NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

MINI PROJECT REPORT

On

BALL & ARROW USING OPENGL

Submitted in partial fulfilment of the requirements
for the completion of 4th SEM.

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

By

Apoorv Singh (16-1-5-057)
Harsh Ganda (16-1-5-067)
Surendra Singh Gurjar (16-1-5-087)
Harshit Agarwala (16-1-5-092)

Under the guidance of

Ms. Upasana Talukdar

Assistant Professor, CSE Department

DEPARTMENT OF COMPUTER SCEINCE & ENGINEERING, NATIONAL INSTITUTE OF TECHNOLOGY, SILCHAR

Acknowledgements

We are highly indebted to Asst. Prof. Ms. Upasana Talukdar, Department of Computer Science & Engineering, NIT Silchar for her guidance and constant supervision as well as for providing necessary information regarding the project and also for her support in completion of the project.

We also extend our sincere thanks and appreciation to all who directly or indirectly co-operated us in completing the project within the time frame.

26th April'18.

INDEX

INTRODUCTION

Overview

This report discusses the result of the work done in designing of "BALL & ARROW GAME using OPENGL".

It is a simple game using OpenGL. The game involves shooting of arrows one after the other in order to hit the balls which are moving in the right end. User has to shoot the balls by pressing the key 'spacebar' on the keyboard.

OpenGL is a library useful for designing and animation purposes. It can be used in C or C++ programming languages and is very user friendly.

Background and Motivation

OpenGL can be found in a large variety of applications today, like to design or develop animation films, cartoons and mostly games. Today's era is an era wherein children and even the adults like to play games. Moreover cartoons are also widely appreciated. Animations are used in daily newspaper, news channel, advertisements and where not to describe products, social issues, political clashes etc.

Objective

Our objective is to develop this project to design a game which can be thoroughly enjoyed and played by all the people who wish.

The main aim is to strike the balls thrice within the given number of arrows which is 10. If the user is unable to accomplish the same, he loses the game.

Methodology

To implement the above goal, the methodology we followed are:–

- At the very beginning we have designed an abstract of the project.
- Various inbuild glut functions like translatef, scalef, solidsphere, strokecharacter etc are been used.
- Two windows are available for both the game and instructions.
- Different functions are created for different purposes like Display(), myHit(), keyboard(), Write(), Draw_Instruct(), etc.
- Result is displayed in the text format at the end of the game.

TOOLS DESCRIPTION

The tool used for this project is OpenGL in C programming language.

Platform

The model can be run in any platform (OS) Linux, Windows provided that the glut should be installed and properly working.

Future Extension:

The game is open to future modifications. Some ideas are:-

1. Adding multiplayer features.
2. Multiple targets.
3. Obstruction to arrow by varying wind speeds.
4. Different types of arrows of varying powers.
5. Variable locations.
6. Making it available on different platforms.

Size of source code:

4.5 kB (4,545 bytes)

DOCUMENTATION

FUNCTIONS USED

Display():

- It plays the game. It initially sets the the red ball to a position given by 'up' variable, and using the ***Translatef()*** function it displaces only the y-coordinate of the ball upwards.

- ***glutSolidSphere()*** renders a 3D sphere with radius, slices and stack as parameters. While the 'spacebar' key is pressed, the function creates an arrow head, and associates a variable 'pos' with it, to translate the arrow towards the right in a single direction. This variable is incremented continually everytime and called with the ***Translatef()*** to redraw the arrow at new positions.

- If the condition for bounds satisfy, that means collision has occurred, and counter1 is incremented to register a hit. The flag 'bang' is set to 1, so that when encountered during the next iteration the following changes can take place: position of sphere is reset to 0, bang is reset to 0 (to prepare for next hit), position of arrow is reset.

- 'up' variable is continually incremented to keep the ball moving upwards for a large number of iterations by calling ***glutPostRedisplay()*** everytime. It marks the normal plane of current window as needing to be redisplayed with the same specifications. The counter1 value is checked for every iteration. Once it has reached a value of 3, ***myHit()*** is called to display that player is a winner.

MyHit():

This function is primarily to display text on screen. We also set the antialiasing for lines and set the line width to prepare to draw our text as Stroke Characters.

Drawhit():

It draws text in Stroke Character font. A stroke font is a 3D font. As opposed to bitmap fonts these can be rotated, scaled, and translated. The GLUT_STROKE_ROMAN font is used here. The text can be placed anywhere on the screen using ***Translatef()*** and scaled to any size as needed using ***Scalef()***.

