Surf Rehab

Title: Surf Rehab

TAVOLARO, Luiz; MOTA, João; NUR, Aaron

Faculdade de Computação e Informática– Universidade Presbiteriana Mackenzie (UPM)
São Paulo – SP – Brazil

{10418215@mackenzista.com.br, 10418226@mackenzista.com.br,
10417095@mackenzista.com.br}

Abstract. *At work, we created a game to help patients in rehabilitation who have partially lost the movement of their lower limbs. Using Python, we developed a game that uses a skateboard for the directional keyboard controls, creating an interactive environment for the player. Our game encourages the user to replicate movements similar to those practiced in regular physical therapy. The game is an endless runner, set in a surfing scenario where the surfer (the main character) must avoid obstacles that appear in their path and try to go as far as possible.*

Keywords: Python, Surf, Rehabilitation, Movement

Resumo. *No trabalho, criamos um jogo com o intuito de auxiliar pacientes em reabilitação, os quais perderam parcialmente o movimento dos membros inferiores. Utilizando Python, desenvolvemos um jogo que utiliza um skateboard para os controles de direcionais do teclado e dessa forma criamos um ambiente de interação com o jogador, sendo assim, o nosso jogo incentiva que o usuário replique movimentos semelhantes aos que ele pratica em uma fisioterapia comum. O jogo é um endless runner, que se passa em um cenário de surf onde o surfista (personagem principal) deve desviar dos obstáculos que surgem em seu caminho e tentar chegar o mais longe que conseguir.*

Palvras-chave: *Python, Surf, Reabilitação, Movimento.*

1. Introdução

Baseado na proposta da disciplina, o grupo buscou algum tema que pudesse ajudar na reabilitação de pacientes que sofreram com a perda parcial do movimento das partes inferiores, sendo as pernas ou tornozelos.

O resultado que a equipe resolveu desenvolver uma forma de integrar os jogos digitais com a recuperação do paciente em seu tratamento e na recuperação de seus movimentos. O jogo desenvolvido pelo time tem em sua jogabilidade elementos que estimulam o paciente de forma semelhante ao tratamento em fisioterapia, visto que foi pensado para ser jogado utilizando um skateboard ao invés de mouse e teclado.

O time escolheu por um tema de surf, já que a prática do esporte necessita de bastante atividade motora nas partes inferiores e muitos praticantes do esporte sofrem com lesões e dores nessa região.

O jogo tem como objetivo desviar dos obstáculos que estão em seu caminho enquanto tenta chegar o mais longe possível, assim além de estimular a movimentação da região requerida, exige que o jogador tenha atenção suficiente para desviar dos obstáculos. Foi desenvolvido em Python utilizando a biblioteca Pygame.

2. Descrição do problema

Atualmente a reabilitação de pacientes cujo perderam parcialmente a habilidade de se locomover pode ser muito doloroso e demorado, sendo assim meios para acelerar esse processo e diminuir a dor do tratamento estão cada vez mais sendo adotados para tentar melhorar a recuperação e a satisfação dos pacientes.

Como dito na introdução do documento, o jogo foi pensado para ser jogado em um skateboard, que simula as teclas de um teclado comum, onde o jogador ao se inclinar para cada um dos lados do skate, ele selecionará os direcionais do teclado.

Desse modo o nosso jogo trabalha os movimentos dos membros inferiores dos pacientes assim como aspectos cruciais para a locomoção no dia a dia como equilíbrio, força coordenação atenção entre outros.

3. Justificativa

Ao contrário das abordagens comuns, inserindo um jogo no tratamento do paciente pode incentivá-lo a jogar mais vezes, de forma que o game tem sua dificuldade progressiva e o jogador precisará práticar repetidas vezes para que consiga o progredir cada vez mais no game, tentando cada vez chegar mais longe e batendo seu record pessoal a cada partida.

