Universidade Federal de São Carlos
Engenharia de Computação

Laboratório de Arquitetura e Organização de Computadores 2

Segundo Relatório

Alunos: Bruna Zamith Santos (RA: 628093)
Marcos Augusto Faglioni Junior (RA: 628301)

Professor: Dr. Luciano Neris

06/2017
São Carlos - SP, Brasil

Conteúdo

Lista de Figuras

1 Introdução

O presente relatório objetiva a apresentação mais detalhada do jogo a ser desenvolvido na disciplina Laboratório de Arquitetura e Organização de Computadores 2, ministrada pelo docente Dr. Luciano Neris, no primeiro semestre de 2017, na Universidade Federal de São Carlos.

O jogo, entitulado ”Clock Colors”, será desenvolvido em linguagem Assembly e fazendo uso da biblioteca Irvine32 [1]. Como objetivo secundário, destaca-se aprofundar o conhecimento na linguagem anteriormente citada e na referida biblioteca, assim como os recursos que esta oferece. Não obstante, será utilizado o montador *Microsoft Macro Assembler* (MASM) [2], o qual suporta as arquiteturas IA-32 e x86-64 para MS-DOS e Microsoft Windows. O ambiente de desenvolvimento - *Integrated Development Environment* (IDE) - será o Visual Studio.

1.1 Etapas de Desenvolvimento

As etapas de desenvolvimento do jogo estão enumeradas a seguir:

1. Construção das Bordas da Tela
2. Construção das Plataformas
3. União das Plataformas e Bordas com o texto do Menu Inicial
4. Implementação do Menu Inicial
5. Criação da Página de Como Jogar
6. Criação da Página de Créditos
7. Contador de Tempo Regressivo
8. Criação do Personagem Controlável pelo Jogador
9. Geração dos Obstáculos na Plataforma, randomicamente
10. Detecção de Colisão
11. Sorteio das Cores Iniciais
12. Variação das Cores da Plataforma
13. Atualização da Altura das Plataformas
14. Criação da Tela de Resultado do Jogo

15. Implementação das Diferentes Dificuldades do Jogo

16. Testes

17. Correção de Eventuais Erros

1.2 Divisão de Tarefas

Para a implementação do projeto, ficou decidido que todos teriam igual participação no desenvolvimento. Assim, ambos os desenvolvedores podem ter contato com as estruturas de dados do jogo e as principais rotinas. Não obstante, evita-se a segmentação do código.

Primeiramente, a dupla se reuniu para decidir a lógica do jogo, as variáveis e as rotinas. Então, para o desenvolvimento, se reunem presencialmente e discutem juntos.

A seguir, o jogo será descrito em alguns aspectos, analisado e discutido. Deste modo, este Segundo Relatório visa o detalhamento das ideias de implementação do jogo.

2 Descrição

2.1 Estrutura de Dados

As estruturas de dados a serem utilizadas no desenvolvimento do jogo são: Vetores, Matrizes e Pilha.

Vetores: Para o projeto, são os *arrays* de caracteres, ou seja, *strings*. São elementos essenciais ao Menu Inicial, à Tela de Resultado e a qualquer informação que será escrita na Tela do Jogo (como o contador regressivo de tempo e as cores sorteadas). Não obstante, também é utilizado como o *array* que armazena as 8 cores disponíveis para sorteio; como o *array* que efetivamente armazenas as 2 cores sorteadas; e como o *array* que armazena as posições das armadilhas.

Matrizes: A tela por si só é uma matriz. Todas as informações exibidas possuem coordenadas X e Y na matriz da tela do jogo.

Pilha: As chamadas de procedimentos são feitas pelas instruções ”Call”e ”Ret”, que por sua vez, fazem uso de uma pilha para armazenar os endereços de retorno.

