Alien X Bomber

Danilo Lins Castro
Eduardo Sankievicz Lima
João Vítor da Silva Lima

Universidade de Brasília, Dep. de Ciência da Computação

Resumo

Este documento detalha o desenvolvimento de "Alien X Bomber", uma releitura do jogo Bomberman de 1983, produzido por Danilo Lins Castro, Eduardo Sankievicz Lima e João Vítor da Silva Lima, e desenvolvido em Assembly utilizando a arquitetura RISC-V e os simuladores FPGRARS e RARS. O propósito deste projeto é empregar o conhecimento adquirido ao longo do semestre na matéria de Introdução a Sistemas Computacionais, ensinada pelo professor Marcus Vinícius Lamar na Universidade de Brasília.

1. INTRODUÇÃO

Bomberman é um jogo lançado inicialmente em 1983 para computador e mais futuramente para NES em 1985, trata-se de um jogo simples de estratégia com um ponto de vista vertical sobre um labirinto com diversos inimigos em cada fase. Assim, o objetivo do jogador é eliminar todos os adversários dentro de certo tempo e evitar entrar em contato

físico com eles. Bomberman veio a se tornar uma franquia mundialmente famosa com títulos de peso e grande importância para a história dos jogos, se consolidando como um ícone cultural.

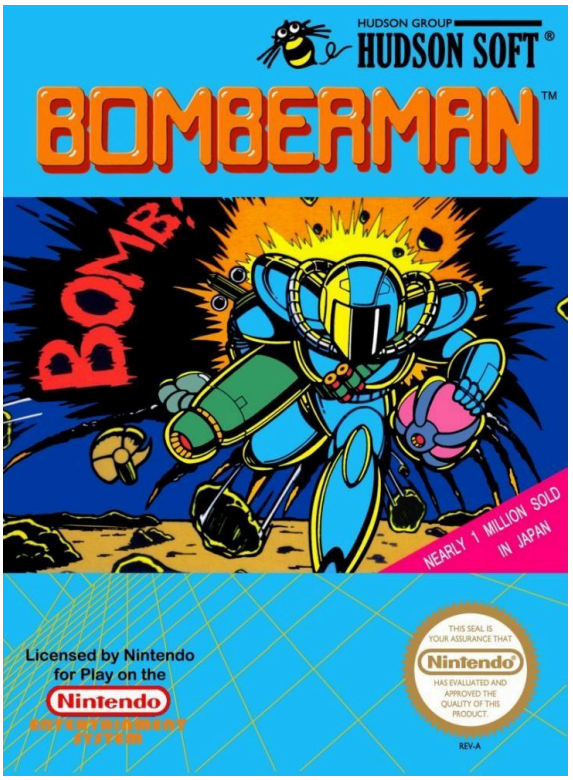

Figura 1: Capa do Jogo Original de NES Bomberman (1985)

Diante disso, o Alien X Bomber é uma releitura que tem como base tanto a jogabilidade quanto o visual do Bomberman clássico, mas com alterações significativas, como o visual dos personagens, objetos e mapas sendo originais, mas com inspiração no universo da franquia de filmes Alien. Portanto, nosso grupo visava fazer uma adaptação limpa e coesa que respeitasse o material original porém também tivesse um belo toque de criatividade.

2. METODOLOGIA

O projeto foi feito em Assembly e simulado no FPGRARS utilizando a ISA 32 do RISC-V. Por isso, as ferramentas utilizadas foram o “Keyboard and Display MMIO Simulator”, o “MIDI” e o “Bitmap 640x480 pixels”. Como resultado, faz-se necessário expor os passos e técnicas usadas durante a criação do projeto, sendo esses, a arte, a música, as fases, os inimigos, a colisão e as funções.

