Муниципальное бюджетное общеобразовательное учреждение Лицей №177 Ново-Савиновского района города Казани

Проектная работа по информатике на тему:

“Игра DoodleJump с использованием библиотеки pygame”

Выполнили:

Ученики 10Г класса

Хайруллин Азат

Капралов Тимур

Проверил:

Учитель информатики

Журкина Алия Ильгизовна

2024

Оглавление

1. Введение
2. Задачи и Актуальность проекта
3. История видеоигр: от научных лабораторий до домашних консолей
4. Техническое задание
5. План разработки игры
6. Описание работы кода – основные компоненты игры
7. Вывод

**Введение**

**Doodle Jump** — Это аркадная игра, в которой игрок управляет персонажем, стремящимся прыгать по платформам, избегая препятствий и собирая бонусы. Проект использует библиотеку Pygame, которая включает в себя все необходимые инструменты для создания 2D игр.

**Актуальность проекта**

Разработка игр является важной частью в сфере программирования, так как она способствует улучшению навыков разработки алгоритмов, работе с графическими библиотеками и улучшению логического мышления. Платформа Pygame предоставляет удобные инструменты для создания 2D-игр, что делает ее популярной среди начинающих и опытных разработчиков.

Проект игры **Doodle Jump** актуален в свете популярности мобильных аркадных игр и предлагает учащимся и начинающим разработчикам уникальную возможность попрактиковаться в создании простых, но увлекательных игр с использованием Pygame.

**История видеоигр: от научных лабораторий до домашних консолей**

Сегодня видеоигры — неотъемлемая часть нашей жизни, но многие не знают, что первые игры появились в научных лабораториях.

* Как все началось

Первой видеоигрой, признанной в истории, была OXO (классические крестики-нолики), созданная в 1952 году британским ученым А.С. Дугласом для демонстрации возможностей компьютеров в рамках его диссертации в Кембриджском университете.

В 1958 году Уильям Хигинботэм создал "Теннис для двоих" — игру на аналоговом компьютере для развлечения посетителей Брукхейвенской лаборатории в Нью-Йорке. В 1962 году Стив Рассел разработал Spacewar!, первую многопользовательскую игру на компьютерах.

* Привет, домашняя консоль!

В 1967 году Ральф Баер создал прототип домашней игровой системы, позднее проданный компании Magnavox, которая в 1972 году выпустила первую домашнюю консоль — Odyssey. Это устройство не получило большой популярности, но стало основой для дальнейшего развития индустрии.

* Что было дальше

Игры продолжали развиваться, и в 1972 году Atari выпустила популярную аркадную игру Pong, а в 1975 году — домашнюю версию. В 1977 году Atari выпустила свою консоль с джойстиками и сменными картриджами, что положило начало второму поколению игровых систем. В это время появляются культовые игры, такие как Space Invaders и Pac-Man.

* Крах видеоигр

В 1983 году индустрия пережила кризис из-за перенасыщения рынка и низкокачественных игр.

* Возрождение видеоигр

В 1985 году Nintendo выпустила консоль NES, которая значительно улучшила качество игр. В 1989 году компания представила портативный Game Boy с игрой Tetris, которая стала мировым хитом и первым игровым устройством, побывавшим в космосе.

Получается, видеоигры, начав с научных экспериментов, стали неотъемлемой частью нашего развлечения и культуры.

**Техническое задание**

**Цель:**  
Разработать игру **Doodle Jump**, в которой игрок управляет персонажем, который прыгает по платформам и стремится получить как можно больше очков, не упав с экрана.

**Основные требования:**

1. **Игровая сцена:**  
   В игре должна быть вертикальная прокрутка, где генерируются платформы.
2. **Персонаж игрока:**  
   Игрок управляет персонажем с помощью клавиш (например, стрелок). Персонаж должен прыгать с платформы на платформу.
3. **Платформы:**  
   Платформы должны появляться случайным образом, с различной скоростью движения, некоторые платформы должны быть статичными, другие — подвижными.
4. **Система очков:**  
   Игрок получает очки в зависимости от высоты, на которую он поднялся
5. **Уровни сложности:**  
   Игра должна увеличивать сложность по мере того, как игрок поднимается выше. Например, платформы могут спустя некоторое время перемещаться по горизонтальной оси.
6. **Графика и звуки:**  
   В игре должны быть графические элементы: фон, спрайты персонажа, платформ, бонусов и препятствий. Также требуется добавить звуковые эффекты для прыжков.
7. **Меню игры:**  
   Стартовое меню, отображение рекорда и результатов.

**План разработки игры**

1. **Подготовка рабочего окружения:**
   * Установить Python и Pygame.
   * Настроить структуру проекта (создать папки для изображений, звуков, исходных кодов).
2. **Создание игровых объектов:**
   * Разработать классы для **персонажа**, **платформ**.
   * Реализовать логику для перемещения и взаимодействия этих объектов.
3. **Обработка ввода:**
   * Реализовать управление персонажем с помощью клавиш стрелок.
   * Реализовать ввод имени игрока для входа в игру
4. **Графика и анимация:**
   * Создать спрайты для персонажа, платформ, бонусов и фона.
   * Реализовать анимацию прыжков и движение объектов.
5. **Добавление звуков и музыки:**
   * Добавить фоновую музыку.
   * Реализовать звуковые эффекты для прыжков, столкновений и других действий.
6. **Игровая механика:**
   * Реализовать систему очков и уровня.
   * Обработать логику падения персонажа, окончания игры и обновления результатов.
7. **Тестирование и отладка:**
   * Провести тестирование на разных этапах разработки.
   * Исправить баги и оптимизировать код.

