

# Projeto Final de PDS2

---

Cauã M., Hugo S., Murilo N. e Otavio L.

# Motivação

Criar demo de jogo 2D com estilo 8-bit visualização top-down, inspirado em “FAITH: The Unholy Trinity”

Contendo:

- Menu inicial e menu de pausa
- Movimentação livre no mapa
- Mapa interativo e variável
- Sistema de colisão
- Inimigos perseguindo o personagem

## PROBLEMA: Estados do jogo

- Evitar loop para cada estado
- Evitar muitos ifs

## **SOLUÇÃO:**

Loop com classe abstrata (polimorfismo)

# Estados do jogo

```
void Game::run() {
    sf::Clock clock{};
    sf::Event event{};
    while (window.isOpen()) {
        GameState& currentState = *stateStack.back();

        float dt = clock.restart().asSeconds();
        while (window.pollEvent(event)) {
            currentState.handleInput(event);
        }
        currentState.update(dt);
        window.clear();
        currentState.render(window);
        window.display();

        maintainStates();
    }
}
```

## PROBLEMA: Alterar o estado do jogo

- Evitar trocar de estado no meio da iteração
- Manter 2 estados simultaneamente (pause não reinicia o estado de jogo)

## **SOLUÇÃO:**

- Fila fifo com estruturas de mudança de estado

# Alterar o estado do jogo

## Game

### Responsabilidades:

- Fila de estados do jogo
- Muda o estado atual
- Loop principal
- Janela de renderização

### Colaborações:

- PendingAction
- GameState

## PendingAction

### Responsabilidades:

- Tipo de ação (push, pop, change)
- Referência ao novo estado

### Colaborações:

- Game

## PROBLEMA: Gerenciar os mapas

- Evitar que todos os mapas estejam carregados na memória
- Customização no mapa sem precisar de recompilar
- Evitar muitos ifs

## **SOLUÇÃO:**

- Armazenar cada área do mapa em um json
- Carregar as entidade apenas da área atual
- Arquitetura desacoplada e escalável

# Gerenciar os mapas

## MapArea

### Responsabilidades:

- Parse json
- Extrai entidades
- Define posição inicial do player nos mapas

### Colaborações:

- MapEntity
- PlayState

## MapEntity

### Responsabilidades:

- Sprite
- Hitbox
- Gatilho

### Colaborações:

- MapArea
- PlayState

# Implementação: Gerenciar os mapas

- 3x3 telas para formar o mapa global
- Gera novo mapa quando player entra em colisão com entidade de gatilho
- Interpreta o esquema fixo de json e extrai as entidades
- Se o json não especificar valores para as bordas e os spawns da área valores padrões são adicionados
- Entidades são carregadas se seu intervalo de vida útil conter o checkpoint atual

## PROBLEMA: Gerenciador de recursos

- Evitar carregar os mesmos recursos várias vezes

## **SOLUÇÃO:**

- Carregar todos os recursos no início do estado
- Armazenar os recursos em um mapa
- Retornar referência ao recurso por um id

# Gerenciador de recursos

## ResourceManager

### Responsabilidades:

- Carrega texturas
- Carrega áudios
- Carrega fontes
- Armazena mídia
- Retorna mídia por nome

### Colaborações:

- MenuState
- PlayState
- PausedState