Departamento de Matemática

FCTUC

Relatório

Visualização Computacional

Professor: José Carlos Teixeira (teixeira@mat.uc.pt)

Trabalho 3 – Animação R3 da época de Natal

Autores:
Ana Catarina Quitério Lourenço
catarinaql@gmail.com
Israel Campiotti
israelcampiotti@gmail.com

Data:
13-01-2017

ÍNDICE

1. SUMÁRIO

O trabalho efectuado visa a utilização dos recursos disponibilizados pela linguagem C e pelo OpenGL no desenvolvimento de uma cena animada em R3 sobre o tema genérico Época de Natal em qualquer parte do mundo com dois modos de operação: automático e manual.

A aplicação gera um jogo em que um boneco de neve num skate deve desviar-se dos obstáculos que aparecem no seu caminho. A decoração das casas, a queda de neve e o som representam o ambiente natalício.

A aplicação deve ser original e ergonómica, concedendo instruções que auxiliem a sua utilização.

O trabalho foi realizado recorrendo às várias bibliotecas da linguagem C e, especialmente, à biblioteca glut.h.

O presente relatório explicita pormenorizadamente a construção do programa criado e o modo de utilização do mesmo.

2. INTRODUÇÃO

2.1 Visão

O trabalho que nos foi proposto tinha por objectivo a utilização dos recursos facultados pela linguagem C e pelo OpenGL no desenvolvimento de uma cena animada em R3 sobre o tema genérico Época de Natal em qualquer parte do mundo.

O tema sugerido é bastante abrangente e permite a criação de cenários bastante diferentes. Optámos por criar um jogo em que um boneco de neve num skate se depara com obstáculos dos quais se deve desviar. O utilizador pode desviar-se de cada obstáculo individualmente, recorrendo às teclas, ou pode optar pela resolução automática, em que o boneco se desvia sem qualquer instrução. o utilizador perde o jogo quando não se consegue desviar de um dos obstáculos.

2.2 Análise e Especificação de Requisitos

Requisitos funcionais

- O programa deve permitir que o utilizador desloque o boneco de neve recorrendo ao teclado.
- O programa não deve permitir que o boneco de neve se desloque para além dos limites da estrada.
- O programa deve permitir uma resolução automática, isto é, o boneco de neve deve desviar-se dos objectos no seu caminho sem qualquer comando do utilizador.
- O programa permite que o utilizador reinicie o jogo e saia do jogo.

Requisitos não funcionais

- O programa deve correr no sistema operativo Windows.
- O programa deve ser desenvolvido recorrendo à linguagem C e ao OpenGL.
- O programa deve recorrer, especialmente, à visualização dinâmica em R3 e às técnicas de iluminação e sombreamento.
- Após ler as instruções, o utilizador deve conseguir usar a aplicação facilmente, não restando dúvidas sobre o modo de funcionamento do jogo.

3. DESENVOLVIMENTO

3.1 Conceção

A implementação dos objectivos foi realizada de forma a criar um jogo que termina quando o boneco de neve não se desvia de um dos obstáculos.

As instruções do jogo estão disponíveis em qualquer momento.

3.2 Arquitetura da solução

O programa divide-se essencialmente em duas partes: desenhar o fundo do jogo e boneco de neve e gerar o movimento dos objectos.

Tendo em vista estes objectivos, foram criados os ficheiros “obstaculos.h” e “display.h”, distribuindo-se por estes as várias funções e variáveis necessárias para implementar a aplicação pretendida, a fim de obter uma melhor organização do programa.

“stcObj.h” contém uma estrutura “Objeto” cujas variáveis dizem respeito as várias propriedades usadas para desenhar cada objecto (tamanho, coordenadas). Contém, ainda, a função “iniciaGlobais”, que inicializa as variáveis globais, uma função “iniciaObj” que permite inicializar uma variável do tipo “Objeto”, atribuindo os valores indicados aos elementos da estrutura e as funções necessárias para desenhar o que aparece na janela: o alfalto (“dChao”), os obstáculos (“desenhaObj”, “desenhaEsfera” e “desenhaCone”), as casas (“desenhasCasa”) e o boneco de neve num skate (“desenhaBoneco”).

