

Criação e Análise de Sistemas de Combate Personalizáveis em um Jogo do Gênero Metroidvania

Fabio Roberto Pereira Filho, Eduardo Henrique Molina da Cruz

¹Campus Paranavaí – Instituto Federal do Paraná (IFPR)
CEP 87703-536 – Paranavaí – PR – Brasil

fabio.filho2002@hotmail.com, eduardo.cruz@ifpr.edu.br

Abstract. *This study explores customizable combat systems in digital games, emphasizing spell design through modular components known as runes. Each rune modifies spell behavior and properties, enabling unique combinations. Through a Metroidvania case study, we evaluate the impact of this mechanic on player engagement, demonstrating that modular systems enhance player experience and support adaptable game designs.*

Resumo. *Este estudo explora sistemas de combate personalizáveis em jogos digitais, com ênfase no design de feitiços por meio de componentes modulares chamados runas. Cada runa altera o comportamento e as propriedades dos feitiços, possibilitando combinações únicas. Por meio de um estudo de caso no gênero Metroidvania, avalia-se o impacto dessa mecânica no engajamento do jogador, demonstrando que sistemas modulares enriquecem a experiência e favorecem designs adaptáveis.*

1. Introdução

O mercado global de jogos digitais, movimentando cerca de US\$200 bilhões anuais e mantendo uma trajetória de expansão contínua [Newzoo 2024], é um dos setores mais dinâmicos da tecnologia, exigindo inovação constante para atrair e reter jogadores em um cenário altamente competitivo [Diniz and Abrita 2021]. Sistemas de combate personalizáveis destacam-se como uma abordagem promissora, oferecendo flexibilidade estratégica que promove imersão e incentiva experimentação [Amorim 2023]. Jogos do gênero *Metroidvania*, caracterizados por mapas interconectados, progressão não linear e aquisição gradual de habilidades que desbloqueiam novas áreas, são ideais para integrar mecânicas modulares, pois amplificam a exploração e a liberdade tática [Soares and Mota 2021]. Essa combinação de personalização e exploração cria experiências envolventes, desafiando desenvolvedores a equilibrar criatividade, consistência técnica e desempenho.

Este estudo analisa sistemas de combate personalizáveis por meio do desenvolvimento do *Manastride*, um jogo 2D *Metroidvania* ambientado em um universo *magepunk* em decadência. Nesse cenário, jogadores navegam por ruínas tecnomágicas, utilizando runas modulares para criar feitiços únicos que se adaptam a diferentes estilos de jogo. O sistema de runas permite personalizar atributos como alcance, velocidade e efeitos secundários, promovendo aprendizado natural e estratégias dinâmicas. O objetivo principal é explorar como essas mecânicas impactam o engajamento do jogador, com foco no sistema de magia e na jogabilidade geral. Os objetivos específicos incluem: (1) analisar mecânicas *Metroidvania* e sistemas modulares em jogos existentes; (2) projetar um sistema de

feitiços personalizáveis baseado em runas; (3) desenvolver um protótipo em *Python* com a biblioteca *Pygame*; e (4) avaliar o impacto dessas mecânicas por meio de testes preliminares e feedback qualitativo.

A metodologia adota ciclos iterativos, inspirados em práticas ágeis, com prototipagem rápida e ajustes baseados em testes internos, garantindo viabilidade técnica e potencial de engajamento. O *Manastride* integra o sistema de runas à exploração característica do gênero, incentivando jogadores a descobrirem combinações que otimizem combate e navegação.

Este artigo está estruturado em: a Seção 2 compara o *Manastride* com jogos relacionados; a Seção 3 detalha o desenvolvimento do jogo; a Seção 4 apresenta resultados e feedback; e por fim a Seção 5 oferece conclusões e perspectivas futuras.

2. Trabalhos Relacionados ao Desenvolvimento de Jogos

Sistemas modulares de combate têm sido amplamente explorados na indústria e na academia, oferecendo abordagens variadas para personalização e engajamento em jogos digitais. A Tabela 1 apresenta uma comparação entre o *Manastride* e quatro jogos que exemplificam diferentes tecnologias e filosofias de design, destacando suas contribuições para mecânicas customizáveis e sua influência no desenvolvimento do *Manastride*.

