GAUNTLET RISC-KNIFE

Israel Honório de Castro Lucas Gabriel de Oliveira Lima Pedro Lucas Pereira Neris

University of Brasília, Dept. of Computer Science, Brazil

Resumo

Este artigo descreve o processo de desenvolvimento do *Gauntlet Risc-Knife,* uma versão do jogo *Gauntlet* feita e executada utilizando o simulador RARS (*Risc-V Assembler and Runtime Simulator*) para o projeto final da disciplina de Introdução aos Sistemas Computacionais (ISC) do curso de Ciências da Computação da Universidade de Brasília (UnB). O simulador RARS emula as funcionalidades da ISA (conjunto de instruções) RISC-V, que, por sua vez, possui licença livre. O RARS utiliza a linguagem de baixo nível Assembly, que é a linguagem de programação mais próxima da linguagem de máquina e, devido ao seu caráter gratuito e livre, é amplamente utilizado para fins educativos.

1. Introdução

Produzido e lançado em 1985 pela empresa Atari Games, *Gauntlet* é um jogo do estilo *arcade* de fantasia e *hack-and-slash*, onde o jogador deve passar pelo mapa no formato de um labirinto evitando obstáculos e derrotando inimigos até chegar a saída.

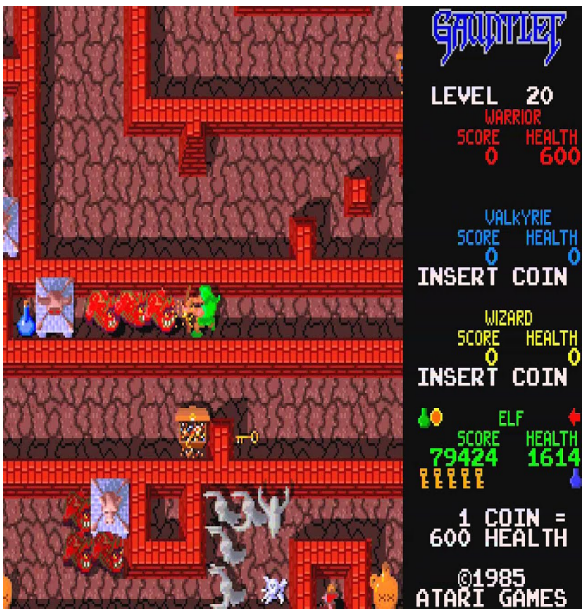

Figura 1: *gameplay* do jogo Gauntlet

4 tipos de personagem baseados no gênero de fantasia estão disponíveis: o guerreiro, o mago, a valquíria e o elfo, cada um com seus respectivos pontos fortes e fracos. Além disso, os inimigos também são baseados na fantasia, tendo fantasmas, demônios e a Morte. O jogo se destaca por ser um dos primeiros jogos em que havia a possibilidade do *multiplayer* (mais de um jogador ao mesmo tempo), suportando até 4 jogadores.

2. Metodologia

Gauntlet Risc-Knife foi desenvolvido e executado utilizando o RARS devido à facilidade de acesso (tanto o Windows quanto o Linux são compatíveis com o RARS, sendo necessário apenas instalar o Java) e familiaridade do grupo com esta ferramenta pelas aulas ministradas em ISC sobre RISC-V e *Assembly*. Além disso, o RARS também disponibiliza ferramentas para a exibição de imagens na tela (*Bitmap Display*) e para a reprodução de áudio (MIDI), o que é conveniente para o desenvolvimento de um jogo.

2.1 Música e efeitos sonoros

O áudio do jogo foi feito utilizando a ferramenta MIDI disponível no RARS. Para a música no começo do jogo, foi feita a tradução das notas da música presentes no *site* Hooktheory para o formato MIDI, utilizando um código Python disponibilizado no GitHub. A música escolhida foi "Viva la Vida", da banda Coldplay. Já os efeitos sonoros foram feitos utilizando a própria ferramenta de MIDI do RARS, que permite a saída em áudio de uma nota musical em um instrumento escolhido.

2.2 Design do jogo

Os mapas foram feitos tendo como base os mapas do jogo original de 1985, com algumas alterações feitas através do aplicativo 'Paint.net'. Já o design do personagem, dos inimigos e dos itens foi feito utilizando os *sprites* prontos do jogo original, sem alterações. Foi escolhido fazer apenas 1 personagem para maior praticidade.

