#pragma comment(lib, "irrklang.lib")//audio library

#include <iostream>

#include <windows.h>

#include <stdio.h>

#include <mmsystem.h>

#include "lodepng.h"

#include <time.h>

#include <stdlib.h>

#include<d:\irrklang\irrKlang-1.5.0\include\irrKlang.h>

#include <glut.h>

using namespace std; //for vector and string useage in texture maping

using namespace std;

using namespace irrklang;

///select Properties -> C/C++ -> Preprocessor -> Preprocessor definitions and append GLUT\_BUILDING\_LIB to the

int menucounter=0;

//\*/ Global Variables/\*//

ISoundEngine\* se;

int newlevel=5;

bool hiscore=false;

bool chek\_fram=true;

bool chek=false;

bool check\_6=false;

bool level = false;

bool instruction = false;

bool return\_menu = false;

bool back1=false;

bool back2=false;

int score =0;

int w\_width = 500, h\_height = 250; //width and height of the Screen

int frameCounter = 0;

bool frameflag = false;

int charNumber = 0;

struct cordinate{//co-ordinates of images stored

float ax;

float ay;

float bx;

float by;

float cx;

float cy;

float dx;

float dy;

} character[9];

char\* adrsound[]={"khurram.wav","nothing.wav","nothing.wav","nothing.wav","noor.wav","nothing.wav","nothing.wav","kumail.wav","noor.wav"};//character sounds

char tempchar;

FILE\* ptr;

int max\_score;

//for rnd function

unsigned short lfsr = 0xACE1u;

unsigned bit;

/////////////////////

//\*/ Self Defined Functions/\*//

GLuint bckgrnd2;

GLuint bckgrndlast;

GLuint backgroundsecond;

GLuint higscore;

GLuint imgtex[9];

bool animat=false;

//texture mapping functions////

void CharacterGulzain\_1();

void background();

void gulzain\_4();

void khurram();

void kumail();

void khadija\_3();

void khadija\_7();

void sir();

void noori\_5();

void noori\_9();

void instructions();

void makingtexture();

GLuint maketextureobject(string filename);

void callingfunction();//main body of game

//\*/ CallBack Functions/\*//

void drawScene();//displays on screen

void update(int);

void initRendering();

void handleKeypress(unsigned char key, int x, int y);//controls keyboard input

void print(float x, float y, float spacechar, int score, char\* string);//to print text

int loop = 0;

float angle = 0.0;

/\*////////////////-------------ANIMATION-------------------\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*/

class Animation

{

int factor,count;

float frame;

float gifs;

GLuint tex;

bool animateflag;

public:

Animation(int x=10)

{

factor=2;count=0;frame=0;

animateflag=false;

gifs=x;

}

void animate()

{

animateflag=true;

}

void settex(unsigned tex)

{

this->tex=tex;

}

void draw(int x, int y, int width, int height)

{

count++;

if((count>=factor)&&(animateflag))

{

count=0;

frame++;

if(frame==10)

{

frame=0;

animateflag=false;

}

}

{

glColor3f(1,1,1);

glEnable(GL\_TEXTURE\_2D);

glBindTexture(GL\_TEXTURE\_2D,tex);

glTexParameterf(GL\_TEXTURE\_2D,GL\_TEXTURE\_MIN\_FILTER,GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D,GL\_TEXTURE\_MAG\_FILTER,GL\_LINEAR);

glBegin(GL\_POLYGON);

glTexCoord2f(0,frame/gifs+1/gifs);//glTexCoord2f(0,frame/gifs);

glVertex2f(x,height+y);

glTexCoord2f(1,frame/gifs+1/gifs);//glTexCoord2f(1,frame/gifs);

glVertex2f(x+width,height+y);

glTexCoord2f(1,frame/gifs);//glTexCoord2f(1,frame/gifs+1/gifs);

glVertex2f(x+width,y);

glTexCoord2f(0,frame/gifs);//glTexCoord2f(0,frame/gifs+1/gifs);

glVertex2f(x,y);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

}

};

Animation anime;

/\*////////////////////////////------------------Class Ended-----------------\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*/

void timer(int val)

{

angle += 2;

if (angle > 180)

angle = 180;

glutPostRedisplay();

glutTimerFunc(25, timer, 0);

}

//\*/ Main Code/\*// Credits: Opengl Rock Tutorials

unsigned rnd()

