Introdução ao Desenvolvimento de Jogos – Turma A
Professora: Carla Denise Castanho (carlacastanho@cic.unb.br)
Monitores: Davi "Doom" Diniz (khenstot@gmail.com)
Leonardo Guilherme (leonardo.guilherme@gmail.com)
Luiggi Monteiro Reffatti (lmreffatti@gmail.com)

Trabalho 2 – TileMap

1. Classe TileSet : Armazenando as Sprites dos Tiles

TileSet
- tileWidth, tileHeight : int - lines, columns: int - tileSet : Sprite* - vTiles : std::vector<Sprite *>* - destRect : SDL_Rect*
+ Tileset(filePath : std::string, lines : int, columns : int); + Tileset(tileWidth : int, tileHeight : int); + addTile(filePath : std::string) : void + render(index : int, posX : float, posY : float) : void + usingSingleFile() : bool + getTileWidth() : int + getTileHeight() : int

A classe TileSet será responsável por armazenar todas as Sprites (Tiles) utilizadas na renderização do TileMap. Ela permite 2 formas de armazenamento desses Tiles: Como um TileSet (imagem única com vários tiles) ou como um vetor de tiles (cada Tile está armazenado em uma imagem separada).

Atributos da classe:

- `tileWidth, tileHeight` : Armazenam a largura e a altura dos tiles.
- `tileSet` : Armazena um arquivo único contendo todo o tileset.
- `lines, columns` : Armazenam o número de linhas e colunas do tileSet (no caso de utilizar arquivo único)
- `destRect` : Armazena o retangulo que sera usado para clipar os tiles do tileSet (no caso de utilizar arquivo único)
- `vTiles` : Armazena os Tiles caso sejam carregados a partir de arquivos separados

Implementando os métodos da classe:

`+ Tileset(tileWidth: int, tileHeight: int, filePath: std::string)`

- Construtor da classe que inicializará o TileSet com uma única imagem de tileset.
- Inicializa atributos da classe: useSingleFile como true, tileWidth e tileHeight com os parametros passados, a vTiles como NULL.
- A Sprite tileset deve ser carregada a partir do `filePath`.
- O número de lines e columns deve ser calculado de acordo com o tamanho da imagem do tileSet carregado e dos valores tileWidth e tileHeight passados.

`+ Tileset(filePath: std::string, columns: int, lines: int)`

- Construtor Opcional: Criar um construtor quase igual ao anterior, mas ao inves de calcular o número de linhas e colunas a partir do tamanho dos tiles, fazer o inverso.

`+ Tileset(tileWidth : int, tileHeight : int);`

- Construtor da classe que será usado para inicializar um tileset vazio. Seus tiles serão adicionados posteriormente usando o método addTile.
- Inicializa todos os atributos da classe: useSingleFile como false, lines e columns como 0, tileWidth e tileHeight com os parametros passados, a Sprite tileSet como NULL e cria um novo vTiles.

`+ addTile(filePath : std::string) : void`

- Testar se o mapa foi inicializado para usar tileSet (arquivo único) ou não. Caso não, carrega o tile e coloca no vTiles.

`+ render(index : int, posX : float, posY : float) : void`

- Testar se o mapa foi inicializado para usar tileSet (arquivo único) ou não.
- Caso sim, deve-se calcular a posição do retângulo de clipping: O index deve ser mapeado para linha e coluna do tileSet, multiplicado pelos respectivos valores de altura e largura, passado para o tileSet através do método clip, e só então deve ser dado o render(posX, posY);
- `Caso não`, renderizar o tile da posição index do vTiles.

`+ usingSingleFile() : bool`

- Retorna True se o Tileset foi inicializado com arquivo único (se tileSet != NULL) ou false se foi inicializado para utilizar um vetor de Tiles (se tileSet == NULL).

`+ getTileWidth() : int`

- Retorna a largura dos tiles.

`+ getTileHeight() : int`

- Retorna a altura dos tiles.

2. Classe TileMap: O Mapa

TileMap
- tileMatrix : std::vector<std::vector <std::vector <int> > > - tileset : Tileset* - mapWidth, mapHeight, mapLayers : int
+ Tilemap(mapWidth : int, mapHeight : int, tileSize : int, layers = 1: int, tileSet = NULL : Tileset *) + Tilemap(mapa : std::string, tileSet = NULL: tileSet *) + load(mapPath : std::string) : void + setTileset(tileset : Tileset*) : void + at(x : int, y : int, z : int = 0) : int& + render(cameraX = 0.0 : float, cameraY = 0.0 : float) : void + width() : int + height() : int + layers() : int

A classe TileMap será responsável por armazenar todas as informações do mapa, o que no caso mais simples são os índices dos tiles para renderização.

Atributos da classe:

- `tileMatrix`: Matriz para armazenar as informações de cada tile.
- `tileSet` : Ponteiro para o Tileset que será usado na renderização.
- `width, height, layers` : Armazenam o número de linhas, colunas e camadas do

Implementando os métodos da classe:

`+ Tilemap(mapWidth : int, mapHeight : int, tileSize : int, layers = 1: int, tileSet = NULL : Tileset *)`

- Inicializa as variáveis do TileMap: width, height, layers e tileset.
- Redimensiona a tileMatriz para as dimensões corretas e a inicializa com -1.

`+ Tilemap(mapa : std::string, tileSet = NULL: tileSet *)`

- Define o tileset e inicializa width, height e layers com 0.
- Chama o método load(mapa) para carregar o tileMap do arquivo.

`+ load(mapPath : std::string) : void`

- Carrega o mapa a partir de um arquivo. O formato desse mapa pode e deve ser definido arbitrariamente.
- Sugestão: Usar um formato que pode ser gerado por um editor de TileMap, como por exemplo o Tiled. Se esse formato se mostrar muito complicado, usar uma simplificação.
- Width, height e layers devem estar definidos no arquivo do tilemap.

`+ setTileset(tileset : Tileset*) : void`

- Redefine o tileset usado na renderização.

`+ at(x : int, y : int, z : int = 0) : int&`

- Retorna uma referência para o conteúdo do tileMap na posição x,y,z. Para facilitar o acesso a matrizes 2D, é definido 0 como padrão em z. Deve funciona como um acessor, igual ao da classe vector.

`+ render(cameraX = 0.0 : float, cameraY = 0.0 : float) : void`

- Checa se o tileSet não é nulo. Caso não seja:
- Itera em toda a matriz, checando se o índice armazenado é < 0. Caso seja, renderiza aquele tile.
- Para renderizar, é necessário passar a posição correta daquele tile na tela (posição do tile * dimensão do tile) e levar em consideração a posição da câmera.

`+ renderLayer(cameraX : float, cameraY : float, layer : int) : void`

- Opcional: Renderiza apenas uma camada do mapa. Istó será útil para aplicar a técnica de parallax no tilemap, que valerá pontos extras no Trabalho 2.

`+ width() : int`

- Retorna a largura do tilemap (número de linhas).

`+ height() : int`

- Retorna a altura do tilemap (número de colunas).

`+ layers() : int`

- Retorna o número de camadas do tilemap.