Universidade Autónoma de Lisboa

Engenharia de Software

2024/2025

Tema: Receitas Eletrónicas

Integrantes do grupo:

- Afonso Soeiro – nº 30012583
- Guilherme Restolho – nº 30012684

Lisboa

Índice

1. Introdução 2
2. Arquitetura e Estrutura do Projeto 3
3. Interface Gráfica (GUI) 4
4. Lógica do Jogo 4
5. Funcionalidades Relevantes e Detalhes de Implementação 4
6. Qualidade de Código e Boas Práticas 4
7. Testes 5
8. Métricas do Projeto 5
9. Dificuldade e Soluções 5
10. Conclusão 6

1. Introdução

O projeto "Park Out" foi desenvolvido como parte da unidade curricular de Programação, com o objetivo de criar um jogo interativo utilizando Python e o framework PySide6. O jogo simula um parque de estacionamento de autocarros onde o jogador deve organizar e movimentar os veículos para que passageiros embarquem corretamente de acordo com a cor correspondente. O desafio é garantir a saída eficiente dos autocarros, minimizando obstruções e otimizando o embarque dos passageiros.

2. Arquitetura e Estrutura do Projeto

Linguagem: Python 3.11

Framework: PySide6

IDE: PyCharm Community Edition

A arquitetura do projeto baseia-se em programação orientada a objetos. As principais classes são:

- CarroWidget: representa o autocarro, com atributos como cor, capacidade, direção e estado de ocupação.
- PessoaWidget: representa o passageiro, com imagem e cor associada.
- Novajanela: representa a janela principal do jogo, orquestrando toda a lógica visual e funcional.

Outras funções de apoio tratam do carregamento de imagens, distribuição aleatória, reinicialização do jogo e execução dos movimentos.

3. Interface Gráfica (GUI)

A GUI foi criada manualmente em PySide6, sem uso do Qt Designer. Características principais:

- Representação visual dos autocarros com imagens e cores distintas.
- Ocupação visível nos autocarros com contador dinâmico.
- Fila de passageiros com atualização em tempo real.
- Botão "Restart" que reinicializa todo o estado do jogo aleatoriamente.

4. Lógica do Jogo

Cada autocarro tem uma direção fixa (up, down, left, right) derivada do nome da imagem e só se move se o caminho estiver livre. A verificação é feita pela função *caminho_livre().*

Ao clicar num autocarro com caminho desobstruído, este desloca-se com animação para a vaga mais próxima. Passageiros da mesma cor embarcam automaticamente se houver lugar.

Quando um autocarro atinge sua capacidade máxima, é removido do tabuleiro.

A função *tentar_entrar_pessoa()* é executada periodicamente via QTimer, simulando o embarque de passageiros.

5. Funcionalidades Relevantes e Detalhes de Implementação

- **Capacidades dos autocarros**: atribuídas aleatoriamente usando *criar_capacidades_exatas(),* com ajuste fino para garantir soma igual ao total de passageiros.
- **Fila de passageiros**: gerada a partir de imagens reais com associação de cor.
- **Movimentação com animação**: utiliza QPropertyAnimation para transições suaves.
- **Estado visual**: inclui indicador de ocupantes dentro do autocarro.

6. Qualidade de Código e Boas Práticas

- Comentários explicativos nas principais funções e blocos de lógica.
- Estrutura modular com separação de interface, modelo e controlo.
- Uso correto de herança com QLabel para customização visual de widgets.

7. Testes

Foram realizados testes manuais e inspeção visual para garantir:

- Movimentos válidos/inválidos.
- Embarque apenas em carros da mesma cor.
- Saída apenas quando o carro está cheio.
- Reinício do jogo gera nova configuração coerente.

8. Métricas do Projeto

Categoria	**Valor**
Total de módulos	4(maim.py, model.py, view.py, controller.py)
Total de linhas de código	710
Linhas de comentários	118

Por estudante:

Membro	Linhas de Código	Linhas de Comentários	Módulos Trabalhados
Guilherme	300	70	Passageiros, Animações, Reinício
Afonso	410	48	Interface, Movimentos, Lógica

Cobertura Funcional:

- mover_carro_para_vaga() ✔
- caminho_livre() ✔
- tentar_entrar_pessoa() ✔
- reiniciar_jogo() ✔

9. Dificuldade e Soluções

Dificuldades:

- Ajustar capacidades para total exato de passageiros.
- Animação sincronizada com estado lógico.
- Geração coerente e aleatória das imagens e cores.

Soluções:

- Função criar_capacidades_exatas() com ajustes iterativos.
- Uso de QTimer e QPropertyAnimation para controlo temporal.
- Funções load_images_from_folder() e prepare_balanced_images_with_colors() para aleatoriedade balanceada.

10. Conclusão

O projeto "Park Out" proporcionou um desafio completo de integração entre lógica de jogo, interação gráfica e gestão de recursos dinâmicos. A experiência consolidou os nossos conhecimentos em programação orientada a objetos, animações em PySide6 e boas práticas de estruturação de projeto.

A aplicação resultante é funcional, estável e visualmente apelativa, cumprindo todos os requisitos propostos pelo enunciado.