Adventures of Pikachu - Jogo inspirado em Adventures of Lolo

Ana Luísa Padilha Alves Bruno Vargas de Souza Harisson Freitas Magalhães

Universidade de Brasília - UnB, CIC, Brasil

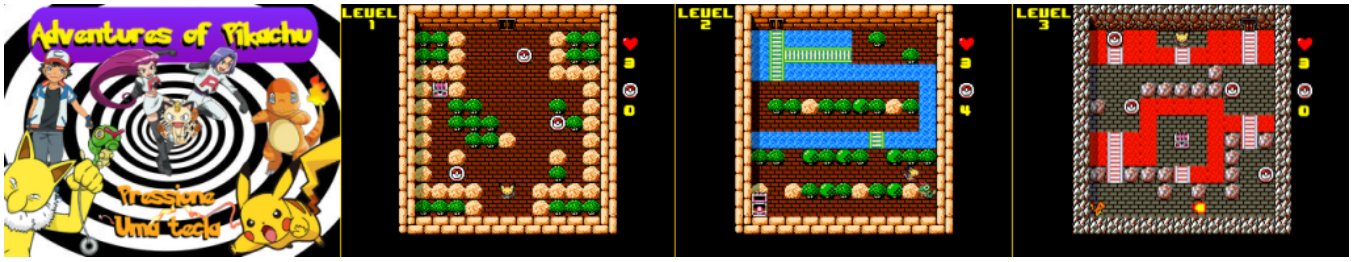

Figure 1: Fases do jogo

ABSTRACT

Este artigo, aborda de forma detalhada o desenvolvimento e criação do jogo Adventures of Pikachu, inspirado em Adventures of Lolo, o qual foi criado em Assembly, uma linguagem de baixo nível, trazendo os resultados e dificuldades obtidos ao longo do processo.

O projeto foi criado com o propósito de colocar em prática o que foi aprendido na disciplina de Introdução aos Sistemas Computacionais usando o RISC-V.

Palavras-chave: Risc-V, RARS, Bitmap , MMIO, MIDI, Pikachu, Matriz, Tiles, Adentures of Lolo, Pokémon, Frames, Assembly.

1 INTRODUÇÃO

Adventures of Lolo foi lançado em 1989 e desenvolvido pela HAL Laboratory para o Nintendo Entertainment System (NES). Trata-se de um jogo de lógica, tendo sido considerado um ótimo jogo para o desenvolvimento cognitivo das crianças. O jogo também faz parte da série Eggerland, constituída por vários jogos de quebra–cabeça. Na história original, Lala, a namorada de Lolo, é raptada pelo grande vilão e o objetivo do jogo é concluir todas fases e salva-la.

Já Pokémon, é uma franquia de mídia que pertencente a The Pokémon Company. Foi criado em 1995 por Satoshi Tajiri, se tornou uma febre nos anos 90 e até os dias de hoje faz um enorme sucesso.

Nossa ideia então foi juntar os dois mundos, criando o jogo "Adventures of Pikachu". Onde, após a Equipe Rocket sequestrar os pokémons, Pikachu tem o dever de avançar os mapas e enfrentar inimigos para salvar seus amigos.

2 METODOLOGIA

Para esse desenvolvimento, usamos o RARS (RISC-V Assembler and Runtime Simulator), que ajuda no processo de familiarização com a linguagem de baixo nível, mesmo com as limitações do mesmo. Dito isso, um dos principais problemas, foi a limitada quantidade de registradores.

2.1 Ferramentas

Usamos o **Bitmap Display** para a construção das imagens na tela, na qual ele sempre trabalha com duas frames, e criamos uma forma de manipular elas, sempre alterando entre uma e outra.

Outra ferramenta de extrema importância foi o **Keyboard and Display MMIO Simulator**, que nos auxiliou a implementar a movimentação do personagem, fazendo com que tenhamos uma interação entre o jogador e a interface.

Além dos citados, também foi usada a Interface de Áudio **MIDI** que foi responsável pela implementação dos efeitos sonoros e pelas músicas, e possui uma variedade de notas e instrumentos, podendo alterar o volume e a duração das mesmas.

Usamos o **PAINT.NET** para construir todas as nossas imagens personalizadas através da pixel art, fazendo pixel a pixel das animações.

2.2 Matriz

```
FASE3: .byte
        25, 20, 24, 24, 30, 30, 30, 24, 24, 28, 24,
        25, 28, 24, 24, 24, 28, 24, 24, 24, 28, 24,
        31, 30, 30, 30, 30, 30, 30, 26, 30, 30, 30,
        31, 30, 30, 26, 26, 26, 26, 26, 20, 30, 30,
        27, 30, 20, 24, 24, 24, 24, 24, 30, 30, 30,
        31, 30, 30, 24, 30, 30, 30, 24, 24, 28, 24,
        25, 28, 24, 24, 30, 21, 30, 24, 26, 28, 26,
        25, 28, 24, 24, 30, 30, 30, 24, 26, 30, 30,
        27, 28, 26, 26, 26, 28, 24, 24, 26, 30, 20,
        31, 30, 30, 30, 26, 30, 26, 30, 26, 30, 30,
        31, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
```

Figura 2: Matriz

Desenvolvemos uma forma de exibir os blocos desejados, e nas coordenadas adequadas, usando um sistema de matrizes que pega os blocos de uma imagem conforme a numeração do vetor e a posição do bloco no **Tiles**.

