Road Fighter

Lucas Amaral De Faria
211055316

Pedro Farias De Oliveira
211055577

Sofia Dy La Fuente Monteiro
211055530

Departamento de Ciência da computação – Universidade de Brasília (UnB)
CIC0003 – Introdução aos sistemas computacionais

Resumo

Road Fighter é uma releitura do jogo de corrida de mesmo nome, “Road fighter”, desenvolvido pela Konami. O projeto foi desenvolvido para a disciplina de Introdução aos sistemas computacionais, utilizando o simulador de linguagem assembly, Rars Risc-V.

Nesse artigo será abordado a metodologia utilizada na criação do jogo, as dificuldades encontradas e os resultados obtidos.

Palavras-chave: Road Fighter, corrida, jogo, simulador Risc-V.

1. Introdução

Road Fighter é um jogo de corrida arcade desenvolvido pela empresa japonesa, Konami. Foi lançado em 1984 para a plataforma de arcade e posteriormente sendo lançado para o videogame Nintendo Entertainment System (NES).

O jogo consiste em uma corrida, com fases de diferentes cenários, cujo objetivo é atingir a linha de chegada, evitando bater nos demais carros, sem que a sua gasolina se esgote.

2. Metodologia

Para a criação do jogo utilizamos o conjunto de instruções (ISA) de código aberto Risc-V para escrever e armazenar o código, e utilizamos também o FPGRARS para executar o jogo, por ele possuir uma execução mais rápida.

2.1 Divisão de Tarefas

Logo no início do projeto, foi decidido que antes de começar de fato a programação do jogo, passaríamos uma semana estudando os jogos criados por alunos em semestres anteriores, para que pudéssemos entender melhor o funcionamento

da linguagem que seria utilizada no decorrer do desenvolvimento do projeto.

Após esse período, foi feita uma reunião onde discutimos a base do funcionamento do jogo e dividimos as primeiras tarefas que cada integrante do grupo iria realizar. A partir desse dia, cada membro do grupo ficou encarregado de uma parte do projeto.

Sempre que obtínhamos alguma dúvida ou progresso, compartilhávamos com os demais, para discutirmos soluções e possíveis melhoras no projeto.

2.2 Movimentação do jogador

Para a movimentação, primeiro se receber uma informação do teclado do jogador, utilizando o *input polling non-blocking*, no qual o programa confere se há uma tecla pressionada. Caso não haja, o jogo retorna para o *loop* principal, onde será criado o movimento da pista.

Com o objetivo de facilitar o desenvolvimento do jogo, apenas as movimentações para as direções esquerda e direita são possíveis. Dessa forma, a movimentação do carro é feita pela animação da pista de fundo, que não possui a necessidade de entrada de dados por parte do jogador.

2.3 Animações

Para criar a percepção de movimento, a solução encontrada foi "empurrar" a imagem de fundo para baixo. Para isso, programamos o jogo para que ele percorresse o *bitmap display* e, para cada linha printada na tela, o programa salva os endereços dessas linhas no frame 0 e salva na linha de baixo no frame 1, para que se possa trocar os conteúdos de posição. Ao repetir isso diversas vezes, cria-se a impressão de que o carro está se movendo.

A partir dessa base conseguimos implementar um limite de linhas a serem movidas, sendo assim, ao atingir tal limite, é carregada na tela a imagem de chegada, mostrando que o jogador venceu.

Para a colisão, salvamos a posição do jogador em um registrador, de forma que antes de se movimentar, o programa confere se os limites da imagem do carro coincidem com as coordenadas de outros elementos no *bitmap display*. Para as animações dessas colisões com esses objetos, separamos cada *sprite* que seria carregado e adicionamos ao programa. Entre as apresentações dessas figuras, colocamos um *time sleep*, ou seja, antes de realizar a próxima instrução, o jogo fica estático por uma determinada quantidade de tempo, para que as imagens possam ser carregadas uma a uma, criando a impressão de movimento.

2.4 Obstáculos

No jogo existem diversos obstáculos (Figura 1), dentre eles podemos citar: poça de óleo no meio da pista, outros carros e caminhões, que aparecem em posições variadas e em momentos aleatórios. Para simular isso, foi criada na memória uma *label* que guarda diversas informações sobre determinados inimigos que aparecerão em uma

fase, por exemplo: tipo de obstáculo, momento em que aparece e a posição em que será mostrado.