Instructions():

It creates a separate instruction page where the instructions are displayed on another window using StrokeCharacter font, and translated to appropriate postions on the screen.

Draw_instruct():
It draws text in Stroke Character font. A stroke font is a 3D font. As opposed to bitmap fonts these can be rotated, scaled, and translated. The GLUT_STROKE_ROMAN font is used here. The text can be placed anywhere on the screen using ***Translatef()*** and scaled to any size as needed using ***Scalef().***

Keyboard():
Keyboard is a keyboard callback function which is used to make our program interactive. It makes the shoot flag = 1 in our program everytime key 'spacebar' is pressed, by recognizing it as an ASCII character.

Main():
The main function performs the required initializations and starts the event processing loop.
All the functions in GLUT have the prefix glut, and those which perform some kind of initialization have the prefix glutInit.

- ***glutInit(int *argc, char **argv)*** - parameters are pointers to the unmodified argc and argv variables from the main function.

- We establish the window's position by using the function ***glutInitWindowPosition*.**

- We choose the window size using the function ***glutInitWindowSize***.

- We define the display mode using the function ***glutInitDisplayMode***. GLUT_RGB - selects a RGBA window. This is the default color mode. GLUT_SINGLE – selects a single buffer window.

- Each window can be created with ***glutCreateWindow***. The return value of ***glutCreateWindow*** is the window identifier.

- ***glutDisplayFunc()*** passes the name of the function to be called when the window needs to be redrawn.

- ***glutKeyboardFunc***- is notify the windows system which function(s) will perform the required processing when a key is pressed. This function is to register a callback for keyboard events that occur when you press a key.

- Creating a menu: ***glutCreateMenu*** creates a menu table on a default right click of mouse. ***glutAddMenuEntry*** adds a menu entry to this menu created.

- When we are ready to get in the application event processing loop we enter ***glutMainLoop***. It gets the application in a never ending loop, always waiting for the next event to process

Init():
Sets the background color for the game.

Source code:

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<GL/glut.h>
#include<time.h>
#include<math.h>

static GLfloat up=-0.2;
static GLfloat pos=-0.2;
int shoot=0,bang=0;
int counter1=0,counter2=0,count=0;
int game,instruct;
char tmp_str[40];
void displost();

void init(void)
{
	glClearColor (1.0,1.0,1.0,0.0);
	glMatrixMode (GL_PROJECTION);
}

void drawhit(const char * message, int x, int y)
{
	glPushMatrix();
	glScalef(0.3,0.2,0.15);
	glTranslatef(x,y,0);
	while (*message)
	{
		glutStrokeCharacter(GLUT_STROKE_ROMAN,*message++);
	}
	glPopMatrix();
}

void myHit()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0,300,0,300);
	glClearColor(1.0,0.0,0.0,1.0);
	glColor3f(0.0,0.0,0.0);
	glLineWidth(8.0);
	drawhit("You Won!!",120,800);
}
```

```
void draw_instruct(const char *message, int x, int y)
{
	int j;

	glPushMatrix();
	glScalef(0.1,0.1,0.0);
	glTranslatef(x,y,0);
	while (*message)
	{
		glutStrokeCharacter(GLUT_STROKE_ROMAN,*message++);
	}
	glPopMatrix();
}

void instructions()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0,200,0,200);
	glMatrixMode(GL_MODELVIEW);
	glClearColor(1.0,1.0,1.0,1.0);
	glColor3f(0.2,0.2,0.2);
	glLineWidth(8.0);
	draw_instruct("How To Play",500,1800);
	glLineWidth(4.0);
	draw_instruct("_________",450,1780);
	draw_instruct("Hit 3 Balls in 10 Shoots",200,1500);
	draw_instruct("Press Spacebar to shoot",200,1200);
	draw_instruct("Press Q to quit",750,100);
	glFlush();
}

void Write(char *string)
{
	glScalef(0.02,0.02,0.0);
	while(*string)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *string++);
}

void display()
{
	int i;
	if(counter1==3)
	{
		myHit();
		glFlush();
```

```
}
else
{
	int j;
	for(j=0;j<100000000;j++);

	glClearColor(0.2,1.0,0.4,1.0);
	glClear(GL_COLOR_BUFFER_BIT);
	glPushMatrix();
	glColor3f(0, 0, 1);
	glRasterPos2f(-0.9, 0.9);
	sprintf(tmp_str, "Arrow count: %d", count);
	Write(tmp_str);
	glPopMatrix();

	if(count>10)
		glutDisplayFunc(displost);
	glPushMatrix();
	glColor3f(0, 0, 1);
	glRasterPos2f(-0.2, 0.9);
	sprintf(tmp_str, "Score: %d", counter1);