“display.h” contém as várias funções necessárias para inicializar os parâmetros de exibição de imagem (“iniciar”), controlar os movimentos do boneco de neve (“timerMoveEsfera”, “decideMovimentoEsfera_Cima”, “decideMovimentoEsfera_Baixo”, “iaMovimento”), controlar a posição dos obstáculos (“setPlace”), verificar se o boneco de neve colidiu com um obstáculo (“Funcao de choque”), desenhar os vários elementos da janela (“Desenhar”), gerar o movimento da cena (“moveCena”), controlar o *input* do teclado (“Teclado”), controlar o input através do menu (“controlaMenu”), controlar a queda dos flocos de neve (“desenhaNeve”, “iniciaPosicaoFloco” e “iniciaNeve”) e gerar os vários textos necessários (“desenhaInstrucoes”, “displayRoman24” e “displayText”).

“main.c” recorre às funções contidas nos ficheiros anteriores para criar a janela do jogo e permitir a interacção com o utilizador.

3.3 Interface com o utilizador

O utilizador tem apenas de usar a janela da aplicação, não sendo necessário usar a consola.

Para tornar a aplicação mais fácil de usar, as instruções devem ser consultadas em qualquer momento do jogo, através do menu.

Quando a janela exibe o jogo, o utilizador apenas pode alterar a posição do boneco de neve, sendo o ambiente que o rodeia constante. Contudo, através do menu, é ainda possível optar pela deslocação automática do boneco, reiniciar o jogo ou sair do jogo.

