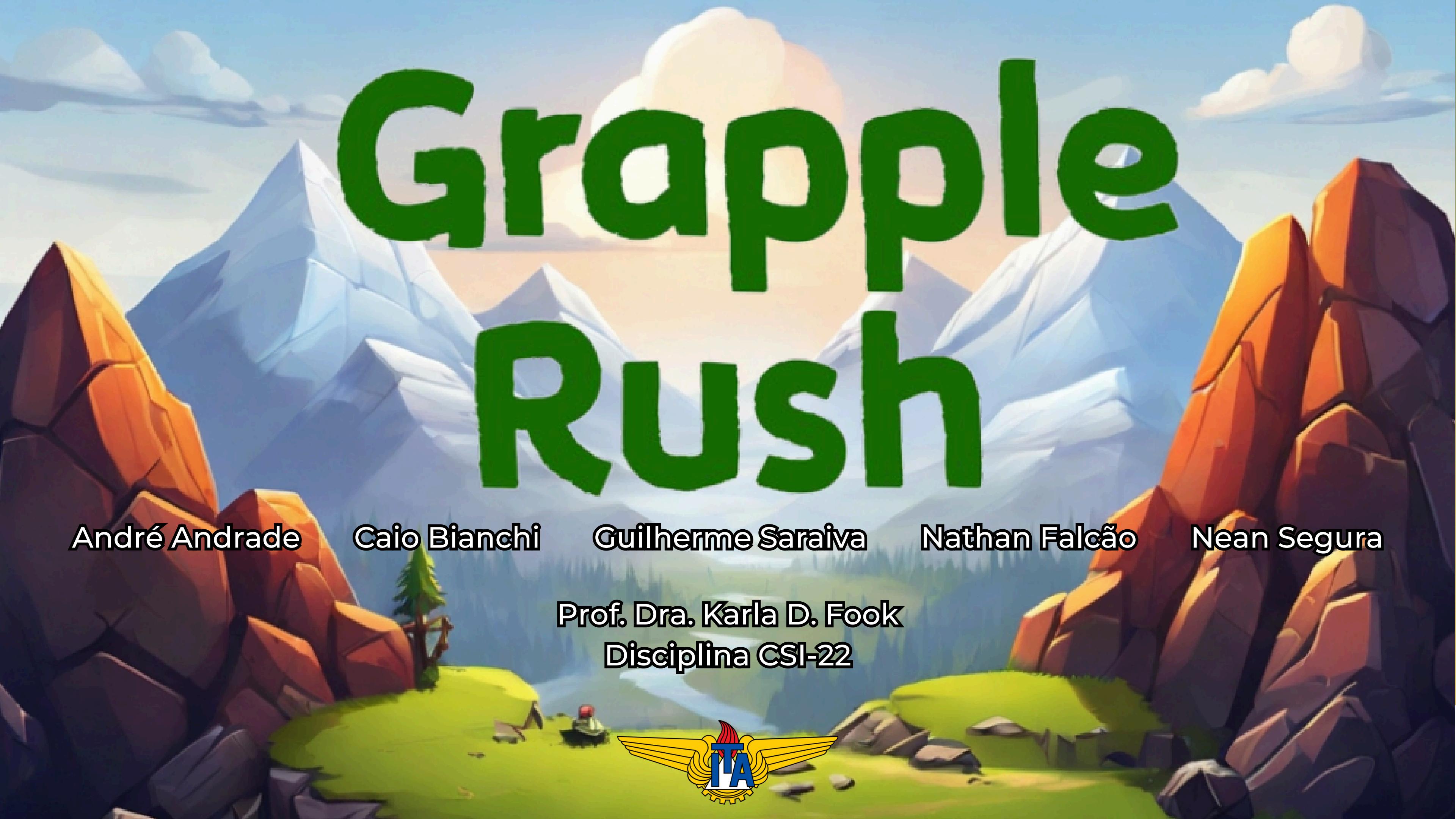


Grapple Rush



André Andrade

Caio Bianchi

Guilherme Saraiva

Nathan Falcão

Nean Segura

Prof. Dra. Karla D. Fook
Disciplina CSI-22



A ideia do Jogo

"*Grapple Rush*" é um jogo de plataforma onde os jogadores usam um gancho de escalada para se mover pelo ambiente e desviar de obstáculos. Se o jogador morrer, ele volta para o início da fase.