{

bit = ((lfsr >> 0) ^ (lfsr >> 2) ^ (lfsr >> 3) ^ (lfsr >> 5) ) & 1;

return lfsr = (lfsr >> 1) | (bit << 15);

}

void main(int argc, char \* argv)

{

se=createIrrKlangDevice();

if(!se){

cout<<"Engine Not Created"<<endl;

}

//Initialize GLUT

glutInit(&argc, &argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(w\_width \* 4, h\_height \* 2.7); //Set the window size

//now make textures

//Create the window

glutCreateWindow("whack-a-nustians");

initRendering(); //Initialize rendering

makingtexture();

//Set handler functions for drawing, keypresses, and window resizes

glutDisplayFunc(drawScene);

glutTimerFunc(25, timer, 0);

glutKeyboardFunc(handleKeypress);

update(0);

glutMainLoop(); //Start the main loop. glutMainLoop doesn't return.

return; //This line is never reached

}

void initRendering()

{

glClearColor(0.0, 0.0, 0.0, 1.0);//writing colour values to the buffer

glMatrixMode(GL\_PROJECTION);/\* i think glMatrixMode() tells openGl that how we are going to treat the image or object .

nOTE: Matrix in glMatrixMode means matrix(plural of matrices) . So the dimensions of an image is always stored in

form of matrix\*/

glEnable(GL\_DOUBLEBUFFER);

glLoadIdentity();

glOrtho(0.0, w\_width, 0.0, h\_height, -10.0, 10.0);/\*It tells openGL to how to look at the image .It is mostly used to create 2D

objects or images\*/

glEnable(GL\_BLEND);/\*glEnable(GL\_BLEND) is used to enable the blending of already present pixel data and incomming data\*/

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);/\*So what this function does to blend already present data in buffer and image data to create a

transparent background . For example try changing values in glClearColor and you will see the

background colour will change . Try changing glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)

to glBlendFunc(GL\_ONE,GL\_ONE) and you will see that image background will no more be transparent\*/

glClearColor(0.0, 0.0, 0.0, 1.0);//updating color data in buffer .

//Reading image file and making texture object

}

void update(int x)

{//repeats the display after every 33 miliseconds

glutPostRedisplay();

glutTimerFunc(33, update, 0);

}

void print(float x, float y, float spacechar, int score, char\* string)//prints text on the screen

{

float j = 0;

int p, k;

GLvoid \*font\_style1 = GLUT\_BITMAP\_TIMES\_ROMAN\_24;//define font style

p = score;//p set to score variable

j = 0;

k = 0;

float d = 0;

int len = (int)strlen(string);//length of string to be printed

for (int i = 0; i < len; i++)

{

glColor3f(5, 5, 5);//text color

glRasterPos2f(x + d, y);//position of text

glutBitmapCharacter(font\_style1, string[i]);//string and font style to be printed in

d = d + 3.6;// position

j = j - 1;

}

while (p > 9)//if score is in 2 digits eg 10

{

k = p % 10;

glRasterPos2f((x - (j\*spacechar)), y);//position of text

glutBitmapCharacter(font\_style1, 48 + k);

j++;

p /= 10;

}

//score in single digit

glRasterPos2f((x - (j\*spacechar)), y);

glutBitmapCharacter(font\_style1, 48 + p);

}

//it prints a 2d string on the opengl window at position x , y in pixels

GLuint maketextureobject(string filename)//resizing an image so that openGL may accept it.

{

vector<unsigned char> image;

unsigned char temp;

unsigned width, height;

GLuint texid;

//Loads the image data

if (0 != lodepng::decode(image, width, height, filename))

cout << "Error file not opened" << endl;

//Making power of two

////////////////////

// Texture size must be power of two for the primitive OpenGL version this is written for. Find next power of two.

size\_t u2 = 1; while (u2 < width) u2 \*= 2;

size\_t v2 = 1; while (v2 < height) v2 \*= 2;

// Make power of two version of the image.

std::vector<unsigned char> image2(u2 \* v2 \* 4);

for (size\_t y = 0; y < height; y++)

for (size\_t x = 0; x < width; x++)

for (size\_t c = 0; c < 4; c++)

{

image2[4 \* u2 \* y + 4 \* x + c] = image[4 \* width \* y + 4 \* x + c];