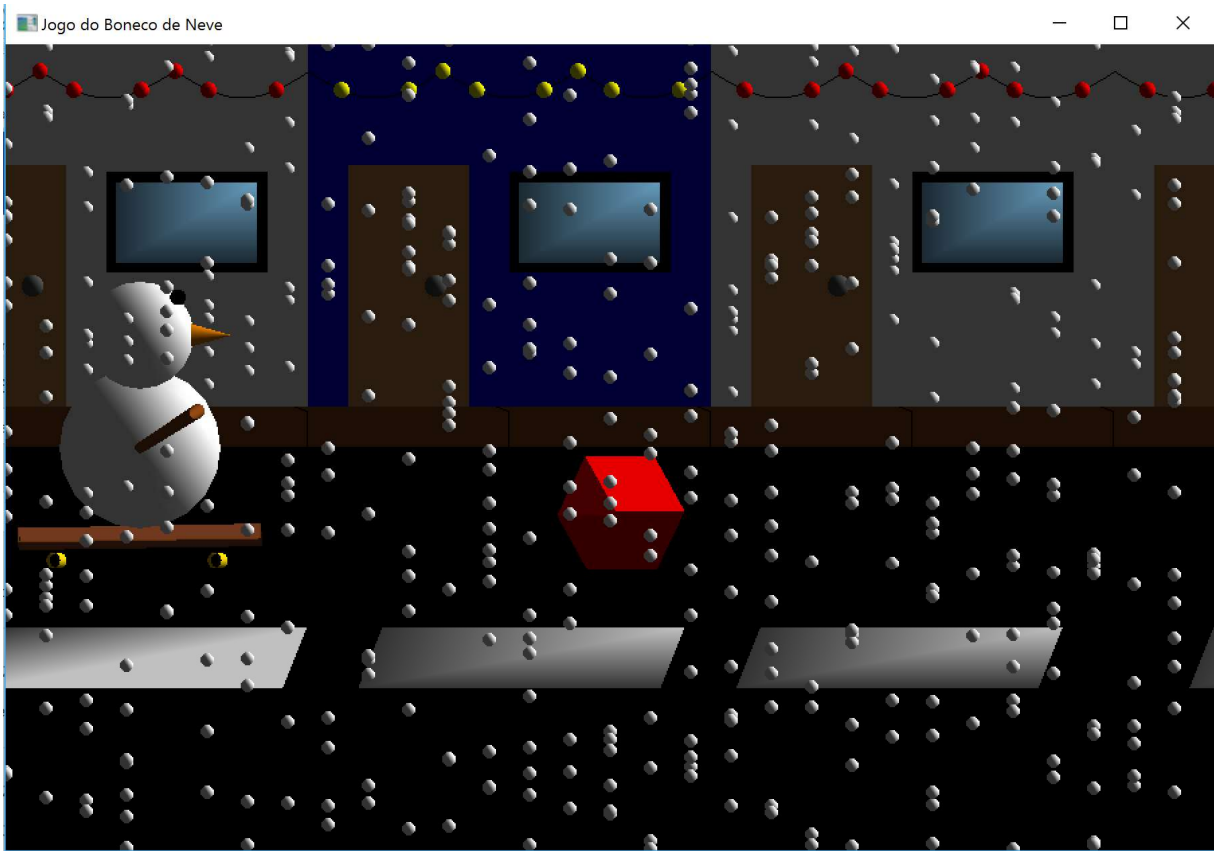

Figura 1 - Início do jogo

3.4 Estruturas de dados

Os principais dados, usados em várias funções, são variáveis globais que representam características como o tamanho da janela e a posição e o tamanho dos vários elementos do jogo ou permitem controlar os movimentos dos objectos, sendo todos do tipo *int* ou *double*.

Os únicos dados que devem ser destacados são os que correspondem aos obstáculos. Estes têm o mesmo tipo de características, que são usadas em várias funções, pelo que criamos uma estrutura "Objeto".

```
Estrutura {
    Int id;    //numero que identifica o objecto
    double Ox;
    double x;          //coordenda x usada para desenhar o objecto
    double y;          //coordenada y usada para desenha o objecto
    double yCima;   //coordenada y quando o objecto se encontra perto de casas
    double yMeio;   // coordenada y quando o objecto se encontra no meio da estrada
    double yBaixo;  // coordenada y quando o objecto se encontra mais afastado das casas
    double radius              //raio do objecto
} Objeto
```

3.5 Principais funções

void FuncaoDeChoque : esta função determina se o boneco atingiu ou não um obstáculo. Se a resposta for não, então é gerado aleatoriamente um novo obstáculo numa nova posição. Caso contrário declaramos o fim do jogo.

void decideMovimentoEsfera_Cima(): esta função é chamada quando o utilizador carrega na tecla 'w' ou quando a resolução automática está seleccionada. Serve para decidir a nova posição do boneco: passa para a posição mais próxima das casas quando se encontra a meio da estrada; passa para o meio da estrada quando se encontra na posição mais afastada das casas.

void decideMovimentoEsfera_Baixo(): analogamente à função acima, esta é chamada pela resolução automática ou quando se pressiona 's'. A nova posição é decidida como sendo o meio da estrada se o boneco está acima e abaixo quando o boneco se encontra no meio.

void timerMoveEsfera(int pos): esta função é responsável pelo movimento do boneco. Dado o valor da variável 'pos' (1 para posição mais acima, 2 para meio e 3 para baixo), que corresponde à posição

para a qual se pretende mover o objecto, determina-se o valor da nova coordenada y usada para desenhar o objecto.

void iaMovimento(): esta função é utilizada para controlar o movimento automático do boneco de neve. Primeiro verifica se o boneco e o obstáculo estão em rota de colisão. Caso estejam, move-se o boneco para uma posição que evite a colisão.

void moveCena(int s): apesar de simples, esta função é responsável por alterar os parâmetros de translação da cena e dos objetos, gerando a sensação de movimento do boneco.

void iniciaNeve(): com esta função definimos a quantidade e espaçamento dos flocos de neve, através de uma matriz.

void iniciaPosicaoFloco(int i, int j): gera coordenadas aleatórias para desenhar os flocos de neve.

void desenhaNeve(): utilizando a estrutura dos flocos de neve, desenha-se o floco enquanto é visível na janela, reiniciando a sua posição quando tal não se verifica. A alteração de y ao longo do tempo gera o efeito do movimento da neve.

