UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas – Departamento de Ciência da Computação

“Echoes of Elementum”: Desenvolvimento de um Jogo de Plataforma do Gênero Metroidvania

Pesquisa Tecnológica

Roger Dornas Oliveira
Orientador: Lucas Nascimento Ferreira

Belo Horizonte, Minas Gerais
2026

Resumo. *Este trabalho documenta a conclusão e a validação prática de Echoes of Elementum, um jogo 2D do gênero Metroidvania desenvolvido sobre uma engine proprietária estruturada em C++, SDL2 e a API gráfica OpenGL. O objetivo principal é a exploração aprofundada de arquiteturas de baixo nível da Ciência da Computação, abdicando do uso de motores comerciais. Esta etapa final foca na expansão sistêmica e no polimento tecnológico do motor, detalhando o gerenciamento avançado de colisões modulares, a implementação de sistemas de progressão (árvore de habilidades e mapa dinâmico) e o aprimoramento da experiência visual (game feel) por meio de partículas e shaders de pós-processamento. A infraestrutura foi validada estruturando-se uma demonstração do jogo, submetida a testes empíricos de usabilidade (playtests). Os dados coletados fundamentaram o refinamento visual e o balanceamento das Máquinas de Estados Finitos da inteligência artificial. Os resultados consolidam uma base tecnológica independente que se provou robusta, escalável e responsiva.*

1. Introdução

A indústria de jogos digitais consolidou-se nas últimas décadas como um dos segmentos mais influentes e lucrativos do entretenimento mundial, superando, em muitos casos, as indústrias da música e do cinema em termos de receita. Dentro desse ecossistema, observa-se uma clara distinção entre as grandes produções, conhecidas como jogos AAA, que dispõem de orçamentos multimilionários e equipes vastas, e o cenário independente (indie). O desenvolvimento indie, impulsionado pela democratização de ferramentas e distribuição digital, tem sido um terreno fértil para a inovação de mecânicas e narrativas.

Embora o mercado atual ofereça motores de jogo (game engines) comerciais robustos, como Unity e Unreal, que abstraem grande parte da complexidade técnica, o desenvolvimento de uma arquitetura própria, partindo do zero, apresenta-se como um desafio acadêmico e técnico de valor inestimável. A decisão de não utilizar engines prontas justifica-se pela necessidade de compreender profundamente o funcionamento de baixo nível de um software de tempo real. Este projeto situa-se na interseção de diversas áreas fundamentais da Ciência da Computação, exigindo a aplicação prática de conceitos de Engenharia de Software para a arquitetura do sistema, Computação Gráfica para a renderização e manipulação de imagens, e Interação Humano-Computador (IHC) para garantir a responsividade e a experiência do usuário.

Neste contexto, o presente trabalho documenta a conclusão e validação do projeto intitulado “Echoes of Elementum”: Desenvolvimento de um Jogo de Plataforma do Gênero Metroidvania. O jogo é construído inteiramente na linguagem C++, utilizando a biblioteca SDL2 para o gerenciamento de janelas e entradas periféricas, e a API gráfica OpenGL, gerenciada pela biblioteca GLEW, para o processamento de vídeo. O desenvolvimento deste projeto teve início como um protótipo na disciplina DCC192 - Desenvolvimento de Jogos Digitais, na qual foi premiado como o melhor jogo da turma, servindo de fundação teórica e técnica para a sua expansão para o POC (Projeto Orientado em Computação). Enquanto a primeira etapa concentrou-se na estruturação do ciclo principal (Game Loop), no gerenciamento de recursos e na migração inicial para um pipeline gráfico programável em OpenGL, esta fase final direcionou seus esforços para o polimento fino da experiência, sofisticação dos sistemas de colisão e combate, expansão do repertório de habilidades e a validação empírica do software por meio de testes com usuários reais.

Devido à elevada demanda de tempo inerente à atividade de level design em mapas interconectados e à necessidade de sucessivas iterações para o balanceamento das curvas de

dificuldade, o escopo de entrega prática deste trabalho delimitou-se à produção de uma demonstração. Esta demonstração técnica, que oferece entre 15 e 30 minutos de tempo de jogo, compreende a área inicial completa do jogo (floresta), o sistema de tutorial dinâmico e a transição para o próximo ambiente (caverna). Embora não tenha sido possível terminar o level design do jogo inteiro, todas as mecânicas fundamentais foram integralmente implementadas, testadas e refinadas, assegurando que a infraestrutura desenvolvida cumpra com os requisitos de estabilidade, escalabilidade e desempenho necessários para um produto comercial de tempo real.

O objetivo geral deste trabalho é projetar, implementar e validar uma demonstração funcional, estável e esteticamente polida do jogo Echoes of Elementum, consolidando sua infraestrutura tecnológica proprietária de baixo nível e demonstrando a viabilidade técnica e a engenharia de software aplicadas ao desenvolvimento independente de jogos digitais.

O restante deste documento está organizado de forma a apresentar, na Seção 2 (Referencial Teórico), o embasamento acadêmico atualizado sobre o gênero Metroidvania, engenharia de software aplicada a jogos de tempo real e os conceitos fundamentais de game feel e IHC utilizados para o polimento da experiência. A Seção 3 (Metodologia e Implementação) descreve a arquitetura interna desenvolvida na engine, detalhando a lógica por trás dos subsistemas de colisão, renderização de efeitos de tela, árvore de habilidades e barramentos de interface periférica.

A Seção 4 (Level Design e Estruturação da Demonstração) expõe o processo prático de criação e interconexão de salas utilizando o editor Tiled e a aplicação empírica das mecânicas. A Seção 5 (Testes, Validação e Refinamento) analisa quantitativa e qualitativamente os dados obtidos através do processo de playtest, justificando as alterações de código efetuadas para o refinamento da experiência. A Seção 6 (Resultados Alcançados) apresenta o resultado final do jogo, usando imagens para exemplificar. Por fim, a Seção 7 (Conclusão e Trabalhos Futuros) sintetiza as principais contribuições técnicas deste projeto, as limitações encontradas e as perspectivas de trabalhos futuros para a expansão do universo de Echoes of Elementum.

2. Referencial Teórico

2.1. Conceitos Fundamentais de Jogos

Para o desenvolvimento de qualquer aplicação lúdica, é essencial compreender o que define um jogo. Academicamente, um jogo pode ser definido como um sistema fechado e formal, onde jogadores engajam-se em um conflito artificial, definidos por regras, que resulta em um resultado quantificável, como vitória ou derrota. Dentro deste sistema, as regras impõem limitações e determinam o que é permitido, criando a estrutura necessária para o desafio. Os objetivos, por sua vez, fornecem a motivação e a direção para as ações do jogador.

Um conceito central no design de jogos é a distinção entre mecânica e jogabilidade. As mecânicas são os métodos invocados pelos agentes para interagir com o mundo do jogo, incluindo verbos de ação como pular, atacar ou interagir. Já a jogabilidade é a consequência da interação dessas mecânicas com as regras e o ambiente, gerando a experiência dinâmica do jogar.

Além disso, aspectos de Interação Humano-Computador são cruciais, especialmente o feedback visual e sonoro. Um jogo deve responder aos comandos do jogador de forma imediata e perceptível, garantindo que o sistema comunique claramente o estado atual do jogo, seja através

de uma barra de vida que diminui ou de uma animação de impacto, elementos fundamentais para a imersão.

