ОГАН ОО «Центр Алые Паруса» - IT-куб Ульяновск

Игра «Snake Game»

Секция «Информатика», номинация «программирование, компьютерные игры»

Автор работы:
Киямов Эмиль Маратович, 8 класс
Научный руководитель работы:
Шувалова Валерия Дмитриевна
Педагог дополнительного образования

89278154444

Игра **“Snake Game”**, написанная с использованием библиотеки pygame (Python)

Об игре

- Классическая аркадная игра
- Управление змейкой стрелками клавиатуры
- Съедайте яблоки, становитесь длинней
- Избегайте столкновений со стенами и своим телом

Основные возможности

- Выбор цвета змейки (зелёный, синий, красный)
- Динамическая система скоринга
- Звуковые эффекты
- Отслеживание лучшего результата

Структура проекта

- **Классы:**
  Head, Body, Apples, Text, Button
- **Функции:**
  main(), los(), main_ui()
- **Ресурсы:**
  изображения, звуки, фон

Основные классы и их функции

Класс Head (голова змейки)

- Отвечает за движение и управление
- Координаты позиции на сетке
- Текущее направление

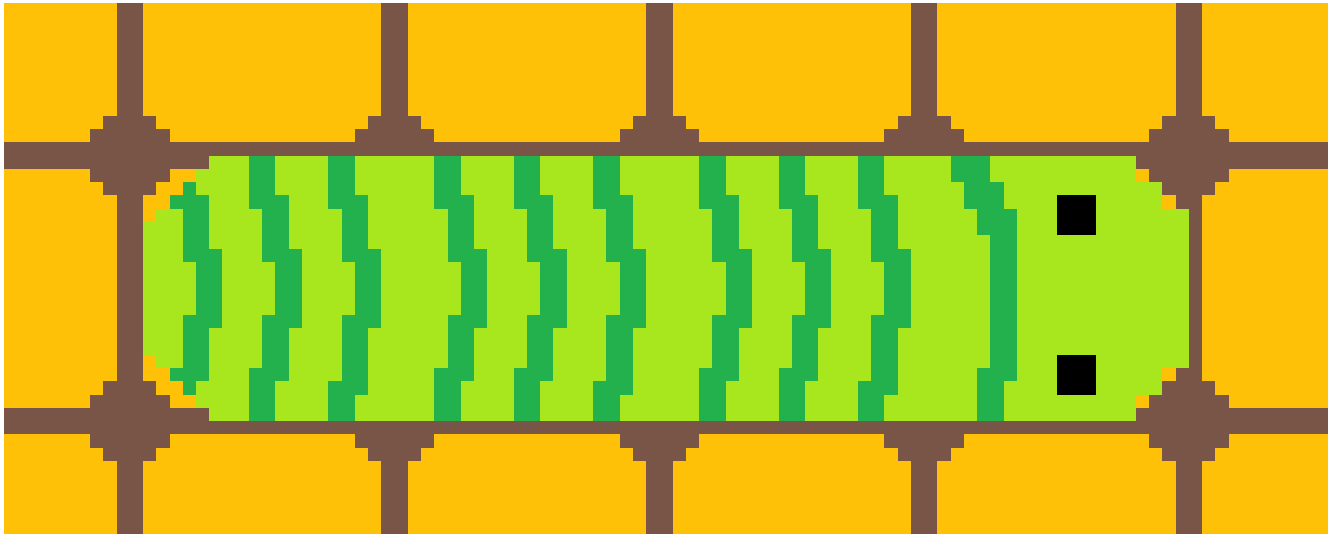
Функция move(), отвечает за движение

```
def move(self):
    self.old_cords = self.cords
    if self.direction == "right":
        self.cords = (self.cords[0] + 40, self.cords[1])
    elif self.direction == "left":
        self.cords = (self.cords[0] - 40, self.cords[1])
    elif self.direction == "up":
        self.cords = (self.cords[0], self.cords[1] - 40)
    elif self.direction == "down":
        self.cords = (self.cords[0], self.cords[1] + 40)

    if self.cords[0] >= W or self.cords[0] < 0 or
self.cords[1] >= H or self.cords[1] < 0:
        self.los()
        die_sound.play()
```

Переменные self.cords и self.old_cords

- Позволяют сохранять текущее положение и предыдущее для корректного отображения змейки на поле



