Departamento de Matemática

FCTUC

Relatório

Visualização Computacional

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

Trabalho 2 – Animação R2 Ecopontos

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

Data:
06-12-2016

ÍNDICE

1. SUMÁRIO 3

2. INTRODUÇÃO 4

2.1 VISÃO 4

2.2 ANÁLISE E ESPECIFICAÇÃO DE REQUISITOS 4

3. DESENVOLVIMENTO 6

3.1 CONCEÇÃO 6

3.2 ARQUITETURA DA SOLUÇÃO 6

3.3 INTERFACE COM O UTILIZADOR 7

3.4 ESTRUTURAS DE DADOS 8

3.5 PRINCIPAIS FUNÇÕES 9

4. APRESENTAÇÃO DE UMA SESSÃO DE UTILIZAÇÃO 13

4.1 ENTRADA 13

4.2 UTILIZAÇÃO DO JOGO 13

5. CONCLUSÕES 17

6. BIBLIOGRAFIA 18

1. SUMÁRIO

O trabalho efectuado visa a utilização dos recursos disponibilizados pela linguagem C e pelo OpenGL no desenvolvimento de uma animação interactiva em R2, referente à colocação de objectos nos três contentores do ecoponto.

A aplicação deve gerar objectos que possam ir para os três contentores de um ecoponto e a(s) propriedade(s) dos objectos que permitem a selecção do contentor deve ser perceptível pelo utilizador. Deve, ainda, ser original e ergonómico, 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. Enumera os requisitos funcionais e não funcionais, menciona as principais decisões conceptuais e a arquitetura da solução, as estruturas de dados e as funções mais relevantes. É, ainda, apresentada uma solução de utilização da aplicação.

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 animação interactiva em R2 relativa à colocação de objectos nos três contentores do ecoponto.

A aplicação deve gerar objectos que possam ir para os 3 contentores de um ecoponto e a(s) propriedade(s) dos objectos que permitem a selecção do contentor deve ser perceptível pelo utilizador. Deve, ainda, ser original e ergonómico, disponibilizando instruções que facilitem a sua utilização.

Tendo em conta os objectivos supramencionado, construímos um jogo em R2 que permite que ao utilizador arrastar os objectos até ao ecoponto certo ou escolher a resolução automática de cada objecto. No segundo caso, o utilizador tem apenas de indicar que tipo de lixo pretende resolver automaticamente e os objectos desse tipo são movidos até ao ecoponto correspondente. O jogo apresenta um botão “INSTRUCOES” que, quando pressionado, exibe as informações necessárias sobre o funcionamento do jogo.

2.2 Análise e Especificação de Requisitos

Requisitos funcionais

- O programa deve permitir que o utilizador seleccione com o rato o objecto que pretende colocar no ecoponto certo.
- O programa deve permitir que o utilizador arraste com o rato os objectos até ao ecoponto correcto.
- O programa deve indicar quando o objecto não é largado no ecoponto certo.
- O programa deve permitir uma resolução automática caso o utilizador não saiba a que ecoponto pertence um objecto.

Requisitos não funcionais

- O programa deve correr no sistema operativo Windows.

- O programa deve ser desenvolvido recorrendo à linguagem C e ao OpenGL.
- Após ler as instruções, o utilizador deve conseguir usar a aplicação facilmente, não sendo exigido conhecimentos para além da informação disponibilizada.

3. DESENVOLVIMENTO

3.1 Conceção

Tendo em conta o tema do trabalho, a implementação dos objectivos foi realizada de forma a criar um jogo destinado a ajudar crianças a fazer a separação e reciclagem do lixo. O formato de jogo torna a aplicação mais apelativa e desafiante para a criança

Deste modo, a aplicação deve ser fácil de utilizar por qualquer pessoa, mesmo que não possua quaisquer conhecimentos na área, pelo que as instruções do jogo devem estar disponíveis a qualquer momento.

Tendo em conta que um jogo didáctico deve facilitar a aprendizagem do assunto em questão, deve ser possível colocar os objectos no ecoponto certo automaticamente quando não se sabe a qual pertence. Apresenta-se ainda uma lista dos objectos já separados, para ajudar a assimilar os conhecimentos.

