ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В нынешнее время персональные компьютеры, смартфоны и другие современные гаджеты стали неотъемлемой частью жизни большого процента населения России, а информационные технологии продолжают активно развиваться. Одним из наиболее быстрорастущих направлений в ИТ в течение длительного времени остается Гейм Девелопмент (от англ. game development — разработка игры).

Компьютерные игры сегодня находятся в числе наиболее популярных видов досуга. По данным за 2018, в мире насчитывается более 2,3 млрд геймеров. Общий объем игровой индустрии по итогам девяти месяцев 2020 года составил 174,9 миллиарда долларов (на 19,6% выше, чем в 2019-м) и значительно превысил объем киноиндустрии (100 миллиардов долларов) и объем индустрии спорта (75 миллиардов долларов). Эти данные дают понять, что компьютерные игры крайне востребованы в среде самых различных социальных групп, и для большого числа людей они стали частью обыденной жизни, чем можно обосновать актуальность данного проекта.

Цель данной проектной работы заключается в создании компьютерной игры.
Задачи проекта:

1. Проанализировать интернет-источники, тематическую литературу по созданию компьютерных игр;
2. Узнать историю возникновения и развития языка Python;
3. Выбрать жанр, вид и праформу для компьютерной игры;
4. Рассмотреть технологию создания компьютерной игры и ее особенности для языка программирования Python;
5. Изучить требуемые технологии и программы;
6. Разработать собственную игру и описать процесс разработки.

Объектом моего исследования является процесс создания компьютерных игр, а предметом исследования проекта — документация по созданию компьютерных игр на языке Python с библиотекой Pygame.
В процессе выполнения моей проектной работы я собираюсь использовать следующие методы исследования: сбор, анализ, обобщение и систематизация материала из научных, научно-популярных статей о компьютерных играх, просмотр обучающих видеороликов, изучение тематической литературы. Сроки выполнения проекта составляют 6 месяцев. Практическим итогом работы станет компьютерная игра.

ГЛАВА 1

1.1 Сущность языка Python

Python — это интерпретируемый объектно-ориентированный высокоуровневый язык программирования. Python используют в различных целях:

- анализ данных;
- построение web-приложений;
- создании игр;
- работе с базами данных;
- решение бизнес-задач и т.д.

Синтаксис Python минималистичен, четок, последователен и лаконичен, в нем придается большое значение читаемости кода, язык не допускает двоякого написания кода. В то же время в поставку Python входит обширная стандартная библиотека, которая включает в себя большое количество полезных функций. Основной упор в Python делается на скорости написания кода, но при этом по производительности Python относительно медленный язык (по сравнению с C, Go, Java). Python поддерживает модули и пакеты, поощряя модульность и повторное использование кода.

Python — объектно-ориентированный язык и практически все данные являются объектами. Python является языком с полной динамической типизацией и автоматическим управлением памятью. Динамическая типизация означает, что тип переменной определяется только во время исполнения.

Одной из интересных и необычных синтаксических особенностей языка является выделение блоков кода пробельными отступами. Также Python отличается отсутствием описания переменных, поддержкой GUI, автоматическим управлением памяти, высокоуровневой структурой данных, поддержкой многопоточных вычислений с глобальной блокировкой интерпретатора

Язык программирования Python является кроссплатформенным языком, то есть позволяет создавать программы для разных операционных систем.

Python поддерживает практически все существующие ОС. Кроме того, Python имеет достаточно простые средства для интеграции с языками C, C++, Java.

Python относится к числу самых востребованных и популярных языков программирования, он также является одним из наиболее активно развивающийся языков. В Интернете доступно большое количество качественных библиотек по различным предметным областям.

1.2 История языка Python

Задумка по реализации языка появилась в конце 1980-х годов, а создание Python было начато Гвидо ван Россумом (Guido van Rossum) в декабре 1989 года, когда он работал над распределенной операционной системой Амеба (Amoeba) в центре математики и информатики в Нидерландах. Ему требовался расширяемый язык, который бы обеспечил поддержку системных вызовов.

Язык Python был задуман как потомок языка программирования ABC, способный к обработке исключений и взаимодействию с операционной системой Амеба. Большое влияние на язык программирования Python также оказали такие языки, как Модула-3, C, C++, Java, Haskell. В качестве названия он выбрал Python в честь популярного британского комедийного телешоу 1970-х “Летающий цирк Монти-Питона”, а вовсе не по названию змеи. С тех пор Python развивался при поддержке тех организаций, в которых Гвидо работал.

