**گزارش پروژه درس گرافیک کامپیوتری – شماره دو**

**محمّدرضا ابوالحسنی 610389084**

شرح فعّالیت ها و تغییرات صورت گرفته :

1. تمامی کارهای مربوط به لود کردن فایل های .obj و نمایش اشیاء که در قسمت های مختلف کد پخش بود، در یک کلاس به نام objHandler جمع آوری شد.

objHandler یک سیستم اتوماتیک برای لود کردن تمام مدل های استفاده شده در بازی دارد. وقتی یک شیء از نوع objHandler ساخته می شود، ابتدا مشخصات همه مدل ها را از روی یک فایل متنی می خواند و ذخیره می کند. این مشخصات شامل نام، آدرس فایل .obj ، یک عدد صفر یا یک که نشان می دهد مدل تکسچر دارد یا خیر و عرض (مؤلفه x) دلخواه برای مدل (جهت scale کردن) می باشد. به این صورت:

ID NAME FILE\_NAME TEXTURE? WIDTH(X)

----------------------------------------------------------------------------------------

00 OLD\_BOX ../obj/box/old\_box.obj 1 02.0

01 SOFT\_BOX ../obj/box/soft\_box.obj 1 02.0

02 COIN ../obj/power/coin.obj 0 00.5

03 MAGNET ../obj/power/magnet.obj 0 00.1

04 CONTAINER\_SHIP ../obj/ship/container\_ship.obj 0 15.0

05 CRUISE\_SHIP ../obj/ship/cruise\_ship.obj 0 05.0

06 JETTY ../obj/structure/jetty.obj 0 06.0

07 T\_CRANE ../obj/structure/t\_crane.obj 0 02.8

08 LIFT\_TRUCK ../obj/truck/lift\_truck.obj 0 06.1

09 BLUE\_TRUCK ../obj/truck/blue\_truck.obj 0 05.0

10 YELLOW\_TRUCK ../obj/truck/yellow\_truck.obj 0 05.0

11 OBSTACLE\_LOW ../obj/obstacle/obstacle\_low.obj 0 02.0

12 OBSTACLE\_TOP ../obj/obstacle/obstacle\_top.obj 0 02.0

13 PIPE ../obj/structure/pipe.obj 0 00.5

14 PIPE\_L ../obj/structure/pipe\_l.obj 0 01.5

15 CONTAINER\_RED ../obj/box/container\_red.obj 0 05.0

16 CONTAINER\_BLUE ../obj/box/container\_blue.obj 0 05.0

17 TOW\_TRACTOR\_B ../obj/truck/tow\_tractor\_b.obj 0 05.0

18 TOW\_TRACTOR\_W ../obj/truck/tow\_tractor\_w.obj 0 05.0

19 ROBOT ../obj/robot/robot.obj 0 01.3

20 ROBOT\_HEAD ../obj/robot/robot\_head.obj 0 00.6

21 ROBOT\_NECK ../obj/robot/robot\_neck.obj 0 00.2

22 ROBOT\_BODY ../obj/robot/robot\_body.obj 0 01.3

23 ROBOT\_BACK ../obj/robot/robot\_back.obj 0 00.4

24 ROBOT\_WHEEL ../obj/robot/robot\_wheel.obj 0 00.7

EOF

سپس مدل ها را با توجّه به این اطلاعات لود می کند، آنها را نرمالسازی می کند، لیست های نمایش (display lists – در گزارش قبل به طور مبسوط به آنها پرداخته شد) را تشکیل می دهد و سپس مدل ها را پاک و حافظه را آزاد می کند و از این به بعد فقط از لیست های نمایش برای رسم مدل ها استفاده می کند.

دسترسی از بیرون به یک شیء از نوع objHandler تنها از طریق سه روال:

string get\_obj\_name(int);

void draw(ObjName);

void get\_dimentions(ObjName, GLfloat\*);

صورت می پذیرد. در گزارش قبل اشاره شد که ObjName یک نوع داده ای ترتیبی (enumerated) است که با مدل ها در تناظر یک به یک قرار می گیرد و عضو آخر آن تعداد مدل ها را بر می گرداند :

enum ObjName {OLD\_BOX, SOFT\_BOX,

COIN, MAGNET,

CONTAINER\_SHIP, CRUISE\_SHIP,

JETTY, T\_CRANE, LIFT\_TRUCK,

...

