Universidade Federal de São Carlos

Laboratório de Arquitetura de Computadores 2 - Turma B
Professor Dr. Luciano de Oliveira Neris

Relatório 3 - O Labirinto do Tesouro

Guilherme Nishi Kanashiro RA: 628298 - Engenharia de Computação
Tiago Bachiega de Almeida RA: 628247 - Engenharia de Computação

São Carlos, julho de 2017

Sumário

1 Introdução

Neste relatório será descrito o projeto e a implementação do jogo ”O Labirinto do Tesouro”, em assembly. Durante a disciplinas de Arquitetura e Organizações de Computadores II foram aprendidos conceitos que foram postos em prática na implementação desse jogo.

1.1 História dos jogos

Jogos eletrônicos são uma forma de entretenimento que vem se tornando cada vez mais popular desde a década de 1960, onde surgiram. Em seus primórdios eram sistemas simples, sendo implementados em equipamentos de baixa capacidade operacional em relação aos mais modernos. Os primeiros jogos desse tipo foram criados apenas como uma brincadeira de programador, tentando fazer algo diferente e divertido com o sistema que possuíam em mãos e também mostrar o poder de processamento do qual dispunham.

Dessa maneira, em 1961, Steve Russel desenvolve o jogo “Spacewar!” para demonstrar o poder de processamento do DEC PDP-1. Embora a ideia de Russel fosse voltada a fins acadêmicos, algum tempo depois Nolan Bushnell cria o Computer Space, uma máquina responsável apenas por rodar o Space War. Sua criação faz tanto sucesso que o leva a fundar a Atari, iniciar a popularização dos consoles e a criar mais jogos para suas máquinas. A Figura 1, a seguir, é uma imagem do SpaceWar sendo executado nessa máquina.

Figura 1: Spacewar em execução

Fonte: Wikipédia

Na década de 1980, porém, devido à grande quantidade de jogos de baixa qualidade que estavam sendo lançados e à popularização do computador pessoal, começou a ocorrer um crescente desinteresse em jogos eletrônicos. Assim, os computadores pessoais estavam cada vez mais desenvolvidos, contemplando um poder maior de processamento, monitores coloridos, um maior número de teclas e outras funcionalidades.

A partir da ideia de computadores pessoais cada vez mais poderosos, a Atari teve a iniciativa de trazer essa busca a maiores capacidades de processamento aos consoles com o Atari 400 (Figura 2) e o Atari 800, com entrada para cartucho. Apesar de ser essencialmente um console, esse equipamento fez muito sucesso como computador pessoal, pois tinha uma capacidade alta de processamento e permitia que o usuário utilizasse disquetes e teclados nele.

Figura 2: Atari 400

Fonte: Wikipédia

A partir desse ponto, nas décadas seguintes ocorreu um crescente desenvolvimento tanto de consoles como de computadores pessoais. Com o passar dos anos ambas as categorias ficaram capazes de reproduzir jogos eletrônicos cada vez mais complexos e também sofreram várias mudanças no que diz respeito às configurações de montagem dessas máquinas, aos periféricos suportados, ao modo de entrada dos programas e jogos, etc. Os computadores pessoais superam a popularidade e o número de vendas dos consoles, apesar disso, o segundo também se encontra em um patamar de altíssima capacidade de processamento.

1.2 Ideia do Projeto

Para colocar em prática a programação em assembly, foi desenvolvido um jogo de maneira semelhante a como era feito nas décadas anteriores. O projeto se trata de um jogo de estratégia focada no planejamento de movimentos do personagem que será controlado pelo jogador, chamado ”O Labirinto do Tesouro”.

Dado um labirinto, o jogador tem seu personagem iniciado no canto superior esquerdo da tela e deve percorrer o cenário coletando uma quantidade definida de tesouros para levá-los ao fim do labirinto (túnel final), realizando um quantidade de movimentos dentro do limite estabelecido pela fase. Além dos tesouros espalhados pelo labirinto, existem túneis conectores que são responsáveis por levar o personagem para outro lugar do cenário gastando menos movimentos.