4. Imagens

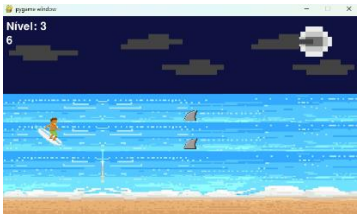

Figura 1 – Nível 1 **Figura 2 – Nível 2** **Figura 3 – Nível 3**

5. Descrição do game

5.1.1 Códigos

5.1.1.1 character_selection ()

```
def character_selection():
    characters = [
        pygame.image.load("img/surfer.png").convert_alpha(),
        pygame.image.load("img/surfer2.png").convert_alpha(),
        pygame.image.load("img/surfer3.png").convert_alpha(),
    ]
    character_names = ["surfer.png", "surfer2.png", "surfer3.png"]
    character_rects = []
    selected_index = 0

    for i in range(len(characters)):
        characters[i] = pygame.transform.scale(characters[i], (100, 150))
        character_rects.append(characters[i].get_rect(center=(SCREEN_WIDTH // 2 - 150 + i * 150, SCREEN_HEIGHT // 2)))

    running = True
    while running:
        SCREEN.fill(BLUE)
        draw_text("Selecione seu personagem", BIG_FONT, WHITE, SCREEN, SCREEN_WIDTH // 2, 50)

        for i, char in enumerate(characters):
            SCREEN.blit(char, character_rects[i])
            if i == selected_index:
                pygame.draw.rect(SCREEN, WHITE, character_rects[i], 3)

        draw_text("Use as setas para mudar, ENTER para confirmar", FONT, WHITE, SCREEN, SCREEN_WIDTH // 2, SCREEN_HEIGHT - 50)

        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                exit()
            if event.type == KEYDOWN:
                if event.key == K_LEFT:
                    selected_index = (selected_index - 1) % len(characters)
                elif event.key == K_RIGHT:
                    selected_index = (selected_index + 1) % len(characters)
                elif event.key == K_RETURN:
                    return character_names[selected_index]

        pygame.display.flip()
```

Essa função tem como objetivo permitir que o jogador selecione o personagem que ele deseja jogar. Ela inicia carregando as imagens das opções de personagens existentes, fazendo tratamentos nas imagens e criando uma lista com o tamanho delas. Em seguida ele desenha a tela, onde ele vai permitir que o jogador escolha o personagem, e em seguida aguarda a próxima ação do usuário.

5.1.1.2 draw_background ()

```
def draw_bg(level=1, scroll=0):
    if level == 1:
        sun_img = suns[0]
        background_img = backgrounds[0]
        cloud_img = clouds[0]
        speed = 1
    elif level == 2:
        sun_img = suns[1]
        background_img = backgrounds[1]
        cloud_img = clouds[1]
        speed = 1
    elif level == 3:
        sun_img = suns[2]
        background_img = backgrounds[2]
        cloud_img = clouds[2]
        speed = 1

    SCREEN.blit(background_img, (0, 0))
    SCREEN.blit(sun_img, (SCREEN_WIDTH - sun_img.get_width(), 0))

    sea_img = seas[0]
    sea2_img = seas[1]
    cur_sea_img = sea_img

    for x in range(100):
        SCREEN.blit(cloud_img, ((x * bg_width) - scroll * speed, 0))
        SCREEN.blit(cur_sea_img, ((x * bg_width) - scroll * 2 * speed, 0))
        if(cur_sea_img == sea_img):
            cur_sea_img = sea2_img
        else:
            cur_sea_img = sea_img
```

Essa função tem como objetivo desenhar o fundo de tela, e sendo assim também dá movimento ao game. Também baseado no nível que o jogador está, ela é responsável por dizer qual será o cenário desenhado

