

Создание 2D игры на языке программирования Python

Языков Кирилл Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе представлена разработка бесконечной двухмерной игры на языке программирования Python с использованием библиотеки Pygame. Особенностью игры является сочетание механики прыжков, характерной для жанра платформеров, и бесконечной генерации препятствий, аналогичной игре Chrome Dino. В процессе реализации создан адаптивный алгоритм увеличения скорости, система случайной генерации препятствий, а также базовая физическая модель персонажа. Архитектура проекта построена модульно: выделены классы для игрока, земли и препятствий, что обеспечивает расширяемость и удобство сопровождения кода. Работа демонстрирует возможности Python для разработки интерактивных приложений и может служить примером для студентов и разработчиков, изучающих основы проектирования игровых информационных систем.

Ключевые слова: Python, Pygame, 2D-игра, бесконечный раннер, платформер, генерация препятствий, разработка игр, интерактивные приложения, проектирование игровых систем.

Creating a 2D Game in the Python Programming Language

Yazykov Kirill Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This work presents the development of an endless two-dimensional game using the Python programming language with the Pygame library. The game features a combination of jump mechanics, typical for the platformer genre, and endless obstacle generation, similar to the Chrome Dino game. During implementation, an adaptive speed-increase algorithm, a random obstacle generation system, and a basic physical model for the player character were created. The project architecture is modular: classes were defined for the player, the ground, and obstacles, providing scalability and ease of code maintenance. This work demonstrates the capabilities of Python for developing interactive applications and can serve as an example for students and developers studying the fundamentals of designing game information systems.

Keywords: Python, Pygame, 2D game, endless runner, platformer, obstacle generation, game development, interactive applications, game system design.

1 Введение

1.1 Актуальность

Разработка компьютерных игр является одной из актуальных и востребованных областей информационных технологий. Игры не только представляют интерес для развлечения, но и служат инструментом для обучения, моделирования процессов и развития алгоритмического мышления. Создание 2D игр позволяет разработчикам освоить основы программирования, работы с графикой, обработкой событий и логикой взаимодействия объектов в игровом пространстве.

1.2 Обзор исследований

В работе S. Watkiss описывается создание нескольких простых игр на Python для понимания структуры разработки и иллюстрации различных техник программирования [2]. J. R. Parker в своей книге рассматривает основные идеи и некоторые дизайнерские приемы при создании игр на языке программирования Python. Также он подчеркивает программные особенности, которые могут стать полезными при разработке игровых проектов [3]. В исследовании Е. Ш. Ефимовой рассматриваются этические аспекты разработки компьютерных игр в контексте стремительного развития информационных технологий. Автор акцентирует внимание на том, что создатели ИТ-продуктов зачастую сосредоточены на стимулировании их потребления, не уделяя достаточного внимания социальным последствиям [4]. В статье А.Д.Бирюковой и М. В. Новиковой рассматриваются все этапы создания компьютерной игры [5].

1.3 Цель исследования

Цель исследования – разработка бесконечной 2D игры на языке программирования Python с использованием библиотеки Pygame.

2 Материалы и методы

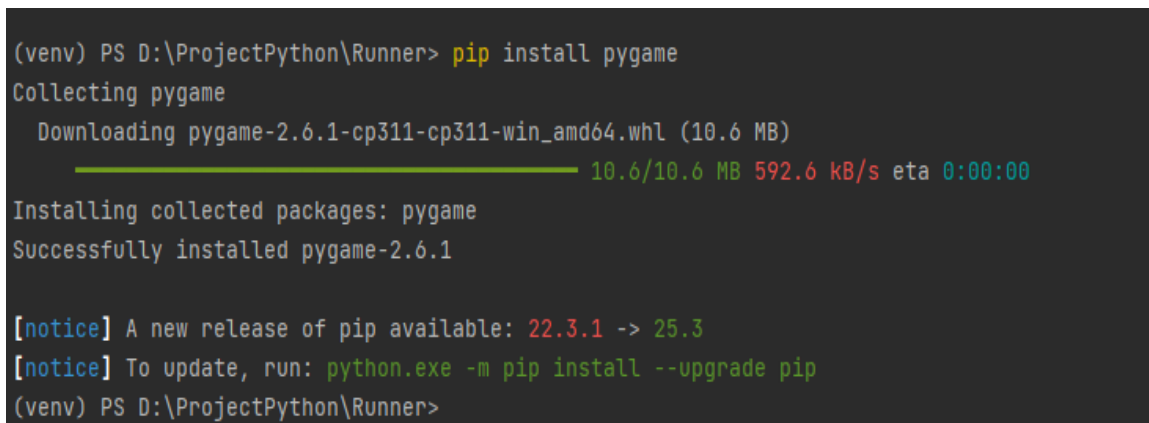
В данном исследовании для разработки игры использовалась интегрированная среда разработки PyCharm. Язык программирования Python был выбран благодаря своей простоте и широким возможностям для создания приложений. Библиотека Pygame предоставила необходимые инструменты для работы с графикой, обработки событий, отрисовки объектов, анимации и управления игровым циклом [1].