O jogador deve então analisar a posição dos tesouros, dos túneis e os caminhos do labirinto de forma que saiba se movimentar para obter os itens e levá-los ao túnel final não ultrapassando o limite de movimentos da fase.

2 Descrição

2.1 Formas de Uso

Para movimentar o personagem o jogador deve utilizar as teclas w,a,s,d para mover a personagem para cima, esquerda, baixo e direita respectivamente. Os tesouros são coletados quando o personagem entra em contato com eles, não sendo necessário pressionar nenhuma outra tecla para tal. Apesar disso, para entrar nos túneis o jogador deve encostar no objeto e pressionar a Barra de Espaço, a tecla de iteração. Os túneis estão dispostos em pares pelo labirinto, sendo que entrar em um túnel de uma determinada cor transporta o personagem para o outro da mesma cor.

Ainda existe um tunel que não faz par com nenhum dos outros. Esse é o túnel final. Ao coletar todos os tesouros o jogador pode entrar nesse túnel e terá completado a fase.

Cada vez que o personagem se desloca a quantidade de movimentos que possui para concluir o labirinto é decrementada em 1. Entrar em túneis também conta como um deslocamento do personagem.

2.2 Formas de Representação Gráfica e Esboço

O cenário da fase será representado por uma vista superior do labirinto, de seus componentes e do jogador.

As paredes do labirinto serão representadas por sequências de um determinado caractere. O caractere definido na implementação lógica foi o “H”, o mapa deste labirinto foi desenhado por sequências dessa letra de forma que o jogador, ao observá-lo, saiba os caminhos que são possíveis de se percorrer (Figura 5). Vale destacar que a letra ”H”só existe na matriz dentro do código. No jogo em execução, tanto o fundo do trecho de tela deste caractere quanto ele próprio foram coloridos da mesma cor, formando um quadrado e dando realmente a ideia de ser uma parede (Figura 8).

Os tesouros estão representados pela letra “O” em cor amarela, remetendo a uma moeda de ouro. Esses caracteres estão dispostos nos caminhos livres do labirinto, de forma que o jogador reconheça que aquilo é um tesouro a ser coletado.

Os túneis estão representados pela letra “T” também nos caminhos disponíveis para o jogador percorrer. Os pares conectados de túneis são representados por dois túneis da mesma cor, de forma que o jogador saiba qual túnel está conectado com qual.

O jogador é representado pela letra “P”, remetendo à palavra “Player”.

O painel de informações do jogo se encontra embaixo do mapa da fase. Nele estão as informações: movimentos limite da fase, movimentos utilizados e tesouros restantes.

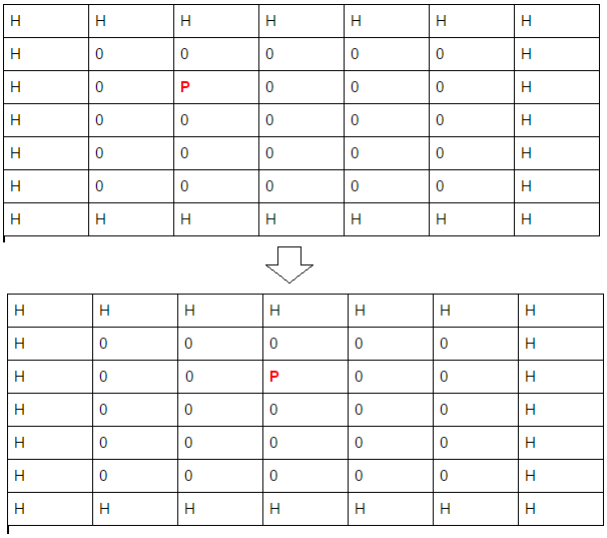
2.3 Etapas do Projeto

Para a implementação do jogo foi feito o seguinte planejamento em etapas:

1. Implementação da gravação dos mapas na memória e sua exibição na tela
2. Implementação dos movimentos básicos do jogador.
3. Implementação da iteração do jogador com os objetos da fase (paredes, tesouros e túneis).
4. Implementação dos estados do jogo, incluindo contagem de pontos, dos movimentos utilizados, verificação de quando a fase foi completada ou não, etc.
5. Implementação do menu inicial e tela de seleção de dificuldade.

