SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA · UFV
CAMPUS FLORESTAL

Multi-Man

CCF 313 - Programação Orientada a Objetos

Aléxia Karoline Augusta Germano Silva EF5373
César Grandis Costa EF5371
Edmarcos Mendes Pinto Filho EF5359
Vinicius Oliveira EF5385
Lucas da Costa Moreira EF5377
Helom Marques EF5892

Florestal - MG
2025

Sumário

1. Introdução

Este projeto apresenta o Multi-Man, uma recriação inspirada no clássico Mega Man, com alguns elementos originais que adicionam a própria identidade dos desenvolvedores ao jogo. O objetivo principal foi resgatar a jogabilidade envolvente do jogo original, mantendo o estilo visual de pixel art e as mecânicas características do jogo.

Além de homenagear um marco na história dos videogames, o desenvolvimento do Multi-Man serviu como oportunidade para aplicar conhecimentos de programação , design e desenvolvimento de jogos. Durante o projeto, foram trabalhados desafios técnicos como a implementação de inimigos, a física de plataforma e a troca de poderes do protagonista, garantindo que a essência do jogo fosse preservada.

O resultado é um jogo que combina nostalgia com inovação, proporcionando uma divertida experiência para os jogadores. O intuito deste documento é detalhar o processo de criação, os desafios enfrentados e soluções encontradas para dar vida a Multi-Man.

2. Desenvolvimento

2.1. Arquitetura do projeto

Em relação a organização da arquitetura do projeto, separamos as pastas em **assets**, para os sprites e áudios, **src** para os arquivos fontes do projeto, com os códigos que estruturam todo o jogo e a pasta **out**, com os arquivos já em bytecode. Na pasta assets também foram incluídos os arquivos de texto que identificam as imagens e timings das animações usadas pelos personagens do jogo.

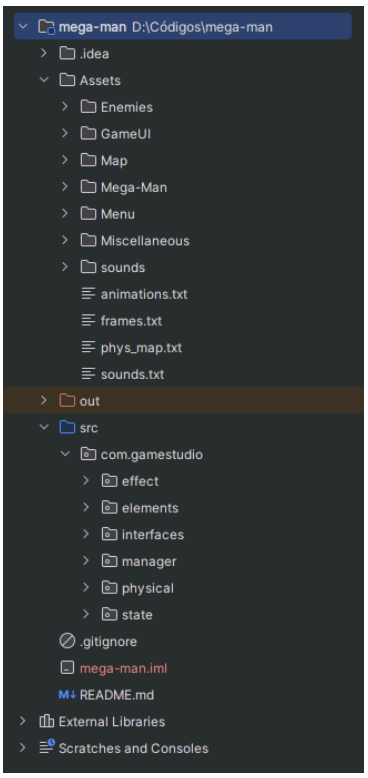

Figura 1 - Estrutura do projeto

2.2. Design

- **Assets:** Englobam todos os recursos multimídia utilizados para a criação do ambiente interativo do jogo. Inclui sprites foram inspirados ou reutilizados de franquias clássicas como Mega Man e Mario. Para adaptar à nossa ideia central, os sprites do Multi-Man passaram por uma recoloração para representar os elementos de fogo, raio e água. A imagem das versões do Multi-Man está representada abaixo.

Figura 2 - Versões do Multi-Man

Além disso, também temos as músicas e os efeitos sonoros, ambos reutilizados do Mega Man II, os sprites dos inimigos, o mapa e todos os outros recursos necessários.

- **Mapa:** A recoloração foi feita no GIMP buscando representar uma paleta outonal. O recorte foi efetuado no Paint.NET, enquanto a manipulação dos tiles foi feita no Aseprite, respeitando uma grade de 16x16 pixels, garantindo um melhor alinhamento para a colisão das entidades.

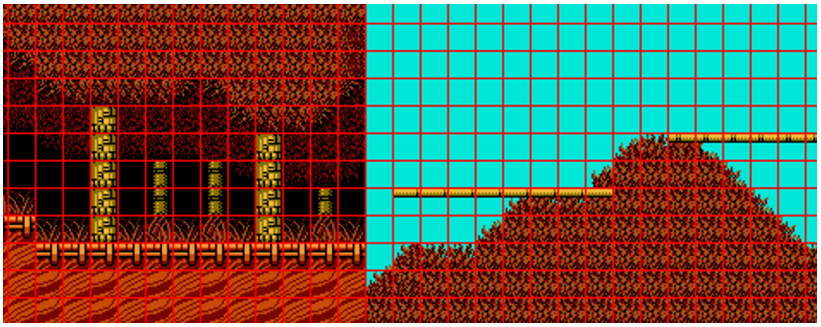

Figura 3 - Mapa de outono

- **Interface:** A interface também é similar ao do jogo supracitado, com os assets da tela de título e a barra de vida importados para este jogo, apenas com uma pequena modificação na tela inicial, indicando que o jogo é uma versão "UFV".

