

**Instituto Tecnológico de Aeronáutica - ITA**

Divisão de Ciência de Computação

**Disciplina de CES-22**

Davi Grossi Hasuda1

davihasuda@gmail.com

Eduardo Henrique Ferreira Silva2

eduardo.hferreiras@gmail.com

Gustavo Nahum Alvarez Ferreira3

[gustavo.nahum@gmail.com](mailto:gustavo.nahum@gmail.com)

1,2,3 Aluno(a) de Graduação em Engenharia do Instituto Tecnológico de Aeronáutica - ITA. Praça Marechal Eduardo Gomes, nº 50. CEP. 12228-900 - São José dos Campos - SP, Brasil.

Projeto de CES-22

**Documentação do projeto de jogo produzido em Python**

1. **Resumo**

Uma vez introduzido o conceito de POO (Programação orientada a objetos), implementá-la é atitude necessária para o completo aprendizado. Assim, o projeto em grupo da disciplina de CES-22 enfocou a implementação desses conhecimentos, a partir da linguagem Python.

**Palavras-chave:** *POO, Python, pygame, GitHub, Git.*

1. **Introdução**

Conforme o tamanho e a complexidade de projetos aumentam, é possível perceber a grande utilidade de linguagens orientadas a objetos, tanto no que se refere a suas funcionalidades adicionais (garantindo um poder ainda maior aos programadores), como também na maior organização e facilidade de acesso e modificação que esses projetos recebem.

Com o intuito de suscitar nos alunos o desenvimento de maiores habilidades e experiência com uma dessas linguagens, o projeto bimestral realizado em Python requisitava a cada grupo de 3 alunos a criação de um jogo usando a framework *pygame*, bem como o uso do repositório online GitHub, a ser manipulado por meio do sistema de versão de controle Git.

1. **Descrição do projeto**

A sugestão proposta pelo grupo de alunos para o tema do projeto teve, como tema, a captura de riquezas em ouro por um *cowboy* do velho oeste norte-americano.

A fim de capturar o ouro, o *cowboy* deve atravessar uma rede de bifurcações binárias, todas alternáveis pelo usuário entre os seus dois estados, desde que o usuário ative a alternância de estado previamente à chegada do *cowboy* – essa alternância é ativada a partir de cliques do mouse.

Ao final da rede de bifurcações, há uma saída disponibilizando o acesso do ouro ao *cowboy*, caso este consiga chegar a essa saída de maneira bem-sucedida. Em todas as demais saídas, e com o intuito de aumentar a dificuldade do jogo, há bombas e dinamites, de tal forma que, caso o jogador chegue ao final da rede em uma dessas saídas, o *cowboy* morrerá, e o jogador terá fracassado no jogo.



Figura 1. Captura de tela do jogo em execução. Destacam-se a existência de um monte de ouro, em uma das saídas, e de dois montes com bombas e dinamites, e outras duas saídas.

1. **Estrutura do projeto**

A estrutura definida para o código visou a sua maior organização para contínuos aprimoramentos e extensões do código pelos membros, bem como sua maior compreensão para futuros leitores do código-fonte. Para isso, prezou-se por uma estrutura de diretórios de natureza concordante, a fim de assumirem conjuntos dotados de coerência interna. Dessa forma, foram produzidos os seguintes diretórios: “Assets” – responsável por reunir os itens externos ao código, isto é, imagens, fontes e a trilha sonora do jogo –; “GameElements” – responsável por reunir os elementos que constituem a estrutura operacional do jogo, ou seja, o *cowboy*/jogador, a estrutura de criação de redes, a classe de criação de bifurcações individuais, a classe de criação de *canvas*/janelas e a estrutura de marcação da pontos do jogador; “GameStates” – responsável por determinar cada etapa do jogo, a partir de classes que definem a ativação de menus, instruções, levels e a finalização do jogo; por fim, há a “main”, responsável por chamar as classes do “GameStates”, e assim controlar todo o fluxo do jogo.

* 1. **Assets**

Dentro deste diretório, há três outras pastas: “Fonts” (fontes), “Images” (imagens) e “Sounds” (trilha sonora do jogo). Dentro de cada uma dessas pastas, há os componentes externos, isto é, não constituídos por código. Eles permitem que a interação com o usuário sejam mais amigáveis, garantindo uma imersão maior com o contexto de velho oeste pretendido, assim como uma interface mais agradável.

* 1. **GameElements**

Neste diretório, há as seguintes classes: Canvas (janela), Fork (bifurcação), Path (caminho), Player (jogador) e Score (pontuação). A classe Canvas é responsável por gerar “Surface”s para a janela de jogo, que deverá ocupar uma resolução de 1920x1080 pixels, e que será mantida para todas as demais impressões que serão lançadas na tela. Para isso, a classe solicita 2 argumentos: as dimensões, em pixels, do comprimento e da altura do Canvas. A classe Fork será responsável pela definição, alternância e impressão no Canvas de cada bifurcação, individualmente; conta, com isso, com os respectivos métodos: init, toggle e draw. A classe Path é encarregada de formar a estrutura de rede por completo, fazendo, para isso, sucessivas chamadas da função Fork para auxiliá-la. A classe Player, por fim, chama a classe Path e, junto a ela, obtém informações a respeito da posição do jogador e do caminho que ele deverá seguir.

* 1. **GameStates**

A este diretório é atribuída a tarefa de gerenciar as diversas etapas do jogo, a saber: a abertura, as instruções, os levels de cada jogo sucessivo, a execução do jogo propriamente dita, e o encerramento do jogo, em caso de derrota do jogador. A classe Start\_Menu é responsável por determinar as configurações da abertura do jogo, seja o plano de fundo, a trilha sonora e até mesmo os textos do jogo. O Menu determinará as confugurações de interação visual e sonora com o jogador, a partir de métodos separados. Game\_Over\_Menu, por fim, fica encarregado pela finalização do jogo e mensagens de derrota do jogador em sua fase final.

1. **Principais dificuldades**

Mesmo após reuniões para se definir a estrutura a ser adotada no código, fez-se necessária a sua reformulação em diversos momentos, devido às dificuldades encontradas durante a sua produção. Houve, assim, uma elevada organização, robustez, versatilidade e extensibilidade do código desde o momento de sua criação, e a qualidade de sua organização e portabilidade somente aumentou com o avanço do projeto.

Possivelmente, a maior dificuldade do projeto estava relacionada às operações matemáticas realizadas, em que as abstrações presentes nos cálculos ligados à estrutura de rede, bem como a grande quantidade de parâmetros associados acabou por elevar a complexidade de projeto das operações desejadas, a exemplo de: a operação de armazenamento dos estados das bifurcações (para cima ou para baixo) em um “Array”, o correto direcionamento do jogados para as bifurcações em seus estados corretos, e a impressão dos tesouros e dos baús.

1. **Conclusão**

Diante de diversos desafios encontrados ao longo do projeto, desde o aprendizado da framework *pygame* de maneira independente de instrução em sala de aula, até a verificação contínua de cada arquivo, classe ou até mesmo método problemático, o grupo pôde evoluir conjuntamente em suas habilidades de programação, projeto e de trabalho em equipe. Dessa forma, conclui-se que o projeto cumpriu sua função de agregar conhecimentos de POO aos alunos, além de oferecer uma forma lúdica e criativa de produzir um projeto que foi concebido do zero.