O jogo possui um editor de níveis integrado.

Objetivo: terminar todas as fases no menor tempo possível.



Tutorial - Instalação

- Clonar o repositório (git clone)

https://github.com/andresafira/Grapple_Rush

- Rodar o arquivo "main.py" dentro da pasta "src"



Tutorial - Gameplay

Movimentação:

- w: cima (pulo)
- a: esquerda
- s: direita

Gancho de escalada:

- Clique (mouse): lançar o gancho no ponto em que estiver
- Espaço: soltar o gancho

Pressione esc para voltar para o Menu.

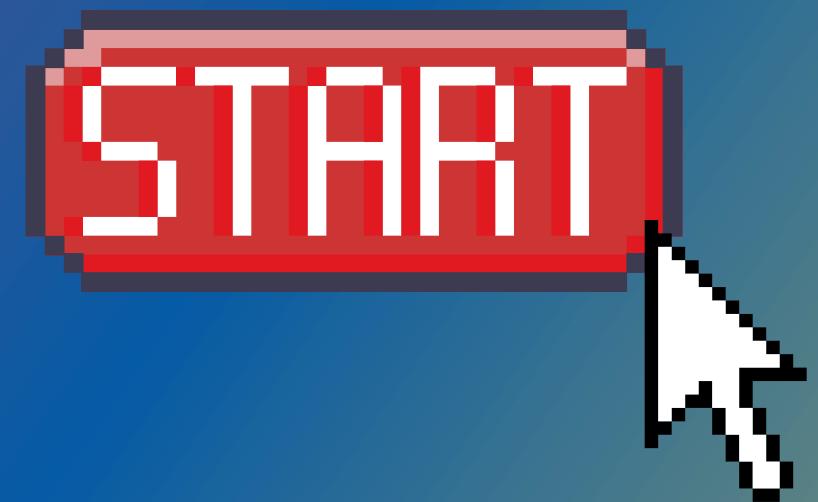


Tutorial - Level Editor

- Up e Down (setas): trocar o level
- Botão esquerdo (mouse): colocar o bloco
- Botão direito (mouse): apagar o bloco



Breve Demonstração



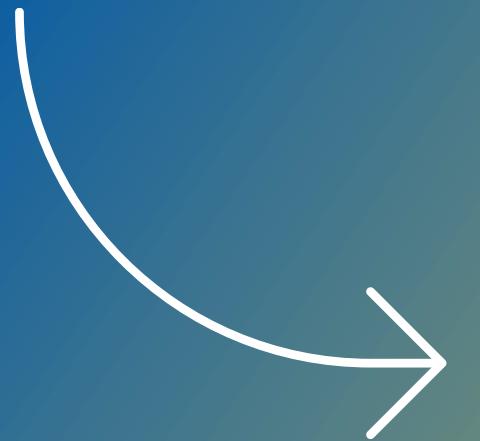
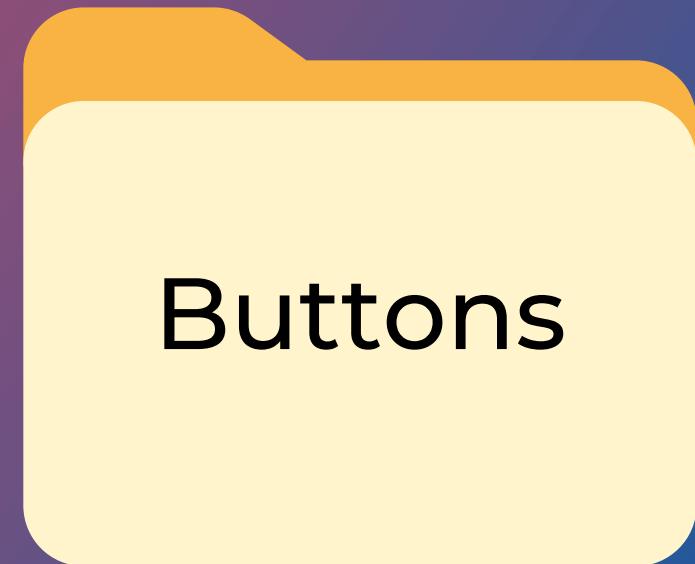
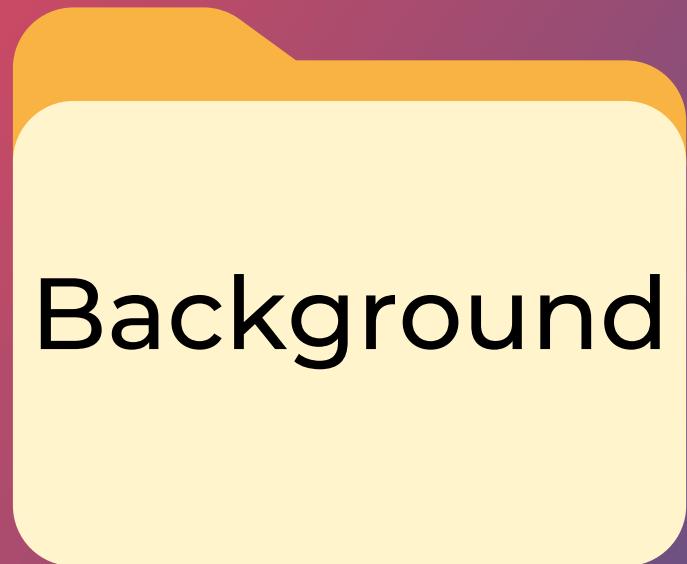
Técnicas de POO