2.1 DIREÇÃO DE ARTE

Figura 2: Sprites do Jogo

A arte foi desenvolvida no Piskel, com o uso auxiliar do Paint.net para eventuais adaptações. Posteriormente, todas elas foram convertidas para o padrão “.data” por meio do script “BMPtoASM”, criado pelo professor. Além disso, todos os sprites têm resolução 32x32, com intuito de facilitar os processos de colisão e de impressão. Quanto ao mapa, utilizamos o aplicativo Tiled, que permite a criação e exportação de modelos de maneira rápida e fácil, e o aplicativo Krita, para modificação visual direta. Sob essa ótica, vale ressaltar a forte influência do filme “Alien, o 8º passageiro”(1979) no design do jogo.

Figura 3: Poster do filme “Alien, o 8º passageiro”

Por fim, cada personagem dinâmico do jogo, como o protagonista, os inimigos e a bomba, possuem um conjunto de frames que estruturam sua animação, cuja passagem é definida pelo tempo e pela movimentação do elemento.

2.2 MAPA E MEMÓRIA

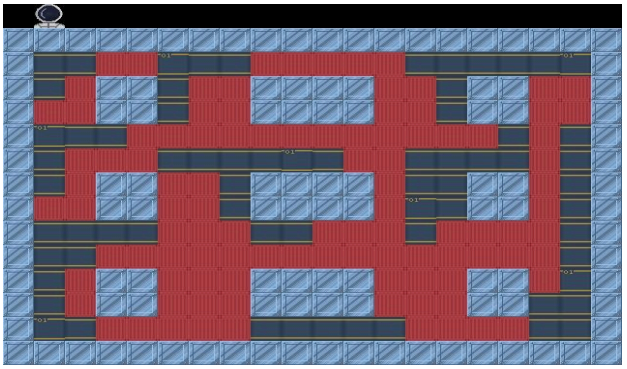

Figura 4: Mapa do Fase 1

Com fito de simplificar a lógica de armazenamento de valores, de modificação de elementos do mapa e de colisão, utilizamos a técnica de “tileset”, que resume cada “tile” a um valor numérico que, no Bitmap, representa uma cor RGB de 8 bits. Assim, para validar ações de jogabilidade, como movimentação dos

personagens, destruição do cenário e impressão dinâmica, basta consultar a matriz que reúne todos esses valores e, se necessário, modificá-la. Dessa maneira, um trabalho extremamente cansativo foi consideravelmente facilitado.

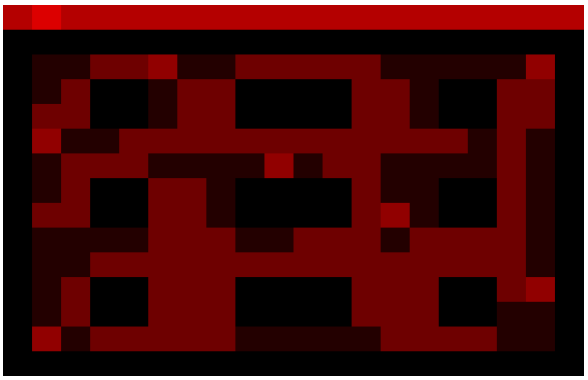

Figura 5: Tileset da Fase 1

Nesse exemplo, a figura 5 é um "tileset" da figura 4. Os quadrados pretos, cujo valor é 0, representam paredes intransponíveis, ou seja, impedem a movimentação; os vermelhos escuros, de valor 1, representam espaços onde os personagens podem andar. Nesse sentido, adicionamos cerca de 20 cores, cada uma exclusiva para um elemento distinto em relação à sua jogabilidade ou à sua aparência.

2.3 BOMBA E EXPLOSÃO

Sendo uma das principais características do jogo original, o desenvolvimento da bomba, da explosão e da destruição do cenário foram, certamente, algumas das nossas principais preocupações. Quanto à bomba, utilizamos um "timer" simples que, ao identificar a passagem de 2 segundos, explode. Além disso, também modificamos o "tileset" quando a bomba é colocada no cenário, a fim de impedir a passagem dos personagens por ela

.