3.2 Arquitetura da solução

O programa divide-se essencialmente em duas partes: desenhar os ecopontos e objectos e colocar os objectos no ecoponto adequado.

Tendo em vista estes objectivos, foram criados os ficheiros “sctObj.h”, “objetos.h”, “resolveLixo.h”, “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) e para distinguir o ecoponto a que pertence. Contém, ainda, uma função “initObj” que permite inicializar uma variável do tipo “Objeto”, atribuindo os valores indicados aos elementos da estrutura.

“objeto.h” contém as variáveis e funções necessárias para desenhar o que aparece na janela: os ecopontos (“desenhaEcoponto”), os objectos a colocar no lixo (“desenhaGafPlast”, “desenhaFolha”, “desenhaGafVidro”, “desenhaLata”, “desenhaCaixa”, “desenhaFrasco”), o fundo (“desenhaSol”, “desenhaNuvens”, “desenhaChao”), o limite entre a área do jogo e a área com informação sobre o jogo (“desenhaBorda”), a área com informação sobre o jogo (“desenhaRetangulo”, “botaoInstrucoes”). As funções que permitem escrever na janela também pertencem a este ficheiro já

que são usadas em "botaoInstrucoes". Na maioria das funções apenas é necessário usar como argumentos as coordenadas da posição em que se pretende desenhar o objecto e o tamanho (largura e altura) do mesmo, embora em algumas funções também seja necessário indicar a cor.

"resolveLixo.h" contém as várias funções necessárias para colocar o lixo no ecoponto certo através de dois métodos: por arrastamento com o rato ("objectoClique", "controlaMouse", "controlaArrasto") e resolução automática, através das opções do menu ("timelLixo", "resolveLixoAmarelo", "resolveLixoAzul", "resolveLixoVerde", "controloAutomatico"). Contém, ainda, funções ("ecopontoCerto", "testaLocal") que permitem determinar se o objecto foi largado no ecoponto certo e retornam o objecto à posição original quando tal não se verifica. A função "controlaMouse" identifica ainda as situações em que o utilizador carrega no botão "INSTRUCOES".

"display.h" contém as várias funções necessárias para inicializar as variáveis globais ("iniciaGlobais", "iniciaVariaveis"), controlar alterações no tamanho da janela ("controlaViewport", "reshape"), controlar as condições do jogo ("cronometro", "mudancaNivel", "fimDeJogo"), permitir que o utilizador interaja com o jogo ("controlaMenu", "controlaKeyboard") e visualizar o conteúdo da área de jogo e da área com informação sobre o decorrer do jogo ("init", "moveNuvemG1", "display").

"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 aparecem assim que se inicia a aplicação e podem ser consultadas em qualquer momento do jogo, carregando no botão "INSTRUCOES". Aparece ainda uma descrição de cada objecto quando se carrega com o botão esquerdo do rato para auxiliar a identificação.

Quando a janela exibe o jogo, os únicos elementos que o utilizador pode alterar são os objectos a colocar no lixo, arrastando-os até ao ecoponto adequado. Contudo, através do menu, é ainda possível colocar o lixo automaticamente no ecoponto certo, reiniciar o jogo ou sair do jogo.

A janela apresenta ainda informação relativa ao nível e tempo decorrido e os objectos que já foram depositados em cada ecoponto.

