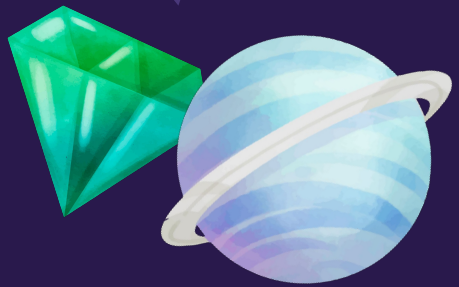
Jungle Cruise

Projeto de CES22
Lucas Maia, Luca Rassi e Gabriel Lucena

Sumário

Objetivo



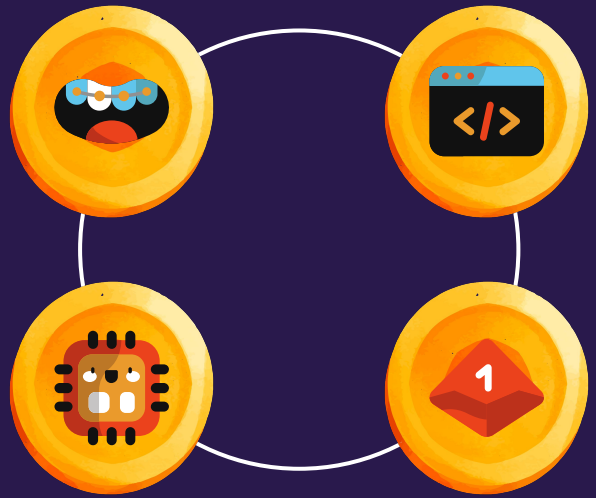
Objetivos do Projeto

Aprendizado

Experiência mais próxima do que enfrentamos no mercado de trabalho

Gestão

Experiência de gestão de um projeto tecnológico em equipe



POO

Desenvolver e aplicar os conhecimentos adquiridos

Passar :)

Ser aprovado no semestre

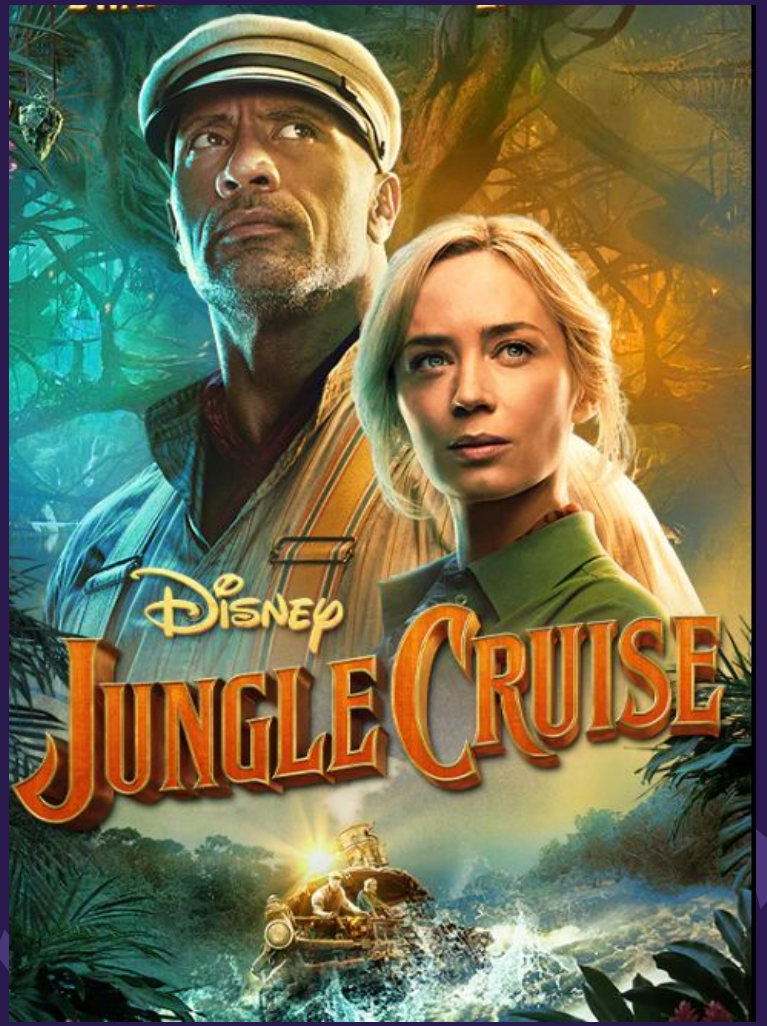
02

Motivação

...

