FUNDAÇÃO GETÚLIO VARGAS

BRYAN SANTOS MONTEIRO

NICHOLAS NOGUEIRA COSTA

NÍCOLAS MATEUS SPANIOL

JOÃO VITOR TOMAZ ALVES FERREIRA

SOFIA AZEREDO DE MOURA MONTEIRO

Relatório para a Avaliação 2 de Linguagens de Programação

Slay the Odyssey

Professor: Rafael Pinho

Rio de Janeiro

2024

1. **Estruturação Inicial**

A primeira etapa do processo consistiu na criação de um diagrama UML para estruturar o jogo. Este diagrama nos forneceu uma visão inicial, permitindo dividir as tarefas entre os integrantes, o desenvolvimento do mapa, criação do esqueleto de combate e o desenvolvimento do menu e das cartas.

A equipe manteve uma comunicação constante para acompanhar o progresso de cada integrante e colaborar na resolução de problemas que surgiram ao longo do processo. Entretanto, o diagrama UML inicialmente utilizado foi rapidamente descartado, pois não representava de forma adequada os desafios encontrados no desenvolvimento. Após criar os esqueletos básicos de cada funcionalidade, o foco passou para implementar elementos mais relevantes, como os efeitos diversos nas cartas, as animações de ataque e de recebimento de dano, a implementação de efeitos sonoros e mais.

2. **Produção Visual**

A criação das sprites foi uma etapa fundamental e exigiu esforço significativo da equipe. As produções iniciais foram realizadas por Nicholas, para teste, mas posteriormente houve maior distribuição da produção. Foram desenhadas cartas, inimigos, Ulisses, o personagem principal, e os fundos de combate, todos de autoria do grupo. Este trabalho visual foi essencial para transmitir a ideia do jogo e garantir a imersão na narrativa da Odisseia.

3. **Desafios Técnicos**

Alguns desafios encontrados inicialmente foram a familiarização com a documentação da biblioteca Pygame, além, é claro, da inexperiência de alguns membros quanto à aplicação prática dos aprendizados. Um dos principais obstáculos práticos enfrentados foi a escalabilidade das telas do jogo. Algumas telas foram produzidas e apareciam em escalas diferentes, o que comprometia a experiência do jogador. Após superar esse empecilho, porém, foi possível integrar as diferentes partes do jogo.

4. **Gameplay e Mecânicas**

A mecânica central do jogo é baseada na construção de um baralho com cartas poderosas, permitindo ao jogador desenvolver estratégias de combate. Após cada batalha, o jogador recebe uma nova carta para adicionar ao seu baralho, o que enriquece as possibilidades

estratégicas para os desafios subsequentes. No mapa de batalhas, é necessário planejar cuidadosamente o caminho a seguir, considerando quais combates evitar ou atrair e se será necessário passar por fogueiras para recuperar vida antes de enfrentar novos desafios.

O sistema de combate é baseado em turnos e funciona com um número limitado de energia disponível para o jogador a cada rodada. Cada carta possui um custo em energia para ser usada e um efeito especial, sendo divididas em cartas de ataque, que aplicam dano nos inimigos, e cartas de defesa, que ajudam o jogador a mitigar ou evitar danos recebidos. Além disso, existem cartas com efeitos especiais que ampliam as possibilidades estratégicas, dependendo da composição do baralho. Um baralho focado em efeitos permite estratégias mais indiretas, exigindo eficiência para derrotar inimigos sem ataques constantes, enquanto um baralho focado em ataques requer maior atenção à gestão de escudo e energia, para evitar a perda significativa de vida ao longo das batalhas.

Após o jogador encerrar seu turno, os inimigos realizam suas ações, que incluem ataques diretos e aplicação de efeitos negativos, como debuffs. Esse ciclo contínuo entre o planejamento no mapa, a construção do baralho e os combates estratégicos compõe a dinâmica do jogo, proporcionando desafios e incentivando o jogador a adaptar suas táticas conforme progride.

4.1. Mapa do Jogo

Para o mapa do jogo, pegamos uma inspiração forte do Slay the Spire, ambos em visual e em funcionalidade. Nosso mapa consiste em duas partes: uma classe MapScreen, que herda a classe abstrata que criamos, Screen, que representa uma tela à qual o jogador tem acesso e uma classe MapNode, que representa um nó no mapa do jogo. Cada nó pode ser uma batalha, um nó de história ou uma fogueira (onde o jogador será curado) e a classe MapNode contém como parâmetro os nós aos quais aquele se conecta. Ou seja, a classe MapNode consiste literalmente em um nó de um grafo direcionado, e toda a lógica de navegar no mapa do jogo consiste em navegar nesse grafo, indo de um nó a outro adjacente.

Fazer o visual do mapa trouxe alguns desafios interessantes, dos quais o principal foi desenhar os caminhos entre os nós do mapa. Inicialmente, cada caminho era desenhado usando uma função do pygame que gera uma linha entre dois pontos. Porém, a fim de manter o visual mais fiel ao jogo original, optamos por desenhar cada "traço" do caminho separado a fim de se assemelhar, de fato, a um mapa. Também queríamos manter o estilo pixel art do

jogo ao contrário de gerar o caminho como uma linha como era feito antes, dando a ideia de que a arte de cada caminho foi feita manualmente.

Nossa ideia foi interessante: desenhamos, na mão, alguns traços de um caminho em ângulos de 0 a PI graus radianos, e escrevemos o código que automaticamente desenha os sprites na tela a fim de formar um caminho entre dois nós. O código recebe os vetores dos dois nós a terem seu caminho desenhado, calcula o ângulo, a distância entre eles e o número de traços a serem desenhados de forma a ter uma distância relativamente constante entre dois traços independente da distância dos nós. Como só desenhamos os traços pra ângulos entre 0 e PI, o código também inverte os sprites quando necessário e se aproveita do fato que, sendo uma linha reta, o ângulo imediatamente oposto ao do sprite pode ser desenhado usando o mesmo sprite sem perda visual. Assim, conseguimos deixar o mapa visualmente agradável sem ter o trabalho de desenhar todos os caminhos na mão, além de nos permitir atualizar a cor dos caminhos ao longo do jogo.

Uma preocupação que tivemos, entretanto, foi na performance. O código que desenha os caminhos é relativamente complexo, podendo deixar o jogo mais lento enquanto o jogador estiver vendo o mapa. Para resolver isso, ao invés de desenhar todas as arestas em cada frame, as desenhamos uma única vez no início do jogo (e redesenhamos as arestas necessárias ao longo do jogo depois que o jogador navega de um nó a outro) em uma textura fixa e apenas colamos (usando blit) a textura na tela a cada frame, o que simplifica bastante o processo. Além disso, como já fazíamos literalmente isso com o fundo do mapa, no lugar de usar uma textura separada apenas desenhamos as arestas sobre o próprio fundo.

5. Conclusão

O grupo confeccionou uma *demo* de jogabilidade funcional e completa, por mais que algumas das ideias originais de maior complexidade tenham que ter sido abandonadas. Apesar das dificuldades e mudanças de plano, no fim, o grupo julgou o jogo resultante desse trabalho como interessante e divertido, além de caracterizar a experiência da proposta como construtiva em diversos sentidos.