3 Результаты

Разработанная игра представляет собой бесконечный 2D раннер, в котором игрок управляет персонажем, преодолевающим случайно генерируемые препятствия. Основной механикой является прыжок над препятствиями. Игра сопровождается подсчётом очков, которые увеличиваются по мере прохождения, а также адаптивным увеличением скорости, усложняющим процесс игры. Игра завершается при столкновении игрока с препятствием. Также реализована система сохранения рекорда в

файл. Архитектура проекта построена модульно: выделены классы для игрока, земли и препятствий, что обеспечивает расширяемость и удобство сопровождения кода.

Перед началом работы необходимо установить PyCharm (IDE), после создать новый проект или уже в открытом перейти в терминал и скачать библиотеку pygame, командой `pip install pygame` (рис. 1).

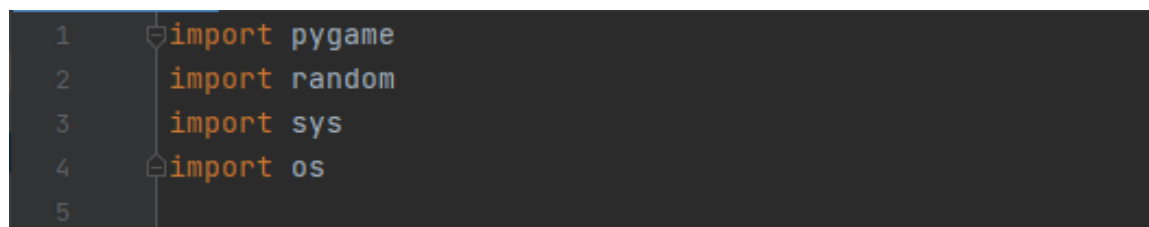


```
(venv) PS D:\ProjectPython\Runner> pip install pygame
Collecting pygame
  Downloading pygame-2.6.1-cp311-cp311-win_amd64.whl (10.6 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.6/10.6 MB 592.6 kB/s eta 0:00:00
Installing collected packages: pygame
Successfully installed pygame-2.6.1

[notice] A new release of pip available: 22.3.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS D:\ProjectPython\Runner>
```

Рисунок 1 – Скачивание библиотеки pygame

После установки необходимо перейти к импорту библиотек (рис. 2).



```
1  import pygame
2  import random
3  import sys
4  import os
5
```

Рисунок 2 – Импорт библиотек в проекте

Следующим этапом следует прописать параметры физики игры, а также параметры окна, земли, препятствий и скорости игры (рис. 3).

```
6    WIDTH, HEIGHT = 900, 400
7    FPS = 60
8
9    GRAVITY = 0.9
10   JUMP_VELOCITY = -14
11
12   GROUND_HEIGHT = 80
13
14   OBSTACLE_MIN_GAP = 300
15   OBSTACLE_MAX_GAP = 650
16   OBSTACLE_MIN_WIDTH = 20
17   OBSTACLE_MAX_WIDTH = 60
18   OBSTACLE_MIN_HEIGHT = 30
19   OBSTACLE_MAX_HEIGHT = 120
20
21   INITIAL_SPEED = 5.0
22   SPEED_INCREMENT = 0.001
23
```

