Componente: Linguagem de Programação - ECT2303
Professor: Sérgio Queiroz de Medeiros
Aluno (a): Ingridy Lopes da Silva - 20210085653 **Turma:** 02A
Aluno (a): Juliane da Silva Santos - 20200075288 **Turma:** 02A

SEGUNDA APRESENTAÇÃO DO PROJETO FINAL

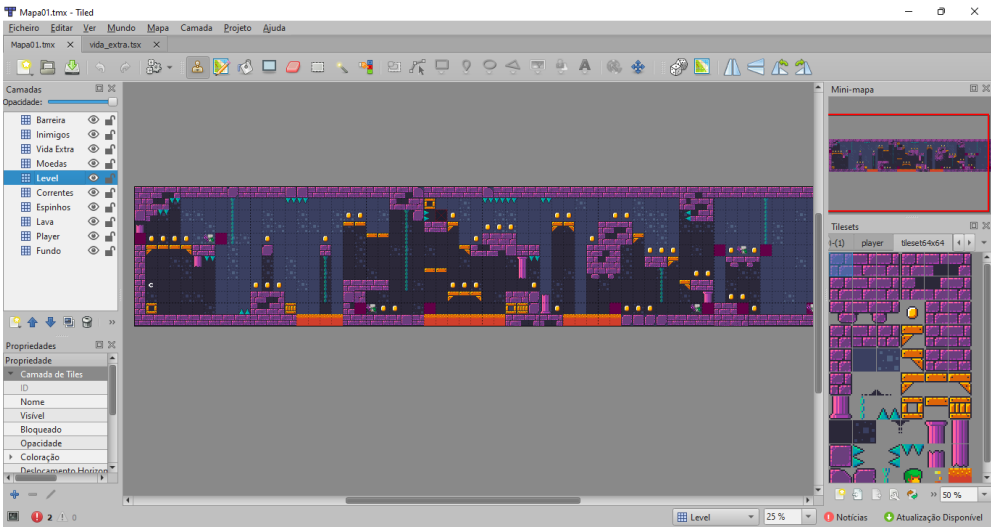
1. INTRODUÇÃO

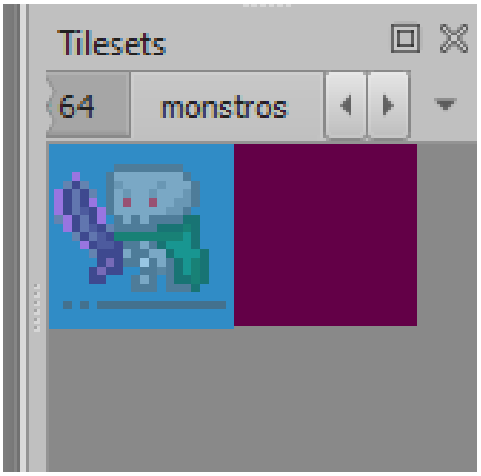
Ainda seguindo a ideia principal do jogo, apresentada na primeira apresentação do projeto final, um jogo de plataforma 2D, onde o personagem principal busca uma forma de voltar a sua vida humana, e para isso é preciso enfrentar vilões para conseguir os elixires de vida.

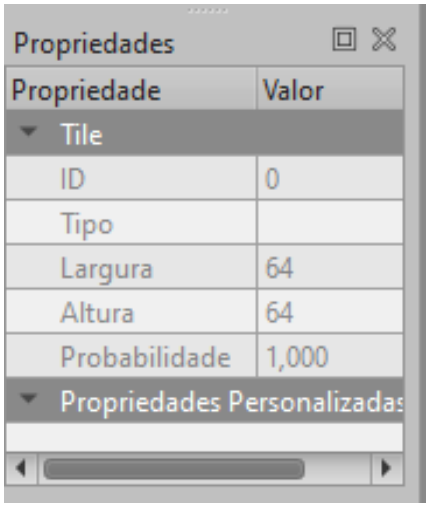
2. CÓDIGO ATUAL

O código atual consta com a movimentação e o poder do personagem, cenário, vilões, moedas para pontuação e sua contagem e por fim uma barra de vida para o player. Para essa etapa, mantemos os cinco arquivos presentes na primeira etapa, adicionando mais cinco arquivos que foram posteriormente integrados ao arquivo principal *main.py*.

