utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

SENET
Board Game

Relatório
Laboratório de Programação

Turma PL3:
Al79895 Bruno Costa
Al78644 Luís Lemos
Al78501 Tiago Costa

Índice

Conteúdo

Objetivos

O propósito deste projeto é desenvolver uma versão do jogo Senet que permita ao usuário jogar contra outro jogador ou um BOT. Para isso, será necessário programar o jogo em Python, de forma funcional e efetiva para a disciplina de Laboratório de Programação.

Descrição do Jogo

O Senet é um jogo de tabuleiro originário do Antigo Egito que era jogado por duas pessoas. O tabuleiro era composto por trinta quadrados, divididos em três filas de dez. Cada jogador tinha um conjunto de peças, que eram movidas ao longo do tabuleiro de acordo com o resultado de um conjunto de quatro “estiletes”.

O objetivo do jogo era mover todas as peças através do tabuleiro e retirá-las do jogo antes do adversário. O movimento das peças era determinado pelos resultados dos dados, mas também havia quadrados especiais que permitiam avançar mais rapidamente ou impediam o avanço do oponente.

O Senet tinha uma forte conotação religiosa no Antigo Egito, sendo considerado uma representação da jornada das diferentes partes da alma através do submundo. Por essa razão, era frequentemente incluído em tumbas e sepulturas egípcias como um meio de ajudar o falecido na sua jornada após a morte.

Metodologia

Como somos um grupo de três pessoas, definimos um plano de trabalho que inclui a definição do escopo do projeto, a atribuição de funções específicas para cada membro do grupo e o planeamento do desenvolvimento do jogo em etapas. Seguimos as metodologias sugeridas para garantir a qualidade do nosso jogo:

1. Definir o escopo do projeto: Começamos por definir o tipo de jogo que pretendem desenvolver, as funcionalidades e as limitações.

2. Escolher as funções de cada membro do grupo: Cada membro do grupo teve uma função definida de acordo com as suas habilidades e experiências no desenvolvimento de jogos em Python/Pygame.

3. Planear o projeto: Criamos um cronograma de atividades, dividindo o projeto em etapas e definindo prazos para cada uma delas.

4. Desenvolver o jogo: Cada pessoa do grupo começou a programar a parte indicada que planeamos anteriormente.

5. Testar e depurar o jogo: Após a conclusão do desenvolvimento de certas funções, testamos o jogo para garantir que ele funcione corretamente e não apresente erros.

6. Revisar o jogo: Analisamos o desempenho do jogo, e ajustamos e melhoramos o código e os gráficos, para melhorar a jogabilidade e a experiência do usuário.

7. Finalizar o projeto: Certificamos-mos que a documentação está completa e atualizada, e preparamos uma apresentação do jogo para apresentá-lo na aula.

8. Avaliar o projeto: Discutimos com os membros do grupo o que deu certo e o que pode ser melhorado para futuros projetos.

Bibliotecas

Estas bibliotecas são essenciais para o desenvolvimento do jogo em Pygame, pois forneceram as funcionalidades necessárias para criar uma experiência interativa e agradável para o usuário. Além disso, a arquitetura do jogo permite uma fácil manutenção do código e adaptação a possíveis mudanças no projeto.

Pygame: é a principal biblioteca utilizada para o desenvolvimento do jogo. Ela é responsável por gerenciar a janelado jogo, exibir imagens na tela, reproduzir sons e capturar eventos do teclado e do mouse.

OS: é uma biblioteca que permite aceder a funcionalidades específicas do sistema operacional. Neste jogo, é usada para localizar os arquivos de som e imagem necessários para o jogo.

Time: é uma biblioteca que fornece funções para medir e controlar o tempo em Python. Neste jogo, é usada para controlar o tempo de espera entre as jogadas dos jogadores e do bot.

Mixer: é uma biblioteca do Pygame que permite a reprodução de sons. Neste jogo, é usada para reproduzir efeitos sonoros, como o som da movimentação das peças, dos cliques e da música de fundo.

Random: é uma biblioteca essencial para tornar o jogo de tabuleiro Senet mais desafiador e imprevisível, tornando cada jogada única e emocionante. Sem a biblioteca "random", o jogo seria previsível e monótono, prejudicando a experiência do jogador.

```
1  import pygame
2  import time
3  import os
4  import random
5  from pygame import mixer
```

Arquitetura e Funções

O jogo é programado em Python e é dividido em várias classes e funções principais que gerenciam diferentes aspetos do jogo:

- A classe "Janela" gerencia a janela do jogo, incluindo a sua criação e configuração, sendo tudo gerido pela função "janela_inicial".

