RELATÓRIO EMAPTALE

Rafael de Pinho André (orientador)
Alex Júnio Maia de Oliveira
Bruno Ferreira Salvi
João Pedro Jerônimo de Oliveira
Thalis Ambrosim Falqueto

RESUMO

Este relatório apresenta o desenvolvimento do projeto "EMApTale", um jogo virtual criado como forma de avaliação do curso de Linguagem de Programação. Inspirado no jogo *Undertale* e nas experiências dos integrantes na Escola de Matemática Aplicada (EMAp) da Fundação Getulio Vargas, o projeto foi concebido para reforçar conceitos de Programação Orientada a Objetos, organização modular e documentação de código. O enredo do jogo narra a jornada de um aluno tentando recuperar notas durante a semana de Avaliações Substitutivas (AS), enfrentando desafios representados por professores em batalhas de turnos. O desenvolvimento envolveu etapas de idealização, criação de sprites e sons, programação do código, testes unitários e elaboração de um relatório final. Cada membro do grupo contribuiu em áreas específicas, como criação de mapas, combate, cutscenes e menus. A estrutura modular e reutilizável do código garantiu organização e eficiência no desenvolvimento. O jogo também oferece uma ambientação interativa e desafios imersivos que refletem a rotina acadêmica do aluno na EMAp.
Palavras-chave: Programação orientada a objetos, jogos 2D, Python, Pygame, EMAp.

1 ESTRUTURA

O jogo virtual "EMApTale", forma de avaliação referente ao curso *Linguagem de Programação*, foi baseado em outra obra eletrônica chamada *Undertale*[1] e nas experiências dos integrantes do grupo na Escola de Matemática Aplicada da Fundação Getulio Vargas - EMAp[2].

O processo de desenvolvimento foi dividido em várias etapas:

- **Idealização do jogo:** Período de escolha do tema e estruturação das partes de cada integrante.

- **Pesquisa dos sprites, sons e artes visuais:** Período de pesquisa dos sprites[3] utilizados em cada parte do jogo, da trilha sonora e da arte visual empregada. Além disso, também foi buscado vídeos informativos sobre a biblioteca *Pygame*.

- **Desenvolvimento do código:** Período de realização do código, com o objetivo de utilizar e reforçar o conceito de Classes e Programação Orientada a Objetos, realização de documentação e testes unitários.

- **Realização de testes:** Período de realização de testes unitários para as principais funções e classes empregadas, além de testar as movimentações e interações presentes no jogo virtual.

- **Relatório:** Confecção de um relatório destacando os principais pontos desenvolvidos na execução do projeto, bem como a estruturação

- **Apresentação:** Apresentação do projeto desenvolvido em grupo para o professor orientador, Rafael Pinho, e para o corpo discente.

Além disso, dividimos as funções para cada integrante da seguinte forma: Alex desenvolveu a parte do combate dos chefes. Bruno criou as cutscenes e as transições dessas para a gameplay. João realizou a estruturação básica das classes e configurou o repositório do projeto, além de criar os menus. Por fim, Thalis confeccionou o mapa, a câmera, as interações, a movimentação e as colisões.

O código foi feito de forma modular e organizado, além de subdividir cada parte em pastas explícitas para uma melhor compreensão no momento do desenvolvimento. A estruturação pode ser encontrada no seguinte repositório do GitHub[4]: `https://github.com/jaopredo/EMApTale`.

2 HISTÓRIA

O jogo se baseia em um aluno da EMAp da Fundação Getulio Vargas do segundo período de Ciência de Dados e Inteligência Artificial que está na semana de AS tentando recuperar as notas baixas tiradas nas avaliações dos cursos. Para recuperar as notas e avançar de semestre, o aluno deverá enfrentar os professores do período e derrotá-los em uma árdua batalha de turnos.