Em relação à explosão, criamos uma lógica de laço "for" que expande seu raio à medida que o "gameloop" é repetido. Quanto à destruição do cenário, recorremos novamente à técnica de "tileset" e modificamos o valor caso fosse uma "tile" destrutível, transformando-a em passagem para os personagens caso o contato fosse confirmado.

Por fim, quanto à colisão com os inimigos e com o jogador, fizemos uma simples lógica de comparação de coordenada: caso as posições do sprite da explosão e do personagem sejam parecidas, o dano é computado.

2.4 INIMIGOS

Para um maior, porém ainda equilibrado, nível de desafio, optamos por adicionar dois tipos de inimigo, que se diferem pela lógica de movimentação e pelo visual. O primeiro tipo, representado pelo sprite de "facehugger" (2ª imagem da figura 2), muda de direção em sentido anti-horário ao identificar uma colisão. Isso, embora torne seu movimento previsível, faz com que ele tenha maior eficácia em fugir das bombas, visto que, ao identificá-las, move-se perpendicularmente, de modo a fugir de seu raio. O segundo tipo, por sua vez, é representado pelo sprite de "chestburster" (3ª imagem da figura 2), que possui uma mudança de orientação randômica, de forma a torná-lo mais imprevisível e ameaçador. Por último, o dano ocorre de maneira semelhante ao da explosão, mas com adição de certos "off-sets" para deixá-lo coeso com o aparente contato dos sprites.

2.5 "POWER-UPS"

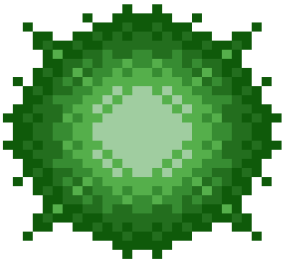

Figura 6: Sprite "power"

Figura 7: Sprite "speed"

Assim como no jogo original, a adição de “power-up” é um elemento essencial que deixa a experiência mais rica e divertida. Sendo assim, desenvolvemos dois deles: o primeiro é um “upgrade” de força, representado pela figura 6, que aumenta o raio da bomba simplesmente manipulando o limite do laço “for” descrito anteriormente; o segundo, por sua vez, acelera o personagem, de maneira que anda duas vezes mais rápido que o normal.

Vale ressaltar que, diferente da força da bomba, o aumento da velocidade não é trivial: deve-se criar condições para impedir seu uso excessivo e acumulativo, visto que isso afetaria completamente a colisão e outros setores do jogo. Por último, para obtê-los, o jogador deve quebrar o cenário, que os gera a depender de uma probabilidade específica.

2.6 FUNÇÕES

Funções são peças essenciais que permitem modularizar e reutilizar blocos de código, de maneira a facilitar a manutenção e compreensão do projeto. Assim, utilizamos inúmeras funções, sendo as principais:

– SET_LEVEL_1 e SET_LEVEL_2: preparam o jogo para os respectivos níveis;
– CHAR_LEFT, CHAR_RIGHT, CHAR_UP e CHAR_DOWN: movimentam o jogador para as respectivas direções;
– KEYPOLL: identifica input do teclado;
– DROP_BOMB: gera a bomba no mapa;
– UPDATE_BOMB: atualiza a bomba a cada “gameloop” e identifica sua explosão;
– UPDATE_EXPLOSION: atualiza a explosão e mede sua duração;
– UPDATE_ENEMY: atualiza, movimenta e analisa a situação de cada inimigo;
– PRINT: imprime o sprite previamente selecionado. (Função baseada no vídeo “RISC-V RARS - Renderização Dinâmica no Bitmap Display”, por Davi Paturi);
– MUSIC_LOOP: toca a música de acordo com o tempo e com as repetições do “gameloop”. Código inspirado no trabalho “Fix it Felix Jr.”, de Márcio Guimarães.