2.3 Elementos do Jogo

- **Personagem principal:** Em relação ao Multi-Man, foram implementados todas as animações para seu funcionamento, incluindo animações de andar, pular, atirar e morrer. Além disso, ao pressionar **F**, **B** ou **E** o personagem muda de poder por um curto intervalo de tempo, possibilitando assim uma diversificação no game.

- **Inimigos:** No jogo, foram implementados três tipos de inimigos comuns, além do boss final. Os inimigos são: Robbit, Battom e Goomba(Personagem do Super Mario). O boss final, que representa um desafio maior, é o WoodMan.

Os personagens mencionados acima podem ser visualizados na figura abaixo, na mesma ordem em que foram citados.

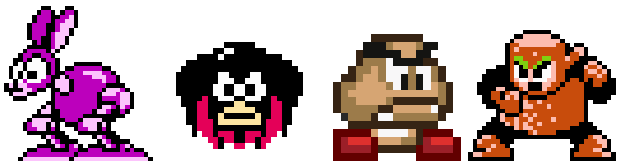

Figura 4 - Inimigos do jogo

- **Fase:** A fase implementada no jogo foi a WoodMan Stage, ambientada em uma floresta. Com plataformas de madeira e inimigos temáticos do jogo original, o cenário proporciona uma experiência envolvente. Além disso, a paleta de cores da fase foi alterada, adicionando um toque mais original ao design visual. A figura abaixo ilustra a fase implementada.

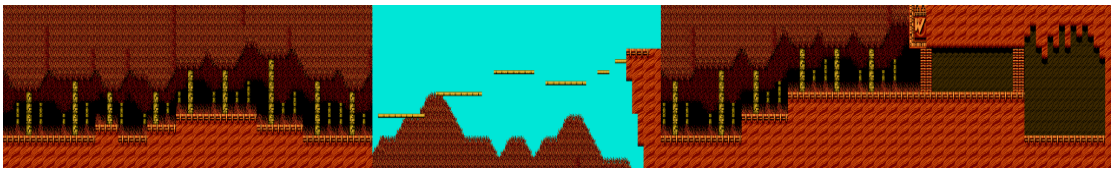

Figura 5 - Fase implementada

- **Itens e Poderes:**

  O jogo possui no total 4 poderes, cada um deles proporciona alguma habilidade diferente ao Multi-Man:

  1. Normal: Versão padrão do personagem principal, possui disparos de plasma que causam 1 de dano;
  2. Fogo: Possui disparos de fogo que causam 5 de dano;
  3. Água: Ao terminar a transformação, o personagem cura um pouco da vida, adicionando 1 ponto por cada 850 milissegundos de ativação da transformação;
  4. Elétrico: Possui o dobro da velocidade das demais transformações.

  Além disso, existe um item que pode ser dropado dos inimigos da fase, tendo uma chance de 25% de ser dropado, caso o item seja coletado, a vida do personagem é regenerada em 5.

- **Interface de usuário:** O jogo conta com um menu inicial simples e funcional, oferecendo duas opções: iniciar o jogo ou sair. Durante a partida, a tela exibe a barra de vida do personagem, garantindo ao jogador uma visão clara do progresso. Quando a batalha com o chefe final começa, uma barra de vida adicional aparece para mostrar o status do chefe. Caso o jogador perca todas as vidas, uma tela de *Game Over* é exibida, oferecendo três opções: reiniciar o jogo, retornar ao menu principal ou sair.

2.4. Mecânica

2.4.1 GameElement

Esta classe representa um elemento genérico dentro do jogo, contendo apenas a posição (x, y) do objeto. Por esse motivo, a maioria das classes do projeto herda desta classe, servindo como base para os demais elementos do jogo.

2.4.2 GameEntity

Esta classe representa uma entidade específica no projeto, com atributos como largura, altura e velocidade, dano etc. Herda da classe GameElement, e diversos componentes da gameplay, como projéteis, inimigos e itens, são derivados dela.