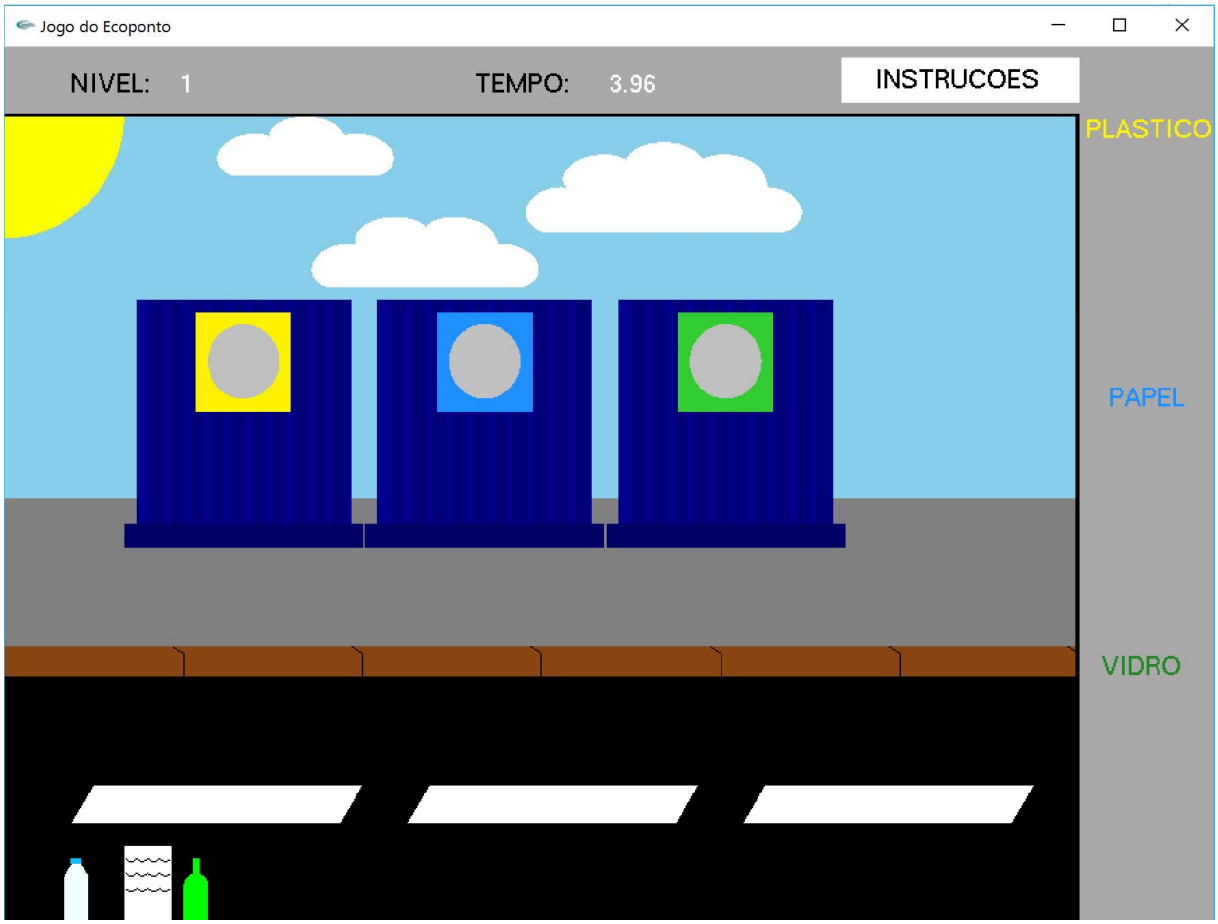

Figura 1 - Início do jogo

3.4 Estruturas de dados

Os principais dados, usados em várias funções, são variáveis globais que representam características como tamanho da janela e a posição e 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 objectos a colocar no lixo. Estes têm o mesmo tipo de características, que são usadas em várias funções, pelo que criamos uma estrutura "Objeto".

```
Estrutura {
        double Ox;      //coordenadas originais do objecto
        double Oy;
        double sW;      //largura do objecto
        double sH;      //altura do objecto
        double dX;      //coordenadas usadas para desenhar o objecto
        double dY;
```

```
  int nobj;          //número correspondente à ordenação no conjunto dos objectos
  int lixo;          //número correspondente ao ecoponto correcto
  int noEcoponto;    //numero que indica se o lixo se encontra
                     corresponde ao ecoponto certo ou não
} Objeto
```