Para o mapeamento foi utilizada uma matriz de caracteres, gravando o mapa na memória com esta estrutura de dados. Sua exibição foi feita em um loop utilizando os comandos de impressão de caracteres da biblioteca Irvine. Os valores dessa matriz são atualizados a cada comando dado pelo jogador, ou seja, supondo que o jogador avance uma posição, o caractere dessa posição na matriz será o caractere do jogador e o anterior voltara a ser o caractere de espaço livre. A Figura 3 exibe esse comportamento.

Figura 3: Esquema da alteração da matriz do mapa



Fonte: Elaborado pelos autores.

Há uma estrutura de repetição onde o programa espera o próximo comando do jogador. Quando o caractere é entrado pelo usuário, ele realiza uma iteração chamando os procedimentos de atualização dos caracteres do mapa e calculando os novos valores para as variáveis do jogo.

Todos os itens enumerados acima foram implementados em procedimentos para facilitar a legibilidade do código e sua manutenção. A programação dos elementos do jogo foi dividida entre os membros da seguinte forma: Tiago ficou com a implementação da movimentação do jogador, da impressão do mapa e da lógica de estados do jogo, incluindo transição entre fases, contagem de pontos e de movimentos utilizados; Guilherme ficou com a criação das fases, da implementação da lógica dos túneis, incluindo a concepção da ideia a ser implementada e da codificação, e da lógica usada para colorir os objetos do jogo.

2.4 Estruturas de Dados e Variáveis Principais

A estrutura de dados utilizada para armazenar a configuração atual da fase será uma matriz de caracteres de dimensões 20x50. A matriz será inicializada com a configuração inicial do jogo e a cada iteração do procedimento principal ela será atualizada com a nova configuração, após o movimento do jogador (Figura 3).

Para facilitar o cálculo das novas posições dos personagens ao entrar no túnel, foi utilizado um vetor contendo as coordenadas x1,y1,x2,y2, representando as posições dos túneis conectados. Ou seja, a cada quatro elementos consecutivos do vetor, temos um par de túneis interligados. Essa estrutura assume a seguinte forma:

```
tuneis BYTE 6,9,47,12,25,10,48,18,20,16,33,14,48,1
```

A seguir estão explicitadas as principais variáveis do projeto:

- (inteiro) movDificuldade: armazena o número de movimentos limite para cada fase, ajustado pela dificuldade selecionada.
- (inteiro) movFeitos: conta quantos movimentos já foram realizados pelo jogador. Caso seja igual ao movDificuldade, o jogador perde o jogo.
- (inteiro) tesourosRestantes: conta quantos tesouros ainda faltam para serem coletados. Quando estiver com o valor 0 e o jogador entrar na saída do mapa, ele passará de fase.
- (caractere) comando: recebe o caractere de movimento inseridos pelo usuário. O programa continua pedindo o caractere até que ele seja uma das teclas ”w”,”a”,”s”ou ”d”, indicando cima, esquerda, baixo e direita, respectivamente. Quando ele recebe a tecla, tenta movimentar o jogador para a direção indicada.

Também foram criadas variáveis de sinal para controlar o fluxo do jogo. Elas servem para auxiliar a comunicação entre os procedimentos, ajudando-os a tomar a decisão certa de acordo com o que foi realizado pelo procedimento chamado anteriormente. Elas são:

- sinalMapa: recebe a direção para o qual o jogador irá se mover. Caso ele colida com uma parede, recebe 1. Caso colida com túnel, recebe 2.
- sinalTesouro: recebe 1 se o jogador colidiu com algum tesouro e 0 se não colidiu.
- sinalTunel: recebe ”T”se o jogador estiver em cima de um túnel, 1 se o jogador usar este túnel e 0 se nenhuma das opções anteriores estiver ocorrendo.
- sinalMenu: recebe 1 se o jogador conseguiu iniciar o jogo pelo menu e 0 se ele selecionou a opção ”sair”.

