Projeto Final de POO: Jogo de Fazenda utilizando a biblioteca curses

João Victor Pimentel Rodeigues
Depto. Engenharia Elétrica - Engenharia de Sistemas
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
joaovpr@ufmg.br

***Abstract*—O objetivo desse trabalho é apresentar o design e a implementação de um jogo com interface de texto (TUI) utilizando a biblioteca "curses". O objetivo principal deste projeto é criar uma experiência interativa envolvente utilizando gráficos ASCII, demonstrando as capacidades de interfaces baseadas em caracteres no desenvolvimento de jogos. O jogo apresenta um mapa simples e finito, um sistema de inventário e várias ações do jogador, incluindo plantar sementes, cortar árvores, construir casas, colher plantações e consumir alimentos. A eficácia da biblioteca curses em fornecer uma experiência de jogo dinâmica e responsiva é avaliada. Os resultados preliminares mostram que os jogos baseados em texto podem oferecer valor de entretenimento significativo e benefícios educacionais em programação e pensamento algorítmico.**

Keywords—curses, classe, desenvolvimento de jogos

I. Introdução

Nos últimos 30 anos, é impressionante que a tecnologia das placas de processamento gráfico (GPUs) deu um enorme salto. A cada nova geração, esses componentes tornam-se mais poderosos, aumentando drasticamente seu desempenho, eficiência e capacidade gráfica comparado à geração anterior. Esta evolução tem impulsionado o mercado de desenvolvimento de jogos, onde grandes estúdios competem para produzir títulos visualmente deslumbrantes, explorando ao máximo o potencial dos melhores hardwares disponíveis no mercado.

Os jogos AAA (Triple-A) são jogos desenvolvidos pelos grandes estúdios, com grandes orçamentos e que são conhecidos por apresentarem os melhores gráficos de jogos para a geração na qual eles são lançados. Gráficos de alta definição, ambientes tridimensionais detalhados e efeitos visuais avançados tornaram-se a "regra", elevando as expectativas dos jogadores quanto à qualidade visual. Entre os fãs mais ávidos existe inclusive uma disputa velada para decidir qual console possui os jogos mais realistas. Mas se, por um lado, sobra beleza e realismo, por outro, há pessoas que dizem que esses jogos pecam falta de originalidade em jogabilidade, art style, mecânicas e inovação.

Na contramão da tendência do mercado, existem os desenvolvedores independentes, que trabalham com menos recursos para desenvolver e promover seu trabalho, mas também sem a pressão dos consumidores. Os jogos indie focam mais em uma jogabilidade inovadora, narrativa envolvente e estilo de arte próprio.

Dentro dessa categoria, os jogos em ASCII representam uma faceta particularmente intrigante. Utilizando apenas caracteres para representar objetos e personagens, esses jogos remetem às raízes da computação gráfica e desafiam a noção de que o avanço tecnológico é sinônimo de melhor experiência de jogo. Em vez de depender de hardware de última geração, os jogos em ASCII focam na criatividade, engenhosidade e na capacidade imaginativa dos jogadores.

Este artigo explora o desenvolvimento de um jogo em ASCII utilizando a biblioteca curses [1] em Python, demonstrando como experiências de jogo significativas e envolventes podem ser criadas sem a necessidade de recursos gráficos avançados. Através deste projeto, foi tentado mostrar que, apesar do avanço inevitável das placas de vídeo e das expectativas associadas, há um espaço vital e inovador para jogos que valorizam a simplicidade e a acessibilidade.

II. Objetivo

O principal objetivo deste projeto é desenvolver um jogo em ASCII utilizando a biblioteca curses, demonstrando que é possível criar experiências de jogo envolventes e interativas mesmo sem o uso de gráficos sofisticados e hardware avançado. Especificamente, o projeto busca:

1. Implementar uma estrutura de código modular e orientada a objetos:

   Desenvolver o jogo utilizando uma abordagem orientada a objetos, com classes bem definidas para representar os diferentes componentes do jogo, facilitando a manutenção e a expansão do código.

2. Criar um ambiente de jogo interativo:

   Criar uma interface interativa que seja acessada através de comandos do teclado. Desenvolver um mapa do jogo onde o jogador pode interagir com o ambiente, coletar recursos, gerenciar um inventário e realizar outras ações, como plantar sementes, cortar árvores, construir casas, colher plantações e alimentar o personagem.