2.2. A Indústria de Jogos

A indústria de desenvolvimento de jogos digitais é classicamente dividida em dois grandes espectros: os estúdios AAA (Triple-A) e os desenvolvedores independentes (Indie). Os jogos AAA são caracterizados por orçamentos massivos, equipes com centenas de especialistas e grandes campanhas de marketing, focando muitas vezes em gráficos de ponta e escopo vasto. Em contraste, o cenário Indie é composto por equipes pequenas, ou até mesmo desenvolvedores solitários, com recursos financeiros limitados. Essa restrição, no entanto, fomenta a criatividade e a inovação, permitindo que jogos independentes explorem nichos de mercado, narrativas experimentais e estilos artísticos únicos que seriam arriscados para grandes corporações.

O desenvolvimento técnico desses jogos é sustentado pelos Motores de Jogo (Game Engines). Uma engine é um ambiente de desenvolvimento de software que abstrai tarefas complexas como renderização gráfica, física e detecção de colisão. Embora o mercado ofereça soluções comerciais poderosas, a criação de motores proprietários continua sendo uma prática relevante, tanto para otimização específica quanto para fins educacionais, permitindo o controle total sobre a arquitetura do software.

2.3. Gênero Metroidvania

O projeto Echoes of Elementum insere-se no gênero conhecido como Metroidvania, um termo derivado das séries Metroid e Castlevania. Este subgênero de jogos de plataforma e ação-aventura distingue-se por apresentar um mapa mundo grande e interconectado, em oposição a fases lineares e discretas. A progressão no jogo não é ditada apenas pela narrativa, mas pela aquisição de novas habilidades ou itens que permitem ao jogador acessar áreas anteriormente inalcançáveis.

Esta estrutura incentiva a exploração e o backtracking (o ato de retornar a áreas visitadas), recompensando a curiosidade do jogador e o domínio sobre a movimentação do personagem. Títulos modernos como Hollow Knight e Ori and the Blind Forest revitalizaram o gênero, elevando os padrões de qualidade visual, fluidez de combate e narrativa ambiental. Estes jogos servem como referências diretas para este trabalho, tanto na estética quanto na complexidade das mecânicas de controle e design de níveis.

2.4. Arquitetura de Software em Jogos

Do ponto de vista da Engenharia de Software, um jogo opera de maneira distinta de aplicações convencionais orientadas a eventos. O núcleo de um jogo é o Game Loop (Ciclo de Jogo), um laço infinito que executa continuamente três tarefas principais: processar a entrada do usuário (Input), atualizar o estado do mundo e dos objetos (Update) e desenhar o quadro atual na tela (Draw). A eficiência deste ciclo determina a taxa de quadros por segundo (FPS) e a fluidez da experiência.

Para gerenciar a complexidade dos objetos em cena (personagens, inimigos, itens), utiliza-se frequentemente uma abordagem baseada em composição, em detrimento de hierarquias de herança profundas. O modelo de Game Objects e Componentes permite que entidades sejam construídas agregando funcionalidades modulares (um componente de renderização, um componente de física, etc.), aumentando a flexibilidade e a reutilização de código.

Para o comportamento de Inteligência Artificial e controle de animações, aplica-se o conceito de Máquinas de Estados Finitos (FSM). Uma FSM permite que uma entidade esteja em

apenas um estado por vez (como "Parado", "Perseguindo" ou "Atacando") e define transições claras entre esses estados baseadas em eventos ou condições. Isso organiza a lógica de comportamento de inimigos complexos e garante que o personagem responda coerentemente aos comandos.

2.5. Tecnologias e Ferramentas Usadas

A implementação técnica deste projeto baseia-se na linguagem C++, escolhida por seu desempenho superior e controle granular sobre o gerenciamento de memória, características essenciais para sistemas de tempo real. Para a interface com o sistema operacional, utiliza-se a biblioteca SDL2 (Simple DirectMedia Layer), responsável pela criação de janelas, captura de entrada (teclado/controle/mouse) e manipulação de áudio.

A renderização gráfica é realizada através da API OpenGL gerenciada pela biblioteca GLEW. Diferente de bibliotecas de alto nível que desenham formas prontas, o uso de OpenGL moderno exige a configuração de uma pipeline gráfica programável, utilizando Shaders (pequenos programas que rodam na GPU) para processar vértices e fragmentos, permitindo efeitos visuais avançados e iluminação dinâmica.

Por fim, para auxiliar no Level Design, utiliza-se a ferramenta Tiled Map Editor, que permite a criação visual dos mapas através de tilesets. A integração entre o editor e a engine do jogo é feita através do formato JSON, que estrutura os dados do mapa e posicionamento de objetos, permitindo que o jogo carregue e construa os níveis dinamicamente durante a execução.

2.6. Game Feel

À medida que a infraestrutura fundamental de um jogo amadurece, a Engenharia de Software deve se alinhar a conceitos de Game Feel. Este termo abrange a sensação tátil e a micro-resposta estética de um jogo em relação às entradas do jogador. O polimento visual não tem a finalidade apenas decorativa, mas atua diretamente na comunicação de eventos mecânicos invisíveis na memória computacional, traduzindo-os para linguagem visual.

A manipulação da câmera (como tremores em impactos), as partículas dinâmicas (como grama e poeira reativas à física do personagem) e os efeitos visuais em tela são essenciais para construir peso, velocidade e impacto. No desenvolvimento de engines modernas através de APIs como OpenGL, isso é largamente alcançado utilizando técnicas de Pós-Processamento via Fragment Shaders. Efeitos como Chromatic Aberration (separação dos canais RGB), borrões direcionais (Blur) e matização de tela transmitem instantaneamente ao jogador estados críticos como vulnerabilidade, paralisia temporal ou dano sistêmico, sem a necessidade de interferir na UI tradicional.

3. Metodologia e Implementação

O desenvolvimento de Echoes of Elementum fundamentou-se em uma abordagem de engenharia de software modular e iterativa. A partir da base construída na primeira etapa da pesquisa, utilizando C++, SDL2 e OpenGL, esta fase final dedicou-se à expansão estrutural da arquitetura, à consolidação das mecânicas do gênero Metroidvania e ao polimento visual e tátil. Esta seção detalha as lógicas e os subsistemas desenvolvidos para comportar todas as funcionalidades do jogo.

3.1. Arquitetura de Atores e Componentes

Para evitar a rigidez de hierarquias de herança profundas, a arquitetura do jogo baseia-se no padrão de composição de Entidades e Componentes. O sistema de atores funciona através de contêineres genéricos que adquirem comportamentos específicos através da agregação de componentes. Dessa forma, um objeto torna-se "renderizável" ao receber um componente de desenho, ou "físico" ao receber um componente de física.

Um componente interessante implementado nessa etapa, foi o CombatBoxComponent, que consiste em um contêiner capaz de gerenciar simultaneamente múltiplos colisores dinâmicos, que são subdivididos em duas categorias lógicas:

- **Hurtboxes (Caixas de Vulnerabilidade):** Retângulos que registram o recebimento de dano. Podem ser aplicados ao corpo inteiro do personagem ou subdivididos para melhor adaptar ao formato do personagem.
- **Hitboxes (Caixas de Dano):** Retângulos responsáveis por aplicar dano às hurtboxes sobrepostas de entidades adversárias.