Embora as funções que ainda não foram referidas sejam importantes para o correcto funcionamento do programa, não nos expandimos na sua descrição já que, essencialmente, são funções cujo objectivo é desenhar os vários elementos do jogo, recorrendo às funções apresentadas nas aulas e expressões matemáticas conhecidas.

4. APRESENTAÇÃO DE UMA SESSÃO DE UTILIZAÇÃO

4.1 Entrada

Quando se entra na aplicação criada, as instruções são imediatamente exibidas. Para ver a janela com o jogo, carrega-se na tecla ESC.

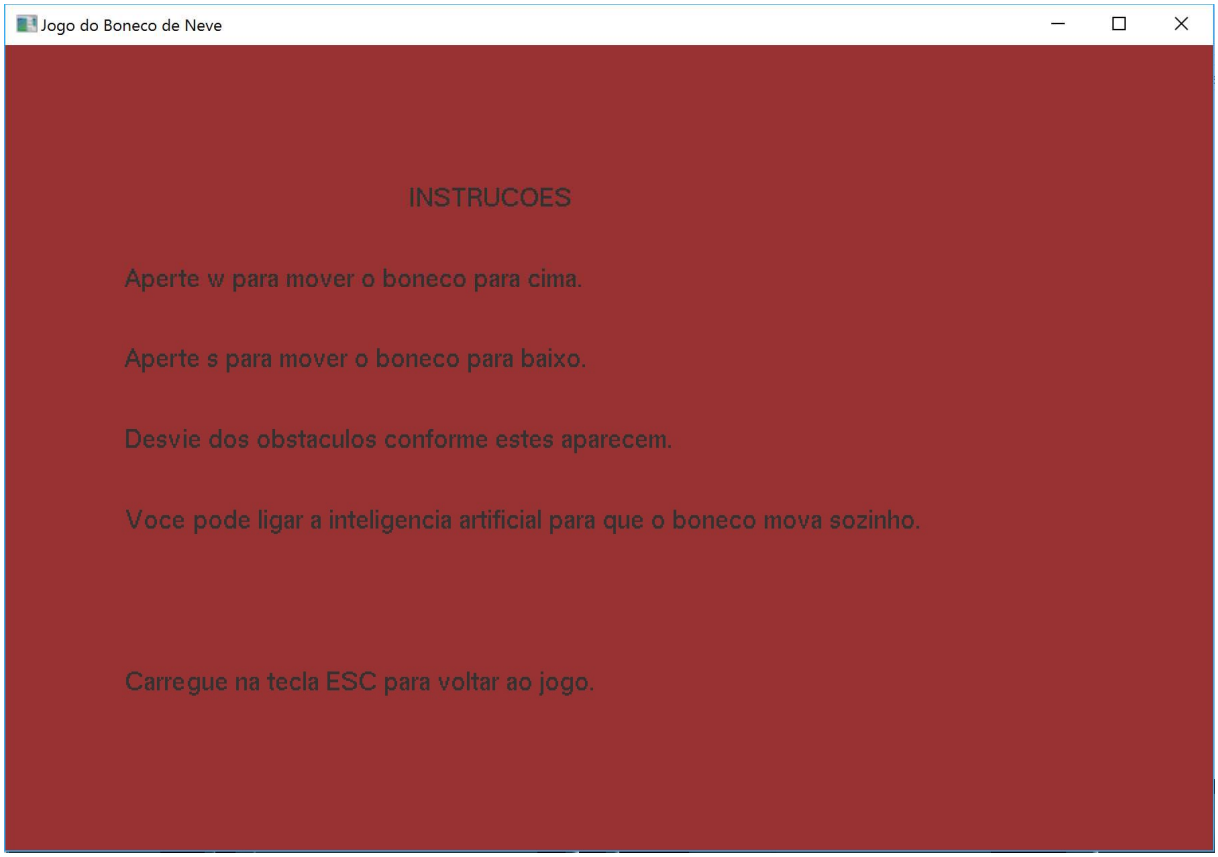

Figura 2 - Instruções

4.2 Utilização do jogo

O utilizador pode controlar a posição do boneco de neve através das teclas ‘s’ e ‘w’, que deslocam o boneco para baixo e para cima, respectivamente.