/\*add other obj names here\*/

NUM\_OF\_OBJS

};

روال اول یک عدد صحیح می گیرد و اسم مدل متناظر با آن را بر می گرداند. روال دوم یک عدد از نوع ObjName می گیرد و مدل متناظر را رسم می کند و روال سوم یک عدد از نوع ObjName و اشاره گر به یک آرایه به طول سه از اعداد اعشاری می گیرد و ابعاد مدل متناظر را در آرایه درج می کند.

در ذیل پیاده سازی این کلاس را می بینیم:

// file: objHandler.h

#include <string>

#include <fstream>

#include <iostream>

#include <iomanip>

#include <GL/glut.h>

#include "../glmwithtexture/glm.h"

using namespace std;

#ifndef OBJ\_HANDLER

#define OBJ\_HANDLER

enum ObjName {OLD\_BOX, SOFT\_BOX,

COIN, MAGNET,

CONTAINER\_SHIP, CRUISE\_SHIP,

JETTY, T\_CRANE, LIFT\_TRUCK,

BLUE\_TRUCK, YELLOW\_TRUCK,

OBSTACLE\_LOW, OBSTACLE\_TOP,

PIPE, PIPE\_L,

CONTAINER\_RED, CONTAINER\_BLUE,

TOW\_TRACTOR\_B, TOW\_TRACTOR\_W,

ROBOT, ROBOT\_HEAD, ROBOT\_NECK,

ROBOT\_BODY, ROBOT\_BACK,

ROBOT\_WHEEL,

/\*add other obj names here\*/

NUM\_OF\_OBJS

};

class objHandler

{

public:

objHandler(string);

~objHandler();

string get\_obj\_name(int);

void get\_dimentions(ObjName, GLfloat\*);

void draw(ObjName);

private:

GLMmodel\* model[NUM\_OF\_OBJS];

string obj\_names[NUM\_OF\_OBJS];

string filenames[NUM\_OF\_OBJS];

bool has\_text[NUM\_OF\_OBJS];

GLfloat default\_widths[NUM\_OF\_OBJS];

GLfloat dimentions[NUM\_OF\_OBJS][3];

GLuint disp\_list\_ids[NUM\_OF\_OBJS];

};

#endif

// file: objHandler.cpp

#include "objHandler.h"

objHandler::objHandler(string desc\_filename){

//read objects descriptions from file

ifstream fin(desc\_filename.c\_str());

fin.ignore(100, '\n'); //first line is trash!

fin.ignore(100, '\n'); //so is the second one!

int p;

for (int i = 0; i < NUM\_OF\_OBJS; i++)

{

fin >> p;

fin >> obj\_names[i];

fin >> filenames[i];

fin >> has\_text[i];

fin >> default\_widths[i];

}

//load actual models

GLfloat minMax[6];

GLfloat trans[3];

GLfloat scl;

for(int i = 0; i < NUM\_OF\_OBJS; i++){

//read .obj file

model[i] = glmReadOBJ(filenames[i].c\_str());

glmVertexNormals(model[i], 90.0, GL\_TRUE);

/\*normalize the model, so that the y-axis

will cuts model's center and minimum

y coordinate will be 0. also scales the

model to feat default width\*/

glmMinMax(model[i], minMax);

trans[0] = -(minMax[0] + minMax[3]) / 2;

trans[1] = -minMax[1];

trans[2] = -(minMax[2] + minMax[5]) / 2;

glmTranslate(model[i], trans);

scl = default\_widths[i] / (minMax[3] - minMax[0]);

if(scl < 0) scl \*= -1;

glmScale(model[i], scl);

//get and store dimentions of the model

glmDimensions(model[i], dimentions[i]);

//make display list and store it's ID

disp\_list\_ids[i] = glmList(model[i], GLM\_SMOOTH|GLM\_MATERIAL);