Ван Россум является основным автором Python и продолжал выполнять центральную роль в принятии решений относительно развития языка с момента его создания и вплоть до июля 2018 года.

1.3 Краткая информация о Pygame

Pygame — это библиотека для языка программирования Python, кроссплатформенный набор модулей, предназначенных для написания игр и мультимедиа-приложений. Библиотека в программировании — это набор готовых подпрограмм, функций, классов и объектов, используемых для разработки программного обеспечения. Модуль — это функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом и предназначенный для использования в других программах. Pygame включается в себя инструменты по реализации:

- компьютерной графики и анимации;
- звука (включая музыку);
- обработки ввода и управления (компьютерная мышь, клавиатура, игровые контроллеры и геймпады)

Pygame была создана Питом Шиннерсом (Pete Shinners) и с начала 2000 года является общественным некоммерческим проектом. Библиотека Pygame создана на основе другой мультимедийной библиотеки Simple DirectMedia Layer (SDL), которая обеспечивает легкий доступ к звуковым и визуальным элементам. SDL была написана Сэмом Лантингой (Sam Lantinga), чтобы упростить задачу переноса игр с одной аппаратной платформы или операционной системы на другую.

Pygame портативна и работает почти на всех платформах и операционных системах. Кроме того, библиотека Pygame выделяется своей простой и удобством в использовании, а также минималистичностью и лаконичностью, выраженной в небольшом объеме кода, модульностью — возможностью использовать отдельно необходимые фрагменты кода.

Благодаря открытому исходному коду Pygame продолжает активно разрабатываться и развиваться, совершенствоваться сторонними разработчиками.

ГЛАВА 2

2.1 Общее описание игры

Для создания игры был выбран жанр аркада. Аркада — это жанр компьютерных игр, который характеризуется коротким по времени, но интенсивным игровым процессом. Среди отличительных особенностей жанра можно выделить следующие:

- условная бесконечность игры;
- простой игровой процесс;
- отсутствие явно прослеживаемого сюжета;
- игровой процесс сосредоточен на одном экране;
- упрощенный интерфейс;
- небольшой промежуток времени, необходимый для их прохождения

Задача игрока состоит в управлении персонажем, который непрерывно перемещается между стенами пещеры. В случае столкновения со стенами персонаж погибает, а игра завершается. Управление производится при помощи клавиатуры — клавиши “пробел”, при нажатии на которую персонаж совершает небольшой рывок вверх. При отсутствии рывков персонаж падает из-за гравитации, и игра также завершается. Помимо цели не разбиться, игроку требуется собирать монеты. [Рис. 1]

2.2 Реализация функциональной части и описание основных алгоритмов

Помимо библиотеки Pygame были использованы библиотеки Math, Random, а также отдельно созданный модуль background:

```
import random
import pygame
import math
import background
```

В функции main была запущена библиотека Pygame, создано окно программы и его заголовок, создан объект, чтобы отслеживать время, а также задана кадровая частота:

```
def main():
    pygame.init()
    display = pygame.display.set_mode(displays)
    pygame.display.set_caption("A ray of light")
    clock = pygame.time.Clock()
    FPS = 30
```

В словаре colours были заданы с помощью кортежей цвета фона, стен и персонажа:

```
colours = {
   "bg": (0, 0, 0),
   "walls": (30, 30, 30),
   "player": (255, 255, 255)
}
```

В переменных displays, playSize, playerRectangle, ph, gravity были заданы размеры окна программы, персонажа, высота стен и сила гравитации:

```
displays = [1200, 800]
playSize = [50, 50]
playerRectangle = [
   round(displays[0]/2 - playSize[0]/2),
```

```
  round(displays[1]/2 - playSize[1]/2),
  playSize[0], playSize[1]]
ph = 20
gravity = -0.01
```

В модуле background были созданы два класса — WallsPiece и Cave. Первый класс отвечает за отрисовку и движение стен пещеры:

```
class WallsPiece:
  def __init__(self, rect, colour):
    self.rect = pygame.Rect(rect)
    self.colour = colour

  def move(self, dx, dy):
    self.rect = self.rect.move(dx, dy)

  def draw(self, display):
    pygame.draw.rect(display, self.colour, self.rect)
```