O jogo foi implementado utilizando técnicas de Programação Orientada a Objetos aprendidas na disciplina e utilizando a Biblioteca Pygame.



Organização dos Arquivos

Guardam as imagens, sprites e trilha sonora:



Hook, Idle,
Run, Jump,
Tiles

Constants

Guarda as constantes de jogo como:

- Dimensões do Player
- Dimensões da Tela
- Constantes de Movimento
- etc

Levels

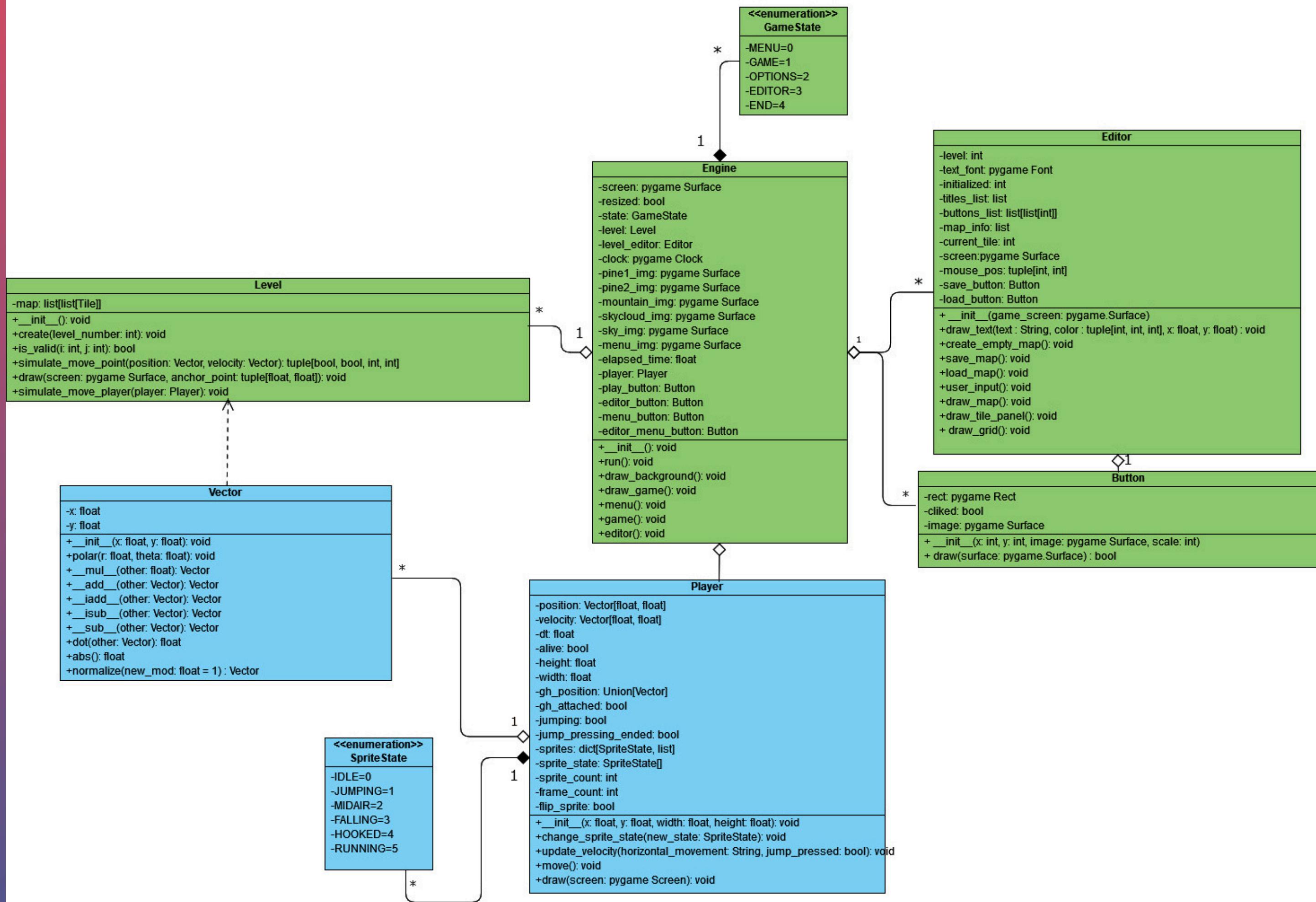
Guarda as informações dos levels do jogo, tanto os nativos quanto os criados pelo level editor.