/\* in case of no texture,

we don"t need the model any more!\*/

if(!has\_text[i]){

/\*we are done with this model,

let's delete it and free memory\*/

glmDelete(model[i]);

}

}

}

objHandler::~objHandler(){

}

string objHandler::get\_obj\_name(int obj\_id){

return obj\_names[obj\_id];

}

/\* dims: array of 3 GLfloats\*/

void objHandler::get\_dimentions(ObjName OBJ, GLfloat\* dims){

dims[0] = dimentions[OBJ][0];

dims[1] = dimentions[OBJ][1];

dims[2] = dimentions[OBJ][2];

}

void objHandler::draw(ObjName OBJ){

if(has\_text[OBJ]){

glmDraw(model[OBJ], GLM\_SMOOTH|GLM\_MATERIAL|GLM\_TEXTURE);

}

else{

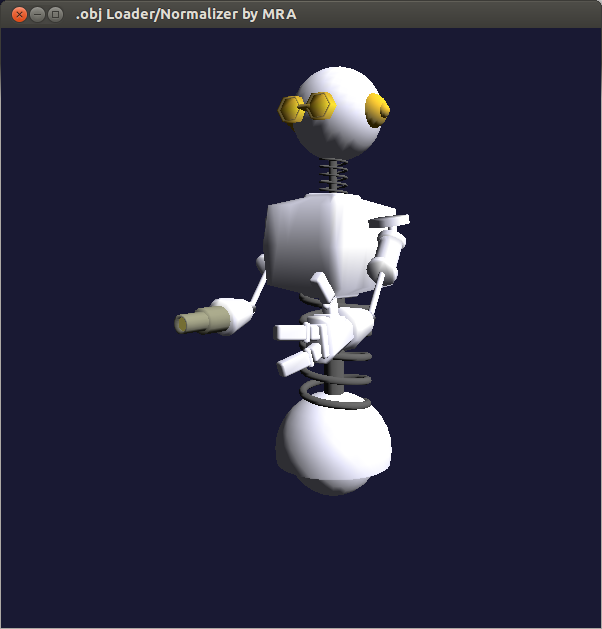
glCallList(disp\_list\_ids[OBJ]);

}

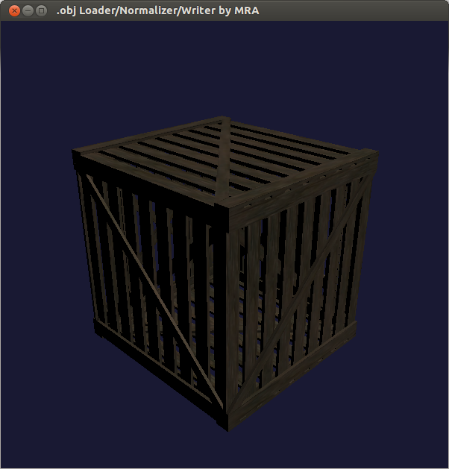
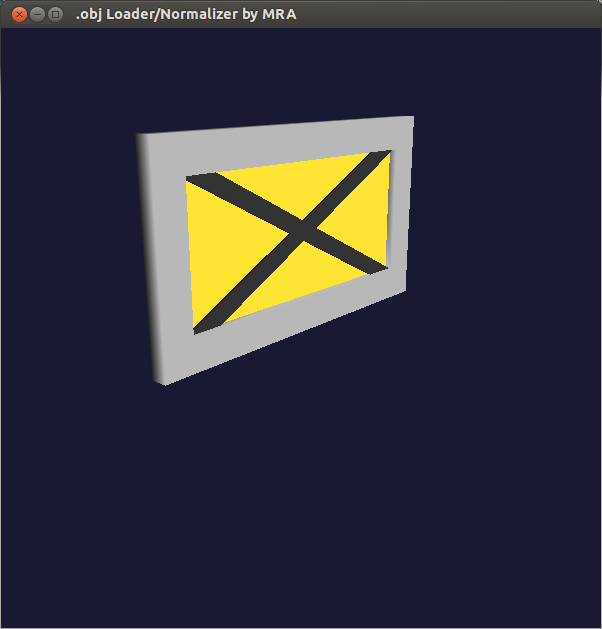
}