Рисунок 3 – Написание физики для игры

Далее необходимо создать счетчик, который сохраняет результат в отдельный файл `highscore.txt`. После прописать инициализацию игры, чтобы при ее запуске начали работу модули, размерами окон и стабильности игры через контролирование ФПС (рис. 4).

```
23
24   HIGH_SCORE_FILE = "highscore.txt"
25
26   pygame.init()
27   screen = pygame.display.set_mode((WIDTH, HEIGHT))
28   pygame.display.set_caption("Infinite Dash")
29   clock = pygame.time.Clock()
30   font = pygame.font.SysFont(None, 28)
31   bigfont = pygame.font.SysFont(None, 56)
32
```

Рисунок 4 – Создание счетчика и инициализация игры

В следующей строке кода нужно указать две функции, которые производят загрузку и сохранение рекорда с последующей перезаписью. Код сначала проверяет наличие файла, а после выводит либо ошибку, либо число, которое находится в файле (рис. 5).

```

32
    1 usage
33  def load_highscore():
34      if not os.path.exists(HIGH_SCORE_FILE):
35          return 0
36      try:
37          with open(HIGH_SCORE_FILE, "r") as f:
38              return int(f.read().strip() or 0)
39      except Exception:
40          return 0
41
    1 usage
42  def save_highscore(score):
43      try:
44          with open(HIGH_SCORE_FILE, "w") as f:
45              f.write(str(int(score)))
46      except Exception:
47          pass
48

```

Рисунок 5 – Чтение и запись файла рекорда игры

Далее необходимо указать все параметры для игрока: размеры, начальную позицию скорость, цвет, проверку прыжка, гравитация прыжка, обновление положения по оси Y, возвращение прямоугольника игрока для коллизий, тень игрока, и глаз игроку (рис. 6).

```

    4 usages
49  class Player:
50      def __init__(self):
51          self.w = 40
52          self.h = 40
53          self.x = int(WIDTH * 0.15)
54          self.y = HEIGHT - GROUND_HEIGHT - self.h
55          self.vy = 0.0
56          self.on_ground = True
57          self.color = (20, 180, 60)
58
    2 usages (1 dynamic)
59      def update(self):
60          self.vy += GRAVITY
61          self.y += self.vy
62          ground_y = HEIGHT - GROUND_HEIGHT - self.h
63          if self.y >= ground_y:
64              self.y = ground_y
65              self.vy = 0
66              self.on_ground = True
67          else:
68              self.on_ground = False
69
    2 usages
70      def jump(self):
71          if self.on_ground:
72              self.vy = JUMP_VELOCITY
73              self.on_ground = False
74
    3 usages (1 dynamic)
75      def rect(self):
76          return pygame.Rect(int(self.x), int(self.y), self.w, self.h)
77
    2 usages (1 dynamic)
78      def draw(self, surf):
79          r = self.rect()
80          pygame.draw.rect(surf, (0,0,0,50), (r.x+3, r.y+3, r.w, r.h))
81          pygame.draw.rect(surf, self.color, r)
82          eye_x = r.x + int(r.w * 0.6)
83          eye_y = r.y + int(r.h * 0.3)
84          pygame.draw.circle(surf, (255,255,255), (eye_x, eye_y), 4)

```

Рисунок 6 – Код параметров игрока

Следующим шагом нужно прописать код для препятствий: позиция параметры (высота и ширина), тип, цвет, возврат коллизий, движение (влево), отрисовку прямоугольного и треугольного препятствия (рис. 7).

```
2 usages
81 class Obstacle:
82     def __init__(self, x, width, height, kind="rect"):
83         self.x = x
84         self.width = width
85         self.height = height
86         self.kind = kind
87         self.color = (200, 40, 40)
88
89     def rect(self):
90         return pygame.Rect(int(self.x), HEIGHT - GROUND_HEIGHT - self.height, self.width, self.height)
91
92     def update(self, speed):
93         self.x -= speed
94
95     def draw(self, surf):
96         r = self.rect()
97         if self.kind == "rect":
98             pygame.draw.rect(surf, self.color, r)
99             pygame.draw.rect(surf, (160, 0, 0), (r.x, r.y, r.w, int(r.h*0.25)))
100         elif self.kind == "triangle":
101             points = [(r.x, r.y + r.h), (r.x + r.w/2, r.y), (r.x + r.w, r.y + r.h)]
102             pygame.draw.polygon(surf, self.color, points)
```

