Fix-It Felix Jr.

Luis Henrique Da Silva, Ana Sarah Nagata, Kamily Silva Andrade Crispim

Departamento de Ciência da computação

Universidade de Brasília (UnB) - Brasília, DF - Brasil

luishenriquen14@gmail.com, anarachan@gmail.com, kamilyunb@gmail.com

Resumo. *Este artigo aborda o desenvolvimento de um projeto utilizando a linguagem de programação Assembly na arquitetura RISC-V, tendo como base o jogo "Conserta Felix" (Fix-It Felix Jr.), criado pela Disney. O estudo visa explorar as técnicas de programação de baixo nível e a adaptação do jogo clássico para a arquitetura RISC-V.*

1. Introdução

O jogo "Conserta Félix" foi desenvolvido pela Disney Interactive Studios em 2012 com a finalidade de promover o filme "Detona Ralph" (Wreck-It Ralph). No jogo, o jogador assume o papel de Félix, um herói que deve consertar as janelas de um edifício danificadas pelo vilão Ralph, utilizando seu martelo mágico, enquanto desvia dos tijolos arremessados pelo inimigo. A experiência de jogar "Conserta Félix" oferece uma imersão nostálgica para os fãs do filme, proporcionando uma conexão entre a narrativa cinematográfica e o universo dos jogos de arcade.

2. Metodologia

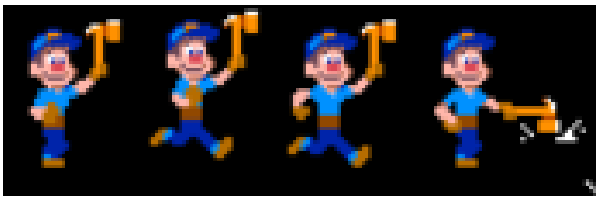
2.1 - Ferramentas utilizadas

O projeto foi desenvolvido utilizando o RARS (RISC-V Assembly Runtime Simulator), um simulador de código Assembly com arquitetura RISC-V, que se mostrou fundamental para a depuração do código. Algumas ferramentas do RARS foram essenciais, como o *Bitmap Display*, que serviu como interface gráfica, e o *Keyboard MMIO*, responsável pela interface com o teclado. Além disso, foi utilizado o FPGRARS, um simulador similar ao RARS, mas com desempenho muito superior e otimizado. O FPGRARS foi especialmente empregado para rodar o jogo e realizar testes de desempenho. A implementação do áudio também foi possível graças ao uso da interface MIDI. Para a manipulação dos sprites do jogo, o aplicativo *Paint.NET* foi utilizado, juntamente com um script de conversão de imagem, essenciais para o processo. Todas

essas ferramentas foram utilizadas com base em tutoriais fornecidos por ex-alunos da disciplina de ISC, bem como em repositórios disponíveis no GitHub.

2.2 - Personagem

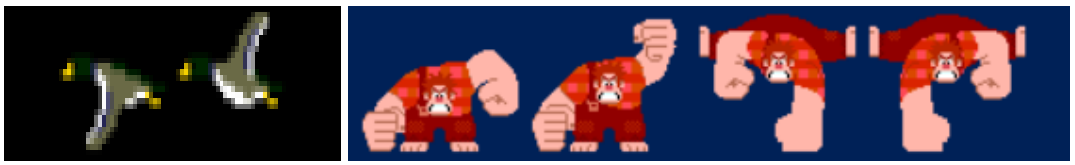
Para o personagem Félix foram utilizadas quatro sprites em formato .data, com tamanhos dependentes da ação executada. A imagem é invertida conforme a direção do personagem, determinada pelas entradas do teclado, sendo, “A” para esquerda, “D” para direita, “W” para cima e “S” para baixo. Além do “P”, que foi um cheat implementado para a mudança de fase. O objetivo principal do jogo é guiar Felix, usando o teclado, para consertar as janelas enquanto evita perder suas três vidas para os obstáculos, do contrário, o jogador perde. O conserto das janelas é realizado ao pressionar a tecla L, o que incrementa a pontuação do jogador. Na fase 2, Félix tem a oportunidade de acessar uma torta que lhe concede power-up, tornando-o imune por alguns segundos.