Essa modularidade permitiu que um único ator possuísse colisores distintos para diferentes partes de seu corpo, viabilizando uma detecção de impactos extremamente precisa.

3.2. Mecânicas de Jogabilidade e Modos Elementais

A jogabilidade central foi projetada para oferecer alta mobilidade e liberdade tática. As mecânicas base de navegação do protagonista incluem movimentação bidimensional fluida, pulo duplo (double jump), avanço rápido (dash), deslizamento em paredes (wall slide), salto na parede (wall jump) e a mecânica de balanço físico utilizando cordas ou ganchos.

O diferencial mecânico do projeto reside no sistema de "Modos Elementais", onde o jogador alterna dinamicamente entre quatro elementos, gerenciados através de um menu radial. Para garantir o planejamento tático e o game feel, a engine foi programada para reduzir drasticamente a escala de tempo (câmera lenta) durante a abertura deste seletor. Os quatro modos implementados e suas respectivas habilidades são:

- **Fogo:** Focado em ataque à distância e utilidade. Permite o disparo de Bolas de Fogo e a invocação do Fire Wisp, um orbe de fogo inteligente que segue o jogador, ilumina passagens escuras e ataca inimigos de forma autônoma e periódica.
- **Gelo:** Focado em controle de grupo e mobilidade aérea. Habilita o disparo de um Jato de Gelo contínuo e a habilidade de Planar (Glide), retardando a gravidade para superar grandes abismos.
- **Raio:** Focado em velocidade extrema e penetração de defesas. Libera o Modo Elétrico (aumentando os atributos de velocidade base do protagonista), o uso do Dash Elétrico (única habilidade capaz de transpor as barreiras elétricas do cenário) e o disparo da Lança de Raio, um projétil que distribui dano em cadeia aos inimigos próximos.
- **Terra:** Focado em poder destrutivo e defesa posicional. Permite o uso do Impacto Sísmico (Ground Slam), essencial para destruir pisos frágeis, e a criação do Pilar de Terra, servindo simultaneamente como bloqueio e como plataforma de escalada.

3.3. Integração de Cenários, Persistência e Sistema de Mapa

Para materializar a progressão não-linear característica do Metroidvania, a construção do mundo foi realizada no Tiled Map Editor, amparada por um analisador léxico (parser) proprietário em C++ capaz de carregar arquivos JSON do mapa em tempo real. A engine suporta carregamento dinâmico de salas, descarregando e instanciando novas áreas de forma transparente quando o jogador cruza os limites da tela.

Os cenários são compostos por instâncias que vão desde plataformas móveis e plataformas de expansão temporal (que crescem e diminuem), até armadilhas como espinhos e poços de lava. O level design também incorpora um sistema de Arenas, onde portas são trancadas até que ondas sucessivas de inimigos sejam derrotadas.

Para auxiliar a navegação nesse complexo ecossistema, implementou-se um Sistema de Mapa Dinâmico, que é revelado progressivamente à medida que o jogador explora novas coordenadas, podendo ser acessado a qualquer momento em uma interface dedicada. Todo o progresso do mapa, juntamente com o desbloqueio de portas, bosses derrotados e atributos do jogador, são armazenados através do Sistema de Salvamento, em que esses dados são salvos em um arquivo json para cada um dos 4 slots de saves disponíveis, para garantir a continuidade temporal entre as sessões.

3.4. Inimigos

O jogo conta com 22 inimigos, sendo 9 deles Bosses. O comportamento dos inimigos é gerenciado por Máquinas de Estados Finitos, com suas transições baseadas em eventos, como distância do jogador, tempo, vida.

Para ilustrar a escalabilidade deste sistema, o trabalho apresenta dois casos distintos. O primeiro é um inimigo simples, a Cobra, que possui um ciclo básico de patrulha e ataque ao detectar o jogador. Além disso, quando um inimigo simples toma um dano, ele muda seu estado para o estado de “Hurt”, interrompendo um possível ataque.

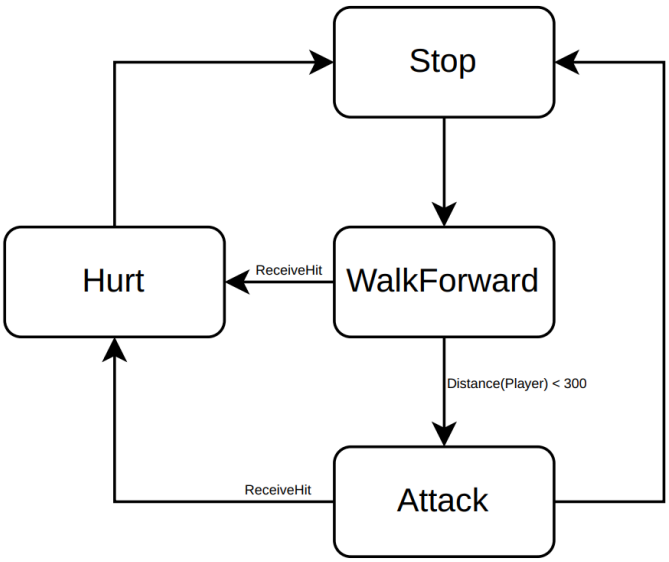

Figura 1 – FSM da Cobra

O segundo caso é de um Boss, a Raposa Guerreira, um inimigo complexo. Sua FSM gerencia múltiplas fases de combate, alternando entre padrões de ataque corpo a corpo, movimentação rápida, ataques a distância e estados de vulnerabilidade.

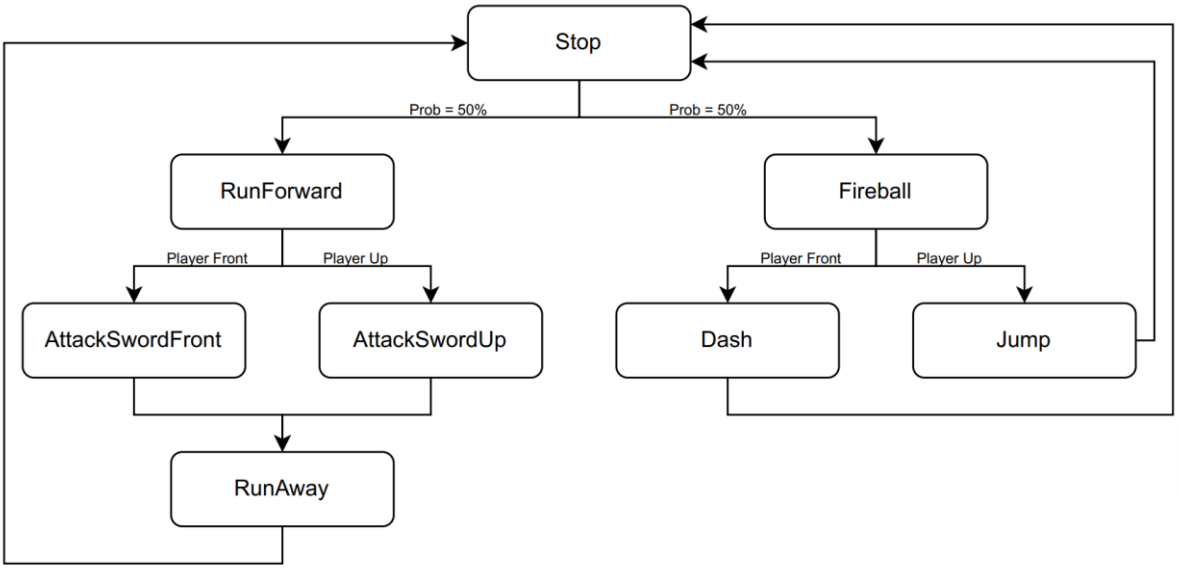

Figura 2 – FSM da Raposa Guerreira

3.5. Interface (UI), Acessibilidade e Progressão

Um dos focos de refinamento foi a reestruturação da Interface de Usuário e dos menus. Para a progressão das habilidades e atributos do jogador, foi desenvolvida uma Árvore de Habilidades (Skill Tree), que permite ao jogador investir os recursos (moedas) coletados no cenário para aprimorar atributos base ou desbloquear de forma permanente as habilidades elementais citadas anteriormente.