[1] Undertale é um jogo do estilo *RPG* criado pelo desenvolvedor *indie TOBYFOX* EM 2015. Pode ser encontrado para compra em `https://store.steampowered.com/app/391540/Undertale/`

[2] Criada em 2011 com o objetivo de desenvolver a matemática contemporânea, adaptada aos desafios da era da informação e do conhecimento.

[3] Um sprite é um personagem ou um objeto dentro dos jogos 2D.

[4] GitHub é uma comunidade de código aberto e fundamental para a forma como é construído software atualmente.

Além disso, a cada professor derrotado, o aluno avança de dia para encarar o próximo desafio, podendo interagir com lugares conhecidos do andar da EMAp, tais como a sala de estudos, a sala dos professores, as mesas de estudo, os sofás, os livros e jornais, a sala do CDMC[5], entre outras.

O personagem conta com vários coletáveis (presentes no cotidiano do aluno) no mapa do andar da escola que o ajudarão na batalha com os professores, tais como livros, simulados, listas, café, bolos, entre outros.

Ao completar a semana de AS, após derrotar todos os professores, o personagem consegue a nota necessária para recuperar as notas em todas os cursos e avançar de semestre.

3 DESENVOLVIMENTO

3.1 Passos Seguidos

No desenvolvimento por detrás do projeto do jogo eletrônico, optamos por utilizar uma programação modular[6] e orientada a objetos[7], proposta e ensinada pelo professor orientador Rafael de Pinho André[8]. Além disso, fizemos uso de algumas estruturas de configuração internas para facilitar o desenvolvimento, tais como arquivos *.json* para armazenar as informações do personagem, dos coletáveis e dos inimigos, módulos com constantes globais e classes de gerenciamento dos eventos contidos no jogo e uma pasta com funções úteis e reutilizáveis no código.

Nos passos iniciais, além da idealização da ideia, nos concentramos em pesquisar os sprites, a trilha sonora, os efeitos sonoros e realizar uma customização das informações dos chefes (nome, vida, ataque, trilha, falas, entre outros). Ademais, confeccionamos o mapa, tendo a inspiração da planta baixa do andar da EMAp.

A seguir, fizemos os sprites do personagem, dos chefes e dos ataques, além da movimentação do personagem no mapa e das interações com os objetos, estruturas e coletáveis.

Após, realizamos os menus de início, de batalha e das ações do personagem em cada turno , o inventário, a tela inicial e o esquema de salvamento de progresso do personagem e a tela de *gameover*, inicializada quando o jogador perde a batalha.

Seguindo, idealizamos as cutscenes do início do novo jogo, as telas de combate, um esquema de armazenar o nome do jogador e um sistema de caixa de texto/falas parecido com de *Undertale*.

A seguir, fizemos a parte do combate de cada chefe, aplicando efeitos característicos de cada um, idealizando os diferentes ataques e um sistema de turnos interativos para cada ação personagem-chefe.

Por fim, realizamos testes unitários de funções de movimentações, de algumas classes dos ataques dos chefes, do inventário e das funções internas de configurações, documentamos o código, realizamos *doctest*, *doctype* e *docstring* e encerramos a parte do desenvolvimento.

3.2 Pastas Do Código

Neste momento, iremos discorrer sobre o conteúdo de cada pasta utilizada no desenvolvimento do projeto, destacando suas funções e como foram reutilizadas ao decorrer do

[5]Centro de Desenvolvimento para a Matemática e Ciências.

[6]Abordagem que divide um programa em módulos independentes e bem definidos

[7]Criada para aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real. Essa abordagem se baseia em classes e objetos.

[8]Currículo Lattes: `http://lattes.cnpq.br/9828097913107361`

código[9].

Primeiramente, na raiz do repositório, estão presentes alguns arquivos de suma importância. O arquivo *main.py* é utilizado para realizar a inicialização do jogo (basta executá-lo para começar a jogatina). Além do *main.py*, há também o arquivo *README.md* que possui informações essenciais para a clonagem do repositório e instalações das dependências.