3.5 Principais funções

Objeto initObj(double Ox, double Oy, double sW, double sH, int nobj, int lixo) : embora esta função não seja muito complexa, servindo apenas para atribuir os valores pretendidos às várias variáveis da estrutura **Objeto**, é essencial na construção do programa, já que é usada várias vezes para alterar os valores da posição usada para desenhar os objectos sem que seja necessário recorrer a diversas linhas de código;

int ecopontoCerto(double x, double y, const int lixo) : esta função permite verificar se um objecto é largado na área indicada do ecoponto adequado, comparando as coordenadas do objecto e as coordenadas que delimitam os quadrados coloridos dos ecopontos. A função devolve o número inteiro 1 quando o objecto é largado no quadrado certo e 0 em caso contrário;

void testaLocal() : quando um objeto ainda não foi depositado no ecoponto certo (obj->noEcoponto != 1), recorre-se à função **ecopontoCerto** para verificar se o mesmo é largado na área pretendida do ecoponto adequado. Quando tal não acontece e o objecto não se encontra na posição original, restaura-se a posição, atribuindo-se as coordenadas originais às coordenadas utilizadas para desenhar o objecto, recorrendo-se à função **initObj**;

int objetoClique(Objeto *objEsc, GLint x, GLint y) : nesta função comparam-se as coordenadas do rato quando se carrega no botão esquerdo com as coordenadas que delimitam a área de um objecto (**Objeto objEsc**), devolvendo o número inteiro 1 quando um objecto é selecionado e 0 caso contrário. Quando um objecto é seleccionado, atribui-se **objEsc** à variável **obj** e guardam-se as coordenadas do rato nas variáveis **startx** e **starty**. É de destacar a importância de recorrer à razão entre a largura e altura da janela e a largura e altura da área de jogo, pois, caso esta razão não fosse considerada nos cálculos, poderiam ocorrer erros na selecção dos objectos após uma alteração no tamanho da janela;

void controlaMouse(GLint button, GLint state, GLint x, GLint y) : quando o botão esquerdo do rato é pressionado, comparam-se as coordenadas do rato com as coordenadas que delimitam o botão "INSTRUCOES", atribuindo valor 1 à variável **mostraInstrucoes** quando o botão é pressionado. Recorre-

se à função **objetoClique** para verificar se algum objeto foi seleccionado, atribuindo-se à variável **moving** o valor desta função. Quando se solta o botão esquerdo do rato, atribui-se o valor 0 à variável **moving**, já que não se pode mover nenhum objecto, e recorre-se à função **testaLocal** para verificar se o objecto foi largado na área destinada ao depósito no ecoponto adequado.

```
PROCEDIMENTO controlaMouse(GLinteiro button, GLintinteiro state, GLintinteiro x, GLintinteiro y) {
    REAL nL <- largura / viewRetCimaLargura
    REAL nH <- altura / viewRetCimaAltura
        SE button = BOTÃO_ESQUERDO E state == BOTÃO_PRESSIONADO
            SE x >= xBotaoInstrucoes/nL E x <= (xBotaoInstrucoes + wBotaoInstrucoes)/nL E y <=
                (yBotaoInstrucoes + hBotaoInstrucoes)/nH E y >= (yBotaoInstrucoes)/nH)
            ENTÃO mostraInstrucoes <- 1
                    elapsedInstrucoes <- elapsedInstrucoes glutGet(TEMPO_DECORRIDO) / 1000.0
            FIMSE
            moving <- objetoClique(&gaf, x, y);
            SE moving = 0
            ENTÃO moving <- objetoClique(&paper, x, y)
            FIMSE
            SE moving = 0
            ENTÃO moving <- objetoClique(&gafV, x, y)
            FIMSE
            SE moving = 0
            ENTÃO moving <- objetoClique(&tin, x, y)
            FIMSE
            SE moving = 0
            ENTÃO moving <- objetoClique(&box, x, y)
            FIMSE
            SE moving = 0
            ENTÃO moving <- objetoClique(&bottle, x, y)
            FIMSE
        SENÃO testaLocal()
                Moving <- 0
                Nome <- " "
        FIMSE
        glutPostRedisplay();
FIMPROC
```

void controlaArrasto(int x, int y) : se a variável moving tem o valor 1, isto é, se um objecto for seleccionado com o rato, as coordenadas utilizadas para desenhar o objecto vão sendo actualizadas recorrendo-se às coordenadas do rato. É, ainda, indicada a descrição a aparecer para cada objecto, usando-se *switch case*;

```
PROCEDIMENTO controlaArrasto(inteiro x, inteiro y)
    SE moving = 1
    ENTÃO obj->dX <- largura / viewUmLargura *(x - startx) + obj->Ox
```

```
            obj->dY <- altura / viewUmAltura * (-y + starty) + obj->Oy;
            ESCOLHA (obj->nobj)
            INÍCIO
                CASO 3: nome <- "Garrafa de plastico"
                CASO 4: nome <- "Folha de papel"
                CASO 5: nome <- "Garrafa de vidro"
                CASO 6: nome <- "Lata"
                CASO 7: nome <- "Caixa de cartao"
                CASO 8: nome <- "Frasco de vidro"
            FIM
            glutPostRedisplay()
    FIMSE
FIMPROC
```