Visando as boas práticas de acessibilidade e UX, foi criado um painel de Configurações robusto, abrigando:

- **Vídeo:** Controle de resoluções de tela e alternância entre Modo Janela e Tela Cheia.
- **Áudio:** Barramentos de áudio independentes para mixagem do volume Geral, Música e Efeitos Sonoros (SFX).
- **Controles:** Um sistema dinâmico de remapeamento de teclas que abrange tanto mapeamentos de Teclado e Mouse quanto de Gamepads (controles).

Como ferramenta de integração do jogador, implementou-se um sistema de Tutoriais via Triggers (gatilhos invisíveis no mapa). Para garantir clareza visual, foi criada uma fonte TTF para esses ícones de teclas de teclado, mouse, e botões de controles Playstation e Xbox. Essa arquitetura escuta ativamente o último periférico utilizado (Teclado, Mouse, Controle do PlayStation ou

Xbox) e altera instantaneamente o ícone do botão correspondente renderizado nas mensagens de tutorial, guiando o jogador sem gerar ambiguidade.

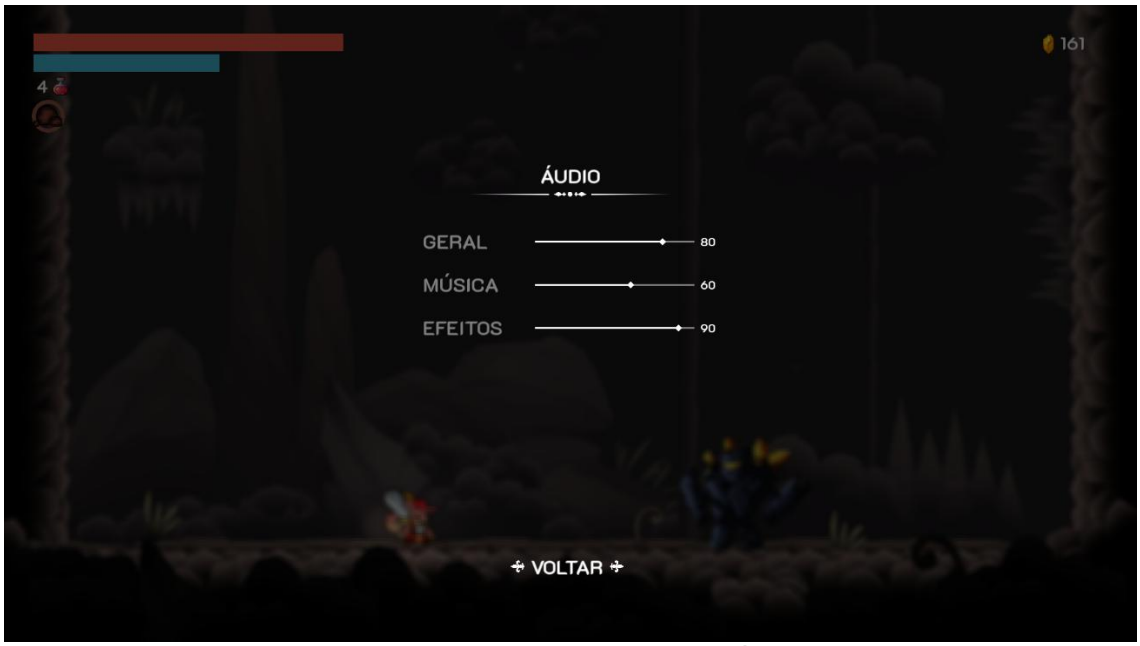

Figura 3 – Menu de ajuste de Áudio

3.6. Polimento Visual: Renderização, Shaders e Partículas

Para atingir o grau de polimento esperado de produções independentes modernas, o pipeline de renderização em OpenGL foi exaustivamente explorado. Os pacotes gráficos de floresta e caverna (obtidos sob licença gratuita do artista Maaot) foram enriquecidos com múltiplas camadas de paralaxe e elementos vivos.

A vitalidade visual do jogo está profundamente atrelada ao Sistema de Partículas customizado. O motor é capaz de instanciar centenas de partículas simultâneas com impacto ínfimo na CPU, sendo utilizado extensivamente como feedback visual em situações como: esguichos de sangue baseados em direcionalidade ao atingir inimigos, explosões, poeira e detritos de grama ao caminhar, pular ou usar o dash, fumaça, o rastro cinético da habilidade do Jato de Gelo, as chamas dinâmicas que orbitam o Fire Wisp, efeitos ao usar a cura, entre outras. Adicionou-se também emissores de partículas atmosféricas para criar efeitos visuais de esporos suspensos no ar, unificando a iluminação das cavernas e florestas.

Por fim, o motor foi provido de renderização de Pós-processamento via Fragment Shaders programados em GLSL. O frame gerado não é enviado diretamente à tela, mas passa por um Framebuffer Object (FBO) onde efeitos matemáticos são aplicados à imagem final. Foram implementados efeitos de:

- **Blur (Desfoque Direcional):** Aplicado no fundo de tela durante a pausa do jogo, focando a atenção na interface;
- **Tela Vermelha e Vinheta:** Aplicada como micro-resposta de tensão e alerta sempre que o jogador sofre dano físico;
- **Chromatic Aberration (Aberração Cromática):** Efeito de distorção e separação dos canais RGB (Red, Green, Blue), ativado temporariamente como penalidade visual e confusão caso o protagonista tenha seus controles invertidos por ataques inimigos específicos.

4. Level Design e Estruturação da Demonstração

Devido à inerente complexidade e à alta demanda de tempo para o balanceamento de um mapa interconectado em sua totalidade, o escopo prático deste trabalho consolidou-se em uma demonstração de início de gameplay. Essa demonstração oferece entre 15 e 30 minutos de jogabilidade contínua, estruturada para validar a progressão do jogador, a curva de aprendizado e o funcionamento da engine.

4.1. Construção do Mundo com o Tiled Map Editor

Para a montagem arquitetônica e estética dos níveis, utilizou-se o Tiled Map Editor. A estruturação topológica de cada sala foi dividida metodicamente em múltiplas camadas, destinadas ao player, checkpoint, inimigos, alavancas, plataformas, triggers, câmera, e várias outras.