- A classe "Menu" está encarregue das funções "menu_principal" e "submenu_jogar_partida" que gerenciam a exibição do menu do jogo e do submenu quando é clicado no botão Jogar Partida, respetivamente.

- A classe "Regras" está encarregue da função "descricao_jogo" que gerencia a exibição das regras do jogo.

- A classe "Settings" está encarregue da função "settings_jogo" que gerencia as configurações do jogo, como o volume da música e dos sons.

- A classe "Sair" é responsável por encerrar o jogo de forma adequada e possui a função "sair".

- A classe "Jogar" está encarregue de várias funções que são responsáveis por gerenciar a lógica principal do jogo como o movimento das peças ("movimento"), as peças que vão para fora do tabuleiro ("fora_tabuleiro"), verificação do destino das peças durante uma jogada ("procura_block"), a escolha da quantidade de estiletes incluindo a atualização da tela e a interação com o usuário ("escolher_cores" e "desenhar_paus") sendo estas funções, todas geridas pela função principal "jogar_partida".

- A classe "Jogarbot" é responsável por gerenciar a lógica do bot do jogo quando o jogador está jogando contra a inteligência artificial e utiliza as mesmas funções da classe "Jogar".

- A classe "Jogar_load" é responsável por carregar os arquivos de som e imagem necessários para o jogo e utiliza também as mesmas funções da classe "Jogar".

Classes e Funções

```python
main.py
SENET Board Game > Code > main.py > ...
1  ##############################################################################
2  #                          SENET BOARD GAME IN PYGAME                        #
3  #  Trabalho Realizado por:                                                   #
4  #  - Bruno Costa al79895                                                     #
5  #  - Luís Lemos  al78644                                                     #
6  #  - Tiago Costa al78501                                                     #
7  ##############################################################################
8  import pygame
9  from janela import Janela
10 from menu import Menu
11
12 # Inicia o pygame
13 pygame.init()
14 pygame.display.set_caption("SENET Board Game")
15
16 # Define a janela como tela cheia
17 janela = pygame.display.set_mode((0, 0), pygame.FULLSCREEN|pygame.NOFRAME)
18
19 # Analisa e atribui as dimensões da janela
20 largura_janela, altura_janela = janela.get_width(), janela.get_height()
21 Janela.janela_inicial(janela, largura_janela, altura_janela)
22
23 # Cria a janela
24 janela = pygame.display.set_mode((0, 0))
25
26 # Chama o menu principal
27 Menu.menu_principal(janela, largura_janela, altura_janela)
```

Código 1 – Código Main Do Jogo

```python
janela.py
SENET Board Game > Code > janela.py > ...
import pygame
import time
import os
import random
from pygame import mixer

class Janela():
    def janela_inicial(janela, largura_janela, altura_janela):
        #-------------------------IMAGENS-------------------------
        imagem_fundo = pygame.image.load(os.path.join('images', 'FUNDOINICIO.png'))
        imagem_fundo = pygame.transform.scale(imagem_fundo, (largura_janela, altura_janela))
        #---------------------------------------------------------

        #-------------------------SONS-------------------------
        mixer.music.load(os.path.join('sounds', 'background.mp3'))
        mixer.music.set_volume(0.2) # Define o volume para 20%
        mixer.music.play(-1)
        #---------------------------------------------------------

        #-------------------------CORES/FONTES-------------------------
        cor_texto = '#ffffff'
        fonte4_path= os.path.join('Fonts', 'roman_sd', 'Roman SD.ttf')
        fonte = pygame.font.Font(fonte1_path, 40)
        #---------------------------------------------------------

        #-------------------------DESIGN-DA-JANELA-------------------------
        mensagem = fonte.render('Clique para começar', True, cor_texto)
        x_mensagem = largura_janela // 2 - mensagem.get_width() // 2
        y_mensagem = altura_janela // 1.4 - mensagem.get_height() // 1.4
```