Geometry

Arquivos com classes e funções auxiliares

- Class Vector

Classes



Engine	
-screen:	pygame Surface
-resized:	bool
-state:	GameState
-level:	Level
-level_editor:	Editor
-clock:	pygame Clock
-pine1_img:	pygame Surface
-pine2_img:	pygame Surface
-mountain_img:	pygame Surface
-skycloud_img:	pygame Surface
-sky_img:	pygame Surface
-menu_img:	pygame Surface
-elapsed_time:	float
-player:	Player
-play_button:	Button
-editor_button:	Button
-menu_button:	Button
-editor_menu_button:	Button
+__init__():	void
+run():	void
+draw_background():	void
+draw_game():	void
+menu():	void
+game():	void
+editor():	void

- A classe principal que representa a engine do jogo.
- Inicializa o Pygame, o jogo no modo MENU, configura a exibição, manipula os estados do jogo, executa o loop do jogo e contém métodos para desenhar o background, o level, o player, o menu e o editor de levels.

Level

```
-map: list[list[Tile]]  
+__init__(): void  
+create(level_number: int): void  
+is_valid(i: int, j: int): bool  
+simulate_move_point(position: Vector, velocity: Vector): tuple[bool, bool, int, int]  
+draw(screen: pygame Surface, anchor_point: tuple[float, float]): void  
+simulate_move_player(player: Player): void
```

- Responsável por representar um nível do jogo.
- Possui métodos para carregar mapas de arquivos JSON, simular movimentos de pontos e jogadores no mapa para lidar com colisões, e desenhar o mapa na tela do jogo.

Editor

```
-level: int  
-text_font: pygame Font  
-initialized: int  
-titles_list: list  
-buttons_list: list[list[int]]  
-map_info: list  
-current_tile: int  
-screen:pygame Surface  
-mouse_pos: tuple[int, int]  
-save_button: Button  
-load_button: Button  
  
+ __init__(game_screen: pygame.Surface)  
+draw_text(text : String, color : tuple[int, int, int], x: float, y: float) : void  
+create_empty_map(): void  
+save_map(): void  
+load_map(): void  
+user_input(): void  
+draw_map(): void  
+draw_tile_panel(): void  
+ draw_grid(): void
```

Player

```
-position: Vector[float, float]  
-velocity: Vector[float, float]  
-dt: float  
-alive: bool  
-height: float  
-width: float  
-gh_position: Union[Vector]  
-gh_attached: bool  
-jumping: bool  
-jump_pressing_ended: bool  
-sprites: dict[SpriteState, list]  
-sprite_state: SpriteState[]  
-sprite_count: int  
-frame_count: int  
-flip_sprite: bool  
  
+ __init__(x: float, y: float, width: float, height: float): void  
+change_sprite_state(new_state: SpriteState): void  
+update_velocity(horizontal_movement: String, jump_pressed: bool): void  
+move(): void  
+draw(screen: pygame Screen): void
```

- Uma classe que define o editor de níveis, definindo as configurações para desenhar o mapa, salvar, adicionar novos elementos e carregar.