Desse modo, durante toda a execução do programa, o jogo percorre essas informações determinando a hora de um obstáculo aparecer ou não aparecer. O que gerou alguns problemas, pois esses objetos estariam se movimentando juntamente com a pista. Para resolvê-los, foi criada uma *label* que atualiza a posição do inimigo a cada animação da pista de fundo. Assim, foi possível criar a percepção de movimento sem perder as informações necessárias para conferir se o jogador colidiu ou não.

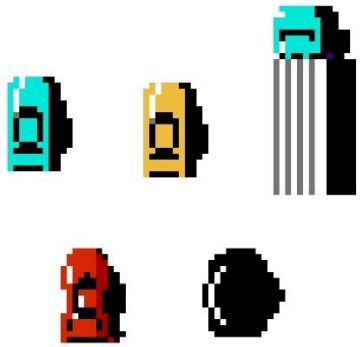

Figura 1 - Obstáculos

A dinâmica de diferenciação dos inimigos foi feita de forma que, ao reconhecer o tipo de obstáculo, o jogo automaticamente redireciona o tipo de reação caso haja a colisão. Desta forma foi possível criar dois tipos resposta do programa dependendo do inimigo que está sendo mostrado em determinado momento.

2.5 Música e efeitos sonoros

Para implementar a música e os efeitos sonoros, usamos uma estrutura parecida com a dos obstáculos. Criamos uma *label* na memória relacionada a quantidade definida de notas e suas respectivas altura e duração. No decorrer do jogo, o programa percorre essas informações, definindo qual nota deve ser tocada em dado momento.

2.6 Níveis

A segunda fase (Figura 2) do jogo foi a última coisa a ser implementada do jogo. Para isso criamos um código com a exata mesma estrutura do primeiro, adicionando apenas algumas modificações como: a troca da imagem do cenário ao fundo, modificamos a quantidade de obstáculos e aumentamos o tamanho do trajeto percorrido pelo jogador, ou seja, o programa terá que percorrer uma maior quantidade de linhas antes da linha de chegada.

Figura 2 – Níveis Diferentes

3. Resultados

O maior obstáculo que encontramos durante o desenvolvimento do jogo certamente foi a linguagem utilizada. Esse foi nosso primeiro

contato com uma linguagem de tão baixo nível, logo tivemos alguns problemas para entender o funcionamento dela e a lógica de desenvolver um jogo.

Outro problema encontrado foi a conversão de imagens de fundo transparente. Ao passar a imagem para o formato BMP, os pixels transparentes se transformavam em branco, o que complicou o processo de conversão da imagem. No final foi necessário converter tais pixels no próprio documento *.data*.

Apesar das dificuldades, conseguimos obter um bom desempenho na criação e implementação do projeto, pois a releitura do jogo ficou extremamente similar ao jogo original. Atingimos a maioria dos objetivos requeridos, conseguimos resolver a maioria dos problemas enfrentados e estamos satisfeitos com os resultados finais.

4. Conclusão

Esse artigo apresentou toda a metodologia, o funcionamento e as dificuldades enfrentadas na criação do jogo "Road Fighter" em assembly Risc-V.

O projeto de desenvolvimento do jogo foi fundamental adquirir conhecimento na linguagem de programação assembly, e nos desafiarmos a implementar esse conhecimento usando o simulador Risc-V.

Apesar de utilizarmos dinâmicas diferentes das do jogo original utilizado como base, acreditamos que o projeto final superou as expectativas.

Dito isso, também admitimos que existem pontos que possamos melhorar com um pouco mais de tempo e experiência com a linguagem. Dentre tais aspectos, podemos citar:

- Mais elementos nos cenários;
- Dinâmica de pontuação do jogador;
- Consertar alguns *bugs* de imagem e de dinâmica do jogo;
- Adicionar outros carros e um sistema de "perder" a corrida.

Referências

Melhores Projetos – RISC-V. Disponível em <https://aprender3.unb.br/mod/folder/view.php?id=604312> Acesso em 10 de abril de 2022.

Monitorias dos Ex-Monitores: Ana Sofia e Victor Lisboa. Disponível em <https://www.youtube.com/watch?v=mBmOkyqejHU&list=PLL0Kob75DU32afhLBN5nY2KzOJ5k6lw-Q&index=2&ab_channel=VictorLisboa:> Acesso em 13 de abril de 2022.

Monitorias dos Ex-Monitores: Ana sofia e Victor Lisboa. Disponível em: <https://web.microsoftstream.com/channel/4333b3d8-9904-4e46-bf1d-dcb73f980368> Acesso em 12 de abril de 2022.