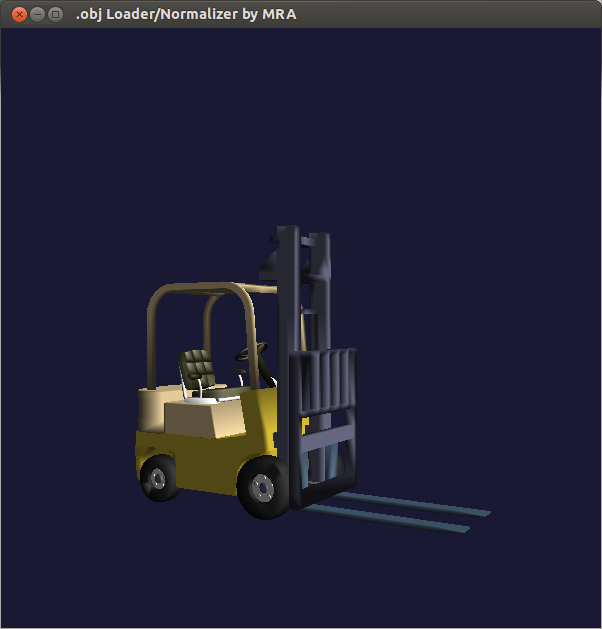
1. Edit و طرّاحی اشیاء محیط بازی:

شخصیت اصلی(robot):

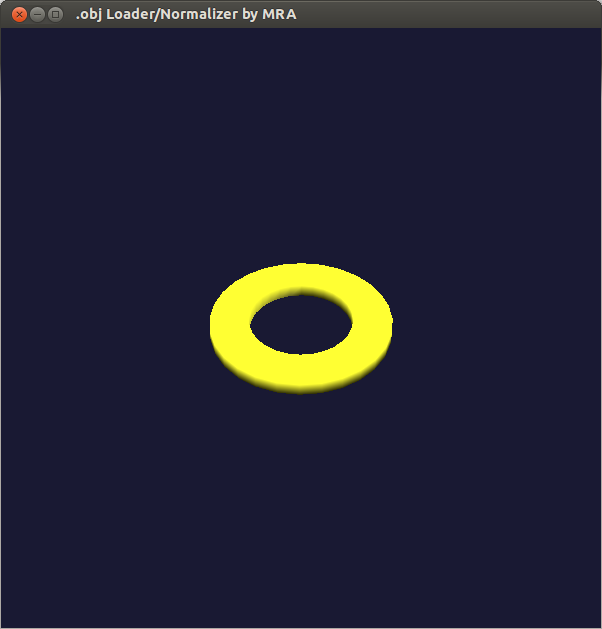
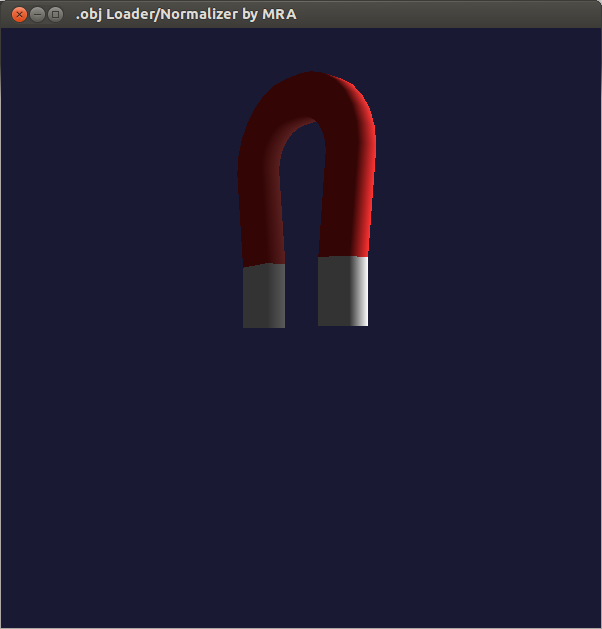


موانع(obstacles):





قدرت ها(powerups):



اشیاء تزئینی (decorative):