2.7 MÚSICA

A escolha da música foi um tanto difícil, especialmente ao considerar que o tema próprio da franquia “Alien” não é muito adequado para um jogo. Dessa maneira, a opção mais válida foi escolher músicas com tema espacial, sendo elas “Space Oddity” e “Starman”, ambas de David Bowie. Para a conversão, foi utilizado o programa “HookTheory to RISCV midi”, de Davi Paturi.

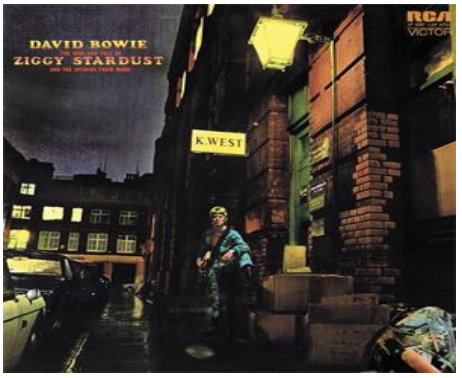

Figura 10: Capa do álbum “Ziggy Stardust and the Spiders from Mars”

Para a reprodução sonora da música, foi utilizada a ferramenta MIDI e suas respectivas syscalls, especialmente a ecall 31. Assim, quando uma nota for tocada, deve-se obter sua altura, duração e o instrumento desejado.

Quanto à implementação em código, utilizamos um vetor para cada nota, contendo sua altura, sua duração e o tempo da nota anterior. Dessa forma, com a ajuda de um timer, a nota seguinte é tocada quando a diferença entre o tempo de toque da nota anterior e o tempo atual fossem iguais à sua duração. Essa lógica, inserida em um loop sincronizado ao “gameloop”, fez com que não houvesse problemas em relação à

execução do jogo enquanto a música fosse reproduzida.

3. RESULTADOS OBTIDOS

Durante o extenso e árduo desenvolvimento do projeto, nos deparamos com diversos problemas, mas que eventualmente foram resolvidos. Por exemplo, pode-se citar dificuldade inata do aprendizado da linguagem Assembly: além de seu baixo nível e de certo nível de ilegibilidade, sua alta complexidade também não auxilia na sua compreensão ou na construção de um código coeso. Além disso, os programas utilizados, especialmente o RARS, não facilitam o processo: perdemos as contas de quantas vezes nosso projeto foi prejudicado não por culpa nossa, mas por instabilidades do próprio simulador. Dito isso, vale ressaltar a ajuda imensa do próprio FPGRARS, uma versão muito superior. Não obstante as dificuldades, o projeto ainda foi finalizado sem mais problemas de peso.

4. CONCLUSÃO

Em suma, a criação dessa releitura de Bomberman, programada em Assembly RISC-V, forneceu grande aprendizado aos estudantes envolvidos e nos permitiu um entendimento melhor da matéria de Introdução a Sistemas Computacionais, todos os elementos requisitados durante a exposição do projeto foram cumpridos e uma significante experiência foi adquirida. Por fim, destacam-se as colaborações de Eduardo Sankievicz na parte artística, de João Vítor Lima no que tange à programação e de Danilo Lins no que faz jus à documentação, ao registro e à música.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Hooktheory to RISCV midi - Conversor da partitura para a transcrição. Disponível em: https://gist.github.com/davipatury/cc32ee5b5929cb6d1b682d21cd89ae83

[2] RISC-V RARS - Renderização dinâmica no Bitmap Display. YouTube, 2021. Disponível em: https://www.youtube.com/watch?v=2BBPNgLP6_s

[3] Krita e Tiled - Criando um Tileset e Matriz para um mapa grande. YouTube, 2025. Disponível em: https://www.youtube.com/watch?v=q2a326oKl6U&list=PLL0Kob75DU32afhLBN5nY2KzOJ5k6lw-Q&index=9

[4] Script e Paint.net - Convertendo imagens para o usar
no RARS. YouTube, 2021. Disponível em: https://www.youtube.com/watch?v=q2a326oKl6U&list=PLL0Kob75DU32afhLBN5nY2KzOJ5k6lw-Q&index=9