void timerLixo(int op) : esta função gera o movimento, não controlado pelo utilizador, de um objecto até ao ecoponto adequado, actualizando as coordenadas a usar para desenhar o objecto, enquanto o este não se encontra no ecoponto certo e utilizando a função *glutTimerFunc* e **timerLixo** recursivamente;

void resolveLixoAmarelo(int nObj) : esta função pretende colocar um objecto de plástico no ecoponto amarelo, começando por verificar de que objecto deste tipo se trata. De seguida, recorre-se à função **timerLixo** para gerar o movimento do objecto até ao ecoponto adequado;

void resolveLixoAzul(int nObj) : esta função é semelhante à anterior, movendo os objectos de papel;

void resolveLixoVerde(int nObj) : esta função é semelhante à anterior, movendo objectos de vidro;

void controlaAutomatico(GLint entry) : quando se indica o tipo de lixo que se pretende colocar no ecoponto, esta função coloca o objecto adequado no ecoponto indicado recorrendo às três funções anteriores e tendo em consideração os objectos presentes em cada nível.

```
PROCEDIMENTO controloAutomatico (GLinteiro entry)
    ESCOLHA (entry)
    INÍCIO
        CASO 0 : SE nivel = 1
                 ENTÃO resolveLixoAmarelo(gafPlast)
                 SENÃO resolveLixoAmarelo(lata)
                 FIMSE
        CASO 1 : SE nivel = 1
                 ENTÃO resolveLixoAzul(papel)
                 SENÃO resolveLixoAzul(caixa)
                 FIMSE
        CASO 2 : SE nivel = 1
                 ENTÃO resolveLixoVidro(gafVidro)
                 SENÃO resolveLixoVerde(fraco)
```

```
                FIMSE
    FIM
FIMPROC
```

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 de dois tipos:

- funções cujo objectivo é desenhar os vários elementos do jogo recorrendo às funções apresentadas nas aulas e expressões matemáticas conhecidas;
- funções que são apenas uma adaptação às necessidades do programa de algoritmos explicados pelo professor.

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.