O polimento visual foi garantido através do uso extensivo de camadas de Backgrounds (planos de fundo) e Foregrounds (primeiros planos). A aplicação de diferentes camadas de decoração, utilizando os tilesets detalhados de floresta e caverna, proporcionou uma forte sensação de profundidade e paralaxe, resultando em um mundo mais imersivo e vivo, complementado pelo sistema de partículas ambientais desenvolvido na engine.

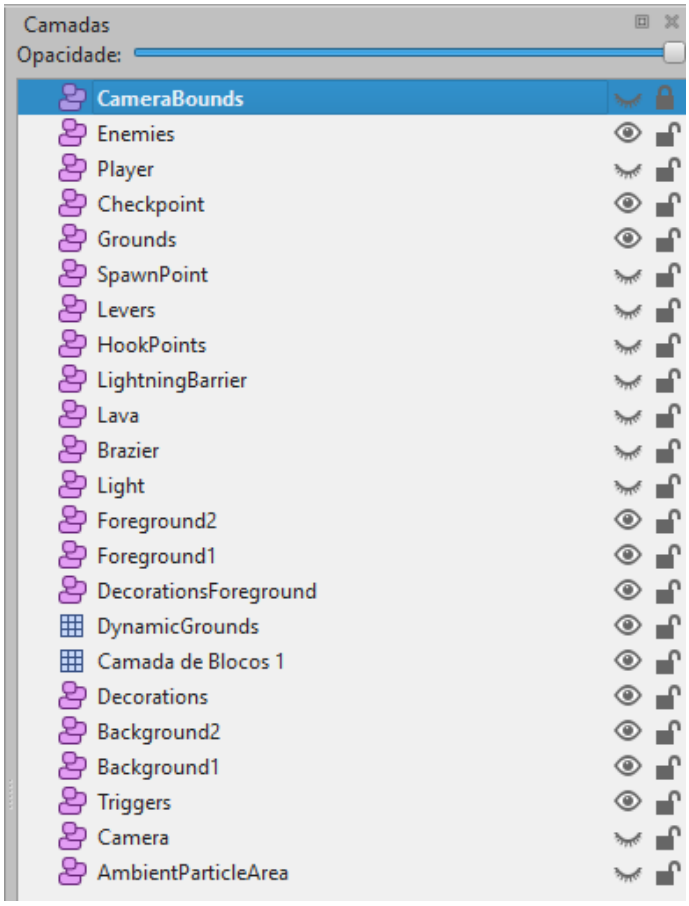

Figura 4 – Camadas usadas do Tiled

Figura 5 – Exemplo de uma sala feita no Tiled

4.2. Estrutura da Campanha Inicial e Curva de Aprendizado

A demonstração desenvolvida é composta por um total de 14 salas interconectadas, sendo 13 localizadas no bioma inicial (Floresta) e uma sala servindo como transição e introdução ao segundo bioma (Caverna).

O design das salas iniciais foi concebido sob a filosofia de tutoriais orgânicos. Em vez de interromper a ação com longos textos explicativos, o jogador é inserido em cenários controlados onde os tutoriais ensinam comandos essenciais de forma contextualizada, como movimentação básica, ataque com a espada, sistemas de cura, interação com checkpoints e o uso da interface de mapa.

À medida que o jogador avança, o level design abandona a linearidade estrita e passa a apresentar bifurcações de caminhos. Esta escolha arquitetônica é fundamental para o gênero Metroidvania, pois instiga a exploração autônoma e a tomada de decisão pelo jogador.

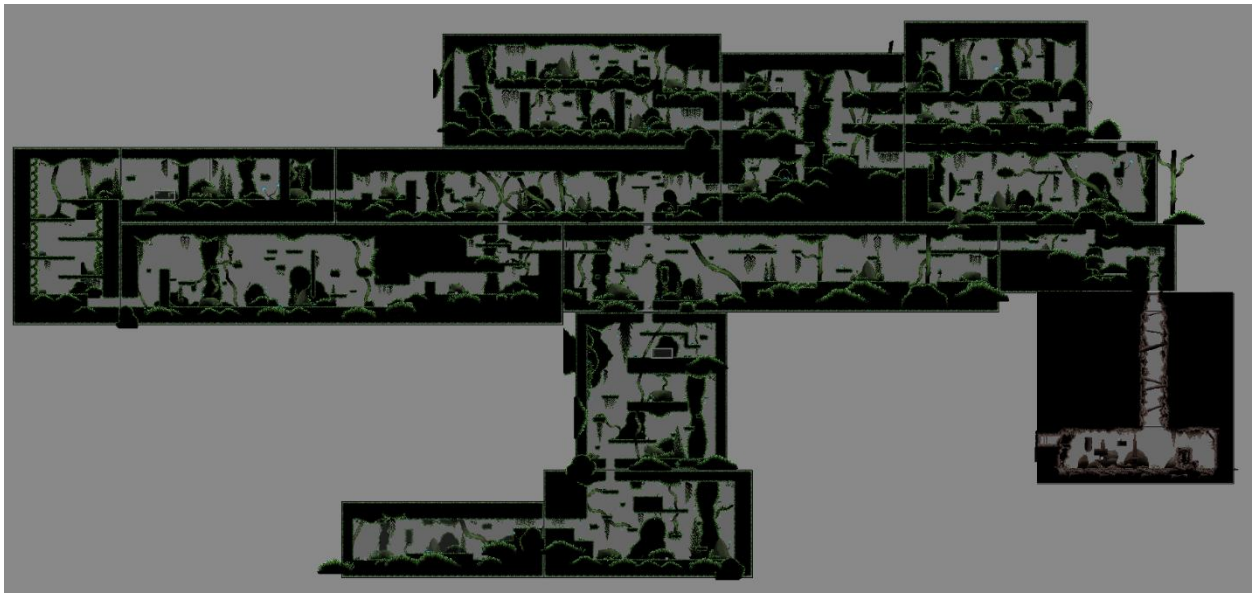

Figura 6 – Mapa completo da demonstração feito no Tiled

4.3. Combate e Progressão Metroidvania

Para manter uma curva de dificuldade gradativa, a área da floresta foi populada com apenas 6 dos 13 tipos de inimigos simples programados no jogo. Estes adversários iniciais possuem Máquinas de Estados Finitos (FSM) de complexidade moderada, permitindo que o jogador se habitue ao timing dos ataques e ao peso do combate sem ser punido severamente.

A fim de testar as habilidades de combate e o controle de espaço do jogador, foram implementadas salas de Arenas. Nesses ambientes, as saídas são bloqueadas automaticamente, exigindo que o jogador sobreviva a múltiplas ondas de inimigos para destravar o caminho.

O clímax estrutural desse início de gameplay ocorre no embate contra o primeiro chefe, o Sapo Guardião. O design deste confronto exige a aplicação de tudo o que foi aprendido nas salas anteriores. A vitória sobre este chefe é o marco de progressão: o jogador é recompensado com a liberação da habilidade de avanço rápido (Dash). Seguindo a lógica de bloqueio e chave (lock-and-key) do gênero Metroidvania, o Dash recém-adquirido é o recurso mecânico necessário para transpor um obstáculo específico na floresta, permitindo que o jogador acesse a 14ª sala, que introduz o bioma da Caverna e encerra a demonstração principal.