Рисунок 7 – Код параметров препятствий

После, необходимо прописать менеджер для препятствий. Особенность данного списка заключается в том, что регулируются места появления, после поражения очищается вся область и начинается все заново, удаляются все препятствия, которые вышли за край экрана, автоматическая генерация нового препятствия, а также создание «коридора» препятствий, чтобы игра казалась бесконечной (рис. 8).

```

102 class ObstacleManager:
103     def __init__(self):
104         self.obstacles = []
105         self.next_x = WIDTH + 200
106
107     3 usages
108     def reset(self):
109         self.obstacles.clear()
110         self.next_x = WIDTH + 200
111
112     2 usages (1 dynamic)
113     def update(self, speed):
114         self.obstacles = [o for o in self.obstacles if o.x + o.width > -50]
115         for o in self.obstacles:
116             o.update(speed)
117         if self.next_x - (WIDTH if len(self.obstacles)==0 else 0) <= WIDTH + 0:
118             w = random.randint(OBSTACLE_MIN_WIDTH, OBSTACLE_MAX_WIDTH)
119             h = random.randint(OBSTACLE_MIN_HEIGHT, OBSTACLE_MAX_HEIGHT)
120             kind = random.choice(["rect", "rect", "triangle"])
121             o = Obstacle(self.next_x, w, h, kind)
122             self.obstacles.append(o)
123             gap = random.randint(OBSTACLE_MIN_GAP, OBSTACLE_MAX_GAP)
124             self.next_x += w + gap
125
126     5 usages
127     def ensure_next(self):
128         while self.next_x < WIDTH + 1200:
129             w = random.randint(OBSTACLE_MIN_WIDTH, OBSTACLE_MAX_WIDTH)
130             h = random.randint(OBSTACLE_MIN_HEIGHT, OBSTACLE_MAX_HEIGHT)
131             kind = random.choice(["rect", "rect", "triangle"])
132             o = Obstacle(self.next_x, w, h, kind)
133             self.obstacles.append(o)
134             gap = random.randint(OBSTACLE_MIN_GAP, OBSTACLE_MAX_GAP)
135             self.next_x += w + gap
136
137     2 usages (1 dynamic)
138     def draw(self, surf):
139         for o in self.obstacles:
140             o.draw(surf)

```

Рисунок 8 – Создание менеджера препятствий

В следующем шаге можно приступить к написанию кода для инициализации объектов и переменных: создается объект игрока, менеджер препятствий, начальная скорость, очки, загрузка рекорда из файла, условие проверки проигрыша игрока и создание сразу нескольких препятствий, чтобы они начинали появляться плавно (рис. 9).

```
137
    1 usage
138 def main():
139     player = Player()
140     obstacles = ObstacleManager()
141     speed = INITIAL_SPEED
142     score = 0.0
143     highscore = load_highscore()
144
145     running = True
146     game_over = False
147     paused = False
148
149     obstacles.ensure_next()
150
```

Рисунок 9 – Разделение калькулятора на основную часть и с хранением истории

Далее необходимо делать подсчет кадров через $dt = \text{clock.tick(FPS)} / 1000.0$, для того чтобы подсчитывать очки и скорость передвижения. А также прописать выход из игры. После этого необязательным шагом является написание кода для взаимодействия с клавиатурой (ESC — выйти, SPACE или UP — прыжок / перезапуск / снять паузу, P — пауза, R — перезапуск только при GAME OVER). (рис. 10).

```

150
151     while running:
152         dt = clock.tick(FPS) / 1000.0
153         for event in pygame.event.get():
154             if event.type == pygame.QUIT:
155                 running = False
156
157             if event.type == pygame.KEYDOWN:
158                 if event.key == pygame.K_ESCAPE:
159                     running = False
160                 if event.key == pygame.K_SPACE or event.key == pygame.K_UP:
161                     if not game_over and not paused:
162                         player.jump()
163                     elif game_over:
164                         player = Player()
165                         obstacles.reset()
166                         obstacles.ensure_next()
167                         speed = INITIAL_SPEED
168                         score = 0.0
169                         game_over = False
170                     elif paused:
171                         paused = False
172                 if event.key == pygame.K_p:
173                     paused = not paused
174                 if event.key == pygame.K_r and game_over:
175                     player = Player()
176                     obstacles.reset()
177                     obstacles.ensure_next()
178                     speed = INITIAL_SPEED
179                     score = 0.0
180                     game_over = False
181