Jungle Cruise

Jogo baseado nas características e ambientação do filme “Jungle Cruise”



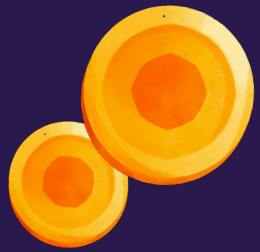
O Jogo

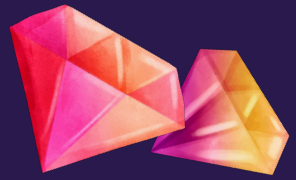
Desvio de obstáculos

Mecânica de jogabilidade bi-direcional

Progressão de níveis

Conforme o sucesso na trajetória, o jogador progride de fase com dificuldade gradualmente aumentada





IMPLEMENTAÇÃO

PyGame

Elementos básicos

```
class Player(Elements):
    def player():
        return True

    def show(self,s):
        # image appearing
        boat = pygame.transform.scale(aux.boat, (s.width*0.1, s.height*0.15))
        s.screen.blit(boat,(self.x,self.y))
```

CLASSE PLAYER: elaborada para o próprio jogador, que no contexto do jogo seria o barco.

Elementos básicos

```python
class Obstacle(Elements):
    def __init__(self, id, obs_type, x, y):
        self.id = id
        self.obs_type = obs_type
        super().__init__(x, y)

    def player():
        return False

    def show(self,s):
        # image appearing
        if self.obs_type == 
            cruise = pygame.transform.rotate(aux.cruise, -90)
            cruise = pygame.transform.scale(cruise, (s.width*0.1, s.height*0.15))
            s.screen.blit(cruise,(self.x,self.y))
        elif self.obs_type == 1:
            stone = pygame.transform.scale(aux.stone, (s.width*0.1, s.height*0.15))
            s.screen.blit(stone,(self.x,self.y))
        elif self.obs_type == 2:
            pirate = pygame.transform.rotate(aux.pirate, -180)
            pirate = pygame.transform.scale(pirate, (s.width*0.1, s.height*0.15))
            s.screen.blit(pirate,(self.x,self.y))
        elif self.obs_type == 3:
            aligator = pygame.transform.rotate(aux.aligator, -180)
            aligator = pygame.transform.scale(aligator, (s.width*0.1, s.height*0.15))
            s.screen.blit(aligator,(self.x,self.y))
```

transform: Any

CLASSE OBSTACLE: Foi também definida uma classe para os obstáculos existentes no jogo, tais como pedras no caminho do jogador (barco), piratas e aligators.

Ambientação

```python
class Environment:
    def __init__(self):
        self.bumped = False
        self.obstacle_speed = 10
        self.obs = 0
        self.y_change = 0
        self.obstacles_passed = 0
        self.score = 0
        self.level = 0
        self.obstacles = {}
        self.obs_id = -1

    def generate_obstacle(self, s):
        self.obs_id += 1
        obs_type = random.randint(0,3)
        obs = Elements.Obstacle(self.obs_id, obs_type, random.uniform(0.15, 0.75)*s.width, -random.uniform(0, 0.5)*s.width)
        self.obstacles[self.obs_id] = obs
        return obs

    def new_level(self,s):
        self.level += 1
        self.obstacle_speed += self.level
        if self.level % 2:
            self.generate_obstacle(s)

    def destroy_obstacle(self, obs):
        self.obstacles.pop(obs.id)
```

CLASSE ENVIROMENTPara caracterizar o ambiente no qual o jogo acontece, foi implementada uma classe Environment, que possui métodos capazes de gerar e destruir obstáculos, além de definir consequências que envolvem tais obstáculos.