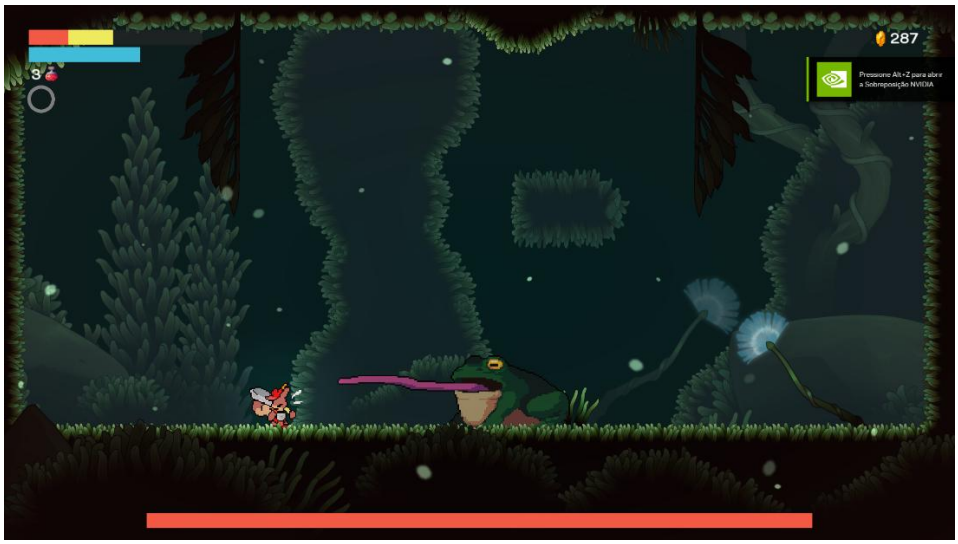

Figura 7 – Luta contra o primeiro boss Sapo Guardião

4.4. O Modo de Jogo Avançado

Embora a campanha principal da demo seja eficaz na validação das mecânicas básicas e do level design, seu ritmo cadenciado impossibilita a demonstração prática de todo o potencial tecnológico desenvolvido para a engine durante a pesquisa. Mecânicas avançadas de navegação, como o Pulo Duplo, Deslizamento e Salto em Paredes, e balanço em Cordas, além da complexidade dos 4 Modos Elementais (Fogo, Terra, Gelo e Raio) e a interação com os outros 9 bosses projetados, não são alcançados no trecho inicial do jogo.

Para mitigar essa limitação, especialmente para as sessões de playtest e avaliação acadêmica da arquitetura de software, foi desenvolvido um Modo Avançado. Este modo funciona como um ambiente de testes isolado (sandbox), no qual o jogador é instanciado com todas as habilidades, ferramentas elementais e melhorias da árvore de habilidades já desbloqueadas.

A partir de um menu dedicado, o jogador pode selecionar cenários específicos para explorar mecânicas isoladas:

- **Tutorial de Habilidades:** Um nível estruturado exclusivamente para testar as movimentações complexas e os poderes elementais (como planar, usar pilares de terra, lanças de raio e fire wisp).
- **Desafios de Chefes (Boss Rush):** Salas configuradas para confrontos diretos contra os chefes avançados do jogo, validando suas complexas inteligências artificiais.
- **Arena dos Esquecidos:** Um cenário hostil com ondas densas e simultâneas de inimigos variados, operando como um teste de estresse tático para o jogador.

5. Testes, Validação e Refinamento

A construção de uma infraestrutura de software robusta não garante, por si só, uma experiência lúdica satisfatória. Em aplicações interativas de tempo real, a percepção e o tempo de resposta do usuário final são os maiores medidores de qualidade do sistema. Dessa forma, a etapa final do desenvolvimento consistiu na realização de playtests (testes de usabilidade e jogabilidade) para validar as mecânicas construídas, o game feel e o balanceamento técnico do projeto.

5.1. Metodologia de Avaliação e Coleta de Dados

Para viabilizar a fase de testes, foi compilada uma versão executável (build) contendo a demonstração da campanha principal (bioma de floresta) e o Modo de Jogo Avançado. Esta versão foi distribuída para um grupo de testadores externos. A coleta de dados foi estruturada através de um questionário elaborado na plataforma Google Forms, contendo perguntas fechadas (para obtenção de métricas quantitativas sobre dificuldade e fluidez) e perguntas abertas (para coleta de feedbacks qualitativos sobre a clareza das mecânicas).

Ao todo, foram obtidos 9 feedbacks detalhados. A análise das respostas permitiu identificar atritos no âmbito de IHC e picos arbitrários na curva de dificuldade, norteando as últimas refatorações de código do motor do jogo, detalhadas nas subseções a seguir.

Figura 8 – Tempo jogado

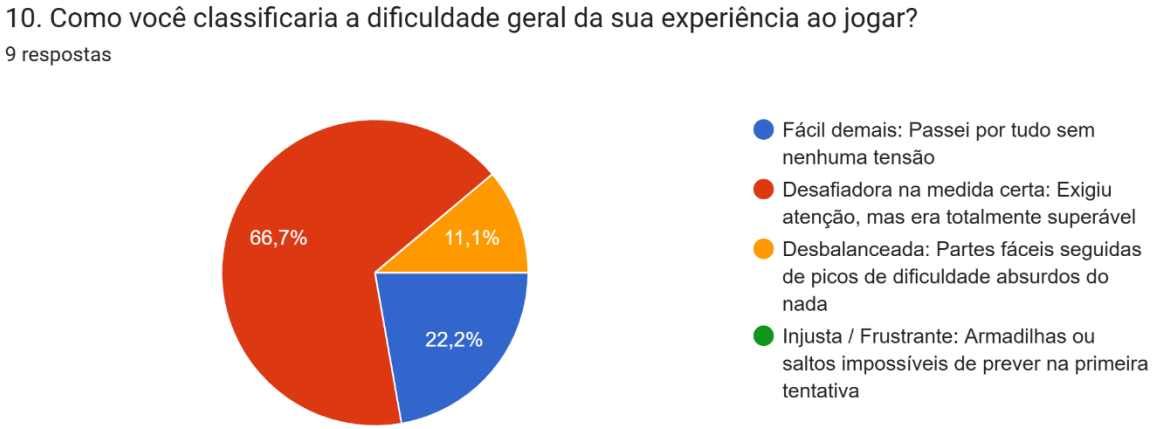

Figura 9 – Percepção de dificuldade

5.2. Melhorando o Sistema de Cura

Na análise das respostas qualitativas do formulário, o problema mais recorrente relatado pelos usuários referia-se à falta de clareza do sistema de cura. Os testadores relataram não saber o momento exato em que a ação de cura terminou, muitas vezes soltando o botão de cura antes de realmente efetuar a cura, gerando vulnerabilidades indesejadas e frustração durante os combates.

Este problema evidenciou uma falha na comunicação visual do estado interno do jogo. Para solucioná-lo, o sistema de partículas foi integrado à mecânica de cura. Foram adicionados efeitos visuais contínuos ao redor do protagonista durante o período em que o botão é mantido pressionado. Assim que o processamento interno conclui o incremento na variável de saúde, um efeito visual de conclusão é disparado. Essa alteração reduziu a ambiguidade cognitiva, permitindo que o jogador tenha certeza visual e imediata de que a ação foi registrada e concluída com sucesso.