```

Рисунок 10 – Код для управления игрой клавиатурой

На следующем этапе нужно сделать привязку управлением мышкой в игре по клику: игра идет – прыжок, игра окончена – перезапуск (рис. 11).

```

181
182     if event.type == pygame.MOUSEBUTTONDOWN:
183         if not game_over and not paused:
184             player.jump()
185         elif game_over:
186             player = Player()
187             obstacles.reset()
188             obstacles.ensure_next()
189             speed = INITIAL_SPEED
190             score = 0.0
191             game_over = False
192

```

Рисунок 11 – Код для управление мышкой

Чтобы мир обновлялся необходимо обновлять игрока и мир, с определенным периодом, скорость будет увеличиваться, прямопропорционально этому увеличиваются и очки, а также каждый раз происходит проверка на столкновение, которое все прерывает (рис. 12).

```

192
193     if not paused and not game_over:
194         player.update()
195         obstacles.update(speed)
196         obstacles.ensure_next()
197         speed += SPEED_INCREMENT
198         score += speed * dt * 10
199         pr = player.rect()
200         for o in obstacles.obstacles:
201             if pr.colliderect(o.rect()):
202                 game_over = True
203                 if score > highscore:
204                     highscore = int(score)
205                     save_highscore(highscore)
206                 break
207

```

Рисунок 12 – Код для обработки истории калькулятора

Чтобы заполнить задний фон игры можно написать код для этого, ниже представлен код для прорисовки кадра определённого размера (холмы, земля, препятствие, текст интерфейса, текст паузы, конец игры), в конце обязательно нужно прописывать обновление экрана `pygame.display.flip()` (рис. 13).

```

207
208     screen.fill((135, 206, 235))
209
210     for i in range(5):
211         pygame.draw.ellipse(screen, (100, 150, 100), (i*300 - (pygame.time.get_ticks()*0.02)%300, HEIGHT - GROUND_HEIGHT - 80, 300, 60))
212
213     pygame.draw.rect(screen, (100, 80, 40), (0, HEIGHT - GROUND_HEIGHT, WIDTH, GROUND_HEIGHT))
214     for i in range(0, WIDTH, 40):
215         pygame.draw.rect(screen, (80, 60, 30), (i, HEIGHT - 20, 30, 8))
216
217     obstacles.draw(screen)
218     player.draw(screen)
219
220     txt = font.render(f"Score: {int(score)}", True, (0, 0, 0))
221     screen.blit(txt, (10, 10))
222     txt2 = font.render(f"Speed: {speed:.2f}", True, (0, 0, 0))
223     screen.blit(txt2, (10, 35))
224     txt3 = font.render(f"Highscore: {highscore}", True, (0, 0, 0))
225     screen.blit(txt3, (10, 60))
226
227     if paused:
228         p = bigfont.render("PAUSED (P)", True, (0, 0, 0))
229         screen.blit(p, (WIDTH//2 - p.get_width()//2, HEIGHT//2 - 40))
230
231     if game_over:
232         go = bigfont.render("GAME OVER", True, (180, 30, 30))
233         info = font.render("Space / Click - Restart. Esc - Exit", True, (0, 0, 0))
234         screen.blit(go, (WIDTH//2 - go.get_width()//2, HEIGHT//2 - 60))
235         screen.blit(info, (WIDTH//2 - info.get_width()//2, HEIGHT//2))
236
237     pygame.display.flip()
238

```

Рисунок 13 – Код для обработки истории калькулятора

Последним этапом следует прописать выход из игры `pygame.quit()` `sys.exit()` и основной запуск игры `if __name__ == "__main__": main()` (рис. 14).

```
238
239     pygame.quit()
240     sys.exit()
241
242 ► if __name__ == "__main__":
243     main()
244
```

Рисунок 14 – Код запуска игры

Код игры готов (рис. 15).