- sinalDificuldade: recebe 1 se foi selecionada a dificuldade Fácil, 2 se Normal foi escolhido e 3 se Difícil.
- sinalFase: recebe 1 se o jogador estiver na primeira fase, 2 se estiver na segunda e 0 se quando terminar a fase atual o jogo for finalizado.

2.5 Descrição dos Procedimentos

Os procedimentos desse projeto tiveram como base as etapas de implementação elencadas na Introdução deste relatório. Seguindo a ideia de implementação apresentada foram criados vários procedimentos, sendo os mais importantes: telaInicial e telaDificuldade, inicializacoes, processaMovimento, atualizaMapa e atualizaContadores.

O procedimento principal se utiliza de um loop que simula a passagem de quadros do jogo. O código vai estar passando sempre pelo bloco de comando principais (dentro do loop) esperando uma reação do jogador e a resposta a essa reação fará com que o jogo continue. Em outras palavras, o código fica repetindo o loop até que algum comando seja inserido, quando acontece ele verifica os estados da partida e toma decisões sobre o jogo.

Abaixo, estão descritos com maior detalhamento os procedimentos citados:

- telaInicial: consiste no menu inicial do jogo. A tela inicial possui as opções ”Jogar”, ”Sair”e ”Guia”. ”Jogar”atualiza as variáveis de sinal para o início do jogo, continuando para a tela de dificuldades; ”Sair”encerra o programa; e ”Guia”abre a tela de instruções.

- telaDificuldade: é a tela onde o jogar escolhe em qual dificuldade irá jogar. Existem as opções de dificuldade Fácil, Médio e Difícil, diminuindo a quantidade de movimentos disponíveis para completar a fase conforme a dificuldade aumenta.

- inicializacoes: configura o início de fase do jogo. Toda vez que uma fase é inicializada, esse procedimento configura as seguintes variáveis e estruturas adequadamente à fase: mapa, tuneis, qntTuneis, TunelFinal, tesourosRestantes e sinalFase.

- processaMovimento: é chamada logo antes de atualizaMapa, pois verifica se é possível mover o jogador para aquela direção. Além disso, também verifica se houve colisão com algum tesouro ou com o fim da fase. Para tanto, recebe o comando inserido pelo jogador e a matriz da fase, faz a comparação entre as posições dos elementos do mapa e do jogador e retorna um valor de controle, dizendo se ocorrerá ou não a atualização da matriz.

- atualizaMapa: é acessado toda vez que o jogador entra com um comando. Esse procedimento recebe o comando e a matriz do mapa. Ele calcula a nova configuração da matriz e exibe na tela o que foi modificado, exibindo então a nova configuração do cenário da fase.

- atualizaContadores: é chamada dentro de atualizaMapa. Ela atualiza o contador de movimentos restantes e de tesouros recolhidos sempre que necessário. Recebe como parâmetros um sinal dizendo se houve ou não colisão com algum tesouro e retorna as variáveis movRestantes e tesourosRestantes atualizadas.

Os procedimentos estão dispostos no código fonte como mostra a seguir, usando a identação para ressaltar qual função é chamada dentro de outra:

```
Principal:

    telaInicial
    telaDificuldade

    Loop do Jogo:

        inicializacoes

        processaMovimento
            atualizaContadores

        atualizaMapa
```

A Figura 6 é um diagrama mostrando todos os estados do jogo de acordo com o que acontece dentro dos procedimentos.

2.6 Configurações Disponíveis Para o Usuário

No menu inicial o jogador poderá escolher entre iniciar o jogo, visualizar as instruções para saber como jogar, ou fechar o programa.

