Desenvolvimento de Jogo em Assembly RISC-V: Uma Abordagem Inspirada em Elementos Cinematográficos

Smash it, Hulk!

Guilherme Silva Cavalcante, Igor Araújo Rodrigues, João Paulo Silva Mendes

Universidade de Brasília, Depto. Ciências da Computação, Brasília - DF

Figura 1: *Screenshot* de tela de inicial do jogo

Resumo

O presente artigo descreve o desenvolvimento de um jogo inteiramente implementado em Assembly RISC-V, que integra interface gráfica, trilha sonora, efeitos sonoros e captação das entradas do teclado. Inspirado no clássico jogo *Fix-It Felix* – popularizado pelo filme *Detona Ralph* – o projeto propõe uma releitura, substituindo o objetivo de conserto pelo de destruição. Com ambientação baseada em cenas emblemáticas do universo Marvel, especificamente a

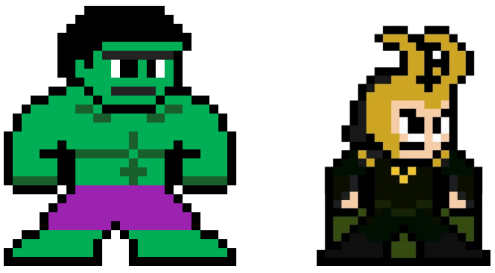
icônica sequência em que o Capitão América convoca o Hulk (“And Hulk, Smash!”), o jogo apresenta como protagonista o Hulk, cuja missão consiste em romper as janelas de um edifício. O antagonismo é representado por Loki, que posiciona-se no topo do prédio e lança projéteis de energia, além da presença dos Chitauri, alienígenas que operam em naves e disparam lasers. O desenvolvimento empregou uma técnica denominada “renderização por demanda”, visando a otimização do desempenho gráfico. Este artigo apresenta a fundamentação teórica, a metodologia aplicada, os desafios superados e os resultados obtidos, concluindo com considerações sobre a aplicabilidade da abordagem adotada e perspectivas futuras.

Palavras-chave: Assembly RISC-V; renderização por demanda; interface gráfica; efeitos sonoros; desenvolvimento de jogos; otimização de desempenho.

1. Introdução

Nas últimas décadas, a área de desenvolvimento de jogos tem expandido-se significativamente, abrangendo desde ambientes de alta complexidade até experimentações em arquiteturas não convencionais. Este projeto insere-se neste contexto ao optar por uma implementação integral em Assembly RISC-V, demonstrando a viabilidade e os desafios de se desenvolver aplicações interativas em baixo nível de abstração. A escolha da linguagem Assembly possibilitou um controle refinado sobre os recursos computacionais, o que se revela especialmente relevante em ambientes com limitações de hardware.

Inspirado no jogo *Fix-It Felix* e adaptado à estética e narrativa dos filmes da Marvel – com ênfase na representação do Hulk e na dinâmica de combate contra Loki e os Chitauri –, o projeto redefine o paradigma de jogabilidade tradicional. Ao inverter a premissa original, em que o personagem conserta as estruturas, a proposta do jogo enfatiza a ação destrutiva, proporcionando uma experiência diferenciada e desafiadora para o jogador. Este trabalho visa documentar, de forma sistemática, os processos e decisões adotados durante o desenvolvimento, destacando as técnicas de renderização e a gestão dos recursos de memória, bem como os impactos destas escolhas no desempenho e na qualidade final do produto.



Figuras 2 e 3: *Sprites* dos personagens do jogo.

2. Metodologia

O desenvolvimento do jogo seguiu uma abordagem orientada por "renderização por demanda", técnica que consiste em atualizar os elementos visuais somente quando há modificações no estado do jogo. Essa estratégia possibilitou uma significativa otimização, uma vez que o fundo das fases é renderizado uma única vez e os sprites dos personagens e demais elementos gráficos são atualizados apenas quando ocorre mudança em suas posições ou estados. Dessa forma, reduziu-se o processamento desnecessário, contribuindo para uma execução mais fluida, independentemente da máquina hospedeira.