3. Explorar a biblioteca curses:

   Utilizar as funcionalidades da biblioteca para criar uma interface de usuário robusta, que permita a manipulação de caracteres na tela, a captura de entradas do teclado e a criação de janelas e menus interativos.

4. Promover a criatividade e a inovação:

Estimular o desenvolvimento de jogos que não dependam de gráficos avançados, mas sim de criatividade e inovação na jogabilidade e nas mecânicas de jogo.

5. Demonstrar a viabilidade dos jogos em ASCII:

Mostrar que, apesar da dominância dos jogos AAA, existem alternativas que atendem a todos os gostos, são mais baratas e que podem oferecer uma experiência de jogo divertida e desafiadora.

III. METODOLOGIA

Existem várias bibliotecas que podem ser utilizadas no desenvolvimento de um jogo na linguagem Python. Entre as opções, a mais conhecida é a biblioteca Pygame [2]. Essa biblioteca oferece muitas funcionalidades ao programador, tais como renderização de gráficos 2D e 3D, detecção de colisão, efeito sonoros e suporte à animação. O Pygame oferece muito mais do que o necessário para esse projeto. Por isso a biblioteca curses foi a escolhida para desenvolver esse jogo.

Curses é uma biblioteca que permite ao usuário ter total controle sobre o terminal de execução. Tratando a tela do terminal como uma matriz, o usuário consegue alterar todas as posições dessa matriz. A biblioteca fornece funcionalidades essenciais para a manipulação de caracteres na tela, captura de entradas do teclado e controle de janelas e cores, permitindo a criação de interfaces interativas e dinâmicas.

A escolha da biblioteca curses foi motivada por sua versatilidade no desenvolvimento de aplicações baseadas em texto, e pelo apelo visual. A biblioteca foi utilizada para gerenciar a interface do jogo, incluindo a renderização do mapa, a exibição de informações do jogador e do inventário, além da detecção de comandos de controle vindos do teclado – o jogo não possui suporte para mouse.

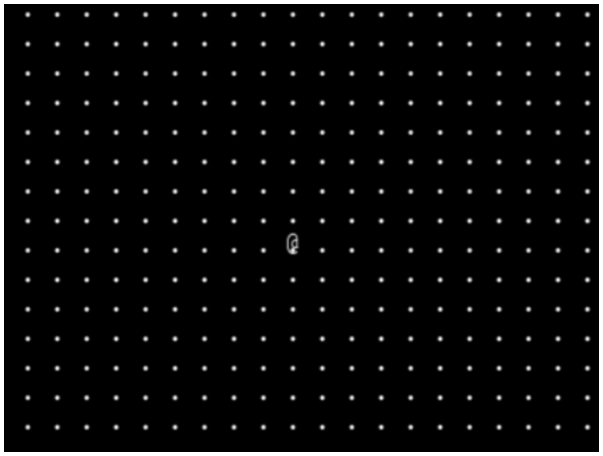

Fig. 1. Tela do jogo feita com a biblioteca Pygame.

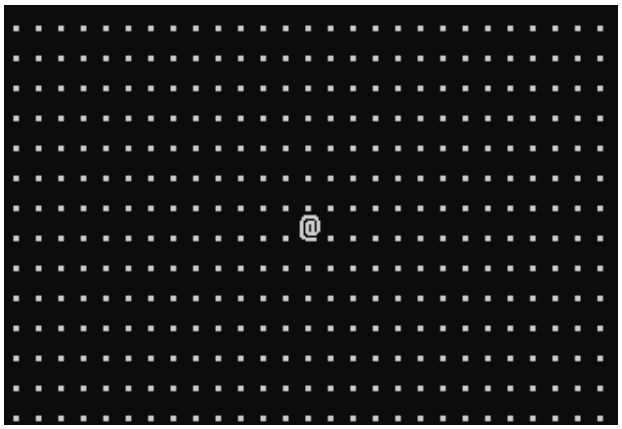

Fig. 2. Tela do jogo feita com a biblioteca curses

A. *O código*

Para iniciar o curses, é necessário usar o método *curses.wrapper()* passando como parâmetro de entrada o nome da função que será responsável por manipular a tela. Quando a função é chamada, ela recebe um objeto que representa a tela e possui métodos para controlá-la.