В классе Cave были созданы функции для контроля за столкновениями персонажа со стенами, для движения и обновления фона:

```
def update(self, movement, player):
  self.move(movement)
  self.check_delete()
  self.check_spawn()
  return self.check_wall_hit(player)

def move(self, movement):
  for wall in self.pieces:
    wall.move(movement[0], movement[1])

def check_wall_hit(self, player):
  corners = [False, False,
          False, False]
  for cave_piece in self.pieces[int(self.scale[0]*0.3):int(self.scale[0]*0.6)]:
    if cave_piece.rect.collidepoint(player.rect.topleft):
      corners[0] = True
    if cave_piece.rect.collidepoint(player.rect.topright):
```

```
            corners[1] = True
        if cave_piece.rect.collidepoint(player.rect.bottomleft):
            corners[2] = True
        if cave_piece.rect.collidepoint(player.rect.bottomright):
            corners[3] = True
    if False in corners:
        return True
    return False
```

Персонаж и монеты также были выделены в отдельные классы, которые были прописаны в основной части программы. В классе Player были созданы несколько функций.
В функции __init__ был прописан угол полета (в радианах), скорость полета игрока, длина “хвоста” персонажа:

```
class Player:
    def __init__(self, rect, colour):
        self.rect = pygame.Rect(rect)
        self.vector = [0, 0]
        self.angle = .5  # in pi radians. ie: 2 = 2pi radians. 0 radians is down, increasing anticlockwise.
        self.speed = 20  # hypo of triangle, where x and y for movement are found via trig with self.angle
        self.colour = colour
        self.tail = []
        self.tail_decay = 1
```

Заданы функции по отрисовке и обновлению положения на игровом окне персонажа и “хвоста” персонажа:

```
def draw(self, display):
    pygame.draw.ellipse(display, self.colour, self.rect)

    for rect in self.tail:
        pygame.draw.ellipse(display, self.colour, rect)

def update(self, objects):
    delta_x = self.speed*math.sin(self.angle * math.pi)
    delta_y = self.speed*math.cos(self.angle * math.pi)
```

```
    self.move(0, 0, [])
    self.update_tail(-delta_x, -delta_y)
    return [delta_x, delta_y]

def update_tail(self, dx, dy):
    for rect in self.tail:
        rect.move_ip(dx, dy)
        rect.inflate_ip(-self.tail_decay, -self.tail_decay)
        if rect.width <= self.tail_decay or rect.height <= self.tail_decay:
            self.tail.remove(rect)

    self.tail.append(self.rect.copy())
```

Функция rollover_angle гарантирует, что угол остается в диапазоне от 0 до 2π:

```
def rollover_angle(self):
    if self.angle <= 0:
        self.angle = 2
    elif self.angle >= 2:
        self.angle = 0
```

В классе Coin был задан цвет монет, созданы функции по их отрисовке, движению, проверке на их столкновение с персонажем:

```
class Coin:
    def __init__(self, rect):
        self.rect = pygame.Rect(rect)
        self.colour = (200, 200, 0)

    def move(self, movement):
        self.rect = self.rect.move(movement)

    def check_collide(self, player):
        if self.rect.colliderect(player.rect):
            return True
        return False

    def draw(self, display):
```

```
        pygame.draw.circle(display, self.colour, self.rect.center,
round(self.rect.width/2))
```

В главной функции main был создан игровой цикл while, контролируемый переменной end. В случае, если происходит столкновение или нажатие клавиши “esc”, переменная принимает значение True, а цикл завершается:

```
end = False
while not end:
  for event in pygame.event.get():
     if event.type == pygame.QUIT:
        end = True
     elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_ESCAPE:
           end = True
```

Также в этой функции была прописана проверка на столкновение персонажа со стеной пещеры:

```
hit = bg.update(movement, player)
if hit:
  end = True
```

ЗАКЛЮЧЕНИЕ

При написании проекта была спроектировано и реализовано программное приложение, служащее для организации игрового процесса. В результате разработки была достигнута основная цель проекта — создана компьютерная игра. [Приложение 2] Для этого были разработаны алгоритмы и использовалось программное обеспечение совместно с применением современных технологий программирования. Проведенное тестирование программы показало ее работоспособность.

В ходе выполнения проектной работы были также решены поставленные задачи:

- описана сущность и история языка программирования Python;
- проанализированы и выбраны наиболее актуальные средства разработки;
- изучены технологии по созданию компьютерных игр (в частности, игровая библиотека Python).