Noita [Games 2020], desenvolvido pela Nolla Games, utiliza uma *engine* própria para implementar um sistema de feitiços altamente modular, baseado na combinação de modificadores que alteram comportamento, trajetória e efeitos de feitiços. A simulação de física de partículas e ambientes destrutíveis permite milhares de combinações únicas. Essa abordagem inspirou o sistema de runas do *Manastride*, mas com maior ênfase em estruturação para progressão no gênero *Metroidvania*.

Frets on Fire [Voodoo 2006], um projeto de código aberto em *Python* com *Pygame*, demonstra modularidade em jogos de ritmo, permitindo personalização de músicas e controles. A simplicidade da biblioteca *Pygame* viabiliza prototipagem rápida, influenciando a escolha tecnológica do *Manastride*. Embora focado em ritmo, o jogo destaca a flexibilidade de *Pygame* para sistemas interativos.

Celeste [Games 2018], desenvolvido em *C#* com *MonoGame*, prioriza mecânicas de movimentação precisas, como *dash* e escalada, que desbloqueiam áreas e estratégias em um contexto *Metroidvania*. A modularidade dos níveis inspirou o design de mapas interconectados do *Manastride*. A escolha de *MonoGame* garante alto desempenho, mas o *Manastride* foca na customização de feitiços, utilizando *Pygame* para maior acessibilidade.

Hollow Knight [Cherry 2017], utiliza *charms* para personalizar habilidades, permitindo ao jogador adaptar estratégias de combate e exploração. Essa abordagem influenciou diretamente o sistema de runas do *Manastride*, que combina feitiços com efeitos táticos para navegação e combate em um universo *magepunk*. A Figura 1 ilustra o mapa interconectado de *Hollow Knight*, destacando a complexidade típica do gênero, um princípio adotado no design do *Manastride* para promover exploração.

Comparado a esses jogos, o *Manastride* integra a simplicidade de *Pygame*, como em *Frets on Fire*, com a personalização tática de *Hollow Knight* e a modularidade de feitiços de *Noita*, mas diferencia-se ao focar em runas que equilibram combate e explo-



Figura 1. Mapa completo de *Hollow Knight*, destacando interconexão típica do gênero. Fonte: <https://scripterswar.com/hollowknight/map-sketch>.

Tabela 1. Comparação entre *Manastride* e jogos relacionados.

Jogo	Gênero	Sistema de Personalização	Diferencial do Manastride
<i>Noita</i>	Roguelite	Feitiços modulares com modificadores (trajetória, efeitos)	Runas integradas à progressão Metroidvania : desbloqueiam áreas narrativas vs. geração procedural aleatória
<i>Frets on Fire</i>	Ritmo	Customização de músicas e controles	Sistema de combate dinâmico vs. mecânicas lineares de ritmo
<i>Hollow Knight</i>	Metroidvania	Charms para personalizar habilidades de combate e exploração.	Feitiços ativos customizáveis + sinergia combate/exploração vs. apenas <i>buffs</i>
<i>Celeste</i>	Plataforma	Modularidade em níveis, sem personalização de combate.	Personalização de magia para combate + navegação vs. apenas movimentação fixa

ração em um contexto *Metroidvania*. Diferentemente de *Celeste*, que enfatiza movimentação, o *Manastride* prioriza a criação de feitiços dinâmicos, utilizando a acessibilidade de *Pygame* para prototipagem rápida e um design que promove aprendizado estratégico e engajamento.

3. Estudo de Caso: Desenvolvimento do Manastride

O *Manastride* é um *Metroidvania* 2D ambientado em um universo *magepunk* em colapso, onde jogadores exploram ruínas tecnomágicas e personalizam feitiços (projétil, *dash*, escudo) por meio de runas modulares. As runas modificam o comportamento dos feitiços, como transformar um projétil em leque de disparos, sequência de múltiplos projéteis, ou modificar atributos como aumento de dano, alcance, velocidade e redução de recarga.

Essa modularidade permite aos jogadores adaptar feitiços ao seu estilo, promovendo engajamento por meio de experimentação criativa e aprendizado estratégico. A integração desse sistema com a estrutura não linear típica do gênero *Metroidvania* visa avaliar seu impacto na imersão, na curva de aprendizado e na diversidade de estilos de jogo, incentivando a descoberta de combinações que otimizem tanto o combate quanto a navegação de áreas normalmente inalcançáveis.