2.2 Variáveis Principais

Os principais dados a serem armazenados são:

- *tempo BYTE ”TEMPO:”, 0*
  Nome do marcador de Tempo
- *pontuacao BYTE ”PONTUACAO:”, 0*
  Nome do marcador de Pontuação
- *biniciar Byte ”INICIAR”,0*
  Nome do botão para iniciar o jogo
- *biniciari Byte ”Pressione w”,0*
  Nome da informação de como acessar o início do jogo
- *bcreditos Byte ”CREDITOS”,0*
  Nome do botão para os créditos
- *bcreditosi Byte ”Pressione a”,0*
  Nome da informação de como acessar os créditos
- *bcomoJogar Byte ”COMO JOGAR”,0*
  Nome do botão para as instruções
- *bcomoJogari Byte ”Pressione s”,0*
  Nome da informação de como acessar as instruções

- *nome BYTE "CLOCK COLORS", 0*
  Nome do Jogo
- *time BYTE 0*
  Armazena o tempo do jogo
- *score BYTE 0*
  Armazena a pontuação do jogo
- *posArmadilhas1 BYTE ?,?,?*
  Posições X das armadilhas da primeira plataforma
- *posXArmadilhas2 BYTE ?,?,?*
  Posições X das armadilhas da segunda plataforma
- *posXArmadilhas3 BYTE ?,?,?*
  Posições X das armadilhas da terceira plataforma
- *posXArmadilhas4 BYTE ?,?,?*
  Posições X das armadilhas da quarta plataforma
- *coresDisponiveis WORD yellow, blue, green, cyan, red, magenta, lightBlue, lightRed*
  Vetor de Cores Disponíveis para as Plataformas
- *coresSorteadas WORD ?,?*
  Vetor de Cores Sorteadas

3 Elementos Específicos

3.1 Procedimentos

3.1.1 Desenha Tela

O jogo conta 5 telas principais, sendo que para cada uma, existe um procedimento específico: Tela de Menu, Tela do Jogo, Tela de Créditos, Tela de Como Jogar e Tela de Resultado do Jogo.

A Tela de Menu (exposta na Figura 1) é impressa ao inicializar-se o jogo, e conterá as opções ”Inicializar o Jogo”, que é selecionada pressionando-se a tecla ”w”ou a tecla ”W”, e chama a Tela do Jogo; ”Instruções”, que é selecionada pressionando-se a tecla ”s”ou a tecla ”S”, e chama a Tela de Como Jogar; e ”Créditos”, que é selecionada pressionando-se a tecla ”a”ou a tecla ”A”, e chama a Tela de Créditos. Deste modo, a Tela do Menu deve ler um caractere do teclado do usuário até que uma tecla válida seja pressionada, e então chama o procedimento que imprime a tela correspondente.

Uma vez acessada e impressa a Tela de Instruções, caso o usuário pressione qualquer tecla, volta-se à Tela do Menu.

Uma vez acessada e impressa a Tela de Créditos, caso o usuário pressione qualquer tecla, volta-se à Tela do Menu.

Uma vez acessada a Tela do Jogo, uma série de procedimentos - a serem detalhados nas próximas seções - é chamada. Contudo, é feita a leitura do teclado do usuário e, caso este pressione a tecla ”q”ou a tecla ”Q”, volta-se à Tela do Menu.

A Tela do Resultado do jogo só é impressa ao final do jogo e contém as seguintes informações: Tempo Total de Jogo, Plataformas Atingidas e Número de Plataformas/Segundo.

Os próximos procedimentos descritos só são chamados se a opção ”Inicializar o Jogo”tiver sido escolhida.