Caso escolha iniciar o jogo serão apresentadas para escolha 3 opções de dificuldade: fácil, médio e difícil. Dependendo da escolha do jogador, o número de movimentos disponíveis irá ser alterado. Caso ele escolha a opção mais fácil, haverá um número maior de movimentos disponíveis para concluir as fases. Caso ele escolha a opção mais difícil, o número de movimentos disponíveis para terminar a fase será mais perto do mínimo possível. A Figura 4 exibe a tela de dificuldade:

Figura 4: Tela de seleção de dificuldade

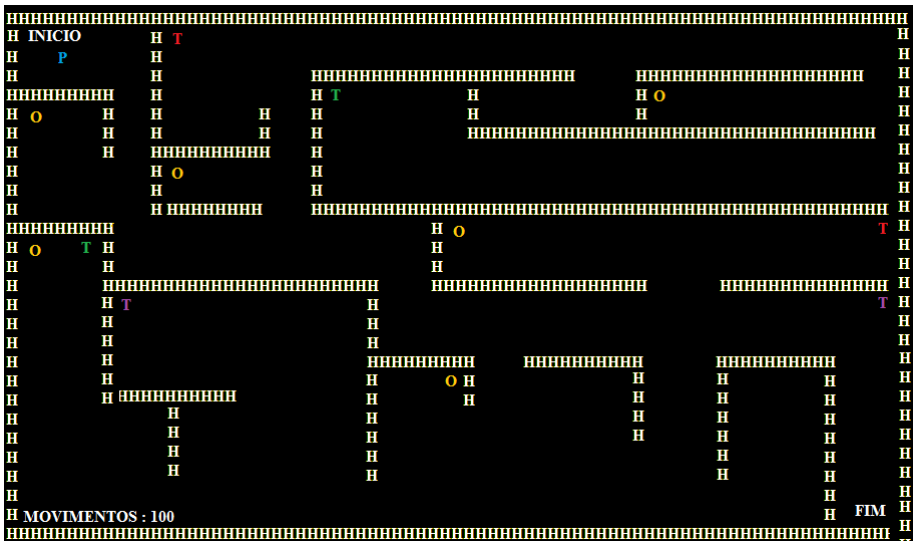
Fonte: Elaborado pelos autores. Captura de tela do jogo em execução.

3 Elementos Específicos

Nesta parte do relatório estão algumas imagens referenciadas durante o texto e mais algumas imagens para fins de exibição do resultado.

A Figura 5 representa a ideia inicial para o jogo em execução, ou seja, como os projetistas imaginaram o jogo antes de programá-lo.

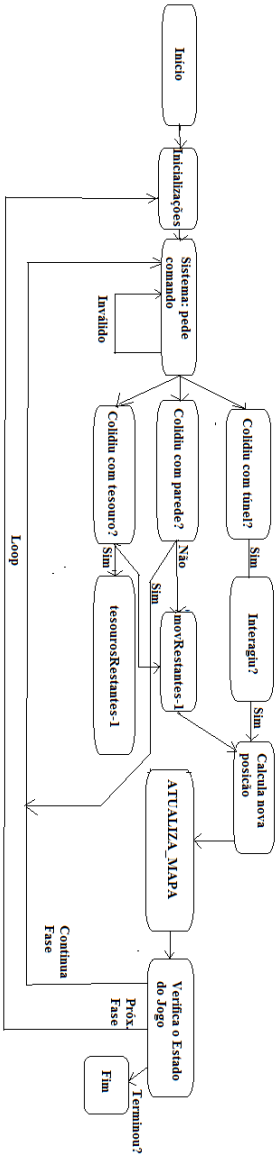
Figura 5: Esboço do jogo em execução



Fonte: Elaborado pelos autores.

A Figura 6 representa o diagrama de estados do jogo, utilizado para explicação nas seções anteriores.

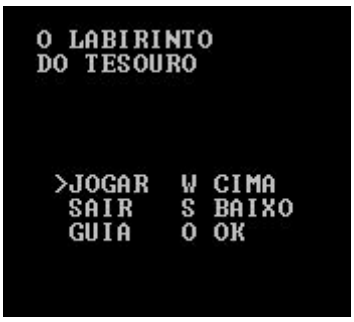
Figura 6: Diagrama de Estados do Jogo



Fonte: Elaborado pelos autores.

As Figuras 7 e 8 são capturas de tela do resultado final do projeto. A primeira representa o menu inicial e a segunda o jogo durante a execução da primeira fase.