}

/////////////////

glGenTextures(1, &texid);

glBindTexture(GL\_TEXTURE\_2D, texid);//creates a new object if not created previously

glTexImage2D(GL\_TEXTURE\_2D, 0, 4, u2, v2, 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, &image2[0]);

return texid;

}

void makingtexture(){

///load all images

bckgrnd2=maketextureobject("level1.png");

backgroundsecond = maketextureobject("level-2.png");

bckgrndlast=maketextureobject("level-3.png");

higscore=maketextureobject("highscore.png");

imgtex[0]=maketextureobject("gulzain.png");

imgtex[1]=maketextureobject("khurram.png");

imgtex[2]=maketextureobject("kumail.png");

imgtex[3]=maketextureobject("khadija.png");

imgtex[4]=maketextureobject("sir.png");

imgtex[5]=maketextureobject("noori.png");

imgtex[6]=maketextureobject("instructions.png");

imgtex[7]=maketextureobject("MENU.png");

///////////////////////////////////////

////co-ordinates of images at all positions

character[0].ax= 130, character[0].ay = 190, character[0].bx=210, character[0].by=190, character[0].cx=210, character[0].cy=130, character[0].dx=130, character[0].dy=130;

character[1].ax= 245, character[1].ay=190, character[1].bx=325, character[1].by=190, character[1].cx=325, character[1].cy=130, character[1].dx=245, character[1].dy=130;

character[2].ax=340, character[2].ay=195, character[2].bx=400, character[2].by=195, character[2].cx=400, character[2].cy=145, character[2].dx=340, character[2].dy=145;

character[3].ax=130, character[3].ay=215, character[3].bx=210, character[3].by=215, character[3].cx=210, character[3].cy=155, character[3].dx=130, character[3].dy=155;

character[4].ax=235, character[4].ay=215, character[4].bx=345, character[4].by=215, character[4].cx=345, character[4].cy=155, character[4].dx=235, character[4].dy=155;

character[5].ax=345, character[5].ay=210, character[5].bx=415, character[5].by=210, character[5].cx=415, character[5].cy=170, character[5].dx=345, character[5].dy=170;

character[6].ax=135, character[6].ay=240, character[6].bx=195, character[6].by=240, character[6].cx=195, character[6].cy=190, character[6].dx=135, character[6].dy=190;

character[7].ax=235, character[7].ay=235, character[7].bx=305, character[7].by=235, character[7].cx=305, character[7].cy=185, character[7].dx=235, character[7].dy=185;

character[8].ax=340, character[8].ay=235, character[8].bx=450, character[8].by=235, character[8].cx=450, character[8].cy=175, character[8].dx=340, character[8].dy=175;

//////////////////////////////////////////////////

}

void background(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, bckgrnd2);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(0, 0);

glTexCoord2f(1, 0);

glVertex2f(0, 260);

glTexCoord2f(1, 1);

glVertex2f(800, 260);

glTexCoord2f(0, 1);

glVertex2f(800, 0);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

void background2(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, backgroundsecond);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(0, 0);

glTexCoord2f(1, 0);

glVertex2f(0, 260);

glTexCoord2f(1, 1);

glVertex2f(800, 260);

glTexCoord2f(0, 1);

glVertex2f(800, 0);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

void backgroundlast(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, bckgrndlast);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(0, 0);

glTexCoord2f(1, 0);

glVertex2f(0, 260);

glTexCoord2f(1, 1);

glVertex2f(800, 260);

glTexCoord2f(0, 1);

glVertex2f(800, 0);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

void CharacterGulzain\_1(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[0]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[1].ax, character[1].ay);

glTexCoord2f(1, 0);

glVertex2f(character[1].bx, character[1].by);

glTexCoord2f(1, 1);

glVertex2f(character[1].cx, character[1].cy);

glTexCoord2f(0, 1);

glVertex2f(character[1].dx, character[1].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

void gulzain\_4(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[0]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[3].ax, character[3].ay);

glTexCoord2f(1, 0);

glVertex2f(character[3].bx, character[3].by);

glTexCoord2f(1, 1);

glVertex2f(character[3].cx, character[3].cy);

glTexCoord2f(0, 1);

glVertex2f(character[3].dx, character[3].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