Código 2 – Código Janela Inicial

```python
menu.py
SENET Board Game > Code > menu.py > ...
import pygame
import os
from jogar import Jogar
from jogarbot import Jogarbot
from load import Jogar_load
from descricao import Regras
from settings import Settings
from quit import Sair
from pygame import mixer

class Menu:
    def menu_principal(janela, largura_janela, altura_janela):
        #-------------------------IMAGENS-------------------------
        # carrega a imagem de fundo
        imagem_fundo = pygame.image.load(os.path.join('images', 'FUNDO.png'))
        # redimensiona a imagem para as dimensões da janela
        imagem_fundo = pygame.transform.smoothscale(imagem_fundo, (largura_janela, altura_janela))
        # Carrega as imagens dos botões
        imagem_botao_normal = pygame.image.load(os.path.join('images', 'button2.png'))
        imagem_botao_hover = pygame.image.load(os.path.join('images', 'button_hover2.png'))
        #---------------------------------------------------------

        #-------------------------CORES/FONTES-------------------------
        # define as variáveis da cor
        cor_texto = '#000000'

        # define as fontes
        fonte4_path= os.path.join('Fonts', 'roman_sd', 'Roman SD.ttf')
        fonte_opcoes = pygame.font.Font(fonte1_path, 25)
```

Código 3 – Código Menu Principal

```python
descricao.py
SENET Board Game > Code > descricao.py > ...
import pygame
import os
from pygame import mixer

class Regras:
    def descricao_jogo(janela, largura_janela, altura_janela):
        #-------------------------IMAGENS-------------------------
        # Carrega a imagem de fundo
        imagem_fundo = pygame.image.load(os.path.join('images', 'REGRAS.png'))
        # Redimensiona a imagem para as dimensões da janela
        imagem_fundo = pygame.transform.smoothscale(imagem_fundo, (largura_janela, altura_janela))
        # Carrega as imagens dos botões
        botao_normal_img = pygame.image.load(os.path.join('images', 'button2.png'))
        botao_hover_img = pygame.image.load(os.path.join('images', 'button_hover2.png'))
        # Redimensiona a imagem para as dimensões desejadas
        botao_normal_img = pygame.transform.smoothscale(botao_normal_img, (200, 70))
        botao_hover_img = pygame.transform.smoothscale(botao_hover_img, (200, 70))
        #---------------------------------------------------------

        #-------------------------CORES/FONTES-------------------------
        # Define as variáveis de cor
        cor_texto = '#000000'

        # Define as fontes
        fonte_path4 = os.path.join('Fonts','roman_sd', 'Roman SD.ttf')
        fonte_texto = pygame.font.Font(fonte_path4, 25)
        #---------------------------------------------------------

        #-------------------------DESIGN-DE-BOTÃO-------------------------
        # Define a área clicável da imagem
```

Código 4 – Código Regras Do Jogo

```python
settings.py
SENET Board Game > Code > settings.py > ...
import pygame
import os
from pygame import mixer

class Settings:
    def settings_jogo(janela, largura_janela, altura_janela):
        #-------------------------IMAGENS-DAS-DEFINIÇÕES-------------------------
        # Carrega a imagem de fundo do menu da escolha de oponente
        imagem_fundo = pygame.image.load(os.path.join('images', 'SETTINGS.png'))
        # Redimensiona a imagem para as dimensões da janela do menu de escolha de oponente
        imagem_fundo = pygame.transform.smoothscale(imagem_fundo, (largura_janela, altura_janela))
        # Carrega as imagens dos botões
        imagem_botao_normal = pygame.image.load(os.path.join('images', 'button2.png'))
        imagem_botao_hover = pygame.image.load(os.path.join('images', 'button_hover2.png'))
        #---------------------------------------------------------

        #-------------------------DESIGN-DAS-DEFINIÇÕES-------------------------
        cor_texto = '#000000'
        cor_texto_telasub = '#ffffff'

        # Adiciona as fontes e os caminhos
        fonte4_path= os.path.join('Fonts', 'roman_sd', 'Roman SD.ttf')
        fonte_opcoes_definicoes = pygame.font.Font(fonte4_path, 25)
        fonte_opcoes_musica = pygame.font.Font(fonte4_path, 22)
        fonte_titulo_definicoes = pygame.font.Font(fonte4_path, 40)

        titulo = fonte_titulo_definicoes.render("Definições", True, cor_texto_telasub)

        # Define o botão Voltar
        largura_botao = 200
```

Código 5 – Código Definições Do Jogo

```python
jogar.py
SENET Board Game > Code > jogar.py > ...
import pygame
import os
import random
from pygame import mixer

class Jogar:
    # Função responsável pela movimentação das peças
    def movimento(i, imagens_pecas, casas_ocupadas, next_pos, next_casa, lancamento):...

    # Função responsável pelas posições das peças fora do tabuleiro
    def fora_tabuleiro(i, imagens_pecas, casas_ocupadas, fora_brancas, fora_pretas, lancamento):...

    # Função responsável por verificar o destino das peças durante o jogo
    def procura_block(casas_ocupadas, next_pos):...

    # Função para escolher aleatoriamente as cores das paus
    def escolher_cores(pau_branco, pau_preto):...

    # Função para desenhar as paus na tela
    def desenhar_paus(janela, paus):...

    # Função para desenhar as paus da peça branca na tela
    def desenhar_paus2(imagem, paus):...

    # Função para desenhar as paus da peça preta na tela
    def desenhar_paus3(imagem, paus):...

    # Função responsável pelo jogo no geral
    def jogar_partida(janela, largura_janela, altura_janela):...
```