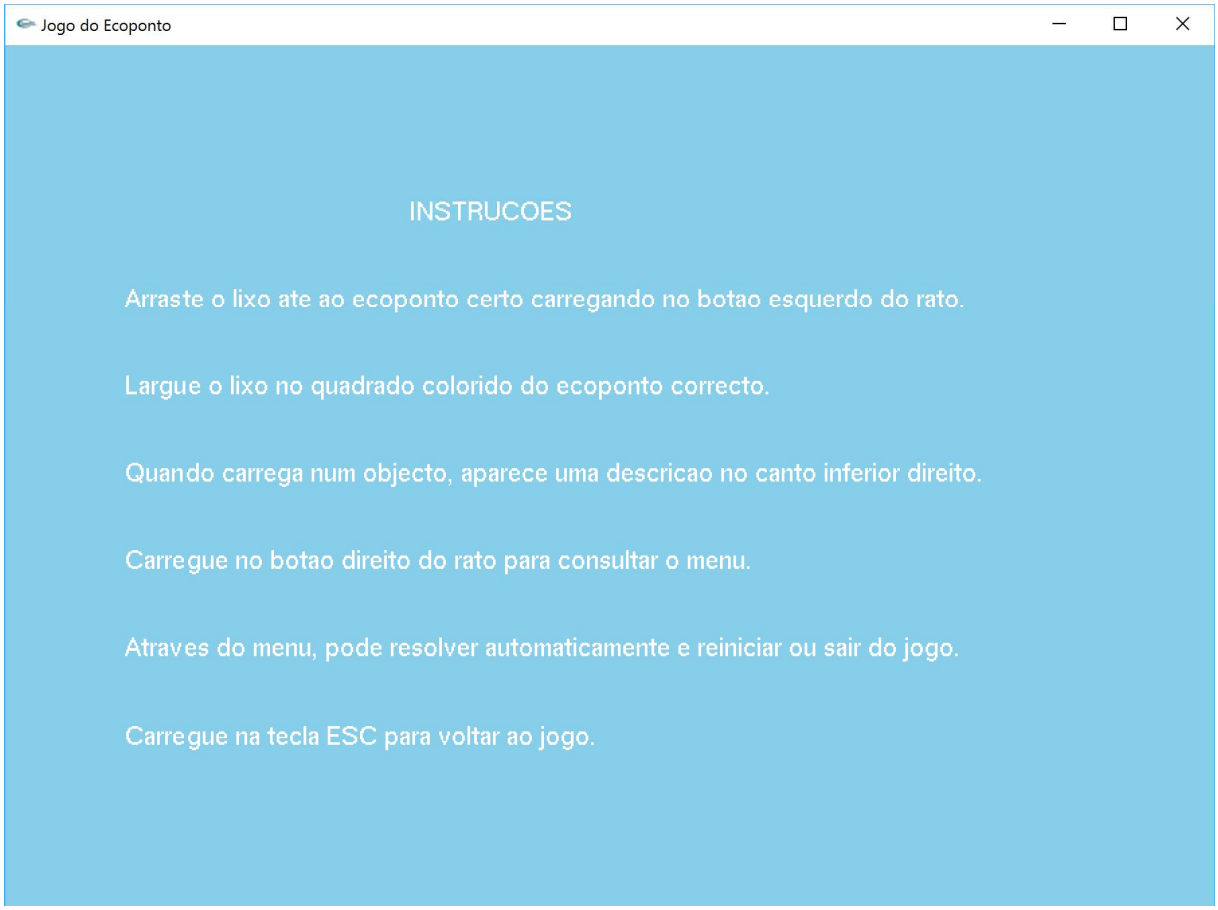

Figura 2 - Instruções

4.2 Utilização do jogo

O utilizador pode arrastar os objectos a colocar no ecoponto com o botão esquerdo do rato. Enquanto um objecto está selecionado, aparece uma descrição no canto inferior direito. Quando os objectos não são largados no ecoponto adequado, voltam à posição original.

