

Project Report: SNEK

I. Game Overview

SNEK is a traditional snake game where you control a snake and grow by eating. The more you grow the harder it gets.

II. Mechanics

As you gain more points your body will grow and your body will be more of an obstacle for you to avoid.

I. Movement

Snake creates a new body in front of itself and deletes the last body. This method creates the illusion of the snake moving. The snake will move at a constant pace until it collides with an obstacle. Movement keys consist of the arrow keys

```
def move_snek(self):  
    if self.new_block == True:  
        body_copy = self.body[:] # copy and add snek body and removes last body  
        body_copy.insert(0, body_copy[0] + self.direction)  
        self.body = body_copy[:]  
        self.new_block = False  
    else:  
        body_copy = self.body[:-1] # copy and add snek body and removes last body  
        body_copy.insert(0, body_copy[0] + self.direction)  
        self.body = body_copy[:]
```

Movement keys consist of the arrow keys

```
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_UP:
        if main_game.snek.direction.y != 1:
            main_game.snek.direction = Vector2(0, -1)
    if event.key == pygame.K_RIGHT:
        if main_game.snek.direction.x != -1:
            main_game.snek.direction = Vector2(1, 0)
    if event.key == pygame.K_DOWN:
        if main_game.snek.direction.y != -1:
            main_game.snek.direction = Vector2(0, 1)
    if event.key == pygame.K_LEFT:
        if main_game.snek.direction.x != 1:
            main_game.snek.direction = Vector2(-1, 0)
```

II. Body Growth

Snake will grow whenever it eats the food, Snake grows by 1 body every food it eats.

```
def add_block(self):
    self.new_block = True
```

III. Collision

Creates collision for snake, food, and the outer walls. If food got randomized into an existing body of the snake, the collision will detect this and re randomize it somewhere else

```
def check_collision(self):
    if self.borgir.pos == self.snek.body[0]:
        self.borgir.randomize()
        self.snek.add_block()

    for block in self.snek.body[1:]: # if borgir spawns on body, rerandomize
        if block == self.borgir.pos:
            self.borgir.randomize()
```

IV. Food Position Randomizer

After snake collects food, it will decide a random position for the next food to spawn.

```
class BORGIR:
    def __init__(self):
        self.randomize()

    def draw_borgir(self):
        borgir_rect = pygame.Rect(int(self.pos.x * cell_size), int(self.pos.y * cell_size), cell_size, cell_size)
        screen.blit(borgir, borgir_rect)  # manages characteristi

    def randomize(self):
        self.x = random.randint(0, cell_number - 1)  # randomizes x and y position
        self.y = random.randint(0, cell_number - 1)
        self.pos = Vector2(self.x, self.y)
```

V. Snake Starting Position

The snake will spawn in a fixed position

```
self.body = [Vector2(5, 10), Vector2(4, 10), Vector2(3, 10)]
self.direction = Vector2(1, 0)
self.new_block = False
```