**Описание кода**

1. **Импорт библиотек**  
   Для начала подключим библиотеку **Pygame**, которая предоставляет инструменты для работы с графикой, обработкой событий в нашей игре. Также подключаем дополнительные модули для работы с математикой, базой данных, модуль работы с системой и прочие.

1. import pygame

2. import random

1. **Инициализация Pygame**  
   Перед тем как использовать любые функции Pygame, мы должны инициализировать библиотеку, а также создать окно игры.

1. pygame.init()

2. screen = pygame.display.set\_mode((800, 600))

3. pygame.display.set\_caption("Doodle Jump")

Здесь создается окно размером 800x600 пикселей, и устанавливается заголовок для окна.

1. **Определение цветов и начальных параметров**  
   Далее мы определяем базовые параметры игры, такие как размеры экрана, скорости, цвета для рисования объектов и начальные параметры (например, начальные координаты игрока).

1. WHITE = (255, 255, 255)

2. GREEN = (0, 255, 0)

3. BLUE = (0, 0, 255)

4. player\_x, player\_y = 400, 500

5. player\_velocity\_y = 0

6. platform\_list = []

1. **Создание класса Player (игрок)**  
   В классе **Player** реализована логика персонажа, который управляется игроком. Мы создаем метод, который будет отвечать за движение персонажа (влево/вправо) и прыжки. Также добавляется метод для рисования персонажа на экране.

1. class Player(pygame.sprite.Sprite):

2. def \_\_init\_\_(self):

3. super().\_\_init\_\_()

4. self.image = pygame.Surface((50, 50))

5. self.image.fill(BLUE)

6. self.rect = self.image.get\_rect()

7. self.rect.center = (player\_x, player\_y)

8. self.velocity = [0, 0]

9.

10. def update(self):

11. keys = pygame.key.get\_pressed()

12. if keys[pygame.K\_LEFT]:

13. self.rect.x -= 5

14. if keys[pygame.K\_RIGHT]:

15. self.rect.x += 5

16.

17. self.rect.y += self.velocity[1]

18. if self.rect.bottom > 600:

19. self.rect.bottom = 600

20. self.velocity[1] = 0

21. else:

22. self.velocity[1] += 1

23.

24. # Управление прыжками

25. if self.rect.colliderect(platform.rect):

26. self.velocity[1] = -15

В этом коде:

* + **update()**: Этот метод обновляет положение игрока, проверяет нажатие клавиш (влево/вправ) и обновляет вертикальную скорость для имитации гравитации.
  + Игрок может прыгать, если его **rect** (прямоугольник) соприкасается с платформой.

1. **Создание класса Platform (платформа)**  
   Платформы случайным образом генерируются на экране. Платформы могут быть статичными или движущимися, и создаются с помощью классов, которые генерируют платформы по случайным координатам.

1. class Platform(pygame.sprite.Sprite):

2. def \_\_init\_\_(self, x, y):

3. super().\_\_init\_\_()

4. self.image = pygame.Surface((100, 20))

5. self.image.fill(GREEN)

6. self.rect = self.image.get\_rect()

7. self.rect.x = x

8. self.rect.y = y

9.

10. def update(self):

11. self.rect.y += 5

12. if self.rect.top > 600:

13. self.rect.top = 0

14. self.rect.x = random.randint(0, 700)

В классе **Platform**:

* + Платформа двигается вниз, а если она выходит за пределы экрана, то появляется снова вверху экрана с случайной горизонтальной позицией.

1. **Генерация платформ и основной игровой цикл**  
   В основной части кода создается несколько платформ и игрок, а также устанавливается основной игровой цикл, который обновляет все элементы на экране и отслеживает взаимодействие с ними.

1. player = Player()

2. all\_sprites = pygame.sprite.Group()

3. all\_sprites.add(player)

4.

5. platforms = pygame.sprite.Group()

6. for i in range(5):

7. platform = Platform(random.randint(0, 700), random.randint(50, 550))

8. platforms.add(platform)

9. all\_sprites.add(platform)

10.

11. clock = pygame.time.Clock()

12.

13. running = True

14. while running:

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

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

17. running = False

18.

19. all\_sprites.update()

20.

21. screen.fill(WHITE)

22. all\_sprites.draw(screen)

23. pygame.display.flip()

24. clock.tick(60)

25.

26. pygame.quit()

В этом цикле:

* + **all\_sprites.update()**: обновляет положение всех объектов (игрока и платформ).
  + **screen.fill(WHITE)**: очищает экран перед отрисовкой новых кадров.
  + **all\_sprites.draw(screen)**: рисует все объекты на экране.
  + **pygame.display.flip()**: обновляет экран.
  + **clock.tick(60)**: ограничивает скорость игры до 60 кадров в секунду.

1. **Завершение игры**  
   Когда игрок выходит из игры (закрывает окно), программа завершает работу, вызывая pygame.quit(), что останавливает все процессы Pygame.

**Основные компоненты игры:**

* **Player**: управляемый персонаж, который прыгает по платформам.
* **Platform**: объекты, по которым персонаж может прыгать.
* **Обработка ввода**: клавиши **влево** и **вправ** управляют движением игрока.

Вывод:

Проект создания игры **Doodle Jump** с использованием библиотеки **Pygame** успешно продемонстрировал основные аспекты и возможности по разработке игр. В ходе работы были реализованы основные игровые механики, такие как управление персонажем, генерация платформ и система очков, что создало увлекательный и динамичный игровой процесс.

В будущем проект можно расширить, добавив большее количество персонажей, улучшенную графику, общую sql базу данных с рекордами всех игроков. Этот опыт стал важной основой для дальнейших исследований и проектов в области разработки игр и программирования.

**Ссылки на проект**

<https://github.com/TimurKapralov/Doodle_Jump_yandexlyceum.git>