Devido à dificuldade de manipular a tela através dos métodos de uma classe, foi decidido que todo o controle da tela ficaria dentro do programa principal main.py. A fim de evitar erros, utiliza-se a função *getmaxyx()* para salvar as dimensões da tela em variáveis. Isso garante que o programa não vai escrever em uma posição fora da tela.

Os métodos mais utilizados para controlar a tela são o *clear(), addstr()* e *refresh()*. A função *clear()* limpa a tela. A função *addstr()* adiciona um char ou uma *string* a partir da posição definida pelo par ordenado passado como parâmetro. Todo o mapa, interface, texto e personagens são controlados por essa função. A função *refresh()* atualiza a página implementando as mudanças que foram feitas pelas duas funções anteriores. Para garantir que o conteúdo da tela seja apresentado corretamente é necessário criar uma hierarquia entre as informações. Caso o caractere '#' seja impresso na posição (0, 0) e, em seguida, o caractere '*' seja impresso na mesma posição, ao dar um *refresh* na tela, o caractere '*' será mostrado pois é a informação mais recente adicionada àquela posição.

Também foram criadas outras duas janelas [3] na tela principal utilizando o método *curses.newwin()*. Esse método cria uma janela dentro do espaço da tela principal. A janela possui os mesmos métodos de manipulação que a tela principal, porém possui coordenadas próprias medidas a partir do vértice superior esquerdo.

A cada vez que um novo jogo é iniciado, um mapa novo é criado, nunca sendo igual a algum mapa anterior. A geração do mapa consiste em iterar sobre todas as posições da tela e definir, a partir da geração de um número aleatório, se o bloco daquela posição será de grama ou será de árvore. Para garantir que a geração do mapa seja aleatória e não repetitiva, foi utilizada a *time()* da biblioteca time do Python. Essa função retorna o tempo passado em segundos desde o dia 01/01/1970 00:00. Esse valor é passado como parâmetro para a função *seed()* da biblioteca random.

Para compor o programa foram utilizadas as classes a seguir:

- **Classe Mapa:** Responsável por criar o mapa do jogo de maneira aleatória. As coordenadas e o caractere de cada posição são guardados em um *dict* e em um arquivo .csv, para o caso de o jogador quiser carregar um save antigo.

- **Classe Jogador:** Armazena informações de status do personagem e possui métodos de todas as funções que o jogador pode realizar.
- **Classe Inventário:** Possui uma lista de itens e um arquivo .csv que salva os itens e suas respectivas quantidades para caso o usuário queira continuar seu último jogo.
- **Classe Blocos:** Possui uma lista com os nomes dos blocos e os símbolos que os representam no mapa, além disso possui métodos para adicionar e remover blocos do mapa.
- **Classe Animal:** Classe mãe da classe Jogador que define métodos comuns ao Jogador e a personagens não jogáveis.

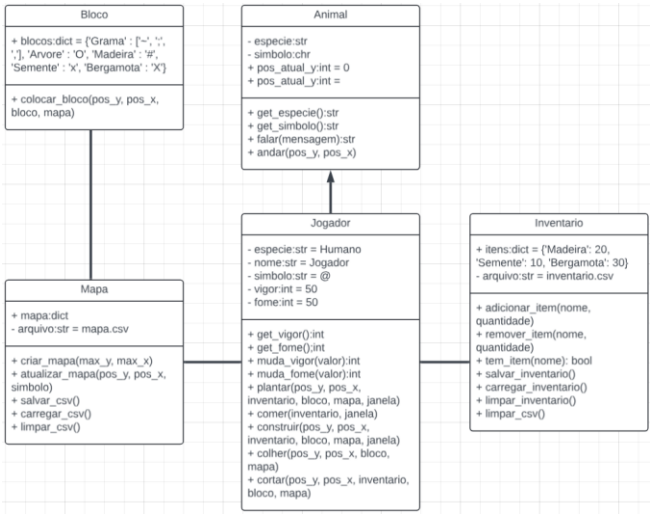

Fig. 3. Diagrama UML de classes

IV. Resultados

O objetivo deste projeto era criar o protótipo de um jogo simples utilizando a biblioteca curses do Python, com algumas funcionalidades planejadas, incluindo o movimento do jogador, a interação com o ambiente e a manipulação de inventário.