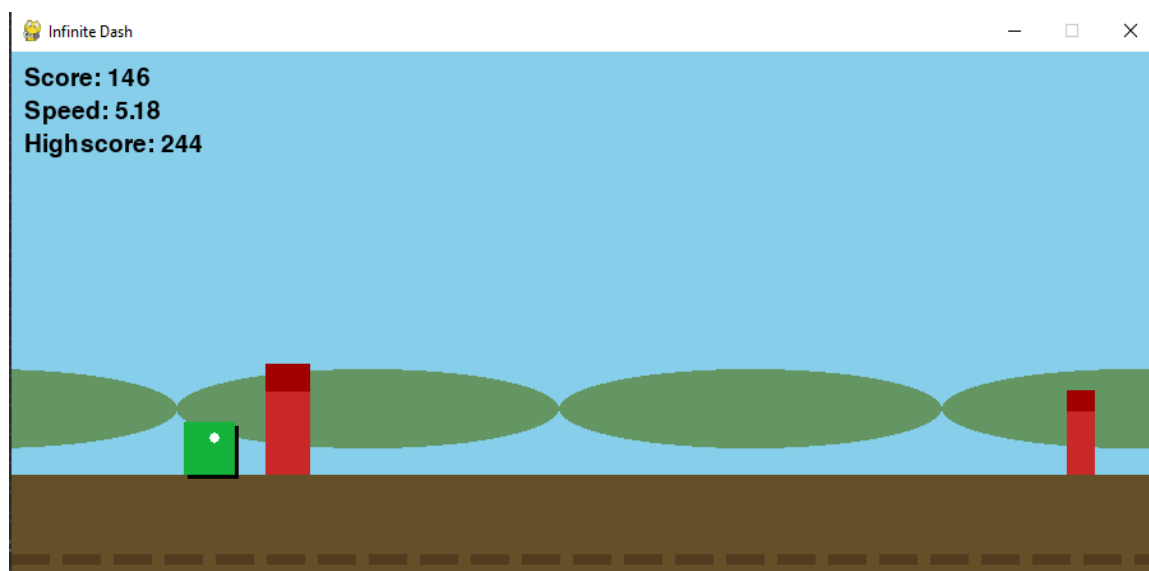


Рисунок 15 – Готовая игра

Прыжок в игре (рис. 16).

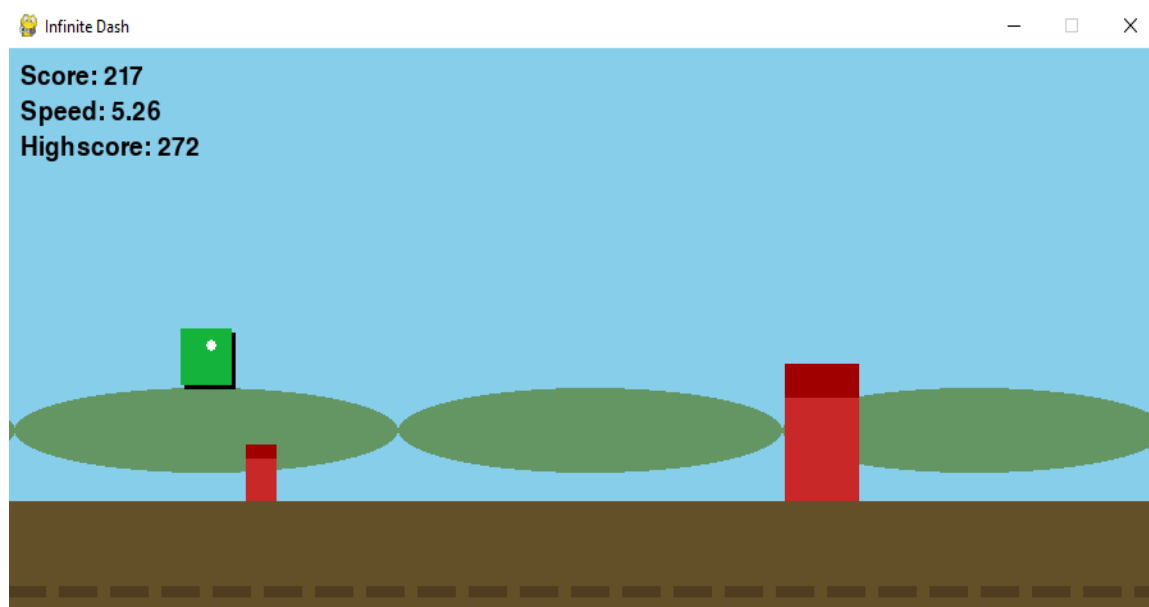


Рисунок 16 – Прыжок в игре

Конец игры (рис. 17).

