UNIVERSIDADE DE BRASÍLIA

DEPARTAMENTO DE CIÊNCIAS DA COMPUTAÇÃO

Prision Corrupt – Um jogo desenvolvido em Assembly RISC-V

Professor: Marcus Vinicius Lamar

Nome: Daniel de Oliveira Morais

Nome: Rafael Mileo Moreira Krauss Guimarães

Nome: Ricardo Pedrosa Ramos Filhos

Resumo

O documento a seguir tem como objetivo detalhar o processo de desenvolvimento, dificuldades encontradas, história e o resultado final do projeto Prision Corrupt, que é uma releitura do jogo Fix-It Felix Jr, feito para a disciplina de Introdução aos Sistemas Computacionais, ministrada pelo professor Marcus Vinicius Lamar.

1. Introdução

O jogo fictício Fix-It Felix Jr, em sua história, foi lançado em 1982 pela empresa de jogos indie TobiKomi, foi um dos jogos de plataforma 2D mais populares já feitos, por sua jogabilidade viciante, gráficos coloridos e gameplay simples. No jogo, o vilão Ralph destrói as janelas de um prédio, e o herói Felix Jr. deve consertá-lo usando seu martelo mágico enquanto evita obstáculos jogados por Ralph e objetos no mapa. O jogo tem uma jogabilidade retrô, semelhante a *Donkey Kong*, aumentando a dificuldade conforme se avança os níveis dentro do jogo.

2 Metodologia

Nosso jogo foi feito utilizando o simulador RARS, linguagem de programação Assembly, na ISA RISC-V e o FPGRARS para rodar o jogo e testá-lo conforme o desenvolvimento do código. Para comunicação utilizamos principalmente o Discord, onde também era feito o versionamento do código, além de ser postado no GitHub.

2.1 Mapas

Nosso jogo se passa em uma prisão, onde os inimigos abrem suas celas e o jogador tem que fechá-las; geramos o sprite com o Dall-E 3 e fizemos ajustes conforme necessitamos no paint.net, as colisões ocorrem nas bordas da prisão de onde não é possível escapar, a colisão é feita no momento da movimentação, ao pressionar uma tecla, o código verifica se a posição (x, y) seria válida (dentro das grades), caso não seja válida, o movimento não acontece. A imagem a seguir mostra um pouco de uma das fases do jogo.

Figura 1- Representação da FASE 2 do jogo

2.2 Personagens

Existem dois tipos de personagem no jogo, o personagem principal, Capitão Nascimento, que no jogo original seria o Felix Jr, e os prisioneiros que aparecem nas portas, utilizando facas. Os sprites dos personagens foram feitos manualmente no paint.net, utilizando como modelo alguns sprites já existentes.

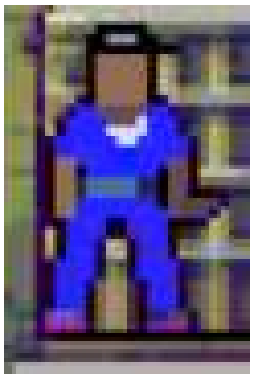

Figura 2 - Policial

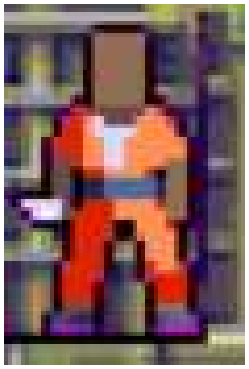

Figura 3 - Prisioneiro

3. Funcionamento do jogo

A jogabilidade do nosso jogo é simples, utilizam-se as teclas AWSD para movimentação do jogador (A-Esquerda, W-Cima, S-Baixo e D-Direita), com a tecla 'F' se desfere o ataque e pode-se eliminar os prisioneiros, equivalente a consertar as janelas no jogo original. Para eliminar os prisioneiros basta atacá-los duas vezes, os prisioneiros possuem uma faca, que utilizam para atacar o jogador, que possui 4 vidas e cada vez que ele recebe um ataque de um dos prisioneiros, perde uma vida. Também existem balas de canhão que caem do céu durante a gameplay, cada vez que o jogador é atingido pela bala de canhão também perde uma vida. Quando o jogador perde todas as vidas, uma tela escrita "Corrupted" aparece, indicando que o jogador perdeu o jogo.