3.1. Metodologia de Desenvolvimento

O desenvolvimento seguiu uma abordagem iterativa, inspirada em metodologias ágeis como Scrum e Kanban [Kniberg and Skarin 2010], com ciclos de 1-2 semanas divididos em planejamento, prototipagem, testes e revisão. O planejamento definia objetivos claros, como implementar a movimentação básica, integrar runas ao sistema de feitiços ou otimizar a física de colisão. A prototipagem criava versões simplificadas para validar ideias, como feitiços com diferentes comportamentos ou animações de *sprites*. Testes internos avaliavam responsividade, equilíbrio e *fun factor*, com ajustes baseados em feedback para melhorar a intuitividade. Por exemplo, testes iniciais revelaram que a física do *dash* exigia maior fluidez para transmitir controle, enquanto iterações posteriores refinaram o impacto visual das runas, como trilhas de partículas em feitiços.

O controle de versão com Git garantiu rastreamento eficiente, permitindo reverter ajustes problemáticos, colaborar de forma organizada e manter um histórico claro de mudanças ¹. O uso do Git foi particularmente crítico durante a integração de novas runas, pois permitiu testar variações experimentais sem comprometer a estabilidade do protótipo, com *branches* dedicadas para cada funcionalidade nova.

A escolha de *Python* e *Pygame* foi motivada pela simplicidade, legibilidade e suporte a jogos 2D, ideais para prototipagem rápida [Python Software Foundation 2025, Pygame Community 2025]. *Python* facilita a escrita e manutenção de código, com uma sintaxe clara e uma vasta comunidade que suporta bibliotecas para gráficos, entrada de usuário e física básica, permitindo iterações rápidas essenciais para refinar o sistema de runas. *Pygame* oferece funcionalidades como renderização de *sprites*, gerenciamento de colisões e controle de entrada, garantindo uma experiência fluida na interação com runas e efeitos dinâmicos. A arquitetura do sistema combina Programação Orientada a Objetos (POO) com elementos de arquitetura dirigida a eventos, promovendo modularidade e baixa acoplamento. Eventos, como colisões ou ativação de feitiços, são gerenciados por uma central de eventos que notifica componentes relevantes, reduzindo dependências diretas e facilitando expansões futuras.

A física do jogo foi modelada com base em princípios realistas, utilizando equações para movimentos sem e com aceleração (gravidade):

$$s_f = s_0 + v \cdot t \quad s_f = s_0 + v_0 \cdot t + \frac{g \cdot t^2}{2} \quad (1)$$

O *GameController* fornece *delta time* para consistência entre diferentes taxas de FPS, enquanto *Pygame* suporta renderização fluida de animações e efeitos, como trilhas

¹Repositório disponível em: <https://github.com/fabiorpfilho/manastride>

de partículas para feitiços. O sistema de feitiços, núcleo do jogo, utiliza runas para modificar o comportamento dos feitiços, permitindo combinações como um *dash* com rastro explosivo ou um escudo que reflete projéteis. A separação de comportamentos por tipo de feitiço facilita a manutenção e legibilidade do código. Além disso, foram também utilizadas as bibliotecas de colisão e animação do *Pygame* para renderização dinâmica.

3.2. Arquitetura do Sistema

A arquitetura do *Manastride* combina Programação Orientada a Objetos (POO) com uma arquitetura dirigida a eventos, projetada para maximizar modularidade, reutilização e extensibilidade. A POO modela entidades do jogo como classes com atributos e comportamentos, utilizando herança para hierarquias e composição para flexibilidade, evitando a rigidez de hierarquias profundas. A arquitetura dirigida a eventos organiza o fluxo do jogo por meio de emissão e escuta de eventos, com uma central de eventos (*event dispatcher*) coordenando notificações para ações como colisões, ativação de feitiços ou mudanças de estado, reduzindo o acoplamento entre módulos e facilitando a adição de novas funcionalidades.

O diagrama de classes, apresentado na Figura 2, ilustra a hierarquia do sistema. A classe abstrata *Object* serve como base para objetos estáticos (plataformas, obstáculos) e dinâmicos (jogador, inimigos), com subclasses específicas como *Player*, *HammerBot* e *Terrain*. A classe *Player* gerencia atributos como posição, vida, mana e estado de animação, enquanto *HammerBot* implementa movimentação de patrulha e dano ao contato. A classe *Terrain* lida com elementos do cenário, como terrenos e plataformas. Classes de alto nível incluem *Level* (geração dinâmica do mundo), *GameController* (orquestração do fluxo do jogo), *Camera* (acompanhamento visual do jogador) e *CollisionManager* (gerenciamento de colisões). O sistema de feitiços é gerenciado pela classe *SpellSystem*, que coordena subclasses para cada tipo de feitiço (*Projectile*, *Dash*, *Shield*) e runas (*Rune*). A integração com eventos permite que ações como lançar um feitiço ou detectar uma colisão disparem notificações, atualizando dinamicamente o estado do jogo.