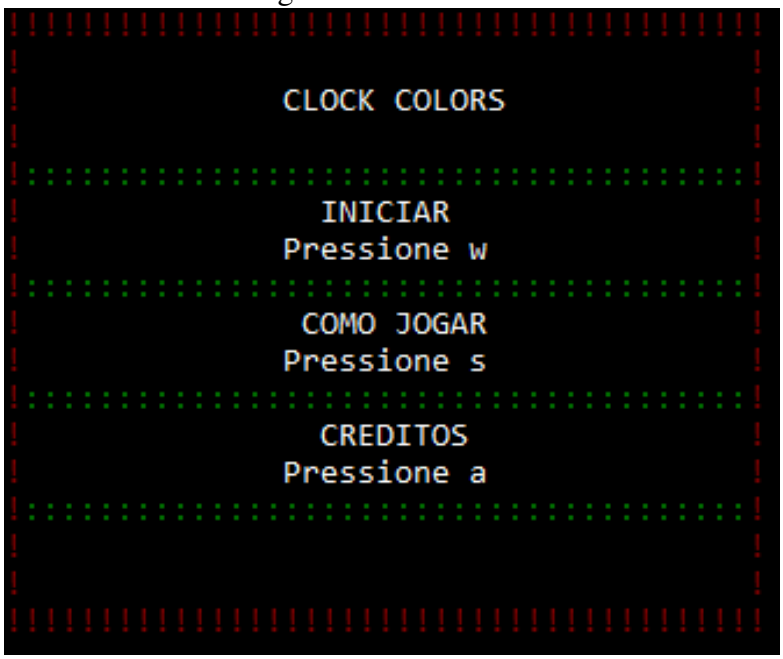
3.1.2 Sorteia Cores Iniciais

O primeiro procedimento chamado é o SorteiaCores. Tal procedimento seleciona, randomicamente, 2 cores dentre as 8 disponíveis e as armazena no vetor de coresSorteadas. As cores disponíveis são: *yellow*, *blue*, *green*, *cyan*, *red*, *magenta*, *lightBlue* e *lightRed*. São cores *default* do Irvine [1].

3.1.3 Contador

Posteriormente, é chamado o procedimento que inicializa o contador com 90 segundos, e a cada segundo, decrementa-o em 1. Além disso, a cada decremento, é feita a

Figura 1: Menu Inicial



comparação do valor do contador com 0. Se for igual, a Tela de Resultado é chamada e o jogo é encerrado.

3.1.4 Sorteia Obstáculos

Próximo procedimento a ser chamado é o que sorteia, randomicamente, as posições no eixo X das 3 armadilhas para cada uma das 4 plataformas. Se duas ou mais posições coincidirem, não é indicado nenhum problema e as armadilhas são sobrescritas. As posições nos eixos Y não são variáveis, sendo sempre uma acima da plataforma correspondente.

3.1.5 Altera a Cor das Plataformas

As plataformas variam entre as 8 cores possíveis. A variação é feita a cada segundo, seguindo a sequência do vetor coresDisponiveis.

3.1.6 Aguarda Comando do Usuário

Durante todo o jogo, é esperado uma entrada do teclado do usuário. O usuário pode entrar com os seguintes comandos:

Tecla 's': Indica movimentação para direita.

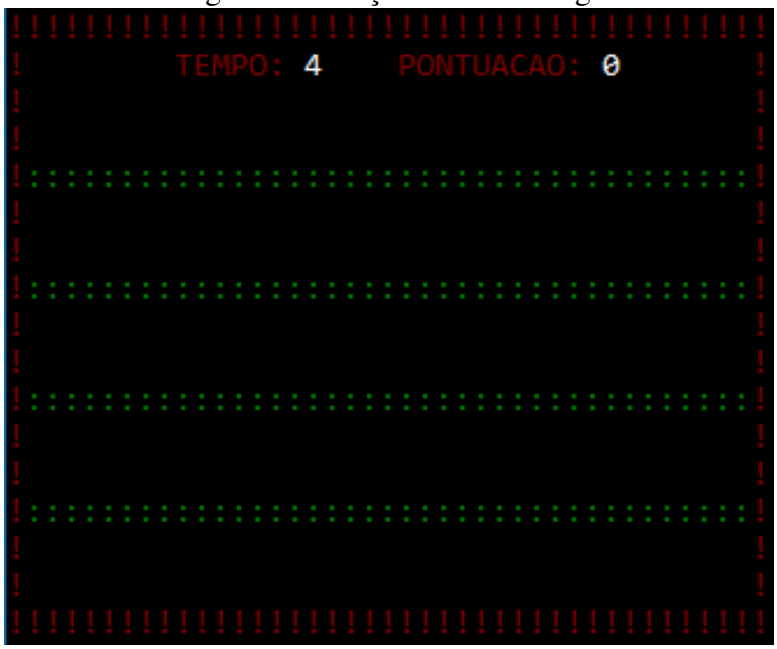
Tecla 'a': Indica movimentação para esquerda.