	Write(tmp_str);

	glPopMatrix();
	glPushMatrix();
	glColor3f(1.0,0.2,0.0);
	glLoadIdentity();
	glTranslatef(0.8,-0.869+up,0.0);
	glutSolidSphere(0.10,20,16);

	if(shoot==1)
	{
		glPushMatrix();
		glLoadIdentity();
		glTranslatef(-0.8+pos,0.0,0.0);
		glColor3f(0.0,0.0,0.0);
		glLineWidth(5.0);
		glBegin(GL_LINES);
		glVertex3f(-0.2,0.0,0.0);
		glVertex3f(0.1,0.0,0.0);
		glVertex3f(0.1,0.0,0.0);
		glVertex3f(0.03,0.05,0.0);
		glVertex3f(0.1,0.0,0.0);
		glVertex3f(0.03,-0.05,0.0);
		glVertex3f(-0.2,0.0,0.0);
		glVertex3f(0.03-0.3,0.05,0.0);
```

```
        glVertex3f(-0.2,0.0,0.0);
        glVertex3f(0.03-0.3,-0.05,0.0);
        glVertex3f(-0.16,0.0,0.0);
        glVertex3f(0.03-0.26,0.05,0.0);
        glVertex3f(-0.16,0.0,0.0);
        glVertex3f(0.03-0.26,-0.05,0.0);
        glEnd();
        glPopMatrix();
}
if(bang==1)
{
        bang=0;pos=-0.2;
        glPushMatrix();
        glLoadIdentity();
        up=0;
        glColor3f(1.0,0.2,0.0);
        glutSolidSphere(1,20,16);
        glPopMatrix();
}
glPopMatrix();
for( i=0;i<200;i+=20)
{
        if(pos>=1.90 && up>0.920 && up<1.080)
        {
                counter1++;
                shoot=0;
                pos=-0.2;
                bang=1;
        }
        if(counter1==3)
                count=0;
        int value=rand()*rand()%20;
        up=(up+value/1500.0+0.005);

        if(up>2)
                up=0;

        if(shoot==1)
        {
                pos=pos+0.015;
                if(pos>2)
                {
                        pos=-0.2;
                        shoot=0;
                }
        }
        glutPostRedisplay();
}
```

```
        glFlush();
        }
}

void displost()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0,300,0,300);
        glClearColor(1.0,1.0,0.0,1.0);
        glColor3f(1.0,0.0,0.80);
        glLineWidth (8.0);
        drawhit("Sorry,",90,850);
        drawhit("Try Again!!",90,650);
        glFlush();
}

void indisplay()
{
        glClearColor(0.8,0.8,1.0,1.0);
        glClear(GL_COLOR_BUFFER_BIT);
        instructions();
        glFlush();
}

void keyboard(unsigned char key,int x,int y)
{
        if (key==' ')
        {
                shoot=1;
                count++;
        }
        if (key=='q' || key=='Q')
        {
                exit(0);
        }
}