Essa arquitetura permite adicionar novos feitiços, runas ou inimigos com mínima alteração no código existente, utilizando composição para agregar comportamentos (runas como componentes plugáveis) e eventos para coordenar interações. O *GameController* sincroniza atualizações usando *delta time*, garantindo consistência em diferentes hardwares. Essa abordagem alinha-se à metodologia iterativa descrita anteriormente, promovendo evolução contínua e adaptação baseada em testes.

3.3. Desafios Técnicos

Diversos desafios técnicos foram enfrentados durante o desenvolvimento, especialmente na integração de mecânicas complexas com *Pygame*. O sistema de colisões apresentou problemas iniciais, com verificações simultâneas de colisões verticais e horizontais causando inconsistências, como o personagem atravessando obstáculos. A solução envolveu ajustar a classe *CollisionManager* para calcular a menor invasão entre retângulos de colisão. Isso garantiu que o personagem respondesse adequadamente ao ambiente, como ao colidir com plataformas ou inimigos.

Animações foram outro desafio significativo, devido ao uso de *sprites* de tamanhos variados, que causavam desalinhamentos visuais, especialmente ao espelhar o personagem para mudar de direção. O problema foi resolvido normalizando as dimensões

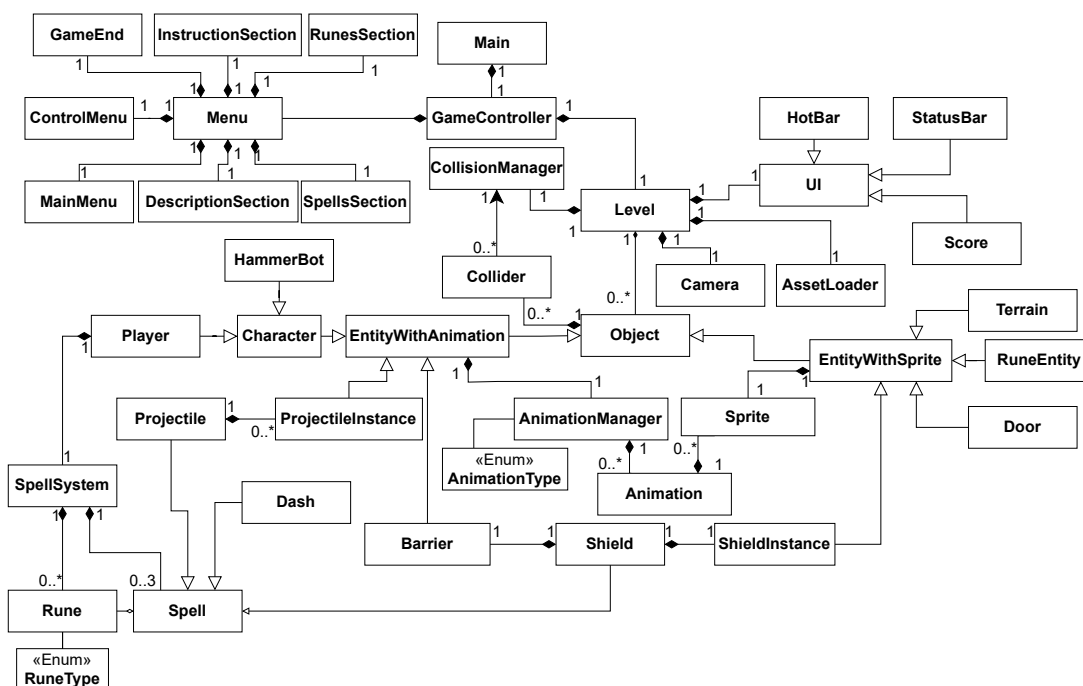


Figura 2. Diagrama de classes do *Manastride*, destacando a hierarquia orientada a objetos.

dos *sprites* e sincronizando a posição do personagem com a caixa de colisão, utilizando funções de transformação do *Pygame*. Esse processo exigiu iterações extensivas para garantir fluidez e coerência visual, com testes iterativos validando cada ajuste.