Um aspecto crucial desta classe é o gerenciamento de estados, uma vez que qualquer entidade do jogo pode estar em diferentes condições, como viva, morta ou machucada. Dessa forma, a classe é responsável por identificar o estado atual da entidade e garantir a transição correta entre os estados.

2.4.3 Projectile

Representa qualquer projétil disparado por alguma entidade do projeto, como esta também é derivada da classe GameEntity, possui interações de colisão com outras entidades, sendo sua diferença principal para uma entidade genérica do projeto é o acréscimo de mais um estado, sendo ele o COLLIDED, que informa quando o projétil colidiu com alguma entidade.

2.4.4 DumbGameEntity

Esta classe herda da classe GameEntity, onde irá representar uma entidade com programação simples dentro do jogo,por exemplo os inimigos do jogo, que possuem uma programação repetitiva, pois normalmente irão apenas seguir o Multi-Man ou seguirão uma direção até colidirem com uma parede.

2.4.5 SmartGameEntity

Diferentemente da classe DumbEntity, esta classe representa uma entidade inteligente dentro do jogo e herda da classe GameEntity. Objetos dessa classe possuem mecânicas avançadas, como disparo de projéteis, programação de colisões específicas para uma interação mais precisa com o cenário e uma mecânica de pulo mais sofisticada.

Um exemplo de aplicação dessa classe são o Mult-Man e o Wood-Man, que se destacam por suas mecânicas avançadas, como disparo, pulo e a execução de diversas ações diferentes durante a gameplay.

2.4.6 Managers

O projeto possui no total 3 managers, onde cada um deles é responsável por gerenciar diferentes tipos de entidades dentro do jogo, possuindo funcionamento semelhantes, abaixo estão todo os managers do jogo:

GameEntityManager: Coordena o funcionamento de toda sas entidades que está no estado de ALIVE do projeto, desenhando todas na tela e as atualizando a cada loop do jogo;

ProjectileManager: Esta classe herda da GameEntityManager, sendo responsável pelo funcionamento de todos os projéteis ativos dentro do jogo, os desenhando na tela e os atualizando;

ItemManager: Esta classe também herda da GameEntityManager, porém é responsável por coordenar o funcionamento de todos os itens no jogo, atualizando o funcionamento de todos eles.

Além disso, o projeto conta com a classe **DataLoader**, responsável por armazenar todos os frames, animações e sons em uma única estrutura. Cada um desses arquivos é mapeado por meio de tabelas hash, o que facilita o acesso a esses assets por outras classes.

Para adicionar esses arquivos à classe, são utilizados os arquivos de entrada frames.txt, animations.txt, phys_map.txt e sounds.txt, que contêm as especificações de cada asset, como seus nomes e diretórios.

2.4.7 Mega-Man

Representando a entidade principal do jogo, MegaMan, é uma classe que herda os atributos de um SmartGameEntity, como posicionamento, sistema de colisões, e estados (morto, vivo, machucado). Além disso, implementando por meio de sobrecarga de métodos, os seus próprios comportamentos, como **update()**, que controla as animações em suas mudanças de estado, **attack()**, que define um ataque específico e suas regras, **run()**, **stopRun()**, **jump()** que representam os movimentos, **getBoundForCollisionWithEnemy()** que define a hitbox (área atacável) do personagem e **draw()**, responsável por desenhar o elemento na tela.

Além dos atributos herdados ou sobrecarregados, também existem variáveis de controle próprias do personagem como **lifeBar** (barra de vida), **isShooting** e **lastShootingTime** (controle do estado de atirar), dentre outros, encapsulados de modo privado para evitar que elementos externos modifiquem indevidamente sua lógica de funcionamento. Também apresenta **setters** de animações que são visíveis somente pelas classes filhas, na momento em que forem modificar a animação original.

2.4.8 Física

Para implementar a física do mapa, foi utilizada uma matriz de 15 x 192. Essas dimensões foram definidas com base no tamanho do asset do mapa, que possui 240px de altura por 3072px de largura. Assim, cada posição na matriz de colisão corresponde a um ladrilho de 16px x 16px dentro do jogo, onde:

0 - ladrilho sem colisão;
1 - ladrilho com colisão;
2 - ladrilho de morte (Se o personagem colidir com este ladrilho, irá morrer).