//////////////////////////////////////////////////////////

void khurram(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[1]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[0].ax, character[0].ay);

glTexCoord2f(1, 0);

glVertex2f(character[0].bx, character[0].by);

glTexCoord2f(1, 1);

glVertex2f(character[0].cx, character[0].cy);

glTexCoord2f(0, 1);

glVertex2f(character[0].dx, character[0].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

anime.draw(100,100,200,200);

}

//////////////////////////////////////////////////////

void kumail(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[2]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[7].ax, character[7].ay);

glTexCoord2f(1, 0);

glVertex2f(character[7].bx, character[7].by);

glTexCoord2f(1, 1);

glVertex2f(character[7].cx, character[7].cy);

glTexCoord2f(0, 1);

glVertex2f(character[7].dx, character[7].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

///////////////////////////////////////////////////////

void khadija\_3(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[3]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[2].ax, character[2].ay);

glTexCoord2f(1, 0);

glVertex2f(character[2].bx, character[2].by);

glTexCoord2f(1, 1);

glVertex2f(character[2].cx, character[2].cy);

glTexCoord2f(0, 1);

glVertex2f(character[2].dx, character[2].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

///////////////////////////////////////////////////

void khadija\_7(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[3]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[6].ax, character[6].ay);

glTexCoord2f(1, 0);

glVertex2f(character[6].bx, character[6].by);

glTexCoord2f(1, 1);

glVertex2f(character[6].cx, character[6].cy);

glTexCoord2f(0, 1);

glVertex2f(character[6].dx, character[6].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

///////////////////////////////////////

void sir(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[4]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[5].ax, character[5].ay);

glTexCoord2f(1, 0);

glVertex2f(character[5].bx, character[5].by);

glTexCoord2f(1, 1);

glVertex2f(character[5].cx, character[5].cy);

glTexCoord2f(0, 1);

glVertex2f(character[5].dx, character[5].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

///////////////////////////////////////////

void noori\_5(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[5]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[4].ax, character[4].ay);

glTexCoord2f(1, 0);

glVertex2f(character[4].bx, character[4].by);

glTexCoord2f(1, 1);

glVertex2f(character[4].cx, character[4].cy);

glTexCoord2f(0, 1);

glVertex2f(character[4].dx, character[4].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

///

void noori\_9(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[5]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(character[8].ax, character[8].ay);

glTexCoord2f(1, 0);

glVertex2f(character[8].bx, character[8].by);

glTexCoord2f(1, 1);

glVertex2f(character[8].cx, character[8].cy);

glTexCoord2f(0, 1);

glVertex2f(character[8].dx, character[8].dy);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

////////////////////////////////////////////////////

void menu() {

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[7]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(0, 0);

glTexCoord2f(1, 0);

glVertex2f(0, 260);

glTexCoord2f(1, 1);

glVertex2f(800, 260);

glTexCoord2f(0, 1);

glVertex2f(800, 0);

glEnd();

glPopMatrix();

glDisable(GL\_TEXTURE\_2D);

}

//////////////////////////////////////////////////////////////////////////////////////////////////////////

void instructions(){

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, imgtex[6]);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(0, 0);

glTexCoord2f(1, 0);

glVertex2f(0, 260);

glTexCoord2f(1, 1);

glVertex2f(800, 260);

glTexCoord2f(0, 1);

glVertex2f(800, 0);

glEnd();

glDisable(GL\_TEXTURE\_2D);

}

////////////////////////////////////////////////////////////////////////////

void handleKeypress(unsigned char key, int x, int y)

{

if(score<0)

score=0;//so score doesnt get less than 0

tempchar = key;

if (key == 'a'){ //to start game

level = true;

}

if (key == 'b'){ //to display instructions

instruction = true;

return\_menu=false;

hiscore=false;

}

if (key == 'e'){ //to return to menu

return\_menu = true;

hiscore=false;

instruction=false;

}

if(key=='c'){

hiscore=true;//check high score

return\_menu=false;

instruction=false;

}

if(chek){

if(level){

if (charNumber==(int)key-48 && check\_6) {//if key of danger character pressed

score=score-2;

chek\_fram=true;

check\_6=false;

}

if (charNumber == (int)key - 48){ //to kill a charachter

chek\_fram=true;

score++;

se->play2D(adrsound[7]);//play sound when correct key pressed

}

if(charNumber!=(int)key-48){//if wrong key pressed then deduct score

chek\_fram=true;

score--;