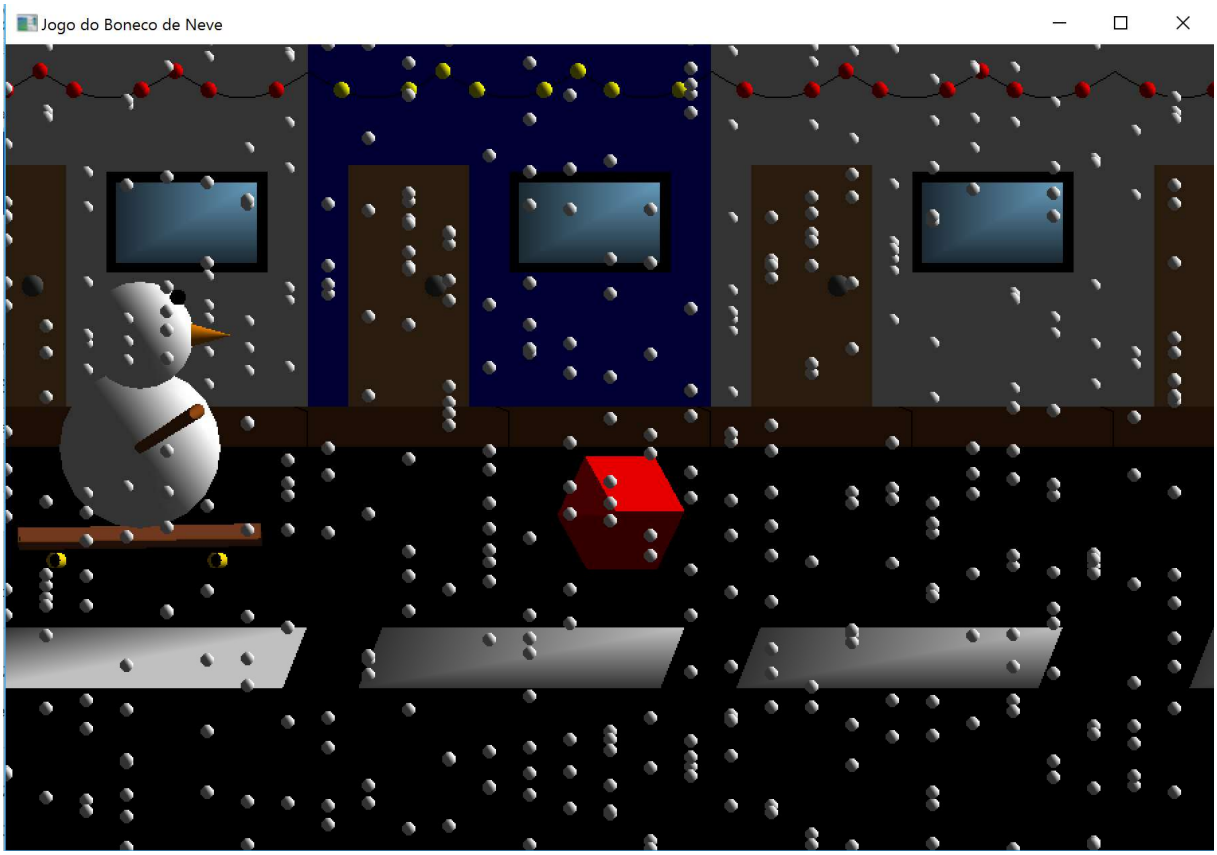

Figura 3 - Jogo

Carregando no botão direito do rato, acede-se ao menu, que permite visualizar as instruções, selecionar a resolução automática, reiniciar o jogo ou sair do jogo.