Основываясь на итогах проделанной работы, можно сделать вывод, что используемые в проекте методы исследования в полной мере смогли поспособствовать в раскрытии актуальности данной проектной работы и достижении поставленных задач.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ И ИНТЕРНЕТ-РЕСУРСОВ

1. Марк Саммерфилд. Программирование на Python. Подробное руководство — 2009. — с. 13-15.
2. Python 3.10.2 documentation. [Электронный ресурс] — Режим доступа: https://docs.python.org/3/
3. A Brief Timeline of Python. [Электронный ресурс] — Режим доступа: http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html
4. Pygame и разработка игр. [Электронный ресурс] — Режим доступа: https://younglinux.info/pygame/pygame
5. Ричард Рус. Игровой дизайн: теория и практика — 2004. — 698 с.

ПРИЛОЖЕНИЕ 1

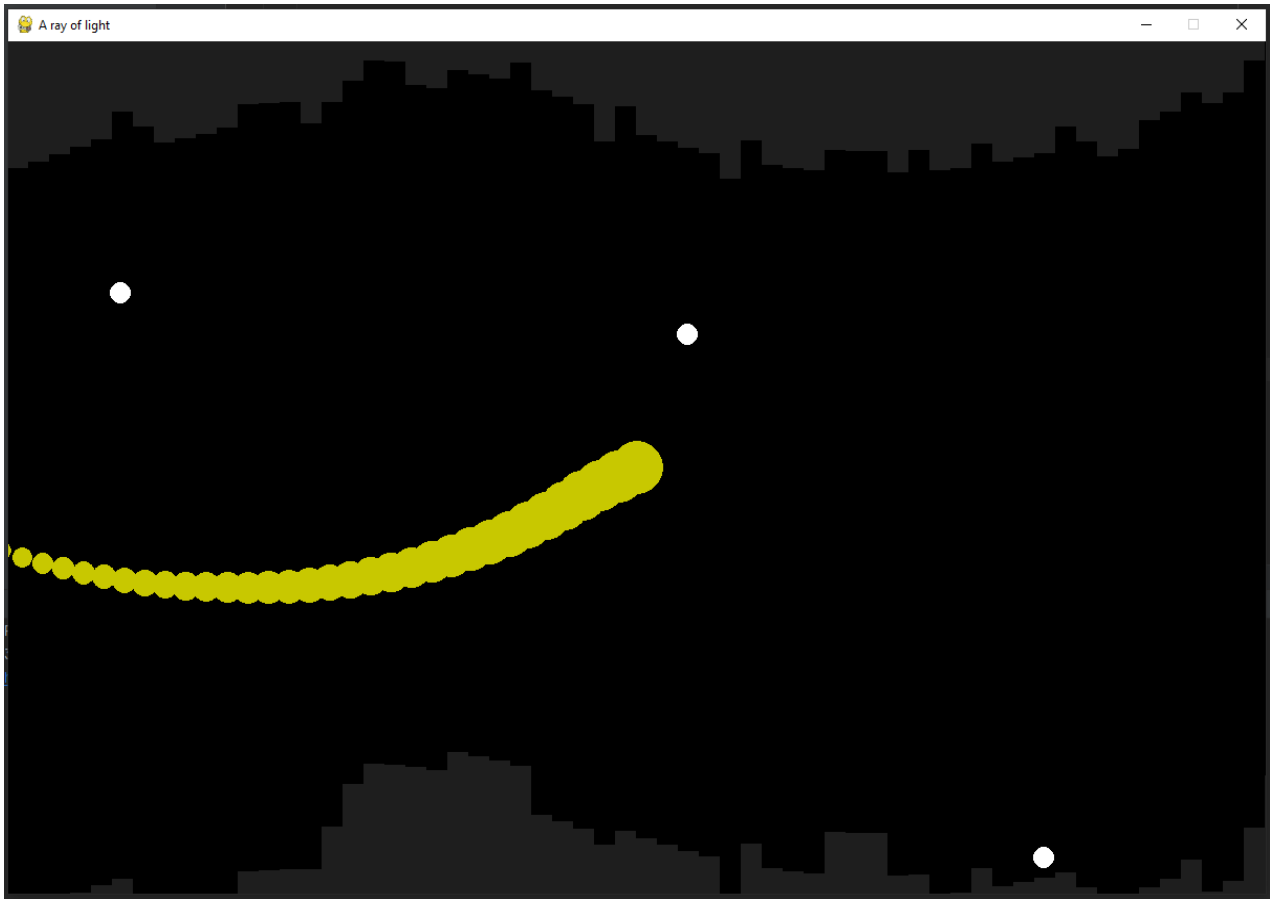

Рис. 1