Código 6 – Código Principal Do Jogo

```python
quit.py
SENET Board Game > Code > quit.py > ...
import pygame

class Sair:
    def sair():
        pygame.quit()
        quit()
```

Código 7 – Código Para Sair Do Jogo

Desafios

Ao longo do desenvolvimento do jogo tivemos inúmeros desafios que tivemos de enfrentar e solucionar, os principais foram:

- Design do tabuleiro: O design do tabuleiro foi um dos principais desafios no desenvolvimento do jogo. O tabuleiro do Senet é composto por trinta quadrados divididos em três filas de dez, com alguns quadrados especiais que permitem avançar mais rapidamente ou impedem o avanço do oponente. Foi necessário criar uma estrutura de dados para representar o tabuleiro e implementar a lógica para permitir que as peças se movessem corretamente ao longo do tabuleiro.

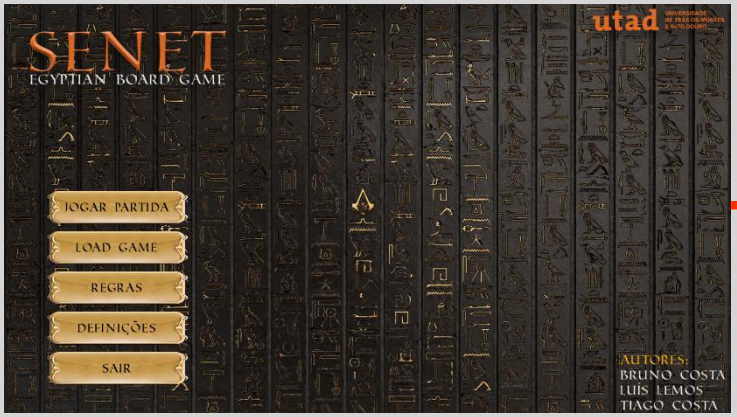
- Lógica do jogo: Outro desafio foi implementar a lógica do jogo de forma precisa e eficiente. O Senet tem regras específicas para o movimento das peças e para a captura de peças do oponente. Foi necessário criar uma estrutura de dados para representar as peças e implementar a lógica para permitir que elas se movessem corretamente de acordo com as regras do jogo.

- Interface do usuário: A interface do usuário também apresentou alguns desafios no desenvolvimento do jogo Senet em Pygame. Foi necessário criar uma interface intuitiva, que permitisse que o usuário interagisse facilmente com o jogo e entendesse as regras e movimentos das peças. Isso incluiu a criação de uma interface gráfica que exibisse claramente o tabuleiro e as peças, bem como a implementação de botões e menus para permitir que o usuário fizesse escolhas e configurações no jogo.

- Inteligência Artificial (BOT): Implementar uma inteligência artificial para o jogo Senet também foi um desafio significativo. A inteligência artificial precisava ser capaz de fazer movimentos estratégicos e tomar decisões com base na posição atual do tabuleiro. Foi necessário criar uma estrutura de dados para representar a inteligência artificial e implementar a lógica para permitir que ela jogasse de forma eficaz contra o usuário.

- Testes e depuração: Por fim, testar e depurar o jogo Senet em Pygame foi um desafio constante. Durante o desenvolvimento, foram encontrados vários bugs e problemas que precisaram ser corrigidos. Isso incluiu testar a lógica do jogo, a interface do usuário e a inteligência artificial para garantir que tudo funcionasse corretamente e sem erros.