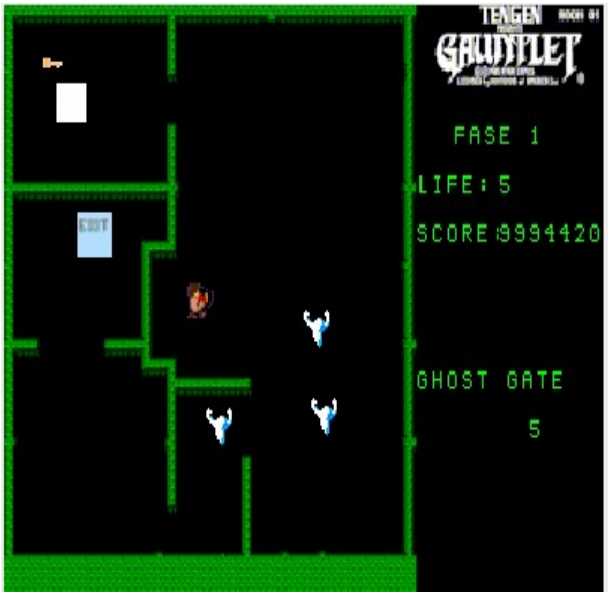

Figura 2: Mapa da fase 1 do jogo *Gauntlet Risc-Knife*

2.3 Posição e movimentação do personagem

O programa recebe um *input* do teclado utilizando uma chamada de sistema. Ao receber esse *input*, o programa busca o endereço deste *input* e realiza um *load* no valor do endereço. Em seguida, o programa realiza a checagem do valor do endereço: caso o valor seja "w", o personagem se movimenta para; caso seja "s", para baixo; caso seja "a", para a esquerda; e caso seja "d", para a direita. Ao processar o valor do input, o programa cobre a movimentação do personagem de

preto no frame 1, e no frame 0 ele atualiza a posição do personagem em 16 pixels na direção em que foi movimentado o personagem. Se não houver nenhum valor no endereço do *input*, a direção do personagem será estática.

2.5 Sistema de dano

Quando um inimigo se move, é chamada uma função que checa a posição dele em relação ao personagem, primeiro na horizontal, se estão na mesma 'linha', e em seguida na vertical, se estão na mesma 'coluna'. Caso estejam pareados iguais, ou caso haja alguma lateral de ambos os personagens que se tocam, utiliza-se uma manipulação, que obterá como resultado a informação se houve ou não o choque do inimigo com o personagem.

2.6 Tiro

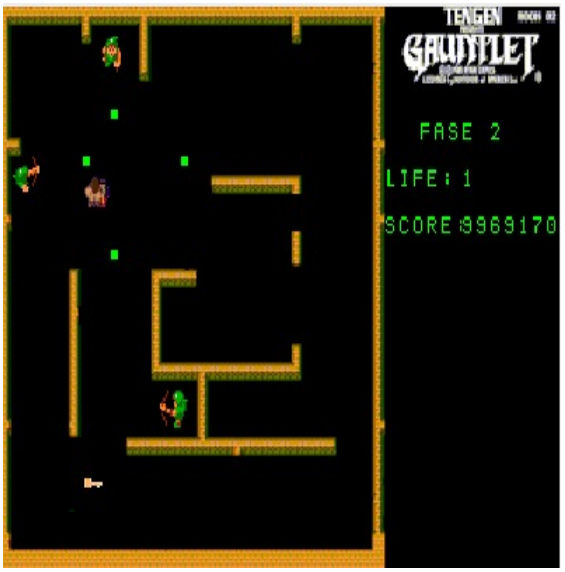

Figura 3: representação do tiro na fase

Para atirar, era verificado se o jogador havia apertado a tecla 'k'. Para fazer a movimentação do tiro, era verificada a posição que o personagem estava através do valor salvo no registrador pelo *input* de movimentação descrito acima. Após essa verificação, o tiro era mostrado na tela indo para a mesma direção que o personagem estava mirando.

2.7 Sistema de chaves e vidas

Para o sistema de vidas, foi utilizado um temporizador que, a cada 1 minuto, retira uma vida do personagem. Também é retirado uma vida caso o personagem seja atingido, como descrito no sistema de colisão acima. Já o sistema de chaves é feito através da verificação da posição da chave no mapa (como descrito na movimentação do personagem). Caso a posição da chave coincida com a do personagem, ela é coletada.

3. Resultados obtidos

3.1 Jogo

Apesar de não ter todas as funcionalidades do jogo original, conseguimos produzir um jogo que consegue realizar de maneira satisfatória conceitos relevantes para o desenvolvimento de um *game*: comandos vindos do teclado, representação da imagem na tela, colisões, movimentação dos inimigos, reprodução de áudio,

diferentes fases e etc, sem uma grande ocorrência de *bugs*. O jogador consegue desfrutar de uma experiência curta, mas bem-feita.
Também foi feita com sucesso a integração com o *Bitmap Display* e o MIDI, ferramentas do RARS.

3.2 Problemas

A maior dificuldade encontrada foi o tamanho do código em Assembly, pois dificultava o tratamento de erros e a alteração do código, além da ocorrência de um erro de *branch target*, quando o endereço passa dos limites de 12 bits de imediato do *Assembly*. Além disso, como são necessários muitos comandos para o funcionamento do código, erros causados por falta de atenção também se mostraram comuns.

4. Conclusão

Com o desenvolvimento do *Gauntlet Risc-Knife*, conseguimos apreender mais conceitos e comandos do RARS, melhorando, portanto, nossas habilidades em *Assembly*. Também conseguimos ter uma noção dos conceitos básicos necessários para o desenvolvimento do jogo, além de termos uma experiência de programação em grupo.

Figura 4: imagem final do jogo

Referências

WIKIPEDIA. Gauntlet
WIKIPEDIA. RISC-V
HOOKTHEORY. Viva la Vida
OPENGAMEART. SpriteSheet similar to Gauntlet II
GITHUB. HookTheory to RISC-V midi