III. Gameplay Elements

I. Sprites

All directions of each type of body of the snake sprites are required.

```
self.head_up = pygame.image.load("Sprites/Snek_up.png").convert_alpha()
self.head_right = pygame.image.load("Sprites/Snek_right.png").convert_alpha()
self.head_down = pygame.image.load("Sprites/Snek_down.png").convert_alpha()
self.head_left = pygame.image.load("Sprites/Snek_left.png").convert_alpha()

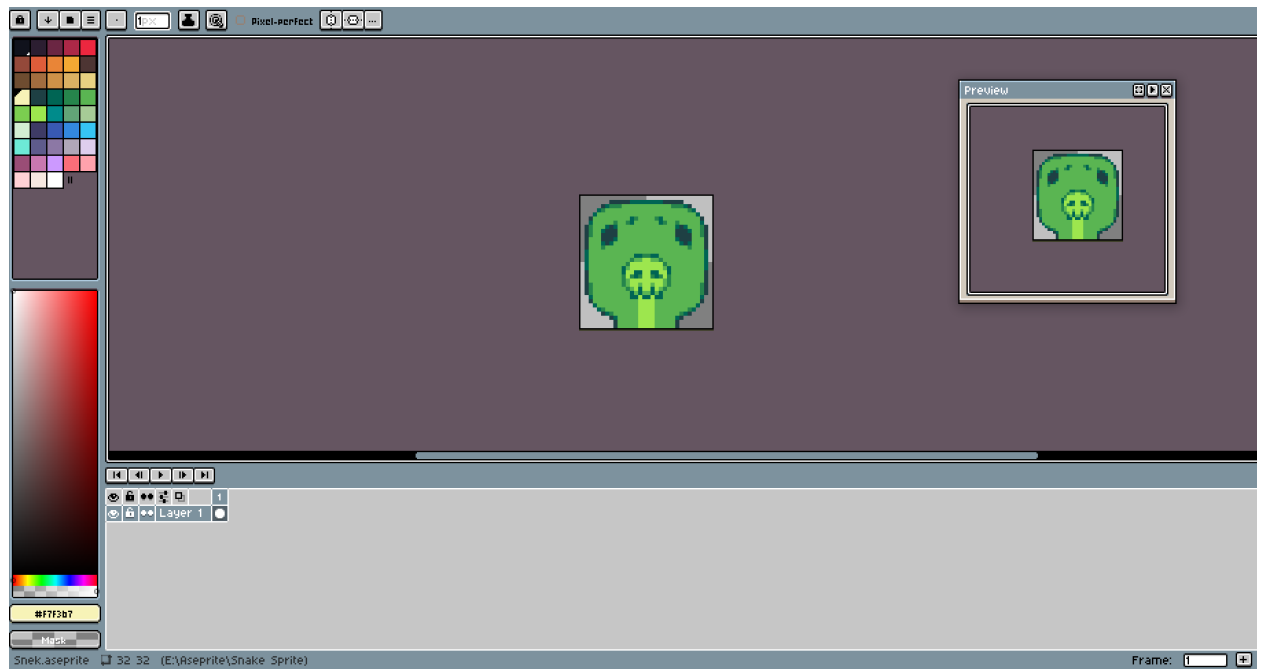
self.tail_up = pygame.image.load("Sprites/tail_up.png").convert_alpha()
self.tail_right = pygame.image.load("Sprites/tail_right.png").convert_alpha()
self.tail_down = pygame.image.load("Sprites/tail_down.png").convert_alpha()
self.tail_left = pygame.image.load("Sprites/tail_left.png").convert_alpha()


self.body_vertical = pygame.image.load("Sprites/body_vertical.png").convert_alpha()
self.body_horizontal = pygame.image.load("Sprites/body_horizontal.png").convert_alpha()

self.up_left = pygame.image.load("Sprites/up_left.png").convert_alpha()
self.up_right = pygame.image.load("Sprites/up_right.png").convert_alpha()
self.down_right = pygame.image.load("Sprites/down_right.png").convert_alpha()
self.down_left = pygame.image.load("Sprites/down_left.png").convert_alpha()


# Snek Sprites
```


Snake Sprites were created within aseprite



Snake head 

Food 

Sneak Body 

Corner Body 

Snake Tail 

The code that helps update the direction of the snake

```
def update_head_graphics(self):
    head_relation = self.body[1] - self.body[0] # selects the block before the head
    if head_relation == Vector2(1, 0):
        self.head = self.head_left # snake faces right
    elif head_relation == Vector2(-1, 0):
        self.head = self.head_right # snake faces left
    elif head_relation == Vector2(0, 1):
        self.head = self.head_up # snake faces down
    elif head_relation == Vector2(0, -1):
        self.head = self.head_down

def update_tail_graphics(self):
    tail_relation = self.body[-2] - self.body[-1] # selects the block before the head
    if tail_relation == Vector2(-1, 0):
        self.tail = self.tail_left # snake faces right
    elif tail_relation == Vector2(1, 0):
        self.tail = self.tail_right # snake faces left
    elif tail_relation == Vector2(0, -1):
        self.tail = self.tail_up # snake faces down
    elif tail_relation == Vector2(0, 1):
        self.tail = self.tail_down
```

II. Checkerboard Background

Detects odd tiles to recolor

```
def draw_bg(self):
    bg_1 = (234, 234, 255)

    for row in range(cell_number):
        if row % 2 == 0:
            for col in range(cell_number):
                if col % 2 == 0:
                    bg_rect = pygame.Rect(col * cell_size, row * cell_size, cell_size, cell_size)
                    pygame.draw.rect(screen, bg_1, bg_rect)
                else:
                    for col in range(cell_number):
                        if col % 2 != 0:
                            bg_rect = pygame.Rect(col * cell_size, row * cell_size, cell_size, cell_size)
                            pygame.draw.rect(screen, bg_1, bg_rect)
```

IV. Interface

The score is calculated by calculating the body gained by subtracting body count by 3, which is the starting body amount

```
score_text = str(len(self.snek.body) - 3)
```

Places scores at the top left, Score uses custom font, add an icon of food right beside the score, and adds background

```
score_surface = game_font.render(score_text, True, (15, 15, 15))
score_x = int(60) # positions the score from top left
score_y = int(40)
score_rect = score_surface.get_rect(center=(score_x, score_y)) # gives score rect
borgir_rect = borgir.get_rect(midright=(score_rect.left - 5, score_rect.centery))
bg_rect = pygame.Rect(borgir_rect.left - 4, borgir_rect.top - 5, borgir_rect.width + score_rect.width + 10, borgir_rect.height + 10)
```