Joy's Chase: A Pac-Man Adaptation

Davi Galvão Guerra, Eric Wu Harris

Departamento de Ciência da Computação
Universidade de Brasília (UnB) – Brasília, DF – Brasil

Abstract. *This article presents the game Joy's Chase, developed by Davi Galvão Guerra and Eric Wu Harris, students at the University of Brasília, as part of the course Introduction to Computational Systems (ISC), taught by Professor Marcus Vinicius Lamar. The project consists of an original reinterpretation of the classic Pac-Man, released in 1980, with the addition of elements and characters from Pixar's Inside Out franchise. The game was implemented in Assembly language, using the RISC-V architecture and the RARS simulator for execution.*

Keywords— *Pac-Man, Assembly, RARS, RISC-V, Inside Out*

Resumo. *Este artigo apresenta o jogo Joy's Chase, desenvolvido por Davi Galvão Guerra e Eric Wu Harris, alunos da Universidade de Brasília, como parte da disciplina Introdução aos Sistemas Computacionais (ISC), ministrada pelo professor Marcus Vinicius Lamar. O projeto consiste em uma releitura original do clássico Pac-Man, lançado em 1980, com a adição de elementos e personagens da franquia Divertida Mente da Pixar. O jogo foi implementado em linguagem Assembly, utilizando a arquitetura RISC-V e o simulador RARS para sua execução.*

Palavras-chave— *Pac-Man, Assembly, RARS, RISC-V, Divertida Mente*

1. Introdução

Este trabalho apresenta uma releitura de um dos maiores clássicos dos jogos eletrônicos, o Pac-Man[1], lançado em 1980 pela empresa Namco e que rapidamente se tornou um ícone da cultura pop.

Em Pac-Man, o jogador controla um personagem amarelo em formato de círculo, que deve percorrer um labirinto comendo todos os pontos enquanto foge de quatro fantasmas coloridos: Blinky, Pinky, Inky e Clyde. Cada fantasma tem um comportamento único, tornando o jogo desafiador e estratégico. O objetivo é limpar o labirinto antes de ser capturado pelos fantasmas. Pac-Man também inclui as Power Pellets, que permitem ao jogador inverter o papel de predador e presa, temporariamente dando ao jogador a habilidade de comer os fantasmas. Sua simplicidade, combinada com uma jogabilidade envolvente, garantiu um sucesso duradouro e uma imensurável influência no design de jogos subsequentes.

Nesse contexto, o jogo produzido busca reproduzir todas as funcionalidades do original, mas com uma temática nova, baseada na franquia Divertida Mente[2]. Divertida Mente (Inside Out) é um filme de animação da Pixar que se passa dentro da mente de uma garota chamada Riley, onde suas emoções — Alegria, Tristeza, Raiva, Medo e Nojinho — são personificadas e controlam suas reações e decisões.

Em Joy's Chase, o jogador assume o controle de Alegria, representada pela cor amarela, mantendo a essência visual do personagem principal do jogo original. Seguindo essa lógica, os fantasmas foram substituídos pelas outras emoções: Tristeza, Medo, Raiva e Nojinho, que perseguem o jogador pelo labirinto. O objetivo do jogo permanece o mesmo: guiar Alegria pelo cenário, recolhendo as "memórias" espalhadas enquanto evita ser capturada pelas outras emoções. Cada emoção adversária terá um comportamento diferente, refletindo suas características do filme e adicionando uma nova camada de desafio à jogabilidade.

Figure 1: Logotipo do jogo

Figure 2: Imagem do filme

2. Metodologia

Nesse projeto, a implementação das funcionalidades foi organizada de forma estratégica, com a distinção entre funções básicas e mais complexas, permitindo um progresso mais eficiente no desenvolvimento do jogo. Essa abordagem foi fundamental, considerando que o aprendizado da linguagem Assembly exigiu tempo e dedicação devido à sua complexidade.

2.1. RARS

O simulador RARS[3] (RISC-V Assembly Runtime Simulator) foi desenvolvido para simular a execução de programas na linguagem Assembly da arquitetura RISC-V. O seu principal objetivo é ser um ambiente efetivo de desenvolvimento para o aprendizado.

O desenvolvimento deste projeto foi realizado utilizando a arquitetura RISC-V, inicialmente com o simulador RARS, e posteriormente migrando para o FPGRARS[4]. A mudança para o FPGRARS se mostrou necessária com o tempo, pois essa ferramenta oferece maior velocidade de execução e melhor otimização.