2.4.9 Inimigos

Em relação aos inimigos, todos eles herdam da classe DumbGameEntity, detalhada acima. Cada inimigo possui características únicas que serão detalhadas individualmente abaixo.

- **Robbit:** Este inimigo verifica colisões com o solo e paredes, ajustando sua posição e invertendo a direção ao colidir. O Robbit possui dois estados, o de IDLE e o estado de JUMPING, esses estados servem para controlar o pulo no personagem e alterar suas animações. O Robbit profere dano ao encostar no jogador.

- **Battom:** Essa personagem possui três estados de iteração: NOACTIVE, ACTIVATING e ACTIVE. Inicialmente o Battom está inativo, para ativar é necessário que o Multi-Man esteja a uma distância de 100 unidades do inimigo. Quando o protagonista entre nessa área o Battom inicia a animação de abrir asas (ACTIVATING) e, ao concluir muda para o estado de (ACTIVE), nesse estado o Battom segue o jogar até que seja derrotado ou deixado para tras. O morcego ataca o jogador ao encostar nele.

- **Goomba:** Este inimigo apresenta um comportamento simples e contínuo, movendo-se automaticamente em uma direção horizontal. Ele possui funções de detecção de colisão com paredes; ao colidir, inverte sua direção de movimento. Além disso, o inimigo ataca o jogador ao entrar em contato, causando dano.

2.4.10 Power-Ups

Os power-ups temporários de fogo, eletricidade e água, presentes no jogo são representados pelas classes FireMegaMan, ElectricMegaMan e WaterMegaMan, respectivamente. Cada uma delas é uma classe filha da classe original MegaMan, em que as mudanças principais são a definição de suas próprias animações, com override dos métodos **draw()**, **update()** e **attack()**, para, respectivamente, desenhar novas animações, atualizar suas

animações e definir um projétil e dano diferente para o ataque. Na figura x, encontra-se o FireMegaMan, como exemplo.

2.4.11 Wood-Man

Por fim, WoodMan é a classe que representa o inimigo final. Dotada de movimentos mais complexos, sistemas de colisão e estados, semelhante ao MegaMan, esta classe também herda de SmartGameEntity, para aplicar tais comportamentos e funcionalidades.

Do mesmo modo que o MegaMan, ele realiza override dos métodos de **update()** e **draw()**, para atualizar estados e exibir suas próprias animações, além dos métodos de movimento run(), stopRun() e jump() e o método attack() em que configura seu projétil, o escudo de folhas.

Além disso, possui estados próprios (INTRO, BEATING_CHEST, IDLE, JUMPING, LEAF_SHIELD_THROW) para além dos básicos de uma GameEntity, para coordenar sua mecanica de movimentos. A classe se comporta de forma a definir o estado de INTRO como o primeiro e dessa forma deixa cada estado em execução por um determinado período de tempo e depois muda para o próximo, configurando suas animações e comportamentos no caminho. O método **setupAttack()**, por exemplo, exclusivo dessa classe, é o responsável por invocar as folhas sempre que o estado mudar para BEATING_CHEST, e após esse estado, ele muda para o LEAF_SHIELD_THROW e chama o método **attack()** juntamente com sua animação.

3. Ferramentas utilizadas

- Aseprite
- VScode
- GitHub
- IntelliJ
- JetBrains
- GIMP
- Paint.NET

4. Resultados

Após o desenvolvimento do projeto, foi possível criar um jogo que contém uma experiência customizada da fase Wood-Man stage, com seus inimigos mais icônicos, bem como novos poderes, além das telas inicial, game-over e vitória.

Ao acessar o jogo pela primeira vez, o usuário encontra um menu com a opção de começar a jogar ou encerrar o programa, como pode ser visto na figura 6. Nele, pode-se navegar pelas opções com as setas direcionais e clicar na tecla **ENTER** para realizar a ação.

Figura 6 - Tela inicial

Ao entrar no jogo, o usuário se encontra em um mundo em que pode se mover com as teclas de direção, **A,W,S,D**, pular com a tecla **SPACE**, disparar com **H**, ou escolher um power-up com **B** (bubble/water), **E** (electric), ou **F** (fire). Durante o percurso, apareceram vários inimigos, cada qual com sua mecânica.