Outros arquivos como *LICENSE, requirements.txt, .gitignore* dizem sobre o tipo de licença que o repositório possui, as dependências necessárias para a instalação e bom funcionamento do código e partes que a plataforma do GitHub deve ignorar, respectivamente.

Começando, a pasta **CLASSES** contém todas as classes fundamentais para o jogo, estruturadas da seguinte forma:

- **battle:** Armazena as classes da batalha, como a classe do player, container do hp, etc.
- **menus:** Armazena as classes do menu da batalha, como os botões "agir", "render", etc.
- **hud:** Aqui ficam os elementos finais do HUD, como a barra de dano.
- **bosses:** Aqui ficam as classes de todos os chefes (bosses), incluindo os ataques de cada boss.
- **effects:** Aqui ficam os efeitos individuais de alguns bosses.
- **map:** Aqui fica toda a estruturação do mapa, câmera, interação, etc.
- **polygon:** Aqui fica a colisão de um polígono, especificamente para cantos não retangulares.
- **sprites:** Aqui fica a classe que renderiza os sprites.
- **text:** Aqui ficam as classes de textos, utilizados tanto na batalha quanto no mapa.

Seguindo, vamos começar pelas pastas mais estruturais, que não armazenam tantas informações brutas, mas que moldam todos o projeto e sua reutilização é essencial para o bom andamento do jogo, além dos testes unitários desenvolvidos:

- **CONFIG:** Na pasta config estão os gerenciadores de combate, de eventos, de texto, de salvamento de progresso, de som e de cutscenes, além de definir o nome do jogo, o caminho base e a quantos frames por segundo o jogo deveria rodar.
- **CONSTANTS:** Nesta pasta estão armazenadas constantes relacionadas aos turnos dos chefes e do personagem, auxiliando na ordenação de cada turno.
- **TESTS:** Aqui estão os testes unitários das funções mais gerais utilizadas, de classes referentes aos ataques dos chefes, classes do inventário e sobre a movimentação do personagem.
- **UTILS:** Na pasta utils estão algumas funções reutilizáveis em certas partes do código.

Prosseguindo, vamos discorrer sobre pastas que armazenam os sons, os sprites e as informações utilizadas para os coletáveis, o personagem e os chefes:

- **INFOS:** Nesta pasta estão os arquivos *.json* utilizados para armazenar as informações dos chefes, dos coletáveis e do jogador.
- **FONTS:** Na pasta fonts estão as fontes de texto utilizadas nas caixas de diálogo (interações no mapa, durante as batalhas, menu inicial, inventário, menu de pausa, entre outros) presentes no jogo.
- **SOUNDS:** Aqui estão todos os sons (trilha sonora e efeitos sonoros) utilizados no projeto.
- **SPRITES:** Nesta pasta estão os sprites utilizados nos chefes, nas cutscenes, nos efeitos e na interface gráfica da batalha. Além disso, há também os sprites do personagem e do mapa.
- **TILESET:** Na pasta tileset estão os sprites dos objetos do mapa e do personagem usado nele. Além disso, estão representados as colisões e um arquivo *.tmx* para facilitar a reprodução do mapa na biblioteca *Pygame*.

Continuando, a pasta **SCREENS** contém classes para telas mostradas durante a gameplay,

[9]Repositório disponível em: `https://github.com/jaopredo/EMApTale.git`

tais como o menu inicial (juntamente com as partes de captar o nome do jogador e fazer o salvamento), as cutscenes inicial e final, a tela de começo do combate, o módulo principal que inicializa o mapa e a cena de gameover.

Por fim, a pasta **REPORT** possui o relatório realizado após a conclusão do projeto, contando os passos que o grupo seguiu, além de detalhes sobre a estrutura do código e da organização do repositório.

Além disso, é importante observar que o código do projeto foi estruturado para facilitar a reutilização e modularização, com funções auxiliares na pasta **UTILS**, constantes definidas na pasta **CONSTANTS** e toda a configuração do jogo centralizada na pasta **CONFIG**. Com isso, a manutenção e a adição de novas funcionalidades ao projeto são facilitadas e o desenvolvimento foi realizado de forma mais eficiente.