O maior obstáculo foi a criação artística, que demandou edição intensiva de *sprites* públicos para alinhá-los ao estilo *magepunk*. Após localizar recursos disponíveis, ajustes manuais foram necessários para garantir consistência visual e compatibilidade com animações e colisões. Ferramentas de edição gráfica, como LibreSprite, GIMP, e Aseprite, foram usadas para padronizar *sprites*, enquanto Tiled foi utilizado para a criação dos mapas. Testes iterativos refinaram a integração, consumindo tempo significativo, mas resultando em um visual coeso que reforça a estética do jogo.

3.4. Detalhamento do Sistema de Runas

O sistema de runas foi projetado para ser intuitivo e altamente flexível, com uma interface de inventário que permite equipar runas. Cada feitiço suporta até três runas, permitindo modificações no comportamento principal e a adição de efeitos complementares, como transformar um projétil em disparos múltiplos, modificar o escudo para criar uma barreira física que impede inimigos de passarem, diminuir a recarga ou aumentar o dano causado ou a duração, dependendo do feitiço. Essa abordagem favorece a criatividade, permitindo que jogadores evoluam de configurações simples para táticas complexas, promovendo aprendizado estratégico. Por exemplo, uma combinação de runas pode criar uma variação do projétil de leque com dano aumentado e recarga reduzida, ideal para combates contra múltiplos inimigos, ou gerar vários projéteis que causam dano de área ao colidir, ampliando o impacto tático.

A implementação utiliza a classe *SpellSystem* para gerenciar a lógica de runas,

com subclasses específicas para cada tipo de feitiço e runa. A arquitetura baseada em eventos notifica o sistema quando uma runa é equipada ou um feitiço é lançado, atualizando dinamicamente os efeitos visuais e mecânicos. A Figura 3 ilustra a interface de seleção de runas, destacando sua clareza visual e acessibilidade, com textos que indicam os efeitos das runas e feedback visual imediato ao equipá-las.

O sistema de runas foi testado com combinações iniciais, como projétil em leque (dano em área), projétil múltiplo (controle de multidões) e projétil básico (dano linear), demonstrando versatilidade. A integração com a exploração *Metroidvania* permite que runas sejam coletadas em áreas específicas do mapa ou de inimigos derrotados, incentivando o jogador a revisitar locais bloqueados, como plataformas altas acessíveis com um *dash* modificado por uma runa que permite alcance extra. Essa sinergia entre personalização e exploração reforça a imersão, enquanto a modularidade do sistema facilita a adição de novas runas, com testes futuros planejados para avaliar combinações avançadas, como runas aplicadas ao *dash* ou escudo, ampliando o potencial estratégico do jogo.

4. Resultados

Esta seção apresenta os resultados do desenvolvimento do *Manastride*, organizados de forma a detalhar o progresso alcançado e as avaliações realizadas. Inicialmente, descreve-se o estado final do protótipo, destacando as funcionalidades implementadas, como sistemas de colisão, movimentação, feitiços, runas e áudio, com ilustrações que evidenciam as interações no ambiente do jogo. Em seguida, são apresentados os resultados de testes realizados com 15 voluntários, com experiência variada no gênero *Metroidvania*, por meio de uma análise que combina métricas quantitativas (notas em escala de 0 a 10) e qualitativas (feedback e sugestões), avaliando o sistema de runas e a jogabilidade geral. Por fim, discute-se o impacto estratégico e didático do sistema de runas, com base em combinações testadas e seu potencial para promover engajamento e aprendizado tático.

4.1. Funcionalidades Implementadas

O protótipo do *Manastride* implementa um sistema robusto de colisão, movimentação fluida, três feitiços básicos (projétil, *dash*, escudo), um sistema completo de runas coletá-



Figura 3. Interface de seleção de runas no *Manastride*, mostrando opções de runas.

veis para todos os feitiços, múltiplos inimigos, uma arena final com três ondas de dificuldade progressiva, e sistemas de pontuação, vida, mana e áudio com *soundtrack* e efeitos sonoros. Os mapas interconectados incentivam a exploração típica do gênero *Metroidvania*, com áreas opcionais que recompensam a exploração com pontos e novas runas. Há a persistência de dados via arquivos locais, que salva a pontuação obtida ao vencer a última fase. A Figura 4 ilustra variações dos feitiços de projétil e escudo, incluindo o projétil em leque, o projétil de múltiplos disparos, o escudo básico e sua variação como barreira física, destacando suas interações com o ambiente.