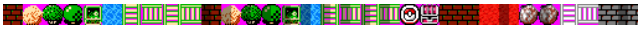

Figura 3: Tiles

Fizemos um sistema na memória de dados, que armazena todos os blocos onde o personagem não pode atravessar, calculando assim, a colisão para cada um deles.

2.3 Movimentação

Para executar o deslocamento do personagem pelo mapa, recorremos aos procedimentos que realizam a tarefa de mostrar o personagem na tela, e apagar logo em seguida, utilizando um mapa como base que armazena as coordenadas anteriores do personagem e exibe o fundo por cima dela.

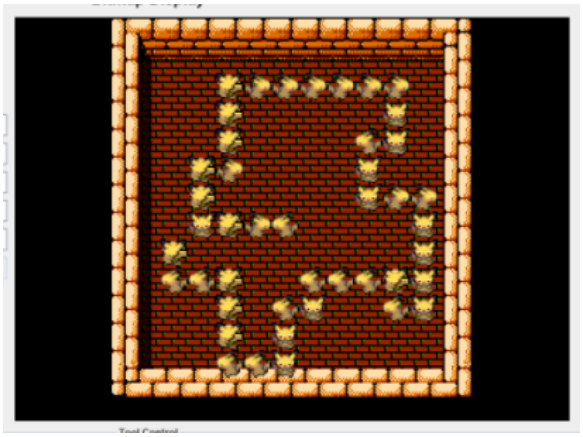

Figura 4: Antes da implementação

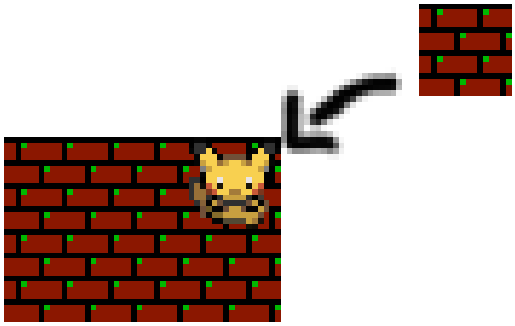

Figura 5: Usando a implementação

2.4 Inimigos

Na fase 2, fizemos com que o jogador utilize a paciência para passar pela fase, na qual executamos uma forma com que o inimigo somente apareça quando o personagem estiver perto dele, então é necessário a atenção.

Figura 6: Inimigo 1 - Caterpie

Depois, na fase 3, elaboramos uma forma com que o inimigo atire projéteis enquanto o personagem se move, de uma forma rápida e através de um temporizador entre um projétil e outro, na qual o jogador não pode ser atingido.

Figura 7: Inimigo 2 - Charmander

3 Resultados Obtidos

3.1 Jogo

Conseguimos implementar muito do que idealizamos no início, apesar de não termos conseguido fazer as 5 fases, estamos satisfeitos com os resultados obtidos.

Fizemos uma breve introdução sobre a história na abertura. Em seguida, inicia-se a primeira fase, simples e introdutória, para que o jogador tenha o primeiro contato com o jogo. Ao decorrer das fases, o nível de dificuldade vai se tornando maior, com a presença dos inimigos e a delimitação de deslocamento do personagem.

Conseguimos criar um sistema de vidas que sofre decremento cada vez que o personagem morre, ao zerar as vidas, é reproduzido na tela a imagem de Game Over, caso contrário, e o jogador consiga passar pelas fases, vai ser exibido na tela, uma imagem que o parabeniza por concluir a sua jornada.

Em ambos os casos, o jogo é reinicializado e o jogador tem uma nova chance para mostrar suas verdadeiras habilidades.

3.2 Dificuldades

Passamos por diversas dificuldades ao decorrer do jogo, que incluem a escassez de registradores, demora ao executar as instruções, o que resultou em alguns travamentos intensos do programa, e a difícil compreensão das músicas, já que as notas musicais atropelam umas as outras, e acaba tendo um resultado diferente do esperado.

Por escolher fazer uma personalização no jogo, tivemos que em alguns momentos desenvolver as nossas próprias animações e imagens, o que acabou tomando parte do nosso tempo.

Ao longo da construção do jogo, alguns erros foram aparecendo. Ao recarregar as fases, não estava sendo possível salvar a matriz na frame seguinte, pois o fundo acabava por pegar a frame contrária, o que resultava em um fundo desconexo.

Figura 8: Fundo Errado

E também obtivemos um erro ao tentar fazer a mudança de fases, por conta da instrução **Branch**, que consegue pular somente até 12 bits, e a melhor solução encontrada, foi a utilização de **Jumps** para fazer o intermédio das Labels.

4 Conclusão

Neste projeto, concluímos que o aprendizado da linguagem de baixo nível torna-se importante para conhecermos como o computador funciona, mas foi bastante difícil para nos acostumarmos no início, entretanto, com o tempo, conseguimos ir aprimorando nossas habilidades.

Ficamos muito felizes com o que conseguimos construir em conjunto, pois o grupo se empenhou bastante e cada um foi muito importante para o resultado final. Nossa relação evoluiu muito desde que começamos o projeto e a interação foi fundamental.

5 REFERÊNCIAS BIBLIOGRÁFICAS

$https://pt.wikipedia.org/wiki/Adventures_of_Lolo$
$https://pt.wikipedia.org/wiki/Pikachu$
Aulas do professor Marcus Vinicius Lamar
https://www.youtube.com/playlist?list=PLL0Kob75DU32afhLBN5nY2KzOJ5k6lw-Q