4 DIFICULDADES

No geral, as principais dificuldades giraram em torno de duas coisas: falta de conhecimento e tempo. Por exemplo, na construção do mapa, ao fazer o mapa no jogo, tinhamos que renderizá-lo para que ficasse no tamanho inteiro da tela. Por falta de conhecimento, foi escolhida uma maneira talvez burra de renderizar o mapa, centrada em multiplicar cada sprite individual do mapa em um tamanho bom que se adequasse a tela do mapa. Além disso, a cada sprite era somado um vetor que jogava cada sprite para um canto da tela, de forma a ocup-lá toda. Funcionou na hora, mas não esperávamos a quantidade de problemas que iriam chegar depois dessa escolha feita na renderização do mapa.

Como problemas diretamente ligados a má renderização do mapa, temos a dificuldade de implementar a câmera, a renderização de cada colisão, interação e item utilizável, o "nascimento"do player no mapa, o lag que obtivemos ao renderizar cada tile a cada vez que o player movimenta, o que pesou o mapa, e vários outros menores que aconteceram e fizeram a equipe perder tempo de trabalho.

Outro problema foi relacionado a usar corretamente o tempo no pygame: para fazer as cutscenes, onde o tempo deve ser cronometrado e corretamente calculado, regular o tempo de imagem, texto e som de forma que ficasse tudo sincronizado custou bastante tempo devio a falta de experiência com esse tipo de aplicação. Em conjunto, os efeitos de esmaecer e escurecer das transições demandou um certo esforço.

Uma dificuldade aparentemente singela foi unificar as classes de inserção de texto dinâmico - classe que desenha letra por letra - pois tínhamos duas em produção, assim foi adicionado as funcionalidades de quebrar linhas sem partir as palavras no meio, funcionalidades esperadas.

Além disso, somado aos problemas supracitados, houveram dificuldades com relação ao posicionamento de cada atributo para criar as classes dos ataques e instanciá-los nas classes dos chefes. Ademais, criar ataques personalizados e escolher trilhas sonoras específicas para cada chefe foi um pouco desafiador.

Além do mais, construir uma estrutura flexível e facilmente manutenível para ser utilizada ao longo do projeto foi uma das maiores dificuldades no desenvolvimento do trabalho. Continuando nessa perspectiva, gerir responsabilidades aos integrantes do grupo foi um ponto de impasse e um grande desafio.

REFERÊNCIAS

[1] The Spriters Resource. *Undertale Spritesheet Collection*. Disponível em: https://www.spriters-resource.com/pc_computer/undertale/. Acesso em: 28 nov. 2024.

[2] The Sounds Resource. *Undertale Sounds Collection*. Disponível em: https://www.sounds-resource.com/pc_computer/undertale/sound/6275/.Acesso em: 29 nov. 2024.

[3] João Pedro. Repositório **EMApTale**. Disponível em: https://github.com/jaopredo/EMApTale.git. Acesso em: 30 de nov. de 2024.

[4] Enter Sandman (Remastered) **Metallica**. Disponível em: https://www.youtube.com/watch?v=XZuM4zFg-60. Acesso em: 04 de nov. de 2024.

[5] Doom OST - E1M1 **Gordon**. Disponível em: https://www.youtube.com/watch?v=BSsfjHCFosw. Acesso em: 04 de nov. de 2024.

[6] Lchavasse - Lunar Abyss **Geometry Dash**. Disponível em: https://www.youtube.com/watch?v=F_TV4vZRSE8. Acesso em: 04 de nov. de 2024.

[7] Thunderstruck **AC/DC**. Disponível em: https://www.youtube.com/watch?v=v2AC41dglnM. Acesso em: 04 de nov. de 2024.

[8] Betrayal of Fear **Geometry Dash**. Disponível em: https://www.youtube.com/watch?v=eXv0tqBtv3E. Acesso em: 04 de nov. de 2024.