5.3. Refatoração de Inteligência Artificial e Balanceamento de Inimigos

Outro ponto crítico levantado pelos dados quantitativos e qualitativos foi o desbalanceamento no combate corporal, especificamente contra os inimigos simples da área inicial. Devido à rigidez da arquitetura de inteligência artificial construída, os ataques dos inimigos não podiam ser interrompidos; se o jogador iniciasse um ataque simultaneamente ao inimigo, o protagonista inevitavelmente sofreria dano, resultando em um ciclo de combate punitivo que desencorajava a agressividade.

Para resolver essa falha de balanceamento sistêmico, foi necessária uma intervenção direta na arquitetura lógica dos adversários. As FSMs dos inimigos simples foram refatoradas para incluir um novo estado prioritário: o estado de Hurt (Dano/Interrupção).

Com essa modificação, sempre que o CombatBoxComponent de um inimigo simples registra uma colisão válida advinda da hitbox do ataque do jogador, a FSM força uma transição imediata para o estado de Hurt. Esta transição cancela qualquer ação agressiva (como patrulha ou ataque) que o inimigo estivesse executando, forçando uma animação de impacto antes de retornar a um estado ofensivo ou neutro. Essa alteração deixou a jogabilidade significativamente mais justa e dinâmica, recompensando os reflexos do jogador e validando a escalabilidade modular do sistema de Máquina de Estados construído.

6. Resultados Alcançados

A conclusão da segunda etapa do projeto Echoes of Elementum cumpriu com êxito o objetivo de consolidar uma infraestrutura tecnológica proprietária e entregá-la sob a forma de uma demonstração funcional e polida. Todos os sistemas propostos, desde a arquitetura de combate e renderização avançada até as ferramentas de interface e level design, foram validados e operam em conjunto mantendo uma taxa de quadros estável e alta responsividade.

Nesta seção, apresentam-se os resultados visuais e funcionais dos principais subsistemas implementados.

6.1. Polimento Visual e Pós-Processamento

O salto qualitativo na renderização foi um dos resultados mais evidentes desta etapa. A adoção dos novos conjuntos de texturas (tilesets) de floresta e caverna, combinada ao Sistema de Partículas customizado, resultou em cenários vibrantes. O motor gráfico agora gerencia com sucesso centenas de partículas simultâneas, aplicadas em poeira de movimento, grama, efeitos atmosféricos e feedback direcional de dano (sangue).

Além disso, a implementação de Shaders para os efeitos de tela validou o domínio sobre a API OpenGL. Os efeitos de pós-processamento atuam diretamente no game feel, comunicando estados críticos ao jogador.

Figura 10 – Cenário do bioma Floresta detalhado e enriquecido com partículas de ambiente

Figura 11 – Efeito de Pós-Processamento (Chromatic Aberration) para indicar controles invertidos

6.2. Subsistemas Auxiliares: Mapa, Interface e Persistência

O desenvolvimento da Árvore de Habilidades agregou um ciclo de progressão contínuo, onde o jogador pode gastar recursos coletados no mundo. O Sistema de Mapa, essencial para o gênero Metroidvania, funciona dinamicamente, mapeando as 14 salas desenvolvidas à medida que o jogador cruza os delimitadores espaciais.

O sistema de persistência (salvamento via JSON) foi expandido e garante que tanto a exploração do mapa quanto os nós desbloqueados na árvore de habilidades sejam preservados perfeitamente entre as sessões.

Figura 12 – Sistema de Mapeamento dinâmico registrando o progresso espacial do jogador

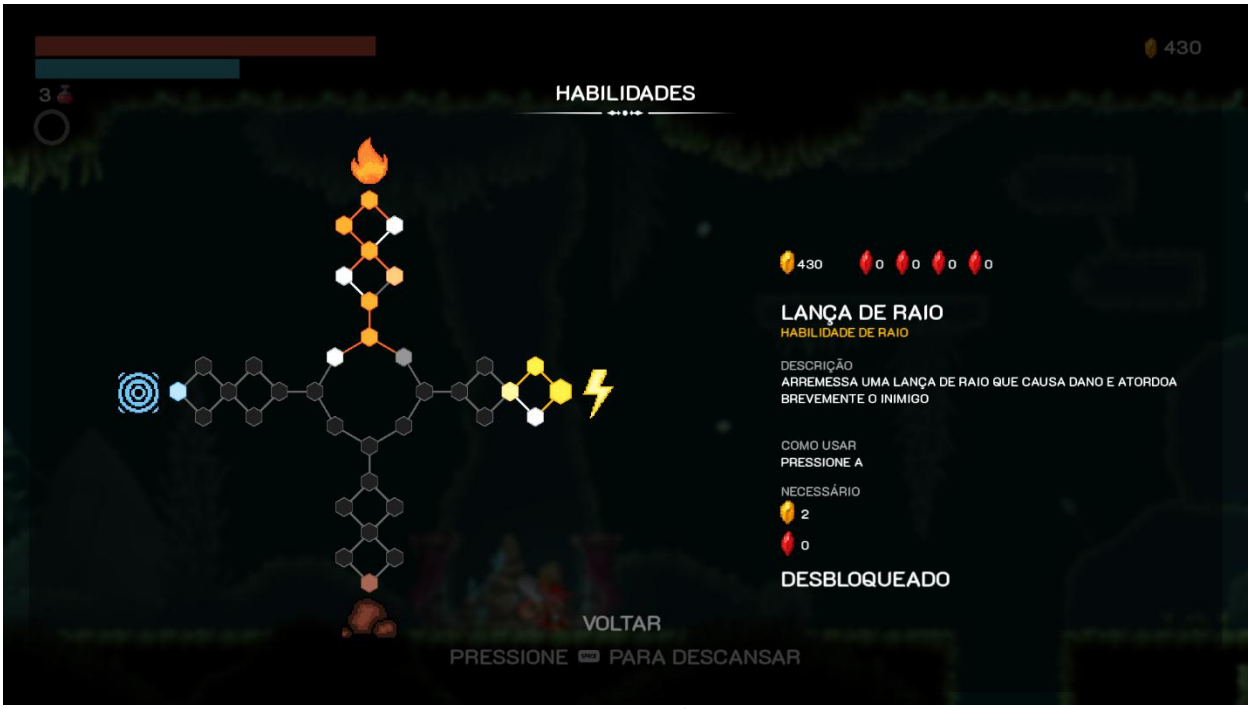

Figura 13 – Interface da Árvore de Habilidades

Figura 14 – Interface de Remapeamento de Botões

6.3. Combate e Modos Elementais

As expansões mecânicas elevaram a complexidade tática do jogo. O novo componente CombatBoxComponent viabilizou um combate preciso, sem registrar colisões sobrepostas incorretamente. A alternância entre os quatro Modos Elementais, auxiliada pela câmera lenta no menu seletor, mostrou-se fluida e encoraja a experimentação.

