Paródia - Road Fighter

Gustavo M. Tonnera

Universidade de Brasília, Departamento de Ciências da Computação, Brasil

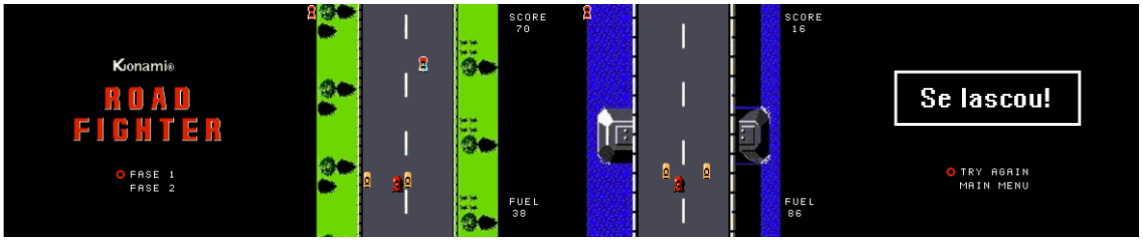

Figura 1: Imagens do jogo produzido

Resumo

Como Projeto Aplicativo da disciplina de Introdução aos Sistemas Computacionais (ISC), foi proposto que os estudantes fizessem uma versão do jogo Road Fighter em Assembly RISC-V. Esse trabalho apresenta as como o autor desenvolveu sua versão.

Palavras-chave: Assembly RISC-V, Road Fighter.

1 Introdução

Road Fighter é um jogo lançado em 1984 pela Konami que consiste no jogador controlar um carro com o objetivo de vencer a corrida. Cada fase do jogo é uma corrida diferente, na qual o jogador deve superar obstáculos, como possas de óleo e outros carros, para alcançar seu objetivo. O jogo apresenta mecânicas únicas: é necessário manter um botão apertado para o carro continuar acelerando, ao bater em um carro ou passar por uma possa de óleo- o carro do jogador começa a derrapar- e alguns carros inimigos se movimentam a fim de dificultar a passagem do jogador. Além disso, a única condição de vitória do jogo é se o jogador conseguiu alcançar o final da corrida, enquanto a de derrota é se ele ficou sem combustível antes de chegar ao final.

Figura 2: Imagem do jogo original

Em função de algumas limitações da plataforma usada para o desenvolvimento do jogo, algumas mecânicas presentes no jogo original tiveram que ser descartadas ou substituídas por outras. Além disso, no intuito de deixar o jogo mais dinâmico e difícil, as condições de vitória e derrota foram modificadas.

A versão desenvolvida inclui duas fases e gráficos baseados no jogo original, mas com música e efeitos sonoros diferentes. Ler as próximas sessões para saber a metodologia e os resultados obtidos.

2 Metodologia

Para a produção da versão de Road Fighter, foi utilizada a linguagem de programação Assembly RISC-V, o Visual Studio Code escrever o código, o Rars15 Custom1 para debugging e o FPGRARS (Fast Pretty Good RISC-V Assembly Rendering System) para executar o programa.

3 Resultados Obtidos

Primeiramente, o código final do projeto é extremamente ineficiente, já que apresenta diversas funções que executam a mesma tarefa, mas com objetos diferentes. Por exemplo, existem uma função que desenha o jogador na tela e outra que desenha a imagem de fundo, ambas desenham imagens na tela, mas desenham imagens diferentes. Isso provavelmente aconteceu devido a falta de experiencia do autor. Entretanto, a ideia do projeto é desenvolver as habilidades dos estudantes tanto em programação, como na linguagem Assembly RISC-V e não desenvolver a melhor versão possível de Road Fighter. A seguir estão descritas as estratégias utilizadas pelas autor para o desenvolvimento do projeto.

3.1 Desenhar imagens na tela

A ISA (Instruction Set Architecture) RISC-V utiliza MMIO (Memory-mapped I/O) para acessar dispositivos de entrada e saída. Portanto, basta armazenar os pixels da imagem que se quer desenhar em endereços específicos para ela ser desenhada na tela. Além disso, essa ISA suporta a construção de dois frames ao mesmo tempo, frame 0 e frame 1. No entanto, só foi utilizado o frame 0 para desenhar as imagens.

Dessa maneira, basta implementar um for loop que comece no endereço inicial do primeiro pixel da

imagem e até o endereço do último pixel e ir armazenando o valor de cada pixel no endereço certo da memória VGA, somando ao endereço da memória VGA a largura da tela e subtraindo esse resultado pela largura da imagem para pular para a próxima linha da tela. Essa estrutura foi empregada em todas as funções que desenham imagens na tela.