Функция direct(), отвечает за направление

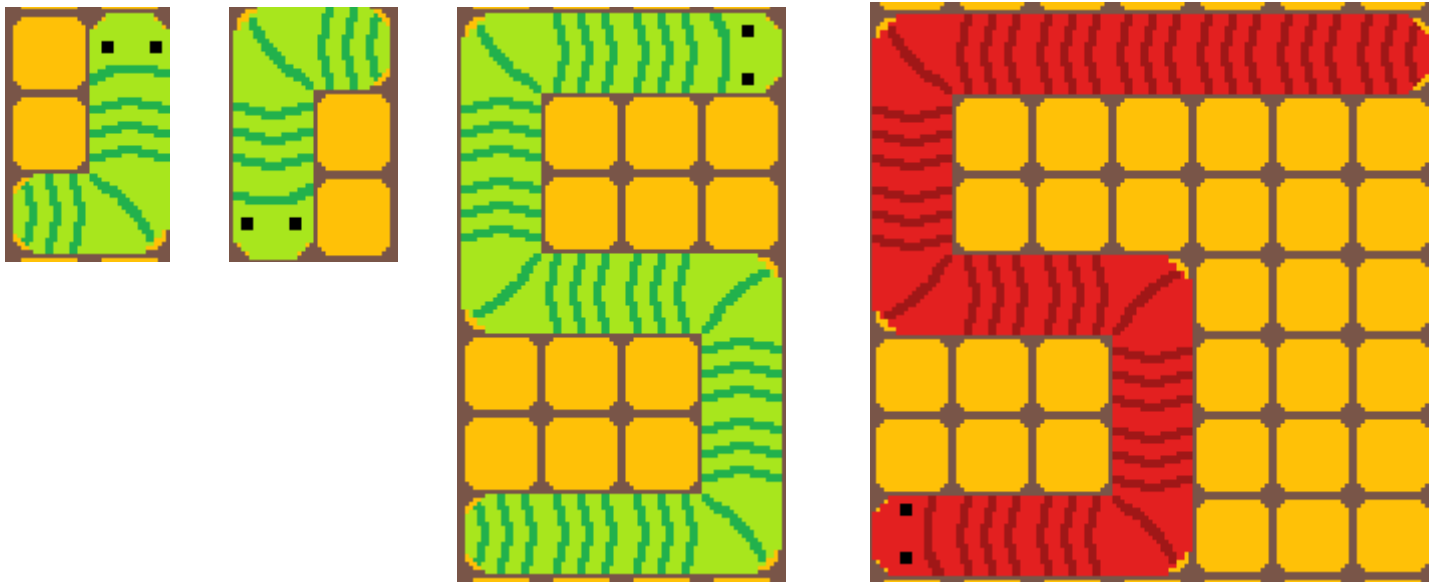
```
def direct(self, d):
    if d == "right" and self.direction != "left":
        self.surf =
pg.transform.rotate(pg.image.load(f'images/{head.color}_head.png'), -90)
        self.surf = pg.transform.scale_by(self.surf, 2)
        return 'right'
    elif d == "left" and self.direction != "right":
        self.surf = pg.transform.rotate(pg.image.load(self.image), 90)
        self.surf = pg.transform.scale_by(self.surf, 2)
        return "left"
    elif d == "down" and self.direction != "up":
        self.surf = pg.transform.rotate(pg.image.load(self.image), 180)
        self.surf = pg.transform.scale_by(self.surf, 2)
        return 'down'
    elif d == "up" and self.direction != "down":
        self.surf = pg.image.load(self.image)
        self.surf = pg.transform.scale_by(self.surf, 2)
        return "up"
    return self.direction
```

Класс Body (тело змейки)

- Каждый сегмент тела следует за предыдущим
- Динамическая смена спрайтов при поворотах
- Корректная работа прямых и угловых сегментов
- Отличие хвоста от обычного тела

Модуль draw()

- Отвечает за корректность отображения сегментов змейки



Функция direct()

- Отвечает за поворот каждого сегмента змейки

```
def direct(self, d):
    self.direction = direct(self, d)

def find_dir(self, index):
    if self.cords[0] < body_parts[index - 1].cords[0]:
        self.surf = pg.transform.rotate(self.surf, -90)
    elif self.cords[0] > body_parts[index - 1].cords[0]:
        self.surf = pg.transform.rotate(self.surf, 90)
    elif self.cords[1] < body_parts[index - 1].cords[1]:
        self.surf = pg.transform.rotate(self.surf, 180)
    elif self.cords[1] > body_parts[index - 1].cords[1]:
        pass
```

Игровой цикл “main()”

- Чтение ввода пользователя (стрелки клавиатуры)
- Обновление позиции головы и тела
- Проверка столкновений и сбора яблок
- Рендеринг всех объектов на экран

Чтение ввода

- Создается маска для считывания клавиш, нажатая клавиша сохраняется в переменной

```
keys =
pg.key.get_pressed()
if keys[pg.K_LEFT]:
    key_pressed = "left"
elif keys[pg.K_RIGHT]:
    key_pressed = "right"
elif keys[pg.K_UP]:
    key_pressed = "up"
elif keys[pg.K_DOWN]:
    key_pressed = "down"
```

Обновление позиции частей тела

- Поворачивает змейку в нужном направлении

```
if cnt >= 5:
    if key_pressed == "left":
        head.direct("left")
    elif key_pressed == "right":
        head.direct("right")
    elif key_pressed == "up":
        head.direct("up")
    elif key_pressed == "down":
        head.direct("down")
    old_positions = [elem.cords for elem in body_parts]
    head.move()
    for i in range(1, len(body_parts)):
        body_parts[i].move(old_positions[i - 1])

    cnt = 0
```

cnt- задержка перед
следующим поворотом

Система коллизий

- Проверка выхода за границы экрана
- Проверка пересечения с телом змейки
- Обнаружение съедения яблока

```
for part in body_parts[1:]:
    if head.cords ==
part.cords:
        head.los()
        die_sound.play()
```

```
if self.cords[0] >= W or self.cords[0] < 0 or self.cords[1] >= H or
self.cords[1] < 0:
    self.los()
    die_sound.play()
```

```
for apple in apples:
    if head.cords == apple.cords:
        last_part = body_parts[-1]
        body_parts.append(Body(last_part.cords))
        occupied = [elem.cords for elem in body_parts] + [a.cords for a in
apples]
        apple.cords = apple_rand(occupied)
        head.score = str(int(head.score) + 1)
        eat_sound.play()
```

Пользовательский интерфейс

- Главное меню с выбором цвета (def main_ui())
- Отображение текущего счёта(score)
- Отображение лучшего результата(highest score)
- Экран Game over с кнопкой перезагрузки(def los())