5.1.1.3 main ()

```python
def main():
    start_screen()
    selected_character = character_selection()
    surfer = Surfer("Player", selected_character)

    all_sprites = pygame.sprite.Group()
    all_sprites.add(surfer)

    level = 1
    scroll = 0

    obstacle_timer = 0
    obstacle_interval = 2000
    obstacle_passed_count = 0
    obstacles = pygame.sprite.Group()
    obstacle_w = 30
    obstacle_h = LANE_HEIGHT * 0.4

    clock = pygame.time.Clock()
    running = True

    while running:
        for event in pygame.event.get():
            if event.type == QUIT:
                running = False
            if event.type == KEYDOWN:
                if event.key == K_ESCAPE:
                    pause_screen()

        scroll = scroll + 2

        obstacle_timer += clock.get_time()
        if obstacle_timer >= obstacle_interval:
            obstacle_timer = 0
            lanes = random.sample(LANE_Y_POSITIONS[:-1], 2)
            obstacle_type = "seagul" if level == 1 else "octopus" if level == 2 else "sharkfin"
            for lane_y in lanes:
                obstacle = Obstacle(lane_y + LANE_HEIGHT // 2, obstacle_w, obstacle_h, speed=2 if level == 3 else 1, obstacle_type=obstacle_type)
                obstacles.add(obstacle)
                all_sprites.add(obstacle)

        if pygame.sprite.spritecollideany(surfer, obstacles):
            running = False

        for obstacle in obstacles:
            if obstacle.rect.right < surfer.rect.left:
                obstacle_passed_count += 1
                obstacle.kill(

        score_color = (0, 0, 0)

        if obstacle_passed_count // 2 >= 5:
            level = 3
        elif obstacle_passed_count // 2 >= 10:
            level = 2
        if level == 3:
            score_color = (255, 255, 255)

        surfer.update_speed(level)

        draw_bg(level=level, scroll=scroll)

        all_sprites.update()
        all_sprites.draw(SCREEN)

        level_text = FONT.render(f"Nivel: {level}", True, score_color)
        score_text = FONT.render(f"{obstacle_passed_count // 2}", True, score_color)
        SCREEN.blit(level_text, (10, 10))
        SCREEN.blit(score_text, (10, 40))

        pygame.display.flip()
        clock.tick(60)

    pygame.quit()
```

Essa é a função principal do jogo. Tem como objetivo comandar tudo que acontece e quando acontece. Cria a tela, o jogador, cria e aplica as regras para o surgimento dos obstáculos e realiza a contagem de pontos.

5.1.1.4 class Surfer ()

```
class Surfer(pygame.sprite.Sprite):
    def __init__(self, name, image_path):
        super(Surfer, self).__init__()
        self.name = name
        self.alive = True
        self.position = pygame.math.Vector2(100, SCREEN_HEIGHT // 2)
        self.speed = 5
        self.image = pygame.image.load(f"img/{image_path}").convert_alpha()
        self.image = pygame.transform.scale(self.image, (60, LANE_HEIGHT))
        self.rect = self.image.get_rect()
        self.rect.center = self.position

    def update(self):
        keys = pygame.key.get_pressed()

        if keys[K_UP] and self.position.y > TOP_LIMT:
            self.position.y -= self.speed
        elif keys[K_DOWN] and self.position.y < BOTTOM_LIMT:
            self.position.y += self.speed

        self.rect.center = self.position

    def update_speed(self, level):
        if level == 1:
            self.speed = 5
        elif level == 2:
            self.speed = 5
        elif level == 3:
            self.speed = 7
```

Esse é a classe do surfista, nela são definidos os atributos necessários e os métodos que são utilizados para sua movimentação e alteração da sua velocidade.

5.1.1.5 class Obstacle ()

```
class Obstacle(pygame.sprite.Sprite):
    OBSTACLE_TYPES = {
        "seagul": ["seagul.png", "seagul2.png"],
        "octopus": ["octopus.png", "octopus2.png"],
        "sharkfin": ["sharkfin.png", "sharkfin2.png"]
    }

    def __init__(self, lane_y, w, h, speed, obstacle_type):
        super().__init__()
        if obstacle_type not in self.OBSTACLE_TYPES:
            raise ValueError(f"Tipo de obstáculo inválido: {obstacle_type}")

        self.images = [
            pygame.image.load(f"img/{img}").convert_alpha() for img in self.OBSTACLE_TYPES[obstacle_type]
        ]
        self.images = [pygame.transform.scale(img, (w, int(h))) for img in self.images]
        self.image = self.images[0]
        self.rect = self.image.get_rect()
        self.rect.x = SCREEN_WIDTH
        self.rect.y = lane_y
        self.speed = 6

        self.animation_index = 0
        self.animation_timer = 0
        self.animation_interval = 2000
```

Nessa classe, definimos o obstáculo assim como todos os atributos necessários para e os métodos necessários para a sua movimentação e animações.