Tecla 'w': Salto para plataforma de cima.

Tecla 'q': Volta para o Menu Inicial.

Se for digitado qualquer outro caractere, o procedimento continua aguardando uma entrada válida do usuário. A Figura 2 mostra um esboço da Tela do Jogo.

Figura 2: Esboço da Tela do Jogo



3.1.7 Verifica Movimento Horizontal Válido

Caso o usuário digite 's' ou 'a', é verificado se ele não está em uma das bordas (paredes) do jogo. Se não estiver, a posição X do jogador é atualizada (se 's', o X é incrementado, se 'a', é decrementado).

3.1.8 Verifica Salto Válido

Caso o usuário digite 'w', são feitas duas verificações:

1. A plataforma logo acima está com uma das duas cores sorteadas.
2. A posição (X e Y) para qual o jogador está se movendo não possue armadilhas.

Caso as duas verificações retornem verdadeiro, então a posição é atualizada e o procedimento "DescePlataformas"é chamado. Caso contrário, o jogo é encerrado e a Tela de Resultado é exibida.

3.1.9 Desce as Plataformas

Em relação ao Relatório 1, foi feita uma modificação: Assim que o jogador digita 'w' e o salto é permitido, ao invés de o jogador subir, as plataformas descem. Desta forma, sempre que é permitido ao jogador saltar, é chamado o procedimento "DescePlataformas", em que cada plataforma recebe a posição da plataforma imediatamente anterior. Além disso, uma nova plataforma (a quarta) é criada novamente, chamando o procedimento que sorteia as armadilhas desta plataforma. A tela do jogo é então atualizada.