}

//killing normal charachter will increase scores

} printf("\n score %d", score);

}

chek=false;

}

void highscore(){//to read highscore from file

//mappingimage

glColor3f(1, 1, 1);

glEnable(GL\_TEXTURE\_2D);//enables openGL capability of texturing an object

glBindTexture(GL\_TEXTURE\_2D, higscore);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

glColor3f(1, 1, 1);

//setting vertex co-ordinates

glBegin(GL\_POLYGON);

glTexCoord2f(0, 0);

glVertex2f(0, 0);

glTexCoord2f(1, 0);

glVertex2f(0, 260);

glTexCoord2f(1, 1);

glVertex2f(800, 260);

glTexCoord2f(0, 1);

glVertex2f(800, 0);

glEnd();

glDisable(GL\_TEXTURE\_2D);

int highscore=0;

ptr=fopen("maxscore.txt","r");

if(ptr==NULL){

printf("Error");

}

fscanf(ptr,"%d",&highscore);

printf("%d",highscore);

print(10, 150, 4.8, highscore, "Your H igh Score is:");//show highscore

fclose(ptr);

}

void callingfunction(){

if(score<0)

score=0;

ptr=fopen("maxscore.txt","w");

max\_score=0;

if(ptr==NULL)

printf("Error");

fscanf(ptr,"%d",&max\_score);

if(score>max\_score){//write/update highscore

max\_score=score;

}

fprintf(ptr,"%d",max\_score);

fclose(ptr);

//change levels of game

if(score<=10){

newlevel=250;

back1=false;

back2=false;

}

if(score<=20 && score>10){

newlevel=250;

back2=true;

back1=false;

}

if(score>50 ){

newlevel=50;

back1=true;

back2=false;

}

if (frameCounter >=newlevel || chek\_fram) { //frame flag becomes true when frame counter becomes 5 and a new random no. is generated

frameflag = true;

frameCounter = 0;

chek=true;

chek\_fram=false;

}

srand(time(NULL)); //seed

if (frameflag){ //new random no. is generated and next charachter appears

charNumber = 1;//+ (rnd() % 9);//not rand()

//frameflag = false;

}

if(frameflag){

if(!chek\_fram){

animat=true;

}

if(chek\_fram){

animat=false;

}

switch (charNumber){

case 1:

khurram();

check\_6=false;

break;

case 2:

CharacterGulzain\_1();

check\_6=false;

break;

case 3:

khadija\_3();

check\_6=false;

break;

case 4:

gulzain\_4();

check\_6=false;

break;

case 5:

noori\_5();

check\_6=false;

break;

case 6:

sir();

check\_6=true;

break;

case 7:

khadija\_7();

check\_6=false;

break;

case 8:

kumail();

check\_6=false;

break;

case 9:

noori\_9();

check\_6=false;

break;

}

frameflag=false;

print(10, 230, 5.2, score, "Score:");

glutSwapBuffers();

}

frameCounter++;

}

//Draws the 2D scene

void drawScene() //actual screen display function

{

if(!level && !hiscore && !instruction){

menu();//display menu

if(menucounter==0)

se->play2D("menu.wav");

if(menucounter==1000){

menucounter=0;

se->play2D("menu.wav");

}

glutSwapBuffers();

}

if(hiscore){

/\*if(menucounter==0)

se->play2D("menu.wav");

if(menucounter==1000){

menucounter=0;

se->play2D("menu.wav");

}\*/

highscore();//display highscore

glutSwapBuffers();

}

if(instruction){

//if(menucounter==0)

/\*se->play2D("menu.wav");

if(menucounter==10000){

menucounter=0;

se->play2D("menu.wav");

}\*/

instructions();

glutSwapBuffers();

} //display instructions

if(return\_menu){ //return to menu from instructions

menu();

/\*if(menucounter==0)

se->play2D("menu.wav");

if(menucounter==1000){

menucounter=0;

se->play2D("menu.wav");

}\*/

glutSwapBuffers();

}

if(level){

if(!back1 && !back2)

background();

if(back1)

background2();

if(back2)

backgroundlast();

return\_menu=false;

instruction=false;//calling background function

hiscore=false;

callingfunction(); //starting game

}

menucounter++;

if(animat){

anime.settex(imgtex[1]);

anime.animate();

glutSwapBuffers();

}

}