Figura 4: *Screenshot* da tela do jogo durante uma *gameplay*

2.1 Limites e colisões

Para delimitar os limites da tela jogável, foram utilizados quatro registradores "fixos", os quais determinam os contornos da área ativa. Outras variáveis de estado – tais como número de vidas, fase atual, posições e estados de invencibilidade – foram armazenadas em espaços específicos da memória, os quais são atualizados conforme a dinâmica do jogo. A técnica de renderização por demanda, entretanto, apresentou desafios particulares, como a questão dos rastros deixados por projéteis que atravessam o Bitmap Display. Para solucionar essa problemática, desenvolveu-se uma função dedicada à substituição dos rastros, restaurando a parte do *sprite* de fundo previamente ocupada.

A mecânica de colisão é implementada através da verificação contínua da distância absoluta entre as coordenadas **x** e **y** do *player* e dos obstáculos, armazenadas em endereços específicos da memória. Durante cada ciclo de processamento, o sistema calcula a diferença entre os valores ***x_player*** e ***x_obstáculo***, bem como ***y_player*** e ***y_obstáculo***, utilizando operações de subtração e módulo para garantir resultados positivos. Se ambas as distâncias absolutas forem menores ou iguais a um limiar pré-definido, uma colisão é registrada.

2.2 Música, Bitmap Display e MMIO

O projeto foi desenvolvido em Assembly para a arquitetura RV32, exigindo um profundo conhecimento do funcionamento da CPU, do gerenciamento de memória e dos periféricos gráficos. A escolha pela

linguagem Assembly, embora aumente a complexidade da implementação, garantiu o acesso direto aos recursos do hardware, permitindo a criação de uma interface gráfica customizada, a incorporação de efeitos sonoros e a manipulação precisa das entradas do teclado. A execução do jogo foi validada inicialmente na tela do RARS – ambiente de simulação para RISC-V – sendo posteriormente testada em outras plataformas compatíveis, evidenciando a robustez e portabilidade da solução.

3. Resultados Obtidos

Após a fase de desenvolvimento e testes, o jogo foi considerado concluído, com a implementação de todas as funcionalidades previstas no planejamento inicial. Os principais resultados obtidos incluem:

- Interface Gráfica e Sonora: A interface foi implementada com sucesso, apresentando elementos visuais consistentes e uma trilha sonora integrada que contribui para a imersão do jogador. Os efeitos sonoros sincronizados às ações, como a destruição de janelas e os disparos dos projéteis, foram adequadamente incorporados, enriquecendo a experiência interativa.
- Mecânica de Jogo: A inversão da premissa do jogo original, ao substituir a função de conserto pela de destruição, proporcionou uma dinâmica inovadora e desafiadora. O jogador, ao controlar o Hulk, deve romper todas as janelas do edifício para progredir, enfrentando o vilão Loki, que se posiciona estrategicamente no topo do prédio e lança projéteis. Além disso, a presença dos Chitauri, com suas naves e ataques a laser, introduziu variações que aumentam a complexidade e a diversidade do jogo.

Figuras 5 e 6: *Sprites de obstáculos que colidem com o jogador*

- Desempenho e Otimização: A técnica de renderização por demanda revelou-se eficiente, permitindo a atualização seletiva dos elementos gráficos e, consequentemente, uma melhoria no desempenho. A renderização única do fundo e a atualização dos sprites apenas quando necessário reduziram o processamento, fazendo com que o jogo rodasse de forma fluida em diferentes

plataformas, mesmo em ambientes simulados.

- Desafios Técnicos Superados: Entre os desafios técnicos, destaca-se a implementação da função de restauração dos rastros deixados pelos projéteis. Essa solução não apenas preservou a integridade visual do ambiente, mas também demonstrou a capacidade de adaptação e resolução de problemas inerentes ao desenvolvimento em baixo nível.

4. Conclusão

O desenvolvimento do jogo em Assembly RISC-V, com sua abordagem de renderização por demanda, representa um esforço significativo no que diz respeito à otimização e à gestão de recursos em ambientes com restrições de hardware. Ao optar por uma linguagem de baixo nível, a equipe de desenvolvimento obteve um controle preciso dos aspectos computacionais, possibilitando a implementação de uma interface gráfica customizada, a integração de efeitos sonoros e a manipulação direta das entradas do usuário.

Referências Bibliográficas

1. Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann.
2. Harris, S. L., & Harris, D. M. (2021). Digital Design and Computer Architecture: RISC-V Edition. Morgan Kaufmann.
3. RARS: RISC-V Assembler and Runtime Simulator. Disponível em: https://github.com/TheThirdOne/rars.
4. Detona Ralph. (2012). [Filme]. Walt Disney Pictures.
5. Marvel Studios. (2012). *The Avengers*. Directed by Joss Whedon