Funcionalidades e Capturas

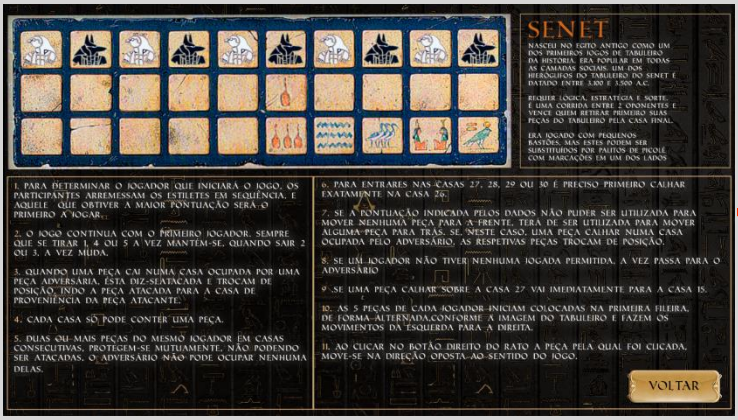


Menu composto por 5 opções:
Jogar: Começar um novo jogo contra o computador ou outro jogador humano.
Load: Carregar um jogo salvo anteriormente.
Regras: Ver as regras do jogo.
Definições: Configurações do jogo, volumes e sons.
Sair: Sair do jogo.

Imagem 1 – Menu Principal do Jogo

Submenu responsável por selecionar o modo de jogo (Contra o Computador ou Contra Outro Jogador) e a opção Voltar para regressar ao Menu Principal do Jogo

Imagem 2 – Submenu Jogar Partida



Tela Dedicada às Regras do Jogo com uma descrição do jogo e as suas regras, conta também com um botão Voltar para regressar ao Menu.

Imagem 3 – Regras Do Jogo

Definições Do Jogo responsável por controlar a Música e os Sons do jogo no geral, novamente com o botão Voltar para regressar ao Menu.

Imagem 4 – Definições Do Jogo

Funcionalidades e Capturas (2)

Imagem 5 – Tela Dos Jogadores 1vs1

Tela responsável por Guardar os Nomes digitados pelos jogadores e Decidir Aleatoriamente quem tem mais estiletes brancos, quem calhar com mais estiletes começa a jogar, a tela desaparece e começa o jogo após 3 segundos de ambos os jogadores terem escolhido os seus nomes.

Tela do Jogo em Execução com o Tabuleiro ocupado por várias peças, uma secção para atirar os estiletes, 2 caixas para armazenar as peças que já saíram do tabuleiro de cada jogador e um texto a indicar o turno do jogador

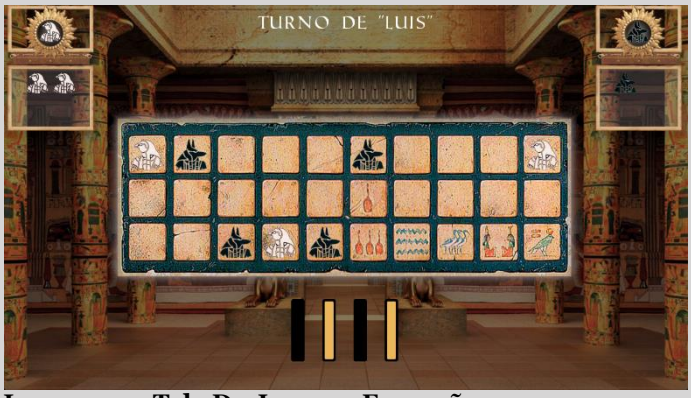


Imagem 7 – Tela Do Jogo em Execução

Imagem 8 – Tela De Pausa

Tela De Pausa do Jogo ativada quando é pressionada a tecla "ESC". Contém as opções: "Continuar" para retomar ao jogo, "Save Game" para guardar o estado atual do jogo, "Menu" para regressar ao menu principal e "Sair" para sair do jogo.

Tela do Vencedor tem como tarefa apresentar o nome do Vencedor, sempre que as 5 peças de um dos jogadores saírem do tabuleiro.

Imagem 9 – Tela Do Vencedor

Conclusão e Avaliação geral

O projeto do SENET em Python apresentou uma implementação bem-sucedida do jogo, com uma interface gráfica de usuário (GUI) amigável e intuitiva. O jogo em si é jogável e possui regras precisas e bem definidas. A implementação da lógica do jogo é clara e fácil de entender, com o código bem organizado e comentado.

No entanto, há áreas em que o projeto pode ser melhorado. Por exemplo, o BOT poderia ser mais aprimorado e melhorado, de forma a funcionar de maneira mais independente. Além disso, o código pode ser otimizado para melhorar o desempenho e reduzir redundâncias e limitações, pois atualmente o código só funciona em computadores com uma escala de resolução de 125%.

De uma maneira geral, o projeto foi bem-sucedido, e apresenta uma interface intuitiva. Apesar disso, ainda há espaço para melhorias, mas a funcionalidade e as regras do jogo foram implementadas de forma precisa e eficiente. Por fim, o projeto é uma realização bem-sucedida e um bom exemplo de como implementar jogos de tabuleiro utilizando a linguagem de programação Python.