O *Manastride* é estruturado em três fases distintas que promovem progressão e desafio crescentes. A primeira fase, introdutória, apresenta inimigos básicos e recompensas, como runas, para familiarizar o jogador com as mecânicas de combate e movimentação. A segunda fase, mais expansiva, oferece um mapa interconectado com áreas opcionais que incentivam a exploração, além de uma porta direta para o nível 3, acessível ao seguir um caminho linear. A terceira fase consiste em uma arena final, onde, após avançar, a saída é bloqueada, exigindo que o jogador enfrente três ondas de inimigos com dificuldade progressiva, que força o jogador a adaptar suas estratégias e combinações de runas conforme o combate se estende para sobreviver ao desafio. Derrotar todas as ondas resulta na vitória do jogo, testando a eficácia das combinações de runas e estratégias desenvolvidas nas fases anteriores.

4.2. Análise do Feedback dos Usuários

Uma pesquisa com 15 participantes, com experiência variada em *Metroidvania* (Figura 5), avaliou o sistema de runas e a jogabilidade. O grupo incluiu novatos e entusiastas, pro-



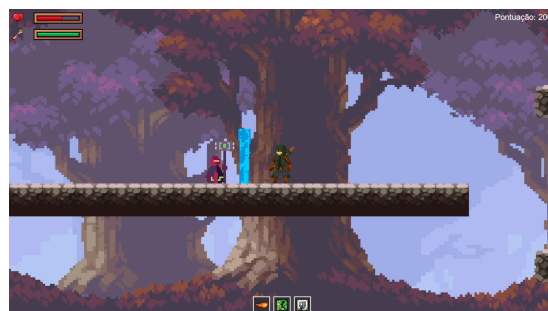
(a) Execução do Projétil em leque— posição inicial



(b) Execução do Projétil em leque— expansão



(c) Projétil de múltiplos disparos e feitiço de escudo básico



(d) Escudo na forma de barreira física que impede passagem de inimigos

Figura 4. Protótipo com feitiço de leque e escudo em execução.

porcionando perspectivas diversificadas.

O protótipo foi bem recebido, com média de 8,9 na avaliação geral (Figura 6), refletindo alto engajamento. Jogadores destacaram a diversão no combate e na experimentação com runas, sugerindo que a modularidade promove experiências envolventes para diferentes níveis de habilidade.

O sistema de runas foi considerado intuitivo, com média de 8,3 (Figura 7). A maioria achou a curva de aprendizado acessível, embora alguns notassem confusão inicial com os controles numéricos ou efeitos das runas, sugerindo a necessidade de tutoriais introdutórios.

A liberdade criativa foi avaliada positivamente por 12 participantes (Figura 8), que apreciaram a capacidade de adaptar feitiços a estilos de jogo variados, promovendo *replayability*. Contudo, alguns sugeriram maior variedade para sustentar o engajamento a longo prazo.

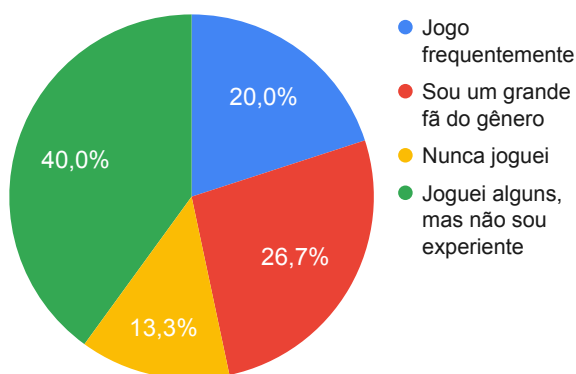


Figura 5. Experiência prévia dos participantes com jogos *Metroidvania*.

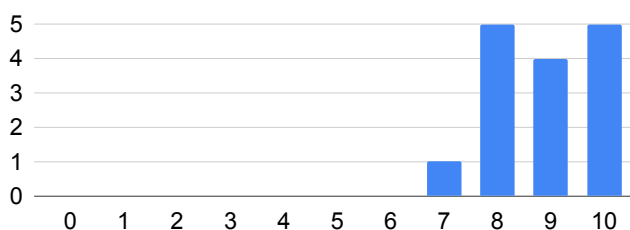


Figura 6. Avaliação geral do protótipo (escala de 0 a 10).