Félix, o protagonista.

2.3- Obstáculos

O vilão Ralph, assim como Felix, possui várias sprites em .data. Sua movimentação é aleatória, possibilitada pelo uso da ecall 41 (randint). Para decidir a posição em que joga-se os tijolos, é verificado que, sempre a animação do Ralph termina de ser executada, a mecânica de criar uma tijolo na mesma coordenada x,y de Ralph é ativada, fazendo-o cair ao longo do eixo vertical. Além disso, durante a segunda fase, é introduzido um novo obstáculo: o pato. Colidir com o pato resulta na perda de uma vida. A movimentação da ave segue um padrão definido.

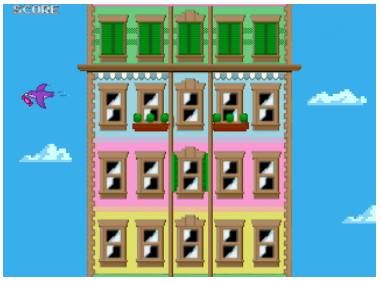


Vilões do jogo.

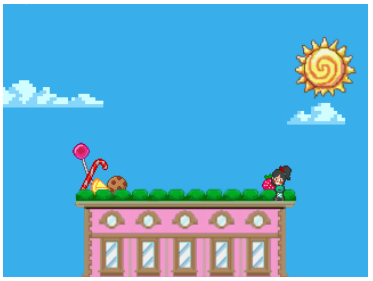
2.4 - Mapas e colisões

Nosso mapa segue uma proposta semelhante à do jogo original *Fix-It Felix*, que se passa em um prédio residencial. No entanto, em nossa versão, o edifício e seus arredores adotam um tema doce, inspirado no universo do *Reino Doce*, o cenário central do filme *Detona Ralph*. Essa escolha tem o objetivo de fazer uma alusão ao ambiente colorido e encantado do filme, que, por sua vez, serviu de inspiração tanto para o jogo original quanto para nossa adaptação. A ilustração do mapa foi criada pela artista independente Rebeca Marques.

fase 1



fase 2



terraço

As colisões no jogo foram baseadas nas coordenadas de cada janela do prédio. O jogador só consegue se deslocar para pontos específicos do mapa, pressionando as teclas de movimentação. Ao fazer isso, é realizado um cálculo que leva em consideração as posições X e Y do personagem e das janelas. Em seguida, verifica-se qual é a janela mais próxima na direção em que o personagem deseja se mover. Se não houver nenhuma janela nessa direção, a colisão é ativada e o Félix não se move. Além disso, existe a colisão com os tijolos, que caem ao longo do eixo Y. Essas colisões também são verificadas por cálculos baseados nas coordenadas do jogador e causam dano ao personagem, fazendo-o perder uma vida a cada impacto. A colisão com a torta, presente na fase 2, é calculada de forma semelhante, mas com uma mecânica um pouco mais complexa. No código, há uma verificação adicional para determinar se a torta está ou não visível na janela. Ao colidir com a torta, o personagem ganha alguns segundos de imunidade aos tijolos, e sua aparência muda, exibindo o rosto do professor Marcus Vinicius Lamar.

2.5 - Música

Através da *ecall* 31 do RARS, é possível gerar notas musicais por um período específico, com uma ampla seleção de instrumentos disponíveis. Para a reprodução de músicas completas, foi implementado um procedimento que lê as informações de notas e durações de uma *label*, tocando uma nota de cada vez, respeitando os intervalos entre

elas. A *ecall* 30 é utilizada para comparar os timestamps da execução atual do *game loop* com o momento em que a última nota foi tocada, garantindo que as notas sejam reproduzidas na ordem correta.