Anteriormente para a criação do mapa, fora utilizada uma matriz feita a mão, desta vez, fizemos a utilização de uma plataforma de criação de mapa, *Tiled*. Onde carregamos o *png* da imagem que iremos utilizar e montamos o cenário com cada bloco e o programa nos retorna um arquivo *csv* onde cada bloco da imagem equivale á uma numeração diferente. Logo após a exportação para o arquivo *csv*, colocamos os arquivos em *desenho_mapa.py*, onde associamos cada camada do mapa com seu respectivo arquivo *csv*, em um dicionário.







Dessa forma, conseguimos adicionar o mapa criado no jogo. Para a leitura do arquivo *csv*, fora criado uma função que nos retorna uma lista de cada linha lida no arquivo, transformando assim, o mapa em uma matriz.

```
def import_csv_layout(path):
    level_map = []
    with open(path) as map:
        level = reader(map, delimiter = ',')
        for linha in level:
            level_map.append(list(linha))

        return level_map
```

Para fazer o desenho das imagens nas posições corretas de cada camada, implementamos na classe *mapa.py* uma nova função, semelhante a do código passado, que lia a matriz através de um laço *for*, a diferença majoritária está na forma como a função associa o arquivo a camada escolhida para poder fazer a adição do *sprite* ao grupo de imagens.

```
fundo_layout = import_csv_layout(desenho_mapa["fundo"])
self.fundo_sprites = self.config_mapa(fundo_layout, "fundo")

level_layout = import_csv_layout(desenho_mapa["level"])
self.level_sprites = self.config_mapa(level_layout, "level")

moedas_layout = import_csv_layout(desenho_mapa["moedas"])
self.moedas_sprites = self.config_mapa(moedas_layout, "moedas")

vida_extra_layout = import_csv_layout(desenho_mapa["vida_extra"])
self.vida_extra_sprites = self.config_mapa(vida_extra_layout, "vida_extra")
```

```
def config_mapa(self, layout, type):
    sprite_group = pygame.sprite.Group()

    for indice_linha, linha in enumerate(layout):
        for indice_coluna, cedula in enumerate(linha):
            if cedula != "-1":
                x = indice_coluna * self.configuracao.tamanho_bloco
                y = indice_linha * self.configuracao.tamanho_bloco

                if type == "fundo":
                    fundo_lista = import_cut('./Imagens/fundo.png')
                    fundo_surface = fundo_lista[int(cedula)]
                    sprite = CorteBloco(x, y, self.configuracao.tamanho_bloco, fundo_surface)
                    sprite_group.add(sprite)

                if type == "level":
                    level_lista = import_cut('./Imagens/tileset64x64.png')
                    bloco_surface = level_lista[int(cedula)]
                    sprite = CorteBloco(x,y, self.configuracao.tamanho_bloco, bloco_surface)
                    sprite_group.add(sprite)

                if type == "moedas":
                    sprite = Moeda(x,y,self.configuracao.tamanho_bloco)
                    sprite_group.add(sprite)
```

Para a colisão do player com os inimigos, e as ameças presentes, foram criadas funções, dentro da classe *mapa.py*, para que fosse possível checar as colisões. Como o player seria único, ele também recebeu uma função própria, visto que, queríamos criar um *sprite* com *GroupSingle()*. A movimentação do inimigo foi feita de forma separada, em uma classe específica para ele. Assim como a criação do poder do personagem e interface do jogo.

Para a execução do jogo utilizamos as teclas de direita e esquerda para movimentação no sentido horizontal do personagem e espaço para fazer o fantasminha voar, com o x fazemos o player soltar seu poder.

3. DIFICULDADES

Um das principais dificuldades no desenvolvimento do jogo, foi a implementação de funções de uma classe em outra. A criação de *sprite* também gerou dificuldades no caminho.

Infelizmente o jogo ainda possui falhas, como o deslocamento da tela, a colisão do player em certos pontos, a animação do inimigo ao virar de posição, uma falha no poder do player quando o mesmo está parado e a falta de menu e outras fases.