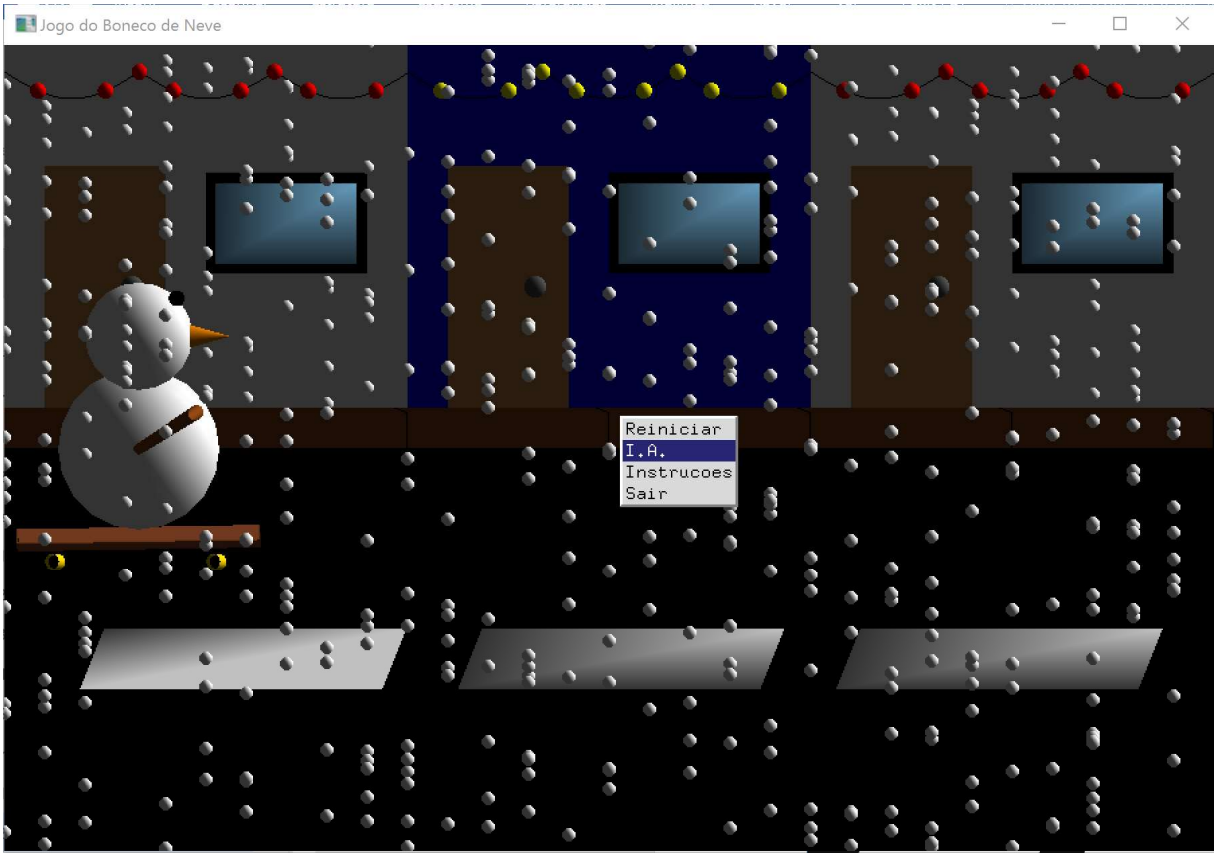

Figura 4 - Menu

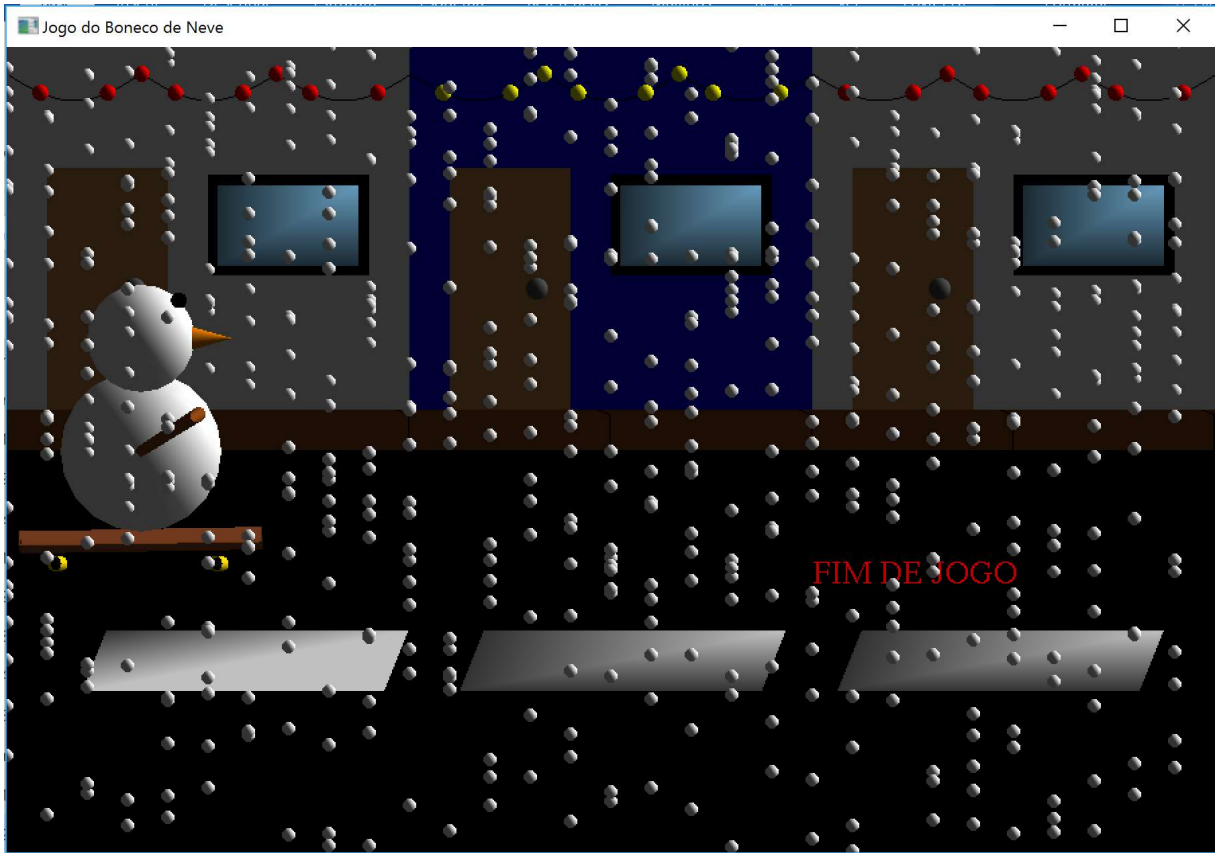

Figura 5 - Fim do jogo

5. CONCLUSÕES

A concretização do presente trabalho possibilitou-nos a consolidação dos conhecimentos adquiridos nas aulas de Visualização Computacional. Permitiu-nos, também, o aprofundamento de algumas áreas do OpenGL, principalmente a visualização em R3 e a iluminação.

Seria simples criar um jogo com mais níveis, aumentando a velocidade de deslocação do boneco ou diminuindo a distância entre os obstáculos ao longo dos níveis.

Julgamos ter conseguido cumprir, de forma original, todos os objetivos do trabalho, criando um programa de fácil utilização. O maior desafio do trabalho foi a implementação da visualização em R3 e da iluminação, já que se tratam de assuntos mais complexos e desafiantes.

6. BIBLIOGRAFIA

Shreiner, D., Sellers, G., Kessenich, J. e Licea-Kane, B., OpenGL Programming Guide, Addison-Wesley Publishing Company, 8 Ed., 1996

Hearn, D.D., Baker, M.P. e Carithers, W., Computer Graphics with OpenGL, 4 Ed., 2014