3.2 Acelerar o carro

Observando as imagens de fundo do jogo original, percebe-se que essas imagens são desenhadas repetidas vezes em sequência com uma distorção: a imagem é divida em duas partes e essas partes trocam de lugar. Isso faz com que o jogador tenha a impressão de que o carro está se movendo, mas na realidade é a imagem de fundo que está "se movendo".

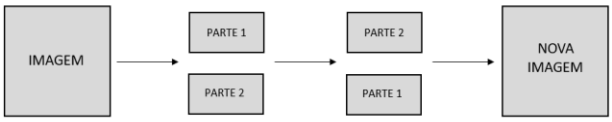

Figura 3: Ilustração da estratégia para acelerar o carro

Essa mesma estratégia foi adotada na versão desenvolvida, uma vez que possibilita um menor uso de imagens para a confecção do projeto.

3.3 Adquirir o Input do jogador

Para adquirir o input do jogador, foi utilizada uma função bem simples presente nos arquivos de exemplos apresentados pelo professor Marcus Vinicius Lamar, professor da matéria de ISC, a qual carrega da memória o valor ASCII (American Standard Code for Information Interchange) da tecla pressionada em um registrador.

```
GET_INPUT:
    li t1, 0xFF200000
    lw t0, 0(t1)
    andi t0, t0, 0x0001
    beq t0, zero, fim_get_input
    lw a1, 4(t1)
    fim_get_input:
        ret
```

Figura 4: Função responsável por adquirir o input

Dessa forma, podemos comparar esse valor com os valores de determinadas teclas para administrar os inputs.

3.4 Administrar o Input do jogador

Como citado na sessão anterior, para interpretar os inputs do jogador, basta comparar os valores ASCII das teclas pressionadas com os das teclas que fazem o carro se mover, os que fazem o ícone dos menus se moverem e o da tecla que confirma opções dos menus.

Para esse projeto, como as setas do teclado não possuem código ASCII, foram utilizadas as teclas "a" e "d" para a movimentação horizontal do jogador, as teclas "w" e "s" para a movimentação do ícone nos menus e a tecla "space" para a confirmação de opções dos menus.

3.5 Movimentação do jogador

No jogo original, a movimentação do jogador é limitada pelas extremidades da rua, se o jogador bater em alguma das extremidades o carro explode. Essa mecânica foi mantida na nova versão, uma vez que é uma das mecânicas mais fundamentais do jogo. Além deixar o jogo mais difícil, o movimento horizontal do carro foi diminuído na nova versão.

Figura 5: Carro explodindo (Jogo original)

3.6 Inimigos

No jogo original, existem vários tipos de carros inimigos, determinados pelas cores dos carros, entretanto, eles sempre aparecem nos mesmos lugares, o que deixa o jogo monótono. A fim de deixar o jogo mais dinâmico, na nova versão os carros aparecem de tempos em tempos em números aleatório entre dois e quatro em posições aleatórias. Dessa maneira, cada tentativa em cada fase do novo jogo é uma partida diferente da anterior. Outra diferença é que no novo jogo há apenas um tipo de inimigo, o carro amarelo.

Figura 6: Inimigos no novo jogo

3.7 Gasolina

Na nova versão do jogo, a gasolina funciona com o mesmo princípio do original: quando o jogador coleta a gasolina, ele aumenta em uma determinada quantidade o combustível restante. A gasolina também apresentava o mesmo problema dos carros inimigos no jogo original, o qual foi corrigido da mesma maneira, porém com um tempo diferente do dos carros.

3.8 Colisões

Para detectar as colisões com os inimigos e com a gasolina no novo jogo, a memória VGA do frame 0 foi utilizada como uma matriz. Essa começa no endereço 0xFF000000 e acaba no endereço 0xFF012C00 para a resolução usada no projeto (320 x 240 pixels), dessa maneira podemos representar a posição de um pixel em coordenadas (x, y), onde o x varia de 0 a 319 e o y de 0 a 239. Armazenando a coordenada do pixel superior esquerdo de um objeto, ou seja, o primeiro pixel a ser desenhado, podemos usá-la para encontrar as coordenadas de outros três pixels, usando o comprimento e a altura da sprite, de maneira que os quatro pixels formem um quadrilátero. Esse quadrilátero é chamado de "hit box" e é usado para detectar as colisões em jogos.