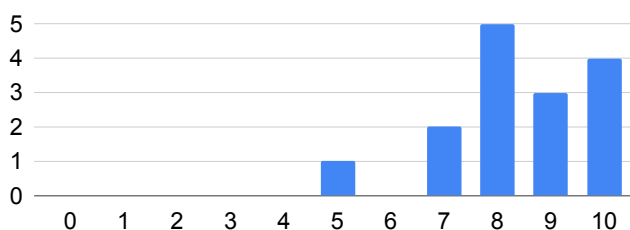


Figura 7. Facilidade de entendimento do sistema de runas (escala de 0 a 10).

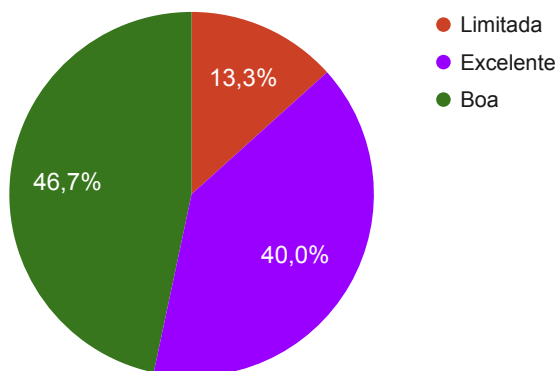


Figura 8. Avaliação da liberdade criativa oferecida pelo sistema de runas.

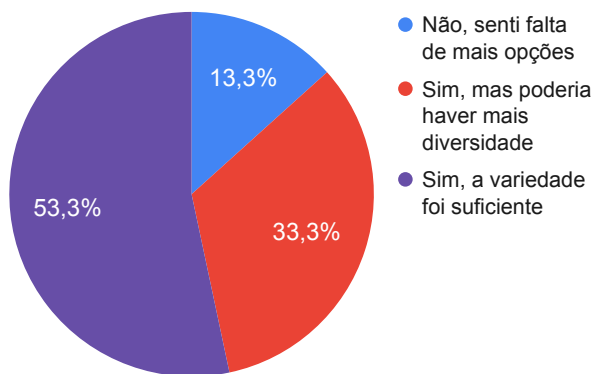


Figura 9. Variedade de runas disponíveis no protótipo.

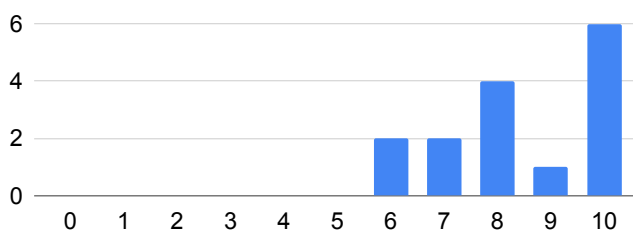


Figura 10. Avaliação da movimentação do personagem (escala de 0 a 10).

A variedade de runas foi considerada suficiente por 8 participantes (Figura 9), mas 7 desejaram mais opções.

A movimentação obteve média de 8,4 (Figura 10), com elogios à responsividade, embora sugestões para melhorar a sensação de peso indiquem áreas para refinamento.

Os participantes valorizaram configurações de projéteis que amplificam dano em área (leque) ou controle de multidões (múltiplos disparos), destacando o apelo estratégico e visual. Sugestões incluíram tutoriais visuais, remapeamento de controles (*dash* para SHIFT), integração de runas ao *dash*, e adição de mecânicas como *wall climbing*. A interface foi bem avaliada (média de 8,5), mas ajustes em clareza foram recomendados para novatos.

4.3. Impacto Estratégico e Didático do Sistema de Runas

O sistema de runas promove engajamento e aprendizado estratégico, permitindo que jogadores evoluam de configurações simples para táticas complexas. A Tabela 2 resume combinações testadas, destacando seus efeitos estratégicos. Testes indicam que 73% dos participantes experimentaram novas abordagens, reforçando o potencial didático do sistema em ensinar planejamento tático.

O sistema de runas alinha-se a conceitos de *fun factor* [Amorim 2023], promovendo agência e progresso. Apesar de limitações, como mapas *placeholder* e número limitado de runas, o protótipo demonstra viabilidade para designs adaptáveis e educativos.

Tabela 2. Exemplos de combinações de runas e seu impacto estratégico no protótipo.