2.2. Arte

A arte do jogo, feita com o auxílio do site Pixilart[5], é baseada no Pac-Man, mas com mudanças para se adequar ao novo tema de Divertida Mente. Os mapas foram planejados para sprites de personagens 16x16, para facilitar alguns aspectos posteriores da produção. O estilo artístico dos mapas tenta se aproximar do original, mas com composições diferentes e design distinto nas paredes, que são preenchidas com memórias, como no filme referenciado.

Figure 3: Imagem do menu

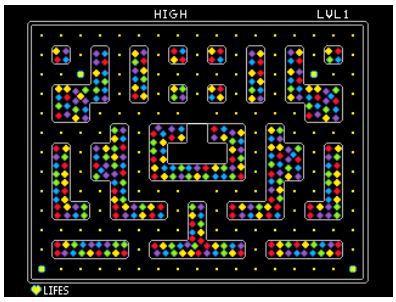

Figure 4: Imagem do mapa 1

2.3. Interface gráfica

A implementação da interface gráfica foi feita a partir do Bitmap Display (320x240, 8 bits/pixel) integrado ao RARS. O processo de renderização foi feito a partir de uma função PRINT criada com o auxílio de projetos[6] e tutoriais de alunos veteranos[7]. Essa função opera baseada em quatro argumentos: o endereço do .data da imagem a ser reproduzida, as coordenadas x e y em que será imprimida a imagem e o frame do bitmap.

De forma simplificada, essa função faz um loop que carrega 4 bytes (1 word) por vez do arquivo .data a ser mostrado na tela e salva no endereço específico do Bitmap Display. Ao serem printadas todas as colunas da respectiva linha, o número da coluna é reiniciado e o número da linha aumenta em 1, repetindo assim o processo até serem impressas todas as linhas do respectivo arquivo.

2.4. Interface com teclado

A implementação da interface com teclado foi feita a partir do simulador KDMMIO integrado ao RARS. O processo de controle de entradas e saídas foi desenvolvido a partir dos materiais[8] que o professor Lamar disponibilizou, apenas com algumas mudanças para se adequar às necessidades.

Em poucas palavras, essa funcionalidade apenas mapeia os botões selecionados e direciona o programa para a label correspondente. Por exemplo, o botão "w", ao ser pressionado, encaminha o programa para a label responsável pela movimentação para cima do personagem controlado pelo jogador.

2.5. Colisão

A implementação da colisão foi uma das funcionalidades mais demoradas e complexas. Foi necessária a adição de várias verificações de colisão, tanto para o jogador quanto para as emoções, então surgiu a necessidade de criar um mapa de colisão. Esse mapa facilitou as checagens, permitindo que, ao verificar o movimento selecionado pelo jogador ou emoção, a cor de um byte do quadrado adjacente indicaria a possibilidade de movimento. Nesse caso, a cor cinza (173) representava as paredes, cor preta (0) indicava caminhos livres, ao passo que as cores amarela (63) e azul (232) assinalavam, respectivamente, os pontos e power pellets.

No caso de colisão entre jogador e inimigos, a verificação era realizada de forma simples, por meio da comparação entre as coordenadas x e y, respectivamente. Caso o jogador se encontre no seu estado natural, a colisão resulta na perda de uma vida e a volta das emoções para sua posição inicial. Em oposição, quando o jogador tiver no estado de poder, a emoção que colidir com o jogador volta à sua posição inicial e a pontuação aumenta.

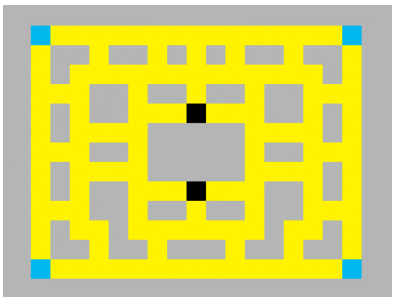

Figure 5: Mapa de colisão 1

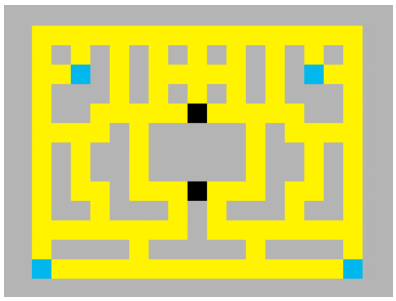

Figure 6: Mapa de colisão 2

2.6. Interface de áudio

A implementação da interface de áudio MIDI foi feita a partir das syscalls padrão do sistema (31 e 32). Para a música, foi utilizado um software de áudio digital[9] para ser possível recriar a música tema "Bundle of Joy"[10] do filme no formato MIDI, para que fosse, posteriormente, convertida[11] para o formato compatível com o RARS. Para os efeitos sonoros do jogo, foram usados os próprios instrumentos disponibilizados pelo simulador, diversificando-os de acordo com a situação vivenciada pelo jogador, como a captura de pontos, morte e eliminação de inimigos.

