Этот код представляет собой реализацию классической игры Змейка с использованием библиотеки Pygame. Игрок управляет змейкой, которая движется по экрану, поедая яблоки и увеличивая свою длину. Игра заканчивается, когда змейка сталкивается с границей экрана или своим собственным телом.

Основные моменты игры:

1. **Змейка** — состоит из сегментов, каждый из которых представлен прямоугольником. Игрок управляет змейкой с помощью стрелок на клавиатуре.
2. **Яблоки** — яблоки случайным образом появляются на экране, и каждый раз, когда змейка съедает яблоко, она растет, добавляя новый сегмент.
3. **Цель игры** — управлять змейкой таким образом, чтобы она съедала яблоки и не сталкивалась с границами экрана или с собой.

1. Инициализация Pygame и настроек экрана

```
pygame.init()
```

- Эта строка инициализирует все модули Pygame, которые необходимы для работы игры.

```
width = 700
height = 700
window = pygame.display.set_mode((width, height))
pygame.display.set_caption("Zmeika")
```

- Устанавливаются размеры окна игры (700x700 пикселей).
- Создается окно с этим размером с помощью `pygame.display.set_mode()`.
- Задается заголовок окна — "Zmeika".

2. Цвета и шрифт для счета

```
snake_color = (148, 0, 211)  # фиолетовый
apple_color = (0, 255, 0)    # зеленый
```

- Задаются цвета для змейки и яблока в формате RGB. В данном случае:
  - Фиолетовый для змейки `(148, 0, 211)`.
  - Зеленый для яблока `(0, 255, 0)`.

```
score_font = pygame.font.SysFont("Arial", 20)
```

- Создается шрифт для отображения счета, шрифт "Arial" с размером 20.

```
game_clock = pygame.time.Clock()
```

- Создается объект для контроля частоты кадров (FPS) игры.

3. Инициализация змейки

```
block_size = 20
speed = 5
snake_length = 3
snake_segments = []
```

- `block_size` — размер блока, из которого состоит змейка (20 пикселей).
- `speed` — скорость игры (количество кадров в секунду).
- `snake_length` — начальная длина змейки.
- `snake_segments` — список, в котором будут храниться сегменты змейки (каждый сегмент представлен прямоугольником `pygame.Rect`).

```
for i in range(snake_length):
    snake_segments.append(pygame.Rect((width / 2) - (block_size *
i), height / 2, block_size, block_size))
```

- Здесь создаются начальные сегменты змейки, каждый сегмент — прямоугольник размером `block_size x block_size`. Начальная змейка состоит из 3 сегментов, расположенных по горизонтали в центре экрана.

4. Управление змейкой и яблоком

```
snake_move = "right"
next_move = "right"
apple_pos = pygame.Rect(random.randint(0, width - block_size),
random.randint(0, height - block_size), block_size, block_size)
```

- Задаются начальные направления движения змейки — "right" (вправо).
- `apple_pos` — позиция яблока на экране, которая генерируется случайным образом.

5. Основной игровой цикл

```
game_over = False
while not game_over:
```

- Начинается основной цикл игры. Игра продолжается до тех пор, пока переменная `game_over` не станет `True`.

Обработка событий

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        game_over = True
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_UP and snake_move != "down":
```

```
            next_move = "up"
        elif event.key == pygame.K_DOWN and snake_move != "up":
            next_move = "down"
        elif event.key == pygame.K_LEFT and snake_move != "right":
            next_move = "left"
        elif event.key == pygame.K_RIGHT and snake_move != "left":
            next_move = "right"
```

- Обрабатываются события, такие как выход из игры (кнопка закрытия окна) или нажатие клавиш. Изменение направления змейки происходит с учетом текущего направления (нельзя двигаться в противоположную сторону).

Обновление направления и движения змейки

```
snake_move = next_move
```

- Обновляется направление движения змейки.

```
if snake_move == "up":
    snake_segments.insert(0, pygame.Rect(snake_segments[0].left,
snake_segments[0].top - block_size, block_size, block_size))
elif snake_move == "down":
    snake_segments.insert(0, pygame.Rect(snake_segments[0].left,
snake_segments[0].top + block_size, block_size, block_size))
elif snake_move == "left":
    snake_segments.insert(0, pygame.Rect(snake_segments[0].left -
block_size, snake_segments[0].top, block_size, block_size))
elif snake_move == "right":
    snake_segments.insert(0, pygame.Rect(snake_segments[0].left +
block_size, snake_segments[0].top, block_size, block_size))
```

- В зависимости от направления движения, первый сегмент змейки (голова) перемещается в нужную сторону, добавляется новый сегмент в начало списка.

Проверка на съеденное яблоко

```
if snake_segments[0].colliderect(apple_pos):
    apple_pos = pygame.Rect(random.randint(0, width - block_size),
random.randint(0, height - block_size), block_size, block_size)
    snake_length += 1
```

- Если голова змейки столкнулась с яблоком (проверка с помощью `colliderect`), создается новое яблоко в случайном месте, а длина змейки увеличивается.

Обрезание хвоста

```
if len(snake_segments) > snake_length:
    snake_segments.pop()
```

- Если змейка стала длиннее, хвост (последний сегмент) удаляется, поддерживая нужную длину.

Проверка на столкновение с границами экрана или телом змейки

```
if snake_segments[0].left < 0 or snake_segments[0].right > width or
snake_segments[0].top < 0 or snake_segments[0].bottom > height:
    game_over = True
```

- Если голова змейки выходит за пределы экрана, игра заканчивается.

```
for segment in range(1, len(snake_segments)):
    if snake_segments[0].colliderect(snake_segments[segment]):
        game_over = True
```

- Если голова змейки сталкивается с любым из ее сегментов (телом), игра заканчивается.

6. Отображение объектов на экране

```
window.fill((0, 0, 0))
```

- Очищает экран, заполняя его черным цветом перед следующим кадром.

Отображение змейки

```
for idx, segment in enumerate(snake_segments):
    if idx == 0:
        pygame.draw.circle(window, snake_color, segment.center,
block_size / 2)
    else:
        pygame.draw.circle(window, snake_color, segment.center,
block_size / 2)
        pygame.draw.circle(window, (148, 0, 211), segment.center,
block_size / 4)
```

- Рисует каждый сегмент змейки как круг, где первый сегмент (голова) рисуется чуть крупнее.

Отображение яблока

```
pygame.draw.circle(window, apple_color, apple_pos.center, block_size
/ 2)
```

- Рисует яблоко как круг в случайной позиции.

Отображение счета

```
score_text = score_font.render(f"Съедено яблок: {snake_length - 3}",
True, (255, 255, 255))
window.blit(score_text, (10, 10))
```

- Отображает текст с количеством съеденных яблок, вычисляемых как разница между текущей длиной змейки и начальной длиной (3).

7. Частота кадров

```
game_clock.tick(speed)
```

- Устанавливает частоту кадров на 5 кадров в секунду (значение переменной `speed`).

8. Завершение игры

```
pygame.quit()
```

- Закрывает Pygame и завершает игру.

Игра представляет собой классическую змейку, где игрок управляет змейкой, которая растет при поедании яблок. Игра завершается, если змейка сталкивается с границей экрана или своим телом. На экране также отображается количество съеденных яблок.