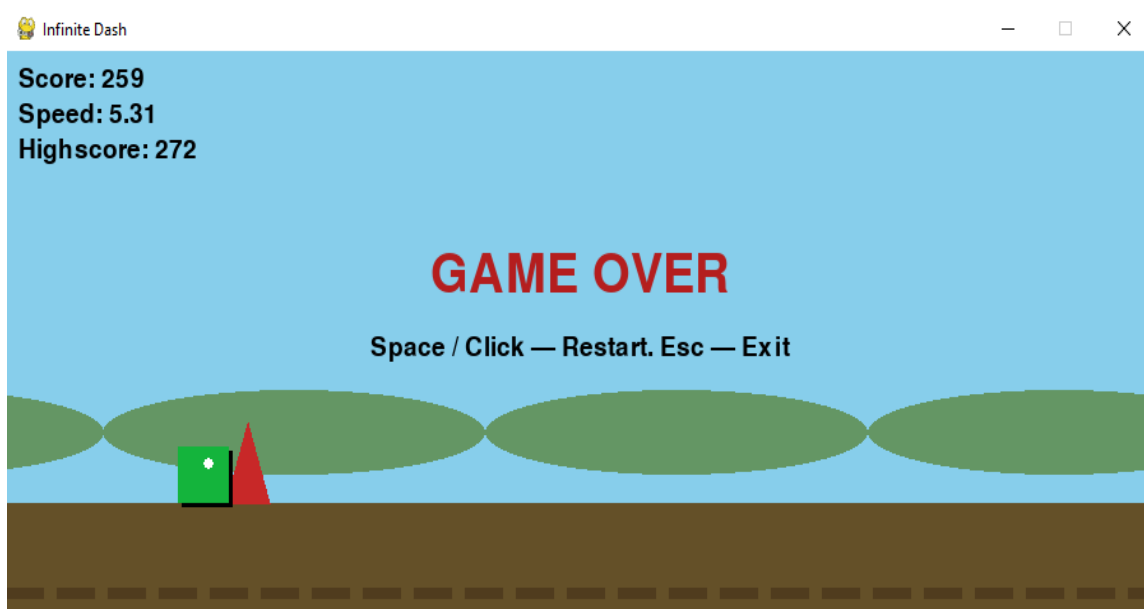


Рисунок 17 – Конец игры

4 Выводы

Создание 2D-игры на языке Python является эффективным и доступным способом изучения основ разработки игр, объектно-ориентированного программирования и взаимодействия с графическими библиотеками. В ходе реализации проекта «Infinite Dash» была использована библиотека Pygame, которая предоставляет готовые инструменты для работы с графикой, обработкой событий, отрисовкой объектов, анимацией и игровым циклом.

Разработанная игра представляет собой бесконечный раннер, в котором игрок управляет персонажем, преодолевающим случайно генерируемые препятствия. В процессе разработки были реализованы основные элементы игровой логики: физика прыжка, гравитация, система столкновений, процедурная генерация препятствий, плавное увеличение скорости, подсчёт очков и сохранение рекордов в файл.

Библиографический список

1. Модуль Pygame для написания игр на языке Python. URL: <http://www.pygame.org> (дата обращения: 18.11.2025)
2. Watkiss S. Beginning Game Programming with Pygame Zero: Coding Interactive Games on Raspberry Pi Using Python. Apress, 2020.
3. Parker J. R. Game Development Using Python. Stylus Publishing, LLC, 2018.
4. Ефимова Е. Ш. Разработка компьютерных игр как этическая проблема // Научные коммуникации. Научная этика. Инженерная этика. Омск: Омский государственный технический университет, 2017. С. 17-21.
5. Бирюкова А. Д., Новикова М. В. Жизненный цикл разработки компьютерных игр // Вестник научных конференций. 2016. №12-4. С. 24-25.