Documento de especificação de requisitos + Relatório Final

1. Introdução

1.1. Objetivo

Fornecer uma visão abrangente dos requisitos funcionais e não funcionais para o desenvolvimento do jogo Sokoban.

1.2. Escopo

● Identificar pelo nome o produto de software a ser produzido

Sokoban Game

● Explicar o que o produto vai e, se necessário, não vai fazer.

Aplicação em Python de um jogo ao estilo “Sokoban” com movimentação padrão utilizando apenas quatro setas (cima, baixo, esquerda, direita).

Trata-se de um puzzle game com o número de fases extensível. As fases serão carregadas através de um arquivo de texto.

● Descrever a aplicação de software que está sendo especificada, incluindo seus benefícios, objetivos e metas.

Jogo de raciocínio para resolver os puzzles (quebra-cabeças) para avaliação final (AV3) da disciplina de “Programação Funcional” devendo seguir as restrições impostas pelo professor no documento original disponibilizado.

(https://docs.google.com/document/d/1Od8FwC_t_A0OBtruWi2ec9p7e7YZJ3_nKQmlD8cWZBM/edit)

2. Descrição Geral

2.1. Requisitos funcionais

[RF 001] Jogo 2D - obrigatório

[RF 002] Fácil jogabilidade - obrigatório

[RF 003] Nº de fases extensível - obrigatório

[RF 004] Movimentação do personagem: O jogador deve poder mover o personagem principal para cima, baixo, esquerda e direita para empurrar as caixas- obrigatório

[RF 005] Regras do jogo: O jogo deve seguir as regras clássicas do Sokoban, como a impossibilidade de empurrar mais de uma caixa ao mesmo tempo e a necessidade de empurrar todas as caixas para os destinos corretos. - obrigatório

[RF 006]Modo single player - obrigatório

[RF 007] Interface gráfica - opcional

[RF 008] Fases armazenadas em arquivo de texto - desejável

[RF 009]Mínimo de 3 fases - desejável

[RF 010] Cores ou formas distintas entre personagem e objetos do cenário - desejável

2.2. Requisitos não funcionais.

[RNF 001]Desenvolvimento em Python - obrigatório
[RNF 002]Desempenho: O jogo deve ser responsivo e fluído, sem atrasos perceptíveis na movimentação do personagem e nas interações com os objetos. - desejável
[RNF 003]Testes e garantia de qualidade: O desenvolvimento deve incluir testes extensivos para garantir a estabilidade e a jogabilidade do jogo. - desejável
[RNF 004]Eficiência de Uso de Recursos - desejável

2.4. Características dos usuários

Os usuários deste jogo são esperados ter um nível educacional básico, experiência em jogos 2D e conhecimentos técnicos mínimos, como a capacidade de operar um computador pessoal.

2.5. Restrições

Linguagem Python

2.6.Suposições e dependências
2.6.1 Fatores Afetando Requisitos:

É assumido que a aplicação será utilizada em sistemas operacionais Windows, Linux e macOS. Mudanças significativas nos sistemas operacionais podem impactar os requisitos.

2.6.2 Atualizações Futuras:

Planeja-se lançar atualizações futuras que podem incluir novos níveis e recursos. Essas atualizações podem influenciar os requisitos, especialmente aqueles relacionados à extensibilidade do número de fases.

RELATÓRIO FINAL

1) Uso dos conceitos estudados na disciplina de Programação Funcional

- **Função Lambda de Alta Ordem:**

  Descrição: A função **quit_action_higher_order** é uma função lambda de alta ordem que aceita uma função (quit) como argumento e retorna uma nova função que imprime "quit" e chama a função passada como argumento.

- **Função Lambda Recursiva:**

  Descrição: A função lambda **is_completed_recursive** é recursiva, sendo utilizada para verificar se o jogo foi concluído, percorrendo recursivamente a matriz.

- **Sucessivas Chamadas de Funções Lambda utilizando Currying:**

  Descrição: A função **curried_move** utiliza currying, permitindo a chamada parcial da função move através de sucessivas chamadas.

- **List Comprehension dentro do Escopo Lambda:**

  Descrição: A função lambda **load_images** utiliza uma List Comprehension para carregar imagens a partir de uma lista de caminhos.

- **Dicionário dentro do Escopo de uma Função Lambda:**

  Descrição: A função lambda **images** retorna um dicionário de imagens.

- **Functor (Map) e List Comprehension:**

  Descrição: A função lambda **load_images** utiliza um functor (map) e uma List Comprehension para carregar imagens a partir de uma lista de caminhos.

- **Monad:**

  Descrição: A classe **Result** implementa um monad, oferecendo tratamento de sucesso ou erro, encadeamento de operações e uma função bind para composição.

2) **Cumprimento dos requisitos funcionais e não funcionais**

Requisitos Funcionais

- Jogo 2D:
  Descrição: O código implementa um jogo Sokoban 2D com auxílio da biblioteca pygame.
  import pygame
  pygame.init()

- Fácil Jogabilidade:
  Descrição: O código permite a movimentação do personagem principal para cima, baixo, esquerda e direita para empurrar as caixas.
  def move(self, x, y, save)

- Nº de Fases Extensível:
  Descrição: O jogo pode ser configurado para carregar diferentes fases a partir de arquivos de texto.
  def load_level(self, filename, level)

- Regras do Jogo:
  Descrição: O jogo segue as regras clássicas do Sokoban, como a impossibilidade de empurrar mais de uma caixa ao mesmo tempo: **def move_box(self, x, y, a, b)**;
  Existe a verificação de conclusão do jogo, ou seja, se todas as caixas estão nas docas, assim como a possibilidade de desfazer o último movimento do personagem.
  def is_completed(self)
  def unmove(self)

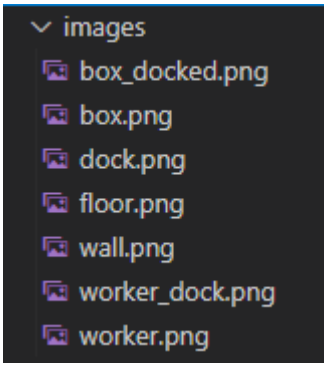
- Modo Single Player:
  Descrição: O código implementa o modo single player.

- Interface Gráfica (Opcional):
  Descrição: Implementação de uma interface gráfica com o auxílio da biblioteca pygame.
  import pygame
  pygame.init()

- Fases Armazenadas em Arquivo de Texto (Desejável):
  Descrição: O código carrega as fases a partir de arquivos de texto (arquivo: **levels**).
  def load_level(self, filename, level)

- Mínimo de 3 Fases (Desejável):
  Descrição:Foram criadas 3 fases/mapas, podendo ser criados mais mapas.

- Cores ou Formas Distintas entre Personagem e Objetos do Cenário (Desejável):
  Descrição: Sprites distintos para cada elemento no jogo



Requisitos Não Funcionais

- Desenvolvimento em Python:
  Descrição: O código é desenvolvido em Python.

- Desempenho:
  Descrição: O código busca ser responsivo, não apresentando travamento nos testes de jogabilidade.

- Testes e Garantia de Qualidade:
  Descrição: Testes manuais, a cada etapa do projeto, feitos pela equipe de desenvolvimento.

- Eficiência de Uso de Recursos:

  Descrição: O código faz uso de uma estrutura de fila (LifoQueue) para rastrear os movimentos do jogador, o que pode contribuir para uma eficiente gestão de recursos.