Figura 7: Tela Inicial



Fonte: Elaborado pelos autores. Captura do jogo em execução.

Figura 8: Jogo em execução durante fase 1

Fonte: Elaborado pelos autores. Captura do jogo em execução.

4 Análise Crítica e Discussão

O projeto foi bastante enriquecedor para a prática da programação Assembly e também para melhor entender os seus conceitos. Houveram desafios interessantes ao grupo, exigindo uma boa organização dos programadores e um bom planejamento da implementação do código. Também foram detectadas possíveis melhorias no código que poderiam ter sido executadas durante a implementação.

Ver o jogo funcionando e sendo divertido de se jogar foi muito satisfatório. Isso mostrou que o grupo conquistou a meta de criar um jogo desafiador, mas prazeroso.

As próximas partes desta seção conterão uma análise crítica e uma discussão sobre tais desafios e melhorias que poderiam ter sido feitas.

4.1 Desafios

O primeiro grande desafio da implementação foi organizar o código de forma que um procedimento conseguisse conversar com o outro. Isso exigiu o desenho do Diagrama de Estados exibido neste relatório e vários algoritmos de forma que fosse possível melhor organizar a codificação. A solução mais viável no momento da implementação foi a utilização das variáveis de sinal, que foram exibidas na Seção 2.4.

Implementar a lógica de transição do personagem pelos túneis também exigiu um bom esforço. O grupo buscou ao máximo criar uma lógica fácil de ser entendida, implementada e alterável, caso necessário em algum momento da implementação. A solução foi criar o vetor de posições exibido também na Seção 2.4.

Por fim, desenhar o mapa foi algo bem desafiador por se tratar de uma parte mais subjetiva do trabalho. Ou seja, o projetista precisou ter uma noção de como seria um mapa desafiador e tentar várias configurações de cenário até chegar em um satisfatório.

4.2 Melhorias

Entre as melhorias que foram detectadas ao final da implementação, mas que não entraram no código fonte, a primeira delas a ser ressaltada é o uso de muitas variáveis e registradores de 32 bits quando seria possível e mais econômico usar tamanhos menores de dados.

Também não foram usados os conceitos de pilha em conjunto com a criação de procedimentos. Isso se deu pelo fato de grande parte do desenvolvimento do jogo ter sido feito antes dos programadores terem conhecimento de tais técnicas. Devido a isso, o código ficou grande, algumas vezes repetitivo e os projetistas tiveram sempre que se lembrar de restaurar o valor antigo dos registradores utilizados dentro de algum procedimento quando saía deste através de algum tipo de inicialização. Tal ocorrido não ocasionou nenhum grande problema durante a implementação, mas o código teria ficado muito mais

otimizado e compacto com as técnicas mais avançadas que poderiam ter sido aplicadas.

Referências

IRVINE, Kip. Assembly Language for Intel-Based Computers. Quarta Edição. Prentice Hall, 2003. 708 páginas.

CONTI, Fátima. História: Primeiros Jogos Digitais Última Modificação: 11 de outubro de 2015. Disponível em: http://www.ufpa.br/dicas/net1/int-h-jo.htm. Data de Acesso: 06 de junho de 2017

CONTI, Fátima. História: Primeiros Jogos Digitais. Última Modificação: 11 de outubro de 2015. Disponível em: http://www.ufpa.br/dicas/net1/int-h-jo.htm. Data de Acesso: 06 de junho de 2017

História dos Jogos de Computador. Disponível em: http://www.ahistoria.com.br/jogos-de-computador/. Data de Acesso: 06 de junho de 2017.

MORAES, Henrique. A História dos Jogos de Computadores. Disponível em: http://www-usr.inf.ufsm.br/~hramos/elc1020/artigo_hist_jogos.pdf. Data de Acesso: 06 de junho de 2017

IRVINE, Kip. CSIS 118B: Computer Organization and Assembly Language. Última Modificação: 06 de julho de 2016. Disponível em: http://www.programming.msjc.edu/asm/Home.aspx. Data de Acesso: 06 de julho de 2017.