Funcionalidades implementadas:

- Movimentação do Jogador:
  - O usuário pode mover o jogador utilizando as setas do teclado.
  - O personagem não sai do espaço delimitado pelas bordas da tela.
  - Foi implementada a colisão entre os personagens e blocos, dando um toque de realismo.
- Inventário:
  - O jogo possui um inventário que permite o gerenciamento dos itens de forma simples
  - As funções possuem tratamento de erros para evitar que o jogador possua quantidades inexplicáveis de itens.
- Interface de Usuário:
  - A interface foi desenhada com diferentes janelas para exibir informações do jogador, inventário e log de mensagens. As cores foram configuradas para melhorar a legibilidade, a experiência do usuário e deixar a interface mais viva.
- Interação com blocos:
  - Foram criadas interações básicas com o ambiente, incluindo plantar sementes, cortar árvores, construir casas e comer itens. As funcionalidades são acionadas por teclas específicas, oferecendo uma interface interativa ao jogador.

Funcionalidades não implementadas:

- Movimentação automática do Jogador:
  - A ideia inicial era fazer a movimentação do personagem ser completamente automática. Ao usuário caberia designaria tarefas pelo teclado e o personagem iria andando até o local de realização da tarefa, semelhante a como é feito no jogo The Sims.
- Crescimento do sementes:
  - A implementação de uma mecânica de crescimento de sementes, onde as plantas cresceriam ao longo do tempo, também não foi concluída. Este recurso exigiria a criação de um sistema de temporização para a evolução das plantas, o que não foi implementado devido às limitações de tempo e complexidade.
- Salvar jogo:
  - Apesar de ter criado arquivos .csv e os métodos para salvar informações como os blocos do mapa e o inventário, especificamente o arquivo do mapa não estava atualizando à medida que a estrutura do mapa era alterada pelo jogador.
- Personagens não jogáveis:
  - Não ter conseguido implementar o movimento automático dos personagens tornou-se irrealizável a criação de NPCs, como inimigos, animais etc.
- Variedade de itens:
  - Devido à falta de familiaridade com ferramentas de manipulação de dados, seria desafiador criar uma lista variada de itens, tendo em mente os métodos que precisam buscar informação da base de dados.

- Classe para a tela:
    - Um objeto que representa a tela é passado para a função principal do programa. Controlar a tela, passando o controle dela de um método para outro deixou o código muito confuso e com muitos bugs.

V. Conclusão

O desenvolvimento deste projeto permitiu explorar as capacidades de criação de jogos em ambiente de terminal, demonstrando que é possível construir experiências interativas com recursos limitados.

A experiência de trabalhar com curses também evidenciou as vantagens e limitações de se criar jogos em ASCII. Enquanto a simplicidade gráfica permite um foco maior na lógica e nas mecânicas do jogo, a ausência de elementos visuais mais ricos pode limitar a imersão do jogador.

Por fim, o projeto alcançou seus objetivos principais e proporcionou um entendimento mais profundo sobre desenvolvimento de jogos em ambientes de texto. Além disso, ressaltou a importância da persistência e da aprendizagem contínua na superação de obstáculos técnicos. Este trabalho não só contribuiu para o crescimento das habilidades técnicas, mas também reforçou a importância de abordar desafios de maneira criativa e adaptável.

O código se encontra disponível em [4].

Referências

[1] curses — Terminal handling for character-cell displays. Disponível em: <https://docs.python.org/3/library/curses.html#>.

[2] PYGAME. Pygame Front Page — pygame v2.0.0. dev15 documentation. Disponível em: <https://www.pygame.org/docs/>.

[3] TECH WITH TIM. Python Curses Tutorial #3 - Windows And Pads. Disponível em: <https://www.youtube.com/watch?v=VzhZ1nTeAsA&list=PLzMcBGfZo4-n2TONAOImWL4sgZsmyMBc8&index=3>.

[4] PIMENTEL, J. joaovic97/PROJETO-FINAL-POO. Disponível em: <https://github.com/joaovic97/PROJETO-FINAL-POO>.