- Representa o personagem controlado pelo jogador.
- Gerencia a posição, velocidade, estado e interações do jogador, incluindo o uso do gancho, movimento e desenho na tela.

<<enumeration>>

Sprite State

-IDLE=0
-JUMPING=1
-MIDAIR=2
-FALLING=3
-HOOKED=4
-RUNNING=5

<<enumeration>>

Game State

-MENU=0
-GAME=1
-OPTIONS=2
-EDITOR=3
-END=4

Vector

-x: float
-y: float

+ __init__(x: float, y: float): void
+ polar(r: float, theta: float): void
+ __mul__(other: float): Vector
+ __add__(other: Vector): Vector
+ __iadd__(other: Vector): Vector
+ __isub__(other: Vector): Vector
+ __sub__(other: Vector): Vector
+ dot(other: Vector): float
+ abs(): float
+ normalize(new_mod: float = 1) : Vector

Button

-rect: pygame Rect
-clicked: bool
-image: pygame Surface

+ __init__(x: int, y: int, image: pygame Surface, scale: int)

+ draw(surface: pygame.Surface) : bool

Funcionamento Explicado

Mecânica da Movimentação

$$a_x = K_p \cdot (v_{target} - v_{current})$$

$$a_y = -g \cdot k$$

$$k = \begin{cases} 0.4 & \text{para personagem em subida enquanto aperta o botão de pulo} \\ 1.2 & \text{para personagem em queda livre} \\ 1.0 & \text{caso contrário} \end{cases}$$

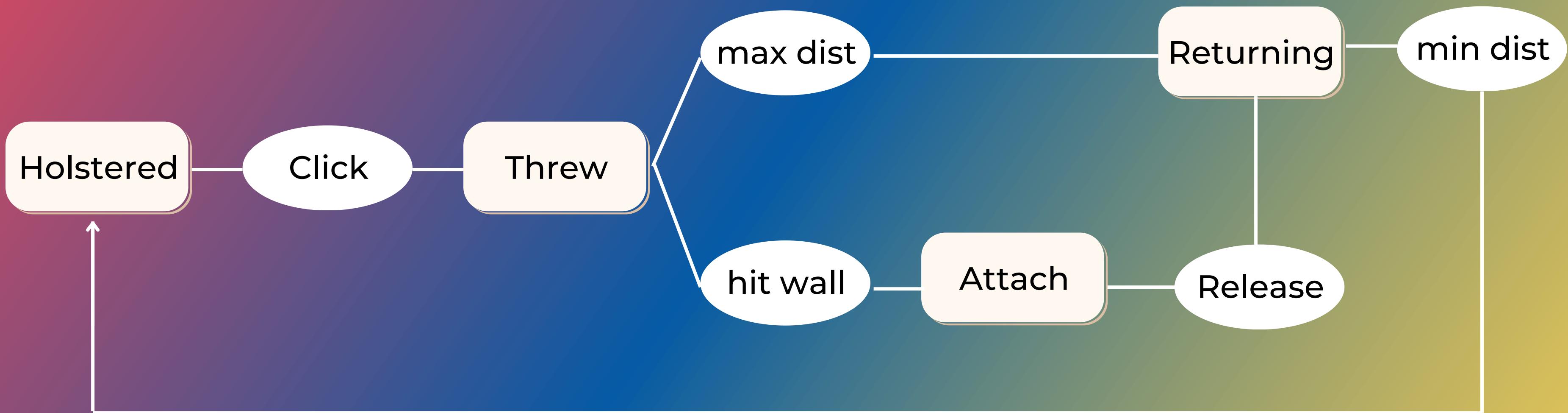


Detecção de Colisão

- Avalia a colisão de vários pontos do player simultaneamente.
 - A função *simulate_move_player* chama *simulate_move_point* para esses pontos
 - A função *simulate_move_point* analisa a posição do ponto em um pequeno tempo futuro
- Se estiver dentro do bloco, a velocidade no eixo de colisão é zerada e a posição é atualizada para o ponto de colisão.



Mecânica do Gancho



Funcionamento do Editor

- Cria uma tela com grid e um espaço para selecionar qual bloco adicionar.
- Para adicionar algo, basta clicar no bloco desejado e então na posição da grid.
- Para remover, basta apertar com o botão direito.
- O mapa é salvo no formato de uma lista de listas em um arquivo json.