int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
        glutInitWindowSize(1500,1500);
        glutInitWindowPosition(0,0);
        glutCreateWindow("Instructions");
        init();
```

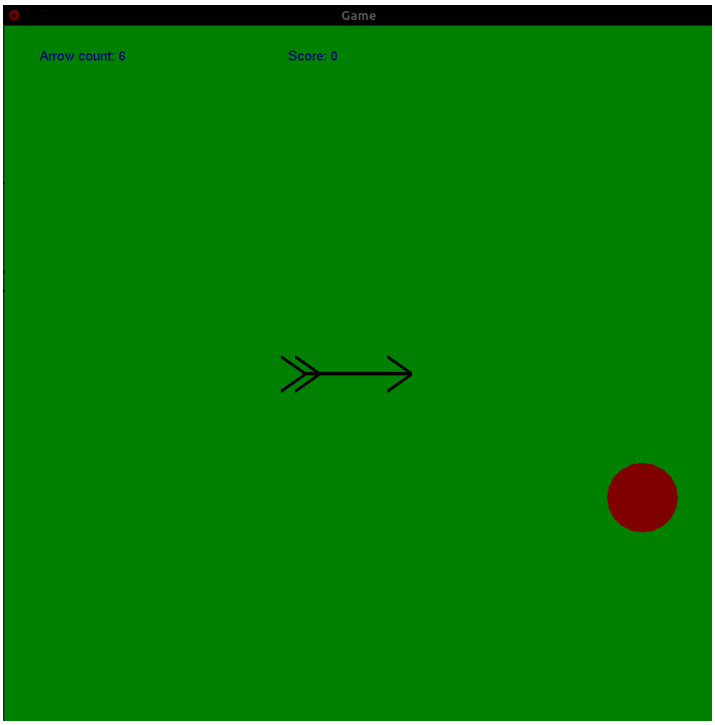
```
	glutDisplayFunc(indisplay);
	glutInitDisplayMode(GLUT_SINGLEGLUT_RGB);
	glutInitWindowSize(1000,1000);
	glutInitWindowPosition(0,0);
	glutCreateWindow("Game");
	init();
	glutKeyboardFunc(keyboard);
	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```

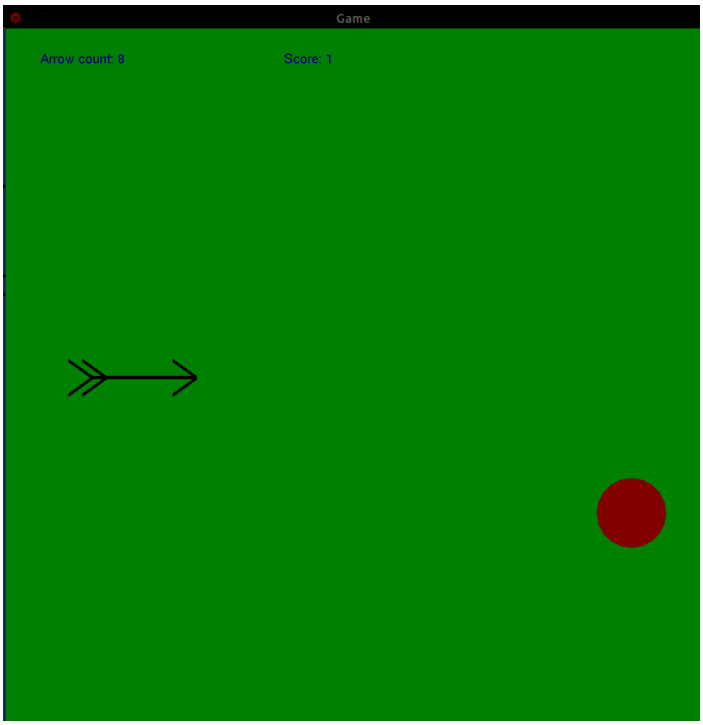
Limitations:-

1. Single Player game.
2. No obstruction to arrows.
3. Single target.
4. Resolution = 1000 X 1000

OUTPUT

SNAPSHOTS


Game
Arrow count: 6
Score: 0


Game
Arrow count: 8
Score: 1


Game
You Won!!


Game
Sorry,
Try Again!!

CONCLUSION

The development of Computer Graphics has made computers easier to interact with and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry.
We started with modest aim with no prior experience in any programming projects as this, but ended up in learning many things, fine tuning the programming skills and getting into the real world of software development with an exposure to corporate environment.
During the development of any software of significant utility, we are faced with the trade-off between speed of execution and amount of memory consumed. This is simple interactive application. It is extremely user friendly and has the features, which makes simple graphics project. Checking and verification of all possible types of the functions are taken care. Care was taken to avoid bugs.
So, we conclude on note that we are looking forward to develop more such projects with an appetite to learn more in Computer Graphics.

BIBLIOGRAPHY:

1. Interactive Computer Graphics: A Top Down Approach with OpenGL- Edward Angel, 5th Edition, Addison-Wellesley, 2008.
2. Online tutorials for game development at NeHe productions.
3. OpenGL Red Book and Blue Book for reference
4. www.opengl.org for OpenGL tutorials.
5. Computer Graphics with OpenGL by Donald D. Hearn, M. Pauline Baker, Warren Carithers.
6. www.stackoverflow.com
7. www.google.com