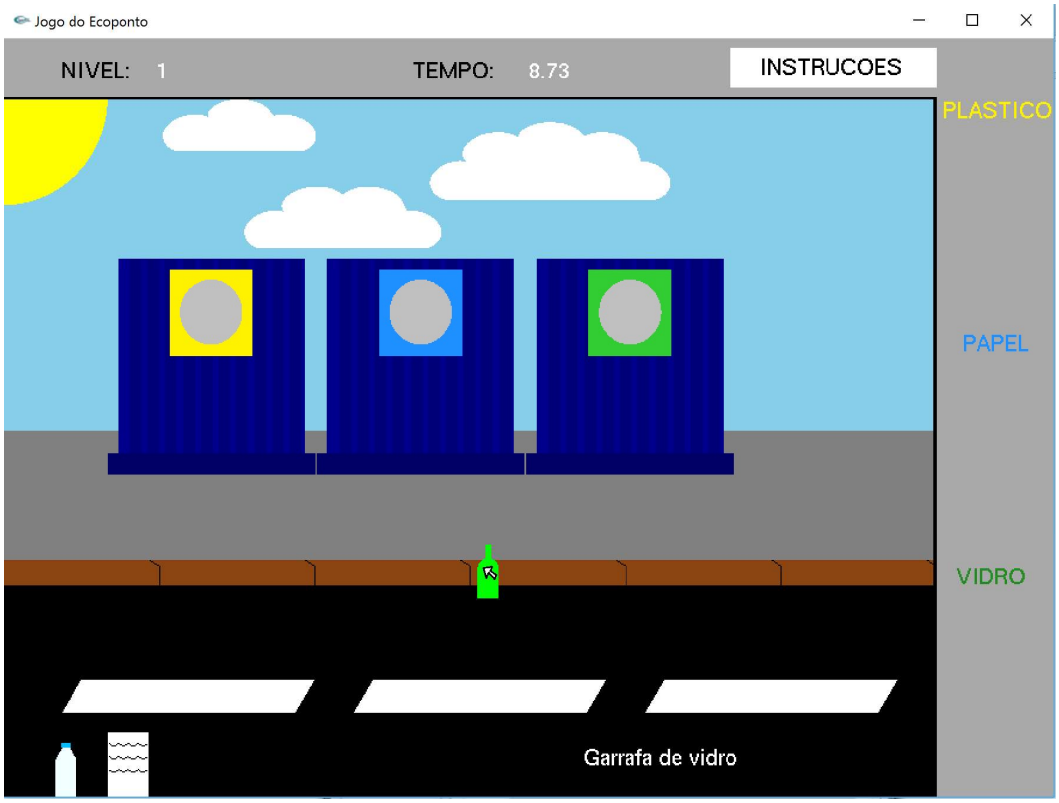

Figura 3 - Arrastamento de um objecto

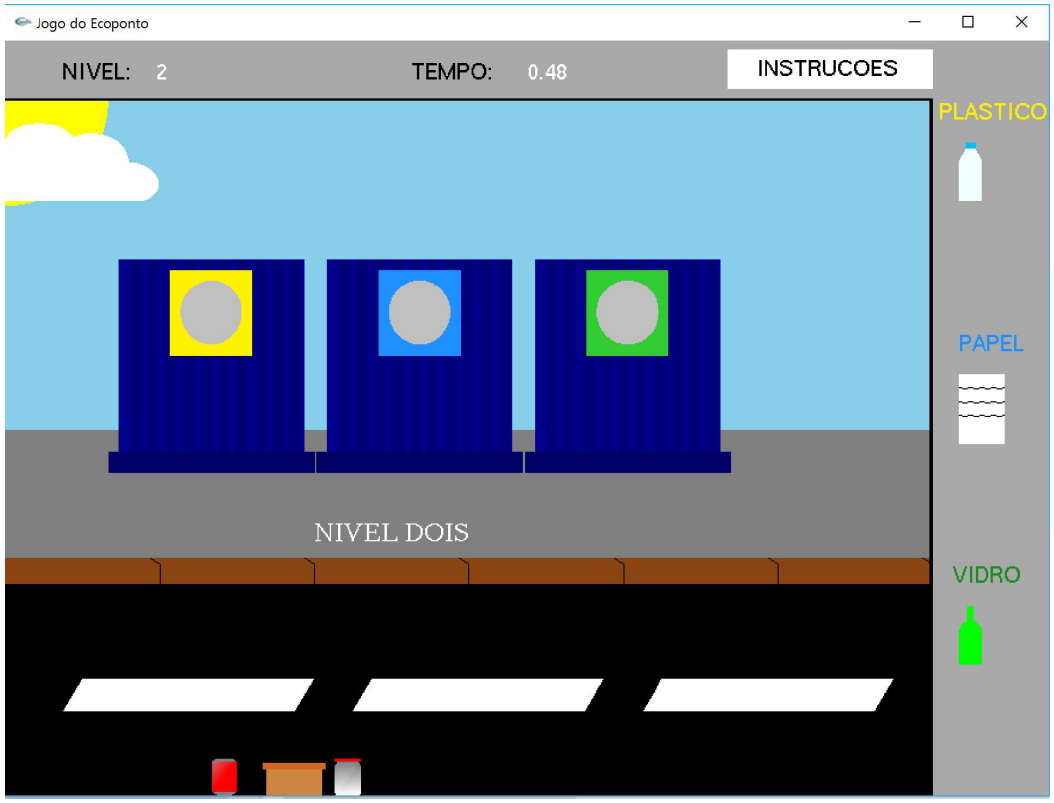

Figura 4 - Mudança de nível

Carregando no botão direito do rato, acede-se ao menu, que permite colocar os objectos de cada tipo no ecoponto adequado, reiniciar o jogo ou sair do jogo.

Figura 5 - Menu e submenu

Informação como nível e tempo de jogo e objectos já separados é sempre visível no decorrer do jogo.