Ao eliminar todos os prisioneiros da fase, aparece uma tela de "Floor completed" indicando o êxito do jogador em completar o nível, e após pressionar a tecla espaço, o jogo inicia a próxima fase. Caso o jogador vença todas as fases, aparece uma tela escrita "Congratulations", indicando que o jogador zerou o jogo.

4. Dificuldades

No processo de desenvolvimento do jogo, ocorreram algumas dificuldades, programar utilizando uma linguagem até então desconhecida por nós e de baixo nível, acarretou na necessidade de um tempo de adaptação e amadurecimento de ideias, que pode ser observada no próprio código do jogo, a seguir listaremos algumas das dificuldades encontradas para a conclusão do nosso projeto.

4.1 Gerenciamento de registradores

O número limitado de registradores no Assembly RISC-V comparado as linguagens de programação de mais alto nível na qual estávamos acostumados, acarretou em um fator a mais para se preocupar durante a produção do código, ter que se atentar a isso enquanto tenta fazer um código funcional é algo que necessitou muito cuidado e revisão de código por erros que aconteciam por causa do número reduzido de registradores e acarretava em problemas cuja causa não era muito clara no primeiro momento. Várias lógicas foram tentadas para contornar esses problemas, lógicas diversas essas que estão presentes no código. No final das contas foi percebida a possibilidade de guardar os valores dos registradores no "negativo" do registrador sp, retornando-os aos valores originais após a execução da função, o que facilitou o

processo de programação e abstraiu a maioria dos problemas com a quantidade limitada de registradores.

4.2 Exibição correta dos sprites na tela

Durante toda a produção do código, ocorreram diversos problemas envolvendo a exibição dos sprites na tela, várias soluções foram aplicadas para resolver problemas como o apagamento dos rastros dos personagens e projeteis, entre elas podemos citar o armazenamento de variáveis de posições anteriores do personagem principal e da bala, assim podemos remover os rastros anteriores e garantir um aspecto orgânico para o jogo. Também pode se destacar a utilização de uma função para implementar o ataque do jogador seguido da morte do prisioneiro na mesma cela em que o jogador se encontra, que foi complicado pois existiam diversos objetos a se considerar dentro da cela, e a ordem que eles eram "printados" no loop teve que ser ajustada especificamente nesse caso.

5. Conclusão

Após um longo processo de desenvolvimento e aprendizado, concluímos o desenvolvimento do jogo com êxito. Tal jornada foi de extrema importância para a evolução e melhor compreensão do que de fato é a área de programação e as diversas aplicações que ela tem.

Ao longo da produção do código, pudemos observar de perto, graças aos conhecimentos obtidos em sala de aula, como a linguagem de programação de baixo nível (nesse caso o Assembly com a ISA RISC-V) é diferente das demais linguagens a quais estamos acostumados. Além disso, diversos aprendizados sobre o funcionamento de um processador de 32 bits foram obtidos.

6. Bibliografia

Repositório Victor Hugo no GitHub

VICTORLISBOA. **GitHub - victorlisboa/LAMAR: LAMAR: Learning Assembly for Machine Architecture and RISC-V**. Disponível em: <https://github.com/victorlisboa/LAMAR>. Acesso em: 03 de janeiro de 2025.

Geração e edição de sprites

OPENAI. **DALL·E 3**. Disponível em: <https://openai.com/index/dall-e-3/>. Acesso em: 23 de dezembro de 2024.

2D Sprites - CraftPix.net. Disponível em: <https://craftpix.net/categorys/sprites/?srsltid=AfmBOoqxw1XjC-BrAvHSqMP4YZOS1QEZ5Wm58BHvr3LYzT4rRifBw_-8>. Acesso em: 23 de dezembro de 2024.

BREWSTER, R. **Paint.NET - Free Software for Digital Photo Editing**. Disponível em: <https://www.getpaint.net/#google_vignette>. Acesso em: 6 de janeiro de 2025.

Site para pegar as músicas

Hooktheory: Create amazing music. Disponível em: <https://www.hooktheory.com/>.Acesso em: 2 de fevereiro de 2025.