Figura 7 - Início da GamePlay

Ao chegar ao fim da fase, ocorre a **Boss Battle** contra o Wood-Man. Esse chefe possui uma mecânica de movimentos própria, e padrões de ataque que precisam ser decifrados para que o jogador saia vitorioso. Power-ups também podem ser usados contra ele, com a mesma

restrição de tempo. Esse inimigo também possui uma barra de vida que precisa ser esvaziada para conceder a vitória. Um exemplo da batalha pode ser visto na figura 8.

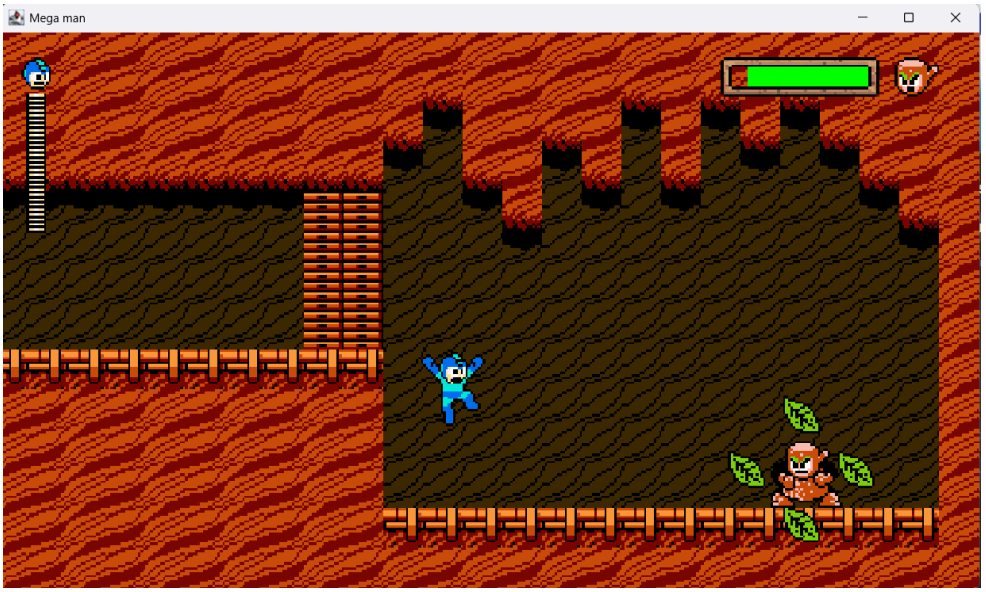

Figura 8 - Vs Wood-Man

Por fim, se durante a batalha contra o chefe ou durante a gameplay o jogador ficar sem vida ou mesmo cair do mapa, uma tela de **Game Over** com a opção de reiniciar o jogo ou sair dele é exibida, como pode ser conferido na figura 10.

Figura 10 - Tela de Game Over

5. Conclusão

O desenvolvimento do Multi-Man foi uma experiência desafiadora e enriquecedora, que permitiu aos desenvolvedores do projeto aplicar conceitos de Programação Orientada a Objetos, como herança, encapsulamento e polimorfismo. A implementação da mecânica de combate e estrutura da fase escolhida demonstrou a necessidade de uma arquitetura organizada dividida em módulos.

Além disso, personalizações como alteração na paleta de cores e implementação do Goomba pertencente ao Super Mario, trouxeram um toque de originalidade ao projeto. Essa experiencia consolidou não apenas os conhecimentos teóricos aprendidos em sala de aula, mas também, reforçou a importância de práticas colaborativas no desenvolvimento de softwares interativos.

6. Referências

[1] ASEPRITE. **Aseprite**. Disponível em: http://www.aseprite.org/. Acesso em: 19 jan. 2025.

[2] GETPAINT.NET. **Paint.NET**. Disponível em: https://www.getpaint.net/. Acesso em: 19 jan. 2025.

[3] GIMP. **GIMP - GNU Image Manipulation Program**. Disponível em: https://www.gimp.org/. Acesso em: 19 jan. 2025.

[4] PHAM, Ngoc Thach. **Rockman**. Disponível em: https://github.com/phamngocthachlt6c/rockman. Acesso em: 19 jan. 2025.

[5] **SPRITES INC**. Sprites INC - /Classic/. Disponível em: https://www.sprites-inc.co.uk/sprite.php?local=/Classic/. Acesso em: 19 jan. 2025.

[6] THE SOUNDS RESOURCE. **Mega Man 2 Sounds**. Disponível em: https://www.sounds-resource.com/nes/megaman2. Acesso em: 19 jan. 2025.