Combinação	Efeito Principal	Estratégia Indicada	Feedback Médio (0–10)
Projétil + Leque	Múltiplos disparos em arco	Dano em área vs. grupos	9,2
Projétil + Múltiplo	Três disparos contínuos	Controle de multidões	8,7
Projétil Básico	Dano linear	Corredores estreitos	8,5

5. Conclusão e Trabalho Futuro

O desenvolvimento do *Manastride* marca um avanço significativo na criação de um sistema de combate customizável no gênero *Metroidvania*, integrando mecânicas modulares que promovem engajamento e criatividade. O protótipo implementa uma base técnica robusta, incluindo um sistema de colisão preciso, movimentação fluida, três feitiços básicos (projétil, *dash* e escudo), um sistema de runas coletáveis no mapa, um inimigo funcional, sistemas de pontuação, vida, mana e áudio com *soundtrack* e efeitos sonoros. Esses elementos, construídos com *Python* e *Pygame*, demonstram viabilidade técnica e potencial para uma experiência envolvente, embora o protótipo ainda utilize um mapa *placeholder* e representações visuais simplificadas, limitando a complexidade típica do gênero.

Testes preliminares com 15 participantes confirmam a acessibilidade e o potencial didático do sistema, que ensina estratégias por meio de experimentação criativa. A modularidade das runas foi elogiada por promover liberdade tática e “replayabilidade”. O feedback qualitativo destacou a diversão no combate e a intuitividade da interface, mas sugeriu melhorias como tutoriais, remapeamento de controles (*dash* para SHIFT ou CTRL), e integração de runas ao *dash*, indicando direções claras para refinamento.

Para trabalhos futuros, planeja-se: (1) refinar o balanceamento das runas, ajustando seus efeitos para maior diversidade estratégica e sinergia entre feitiços; (2) expandir a arena final com novos chefes ou desafios narrativos, enriquecendo a história e testando combinações avançadas de runas; (3) otimizar *cooldowns* de feitiços e remapear controles com base no feedback, visando maior equilíbrio e conforto; (4) Analisar a adequação do remapeamento de controles como sugerido e (5) realizar testes expandidos com jogadores de diferentes perfis para avaliar engajamento, impacto das runas e *fun factor*, refinando o protótipo com base em feedback iterativo.

O *Manastride* contribui para o estudo de sistemas modulares ao demonstrar como a simplicidade de *Pygame* pode suportar mecânicas complexas, alinhando-se a jogos como *Hollow Knight* e *Noita* em sua ênfase na personalização. O projeto destaca o potencial educativo de sistemas de runas, incentivando aprendizado estratégico e criatividade, enquanto aponta para a necessidade de maior complexidade para alcançar o padrão do gênero.

Referências

- Amorim, C. F. (2023). A indústria de jogos eletrônicos: oportunidades e desafios para o cenário de e-sports. *Ciências Exatas e da Terra*, 27(122/123).
- Cherry, T. (2017). *Hollow knight* [jogo eletrônico]. Disponível para PC, Nintendo Switch, PlayStation 4 e Xbox One.
- Diniz, R. G. and Abrita, M. B. (2021). A indústria de games no território brasileiro: um estudo baseado em dados e indicadores recentes. *Formação (Online)*, 28(53).
- Games, M. M. (2018). *Celeste* [jogo eletrônico]. Disponível para PC, Nintendo Switch, PlayStation 4, Xbox One e outras plataformas.
- Games, N. (2020). *Noita*. Disponível em: <https://noitagame.com/>. Acesso em: 30 maio 2025.
- Kniberg, H. and Skarin, M. (2010). *Scrum and Kanban: Making the Most of Both*. InfoQ, C4Media, Inc.
- Newzoo (2024). Newzoo's global games market report 2023: May 2024 update.
- Pygame Community (2025). Pygame documentation. <https://www.pygame.org/docs/>. Domínio público.
- Python Software Foundation (2025). What is python? executive summary. <https://www.python.org/doc/essays/blurb/>. Domínio público.
- Soares, J. P. and Mota, R. R. d. (2021). Guiando com uma mão invisível: Explorando metroidvanias e sua não-linearidade guiada. In *Proceedings of SBGames 2021*, Curitiba, Brazil.
- Voodoo, U. (2006). *Frets on fire*. Disponível em: <https://fretsonfire.sourceforge.net/>. Acesso em: 30 maio 2025.