2.7. IA dos inimigos

Tal como no jogo original, o jogo produzido possui quatro inimigos com diferentes comportamentos, de forma que reproduzisse as características de cada personagem do filme.

2.7.1 Raiva

O Raiva se movimenta de forma ininterrupta até colidir com uma parede, quando troca de direção. Em modo de perseguição, sua movimentação segue o sentido anti-horário, enquanto em modo de fuga, ocorre a inversão da direção. A ideia por trás dessa movimentação é fazer com que as extremidades do mapa sejam território hostil para o jogador.

2.7.2 Medo

O medo se movimenta de forma pseudoaleatória, com o uso da ecall 42, que gera um número em uma faixa de 0 a 3, para representar cada direção. A ideia por trás dessa movimentação fortuita é tornar o jogo desafiador e mais imprevisível.

2.7.3 Tristeza

A tristeza, quando em modo de perseguição, tenta reduzir ao máximo a distância com o jogador, priorizando sempre o eixo de movimentação com maior diferença. No entanto, em seu modo de fuga, ela realiza o contrário, tentando aumentar a distância no eixo de menor diferença, o que a leva até o canto diametralmente oposto ao jogador. O propósito dessa movimentação é transformar ela na principal ameaça ao jogador, visto que está sempre no encalço da Alegria.

2.7.4 Nojinho

A Nojinho, quando se encontra em perseguição, espelha os movimentos do jogador, indo para a direção oposta a dele. Todavia, ao entrar no estado de fuga, a personagem copia a movimentação do jogador, o que tende a realizar seu deslocamento para as extremidades do mapa. Esse comportamento atípico tem a função de criar bloqueios de passagem dinâmicos, uma vez que sempre se encontra em movimento quando o jogador se desloca.

3. Resultados Obtidos

O desenvolvimento do projeto foi um árduo desafio com diversas adversidades ao longo do percurso. Em primeira instância, a falta de conhecimento com o Assembly se mostrou um desafio nos primeiros momentos do trabalho, devido a inexperiência com a linguagem e sua complexidade inerente, o que se tornou nítida ao surgirem os problemas de branch ao longo do código, uma vez que é possivel realizar apenas saltos de 12 bits, o que é um empecilho em um código extenso.

A implementação de todas as funcionalidades necessárias para o sucesso do projeto mostrou-se um desafio, mas que foi superado com estudo e dedicação, o que resultou em uma releitura divertida e com personalidade de dois universos, que pareciam distantes, mas que se uniram perfeitamente.

4. Conclusão

Nesse artigo, foi abordada a criação de uma releitura original do clássico Pac-Man na linguagem Assembly RISC-V. Foi um enorme desafio que resultou em um grande aprendizado e experiencia no uso de uma linguagem de baixo nível.

Referências

(1) Wikipedia do Pac-Man
https://pt.wikipedia.org/wiki/Pac-Man
(2) Trailer do filme Divertida Mente
https://www.youtube.com/watch?v=yRUAzGQ3nSY
(3) GitHub do RARS
https://github.com/TheThirdOne/rars
(4) GitHub do FPGRARS
https://github.com/LeoRiether/FPGRARS
(5) Pixilart
https://www.pixilart.com/
(6) GitHub LAMAR
https://github.com/victorlisboa/LAMAR
(7) RISC-V RARS – Renderizacao dinamica no Bitmap Display
https://www.youtube.com/watch?v=2BBPNgLP6_s&t=982s
(8) Materiais do Professor Marcus Vinicius Lamar da Linguagem Assembly e I/O (sem link)
(9) FL Studio
https://www.image-line.com/ fl-studio/
(10) Musica Bundle of Joy
https://www.youtube.com/watch?v=32lI1Z1MBx0
(11) Tradutor de MIDI para RISC-V
https://github.com/Zen-o/Tradutor_MIDI-RISC-V