Para possibilitar a reprodução simultânea de efeitos sonoros, foi criado um script baseado em estados, onde cada efeito é representado por uma *label* lida a cada novo frame do *game loop*. Em cada *label*, são armazenadas informações essenciais, como o estado do efeito sonoro (1 para reproduzido, 0 para não reproduzido), o índice da nota atual, o timestamp da última nota tocada, a duração da nota, o volume e o instrumento utilizado. Para iniciar um efeito sonoro, basta alterar o estado do efeito desejado para "1" e reiniciar os contadores e índices no código. Na próxima execução do procedimento, o efeito será tocado.

2.6 -Game Loop

O game loop começa tocando os efeitos sonoros. Em seguida, verifica se há uma cutscene ativa e, caso haja, a executa. Depois, analisa se a animação de movimentação do personagem está ativada e, se estiver, chama a função correspondente. Alterna os frames do personagem e carrega o background correto conforme a fase atual. Em seguida, imprime as janelas e verifica se a torta deve ser criada ou exibida na fase 2, além de testar colisões com Félix. Depois disso, movimenta o Ralph, verifica sua animação e imprime seu sprite, alternando entre estado inativo e animado. O jogo então imprime o personagem Félix com o sprite correto, considerando se ele está parado ou pulando, e verifica se ele está invencível devido ao power-up da torta. Se estiver, imprime um efeito visual e testa o tempo restante da invencibilidade. Em seguida, verifica se o controle de patos deve ser ativado na fase 2. O loop então atualiza o frame na memória, desenha as vidas, imprime o número da fase e testa colisões entre Felix e os tijolos. Logo após, exibe a pontuação do jogador e, por fim, captura as teclas pressionadas antes de reiniciar o ciclo.

3 -Resultados Obtidos

Mesmo diante de uma proposta desafiadora e de um longo período de dedicação ao desenvolvimento do jogo, conseguimos concluir o trabalho atendendo a todas as especificações. Nossa trajetória foi marcada por desafios, especialmente devido à complexidade da linguagem de baixo nível, mas contamos com o suporte essencial dos monitores, que foram fundamentais para nosso sucesso.

Um dos problemas enfrentados foi o aparecimento inesperado de tijolos em posições erradas na tela, causado por um contador mal gerenciado que acessava uma região indevida da memória. Felizmente, conseguimos identificar e corrigir esse erro. Além disso, a organização foi um fator crucial para o gerenciamento eficiente dos registradores e a manutenção da legibilidade do código.

Também tivemos que corrigir algumas inconsistências nos saltos entre funções, pois o comando ret nem sempre se comportava conforme o esperado. O tempo foi outro fator determinante no processo de tomada de decisões, pois, apesar do desejo de implementar mais elementos ao jogo, o prazo nos levou a optar por uma versão mais simplificada, porém totalmente funcional.

tijolos aparecendo no lugar errado.

6. Conclusão

Neste artigo, foi apresentado o processo e o resultado do projeto desenvolvido na disciplina de Introdução aos Sistemas Computacionais (ISC). Aplicamos os conceitos estudados em sala de aula, utilizando a linguagem de baixo nível Assembly RISC-V para a implementação do jogo.

7. Referências

-Github com material de apoio da matéria de ISC

https://github.com/victorlisboa/LAMAR?tab=readme-ov-file

-Playlist de videoaulas do professor Marcus Vinícius Lamar

https://www.youtube.com/playlist?list=PLALgy7vIdqUb5INNKsUwkw6Avpo-h_qpm

-Playlist do ex-aluno Thales Menezes

https://www.youtube.com/playlist?list=PLL0Kob75DU32afhLBN5nY2KzOJ5k6lw-Q

-Jogo Fix-it Félix

https://online.oldgames.sk/play/genesis/fix-it-felix-jr/10107

-Site usado para a música
https://www.hooktheory.com/