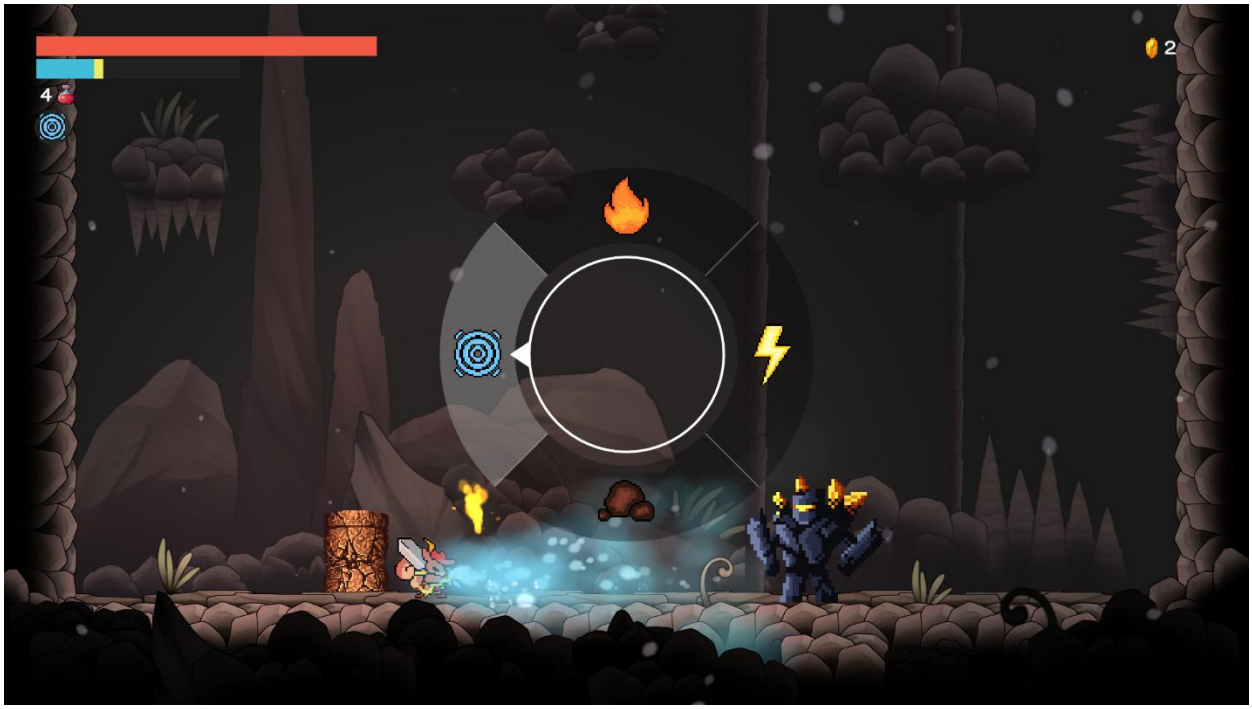

Figura 15 – Menu de seleção de Modos Elementais e jogador usando as habilidades de Fire Wisp, Jato de Gelo, Pilar de Terra e Modo Elétrico

Figura 16 – Habilidade de Bola Fogo

Figura 17 – Habilidade de Planar

Figura 18 – Habilidade de Lança de Raio

Figura 19 – Habilidade de Impacto Sísmico

7. Conclusão e Trabalhos Futuros

O presente trabalho cumpriu com êxito o objetivo de projetar, implementar e validar um motor de jogos proprietário de tempo real, culminando na entrega de uma demonstração funcional e polida do jogo Echoes of Elementum. A decisão de construir uma engine customizada em C++, SDL2 e OpenGL, prescindindo de soluções comerciais prontas, consolidou-se como um desafio acadêmico de imenso valor, exigindo o domínio e a aplicação prática de múltiplos pilares da Ciência da Computação, como Arquitetura de Sistemas, Computação Gráfica e Interação Humano-Computador.

Durante a segunda e última etapa deste POC, a infraestrutura básica foi expandida para suportar sistemas de alta complexidade. A adoção do padrão de Entidades e Componentes provou sua escalabilidade, especialmente com o desenvolvimento do CombatBoxComponent, que permitiu uma detecção de colisão granular e justa. O domínio sobre a API OpenGL foi evidenciado não apenas pela estabilidade na renderização de centenas de partículas dinâmicas, mas também pela aplicação técnica de Fragment Shaders para efeitos de pós-processamento de tela, ferramentas que elevaram significativamente o game feel e o aspecto visual do projeto.

Do ponto de vista do design de software orientado ao usuário, o projeto alcançou um alto nível de maturidade em acessibilidade e UX. A implementação de sistemas de remapeamento dinâmico de controles, tutoriais responsivos com renderização vetorial, a Árvore de Habilidades interligada à persistência de dados em JSON e o mapeamento automático do cenário validam a robustez da interface programada.

Adicionalmente, a realização de playtests estruturados com usuários reais demonstrou o compromisso do trabalho com as metodologias de validação de software. Através da coleta e análise de métricas, foi possível identificar gargalos cognitivos e de balanceamento, resultando em

refatorações vitais, como a inserção do estado de vulnerabilidade (Hurt) nas Máquinas de Estados Finitos dos inimigos e a adição de feedback visual na mecânica de cura.

Por fim, o desenvolvimento do Modo de Jogo Avançado e a estruturação das 14 salas interconectadas comprovam que, embora o jogo completo exija mais tempo de produção de conteúdo, o alicerce tecnológico está concluído. Echoes of Elementum não é apenas um protótipo, mas uma base sólida, otimizada e testada, capaz de suportar as demandas de um produto comercial independente.

Com a base tecnológica perfeitamente estruturada e todas as mecânicas elementais implementadas, o principal gargalo para a finalização de Echoes of Elementum reside na produção criativa de conteúdo (Level Design e Arte). Dessa forma, propõem-se as seguintes atividades para o avanço do projeto visando um eventual lançamento comercial:

- **Expansão do Level Design e Biomas:** Mapear e construir as rotas interconectadas de todo o jogo, integrando completamente o bioma de Caverna e desenvolvendo novas áreas temáticas para viabilizar as rotas de backtracking clássicas do gênero Metroidvania.
- **Integração do Catálogo de Inimigos:** Distribuir e balancear os 13 tipos de inimigos simples e os 9 bosses, cujas FSMs e mecânicas já foram desenvolvidas e testadas no Modo Avançado, pelo mapa principal, ajustando a curva de dificuldade progressiva.
- **Aprofundamento Narrativo:** Colocar no jogo uma história e objetivo para o jogador, através de diálogos, NPCs e de itens colecvionáveis.
- **Polimento Sonoro:** Substituir e refinar os efeitos sonoros temporários e a trilha musical, garantindo que o áudio espacial e os feedbacks acústicos estejam em perfeita sincronia com o aprimoramento visual e o game feel já alcançado no projeto.
- **Portabilidade:** Investigar a compilação cruzada do código-fonte nativo em C++ para viabilizar a execução do jogo em sistemas operacionais baseados em Linux e, futuramente, em plataformas de consoles dedicados.

8. Referências

SIMPLE DirectMedia Layer. SDL Wiki Documentation. Disponível em: https://libsdl.org/

DE VRIES, Joey. LearnOpenGL. Disponível em: https://learnopengl.com/

LAZY FOO' PRODUCTIONS. Beginning Game Programming v2.0. Disponível em: https://lazyfoo.net/tutorials/SDL/index.php

NYSTROM, Robert. Game Programming Patterns. Disponível em: https://gameprogrammingpatterns.com/

LINDEIJER, Thorbjørn. Tiled Map Editor. Disponível em: https://www.mapeditor.org/

MAAOT. Página do criador no Itch.io. Itch.io. Disponível em: https://maaot.itch.io/