3.2 Diagrama de Estados

Figura 3: Diagrama de Estados do Jogo - Parte 1

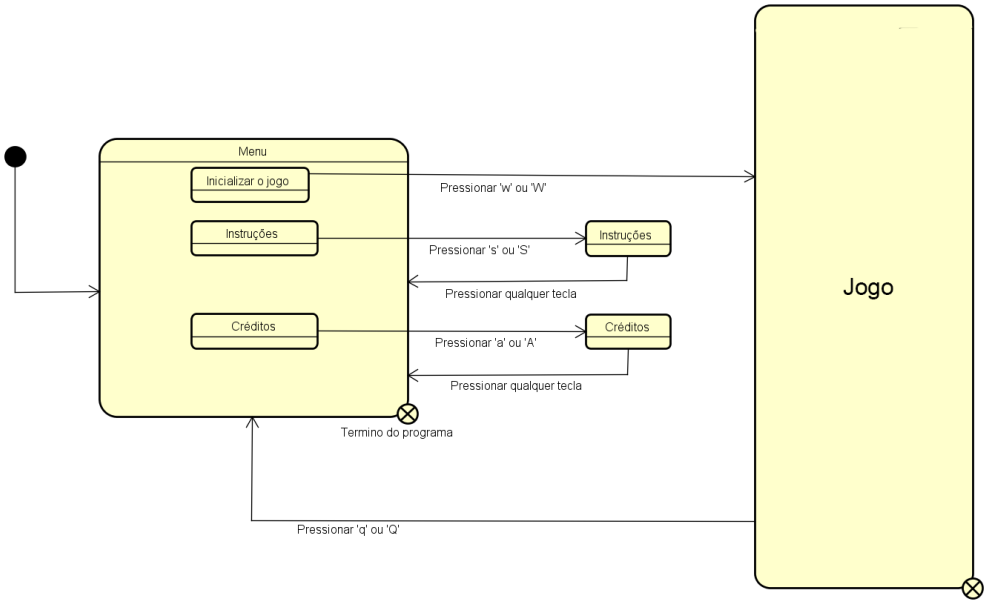
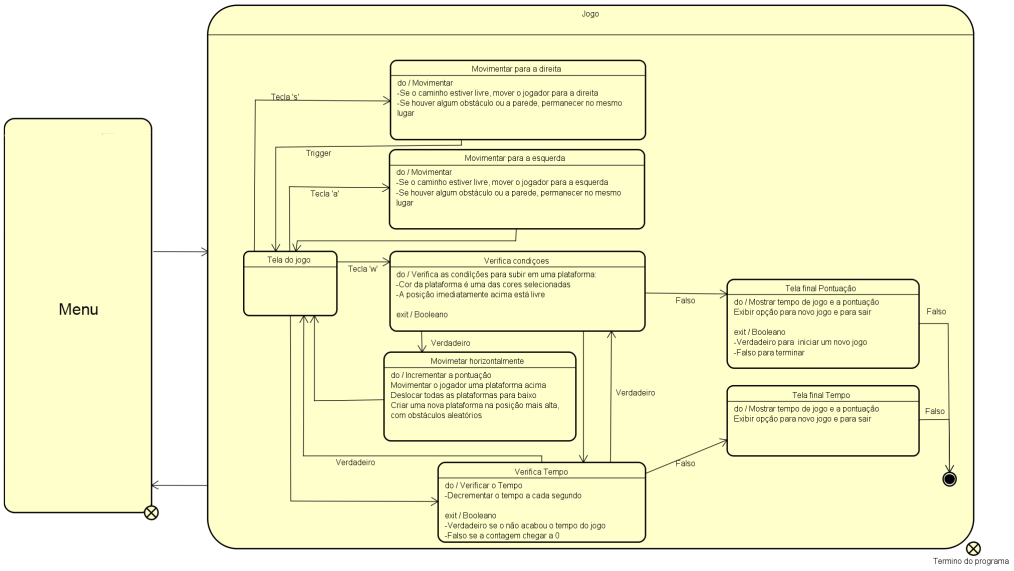


Figura 4: Diagrama de Estados do Jogo - Parte 2



3.3 Configurações Disponíveis ao Usuário

Após implementação completa do jogo, visa-se implementar a opção de seleção de dificuldades. Assim, o usuário poderá, no Menu Inicial, optar entre duas dificuldades do jogo: Normal e Expert.

Normal: A dificuldade Normal é implementada como exposto até o presente momento.

Expert: A dificuldade Expert é implementada como a Normal, com três alterações: Serão 5 armadilhas por plataforma, apenas 1 cor sorteada e as cores das plataformas variam mais rapidamente.

4 Discussão e Análise

A maior dificuldade encontrada até o momento de escrita deste relatório foi entender como as cores e o contador variariam a cada segundo. Contudo, após testes iniciais, foi encontrada uma forma de trabalhar com esta característica do jogo. Ainda, um problema enfrentado era o *delay* na atualização da tela, causada por uma condição conhecida como "*flickering*". Uma solução enconrada foi renderizar uma cena em um *buffer* da memória e então dar um *output* do *buffer* para o *console*.

Além disso, duas alterações foram feitas em relação ao relatório anterior: 1. O usuário não mais sobe na tela, mas sim as plataformas que descem. Isto foi feito pensando que facilitaria a jogabilidade, dado que não mais precisaria atualizar a tela inteira assim que o jogador chegasse ao topo. 2. Propõe-se a implementação de uma opção de escolha de dificuldades do jogo, como descrito na Seção 3.3.

No geral, acredita-se que os resultados alcançados até o momento de escrita deste segundo relatório são satisfatórios, com todas as informações necessárias para dar prosseguimento ao desenvolvimento do projeto.

Referências

[1] Irvine Library Help. Disponível em: http://programming.msjc.edu/asm/help/index.html?\\page=source%2Fabout.htm. . Acesso em: 26 abr. 2017.

[2] The MASM32 SDK. Disponível em: http://www.masm32.com/. Acesso em: 26 abr. 2017.