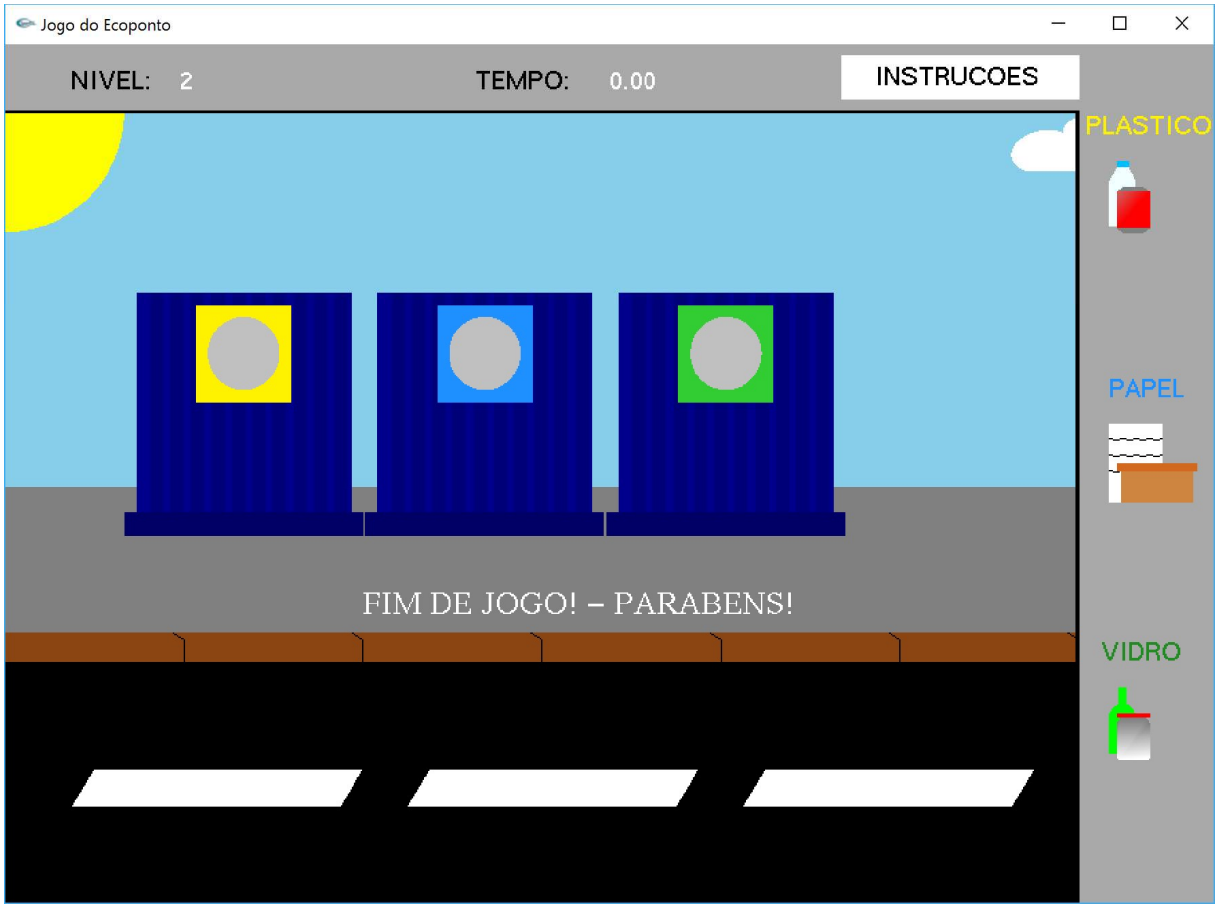

Figura 6 - Fim do jogo

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 utilização do rato e a geração de movimentos.

Embora o jogo criado contenha apenas dois níveis, tornar-se-ia simples expandir o jogo, já que apenas seria necessário desenhar mais objectos e, de seguida, usar as funções já criadas.

Julgamos ter conseguido cumprir, de forma original, todos os objetivos do trabalho, criando um programa de fácil utilização. Destacamos, também, a aplicação didática do mesmo, já que a sua utilização possibilita, que de forma lúdica, os utilizadores apreendam e/ou consolidem conhecimentos já adquiridos, potenciado assim a formação de cidadãos mais conscientes e responsáveis.

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