Eventos

```
def text_object(text, font):
    textSurface = font.render(text, True, (255,255,255))
    return textSurface, textSurface.get_rect()

def event_message(s,msg):
    myfont = pygame.font.SysFont("None", int(s.height/6))
    game_over = myfont.render(msg, 1, (0,0,0))
    dim = game_over.get_size()
    s.screen.blit(game_over, (int((s.width-dim[0])/2),int((s.height-dim[1])/2)))
    pygame.display.update()

def quit_event(event):
    if event.type == pygame.QUIT:
        pygame.quit()
        quit()
        sys.exit()
```

MÉTODOS DE MENSAGEM:
Display de mensagens na tela para o jogador.

Interface

CLASE SCREEN: A implementação da interface se resume a uma classe nomeada de Screen, alguns dos métodos mais importantes dela são:

```python
def intro_page(self):
    background_font_path = "font/quicksilver-fast-font/QuicksilverFastRegular-DO3oE.ttf"
    intro_image = pygame.image.load("images/background.jpg")
    intro_image = pygame.transform.scale(intro_image,self.dimensions)

    self.screen.blit(intro_image,(0,0))
    font = pygame.font.Font(background_font_path, int(50*self.height/600))
    dim = font.size("Jungle Cruise Game")
    title = font.render("Jungle Cruise Game", True, (192,192,192))
    self.screen.blit(title,(int((self.width-dim[0])/2),int(50*self.height/600)))

    self.intro_loop()

def intro_loop(self):
    intro = True
    while intro:
        for event in pygame.event.get():
            events.quit_event(event)
        self.mouse = 0
        self.click = 0
        self.mouse = pygame.mouse.get_pos()
        self.click = pygame.mouse.get_pressed()

        self.screen_button('start')
        self.screen_button('instruction')
        self.screen_button('quit')

        pygame.display.update()
```

```python
def screen_button(self,button):
    button = aux.buttons[button]
    dim = button['dim']
    x0 = self.width*dim[0]
    y0 = self.height*dim[1]
    x1 = self.width*dim[2]
    y1 = self.height*dim[3]
    if self.mouse[0] > x0 and self.mouse[0] < x1 and self.mouse[1] > y0 and self.mouse[1] < y1:
        pygame.draw.rect(self.screen, button['alt_color'], (x0,y0,x1-x0,y1-y0))
        if self.click == (True, False, False):
            for f in button['fs']:
                exec(f)
    else:
        pygame.draw.rect(self.screen,button['color'],(x0,y0,x1-x0,y1-y0))

    smallText = pygame.font.Font("freesansbold.ttf", int((y1-y0)/3))
    textSurface, textRect = events.text_object(button['text'], smallText)
    textRect.center = ((x0+x1)/2,(y0+y1)/2)
    self.screen.blit(textSurface,textRect)
```

Interface

```
def countdown(self):
    countdown = ['3','2','1','GO!']
    self.env = Environment()
    pygame.display.update()
    for count in countdown:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                events.quit_event(event)

        self.screen.fill((30,116,187))
        self.background()
        self.score_card()
        largetext = pygame.font.Font("freesansbold.ttf",int(115*self.height/600))
        textSurf,textRect = events.text_object(count,largetext)
        textRect.center = ((self.width/2),(self.height/2))
        self.screen.blit(textSurf,textRect)

        pygame.display.update()
        self.clock.tick(1)

    self.game_page()
```

```
def background(self):
    self.screen.fill((30,116,187))
    forest = aux.forest
    forest = pygame.transform.scale(forest, (self.width*.15, self.height))
    self.screen.blit(forest, (0,0))
    self.screen.blit(forest, (.85*self.width,0))

# function for score card
def score_card(self):
    font = pygame.font.SysFont(None, 35)
    passed = font.render("Passed: "+str(self.env.obstacles_passed), True, (255,255,255))
    score = font.render("Score: "+str(self.env.score), True, (255,255,255))
    level = font.render("LEVEL: "+str(self.env.level), True, (255,255,255))
    self.screen.blit(passed, (0,int(50*self.height/600)))
    self.screen.blit(score, (0,int(100*self.height/600)))
    self.screen.blit(level, (0,int(150*self.height/600)))
```

04

O pós-projeto...

O que ainda poderia
ser feito e melhorado

O que poderia ser otimizado

Obstáculos

Sobreposição no late-game

Interface

Interface e artes mais bonitas

Sons

Desenvolver um pouco mais a ambientação sonora