Dessa maneira, uma colisão ocorre quando: $x_0 \leq a \leq x_1$ e $y_0 \leq b \leq y_1$, sendo (x_0, y_0), (x_1, y_0), (x_0, y_1) e (x_1, y_1) as coordenadas dos pixels superior direito, superior esquerdo, inferior esquerdo e inferior direito, respectivamente, e (a, b) um pixel de outra sprite.

Para detectar as colisões com as extremidades da rua, as imagens de fundo das fases foram produzidas de forma que as extremidades em mesmas coordenadas x. Assim, basta comparar a posição x do jogador com essas coordenadas x para todas as fases.

Figura 7: Hit box do jogo League of Legends

3.9 Condições de vitória

No jogo original, o jogador precisa alcançar o final da fase para ganhar. Na nova versão, o objetivo do jogo foi levemente alterado: em vez de passar das fases, o principal objetivo do jogador é alcançar a maior quantidade de pontos durante a partida. Quanto mais o jogador avançar na fase, mais pontos ele ganha. Essa mudança foi feita para adequar o jogo ao novo modelo de geração de inimigos e combustível.

Além disso, foi implementado um número máximo de vezes que o jogador pode colidir com os outros carros, o qual depende da fase, a fim de deixar o jogo mais complicado.

3.10 Efeitos sonoros, música e imagens

As imagens de fundo usadas para fazer as fases foram produzidas usando as imagens das duas primeiras fases do jogo original. As sprites dos carros e da gasolina usadas são as do jogo original. A música tocada durante as o jogo é Tokyo Drift os efeitos sonoros de batida e explosão foram feitos usando o sintetizador de áudio do Rars15 Custom1.

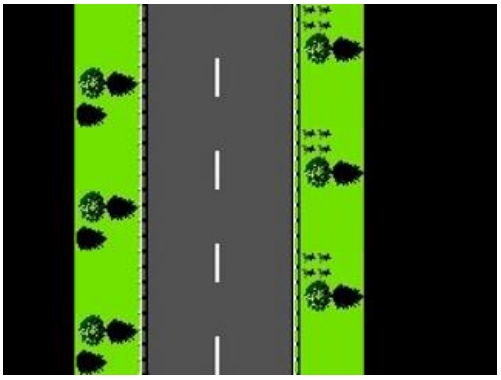

Figura 8: Imagem de fundo da fase 1

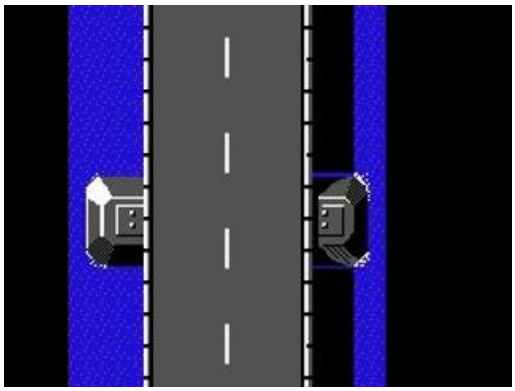

Figura 9: Imagem de fundo da fase 2

Figura 10: Sprites usadas

3.11 Dificuldades

Com certeza a maior dificuldade no desenvolvimento do projeto foi o debugging do código. Apesar do Rars15 Custom1 possuir ferramentas que auxiliem nessa tarefa, essa IDE é muito lenta. Portanto, foi necessário criar arquivos de teste para testar as funções separadamente e depois ver como ela se comportava com o resto do código, o que torna o processo bem demorado.

Outra dificuldade foi a linguagem em si. Ainda que o professor Marcus Vinicius Lamar tenha ministrado aulas ensinando a sintaxe da linguagem, como programação precisa de prática para ser aprendida com proficiência, as primeiras semanas de desenvolvimento foram bem ineficientes.

Independentemente dessas dificuldades, passadas as primeiras semanas, o projeto fluiu sem muitas outras dificuldades, com exceção da função que verifica as colisões entre o jogador e os carros inimigos, a qual também gerou certas dificuldades para ser desenvolvida.

4 Conclusão

Dessa forma, o projeto conseguiu cumprir com seu objetivo: desenvolver habilidades de programação e conhecer a linguagem Assembly RISC-V. Foi feita uma versão do jogo Road Fighter com algumas alterações nas mecânicas de movimento, geração de inimigos e da gasolina, nas condições de vitória/derrota e no objetivo geral do jogo, apesar das dificuldades citadas na sessão 3.11.

5 Referências

Aulas e slides do professor Marcus Vinícius Lamar;

Monitoria dos monitores: Ruan, Eduardo e Leandro;

Jogo original.