p.usr\_y + p.usr\_height - 5 >= barrier.y:  
 if barrier.x <= p.usr\_x + p.usr\_width - 5 <= barrier.x + barrier.width:  
 if self.check\_health():  
 self.object\_return(barriers, barrier)  
 return False  
 else:  
 return True  
 elif self.jump\_counter >= -1:  
 if p.usr\_y + p.usr\_height - 5 >= barrier.y:  
 if barrier.x <= p.usr\_x + p.usr\_width - 35 <= barrier.x + barrier.width:  
 if self.check\_health():  
 self.object\_return(barriers, barrier)  
 return False  
 else:  
 return True  
 else:  
 if p.usr\_y + p.usr\_height - 10 >= barrier.y:  
 if barrier.x <= p.usr\_x <= barrier.x + barrier.width:  
 if self.check\_health():  
 self.object\_return(barriers, barrier)  
 return False  
 else:  
 return True  
  
 return False  
  
 def count\_scores(self, barriers):  
 above\_cactus = 0  
  
 if -20 <= self.jump\_counter < 25:  
 for barrier in barriers:  
 if p.usr\_y + p.usr\_height - 5 <= barrier.y:  
 if barrier.x <= p.usr\_x <= barrier.x + barrier.width:  
 above\_cactus += 1  
 elif barrier.x <= p.usr\_x + p.usr\_width <= barrier.x + barrier.width:  
 above\_cactus += 1  
  
 self.max\_above = max(self.max\_above, above\_cactus)  
 else:  
 if self.jump\_counter == -30:  
 self.scores += self.max\_above  
 self.max\_above = 0  
  
 def game\_over(self):  
 if self.scores > self.max\_scores:  
 self.max\_scores = self.scores  
 stopped = True  
 while stopped:  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 quit()  
  
 print\_text('Game over, press Enter to play game or Esc to exit', 20, 300)  
 print\_text('Max scores: ' + str(self.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)  
  
 def show\_health(self):  
 show = 0  
 x = 20  
 while show != self.health:  
 display.blit(health\_img, (x, 20))  
 x += 40  
 show += 1  
  
 def check\_health(self):  
 self.health -= 1  
 if self.health == 0:  
 return False  
 else:  
 return True