VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM-590014

A Computer Graphics and Visualization

Mini-Project Report

On

"STORM"

A Mini-project report submitted in partial fulfilment of the requirements for the award of the degree of ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belgaum.*

Submitted by:
SURAJ MANDAL (1DT14CS100)
AND
TWINKLE AGARWAL (1DT14CS106)
Under the Guidance of:
Mr. Manjunath D. R.

(Asst. Prof. Dept of CSE)

Department of Computer Science and Engineering
DAYANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT
Kanakpura Road,Udayapura, Bangalore

2016-2017

DAYANADA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT,

Kanakpura Road,Udayapura, Bangalore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Mini-Project on Computer Graphics and Visualization work entitled **"IMPLEMENTATION OF STORM GAME USING OPENGL"** has been successfully carried out by **SURAJ MANDAL(1DT14CS100) and TWINKLE AGARWAL(1DT14CS106)** a bonafide students of **Dayananda sagar academy of technology and management** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during academic year 2016-2017. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

GUIDES:

Mr. MANJUNATH D.R
Asst. Prof. Dept of CSE

Dr. C. NANDINI
Vice Principal & HOD, Dept. of CSE,DSATM

Examiners:

1:

2: **Signature with Date**

ABSTRACT

We have proposed a Bird firing which aims at the target and shoots, the targets are moving blocks of objects. The player can switch sides of the shooter by pressing the UP and the DOWN key. The UP key is pressed to aim upwards and the DOWN key is pressed to aim downwards. When a proper orientation is achieved, the player can press ‘S’ to shoot. On shooting the targets points are increased, the player also gets to know the numbers of targets missed at a particular instance.A maximum of 10 chances are given to the player to shoot and score maximum points. The player can anytime refer the instruction by pressing ‘H’ and can quit the game by a right click on the mouse.

ACKNOWLEDGEMENT

It gives us immense pleasure to present before you our project titled **'IMPLEMENTATION OF STORM GAME USING OPENGL'.** The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of those who made it possible. We are glad to express our gratitude towards our prestigious institution **DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT** for providing us with utmost knowledge, encouragement and the maximum facilities in undertaking this project.

We wish to express a sincere thanks to our respected principal **Dr. B. R. Lakshmikantha** for all their support.

We express our deepest gratitude and special thanks to **Dr.C.Nandini** , **Vice Principal & H.O.D, Dept. Of Computer Science Engineering**, for all her guidance and encouragement.

We sincerely acknowledge the guidance and constant encouragement of our mini- project guides, **Assistant Prof. Mr. Manjunath D. R.**

SURAJ MANDAL (1DT14CS100)
AND
TWINKLE AGARWAL (1DT14CS106)

TABLE OF CONTENT

Chapter-1

INTRODUCTION

1.1 COMPUTER GRAPHICS

Computer graphics are graphics created using computers and more generally, the representation and manipulation of image data by a computer. "almost everything on computers that is not text or sound".

Now, we can create image using computer that are indistinguishable from photographs from the real objects

The various applications of computer graphics are:

- Display of information
- Design
- Simulation and animation
- User interfaces

The phrase “Computer Graphics” was coined in 1960 by William Fetter, a graphic designer for Boeing

Today, we find computer graphics used in various areas that include science, medicine, business, industry, art, entertainment etc

The main reason for effectiveness of the interactive computer graphics is the speed with which the user can understand the displayed information

The current trend of computer graphics is to incorporate more physics principles into 3D graphics algorithm to better simulate the complex interactions between objects and lighting environment.

1.2 OpenGL TECHNOLOGY

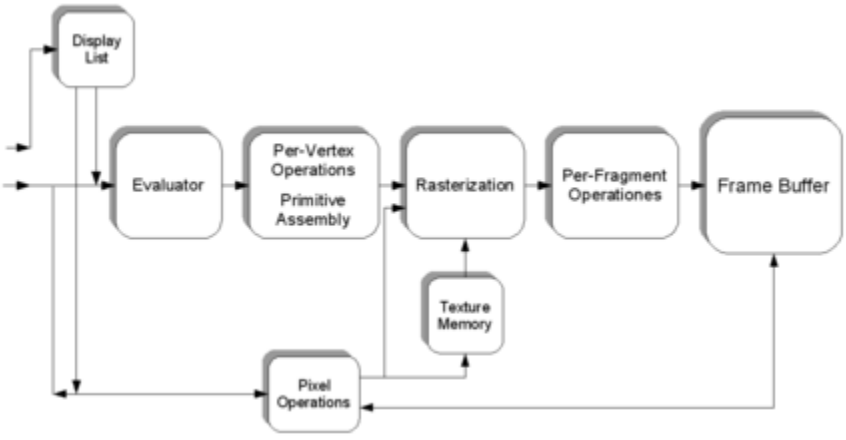
OpenGL (**Open G**raphics **L**ibrary) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, flight simulation, and video games.

OpenGL is a software API to graphics hardware, designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms, Intuitive, procedural interface with c binding, No windowing commands! And No high-level commands for describing models of three-dimensional objects.

OpenGL serves two main purposes:

1. To hide the complexities of interfacing with different 3D accelerators, by presenting the programmer with a single, uniform API.
2. To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary).

OpenGL has historically been influential on the development of 3D accelerators, promoting a base level of functionality that is now common in consumer-level hardware:
Rasterised points, lines and polygons as basic primitives



Simplified version of the Graphics Pipeline Process; excludes a number of features like blending, VBOs and logic ops

- A transform and lighting pipeline
- Z-buffering
- Texture mapping
- Alpha blending

A brief description of the process in the graphics pipeline could be:

- Evaluation, if necessary, of the polynomial functions which define certain inputs, like NURBS surfaces, approximating curves and the surface geometry.

1. Vertex operations, transforming and lighting them depending on their material. Also clipping non visible parts of the scene in order to produce the viewing volume.
2. Rasterisation or conversion of the previous information into pixels. The polygons are represented by the appropriate colour by means of interpolation algorithms.
3. Per-fragment operations, like updating values depending on incoming an game d previously stored depth values, or colour combinations, among others.
4. Lastly, fragments are inserted into the Frame buffer.

Many modern 3D accelerators provide functionality far above this baseline, but these new features are generally enhancements of this basic pipeline rather than radical revisions of it. As OpenGL is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

1.3 PROJECT DESCRIPTION

We have proposed a 2D shooting game wherein the shooter is moved up and down to aim at the targets by pressing the UP and the Down key respectively, the target is shot when the user presses the ‘S’ key. A maximum of 10 chances are allowed, post which the game ends.

1.4 FUNCTIONS USED

This project is developed using CodeBlocks and this project is implemented by making extensive use of library functions offered by graphics package of OpenGl, a summary of those functions follows:

void glVertex[234] [sfid](TYPE xcoordinate, TYPE ycoordinate, ...):
Specifies the position of a vertex in 2, 3, or 4 dimensions. The coordinates can be specified as shorts s, ints i, floats f or doubles d.

void glBegin() :
Specifies the primitives that will be created from vertices presented between glBegin and subsequent glEnd.. GL_POLYGON, GL_LINE_LOOP etc.

void glEnd(void) :
It ends the list of vertices.

void glColor3[b i f d ub us ui](TYPE r, TYPE g, TYPE b):
sets the present RGB colors.

void glClear(GLbitfield mask):
glClear takes a single argument that is the bitwise *or* of several values indicating which buffer is to be cleared.
GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT, GL_ACCUM_ BUFFER_BIT, and GL_STENCIL_BUFFER_BIT. Clears the specified buffers to their current clearing values.

void glClearColor(*GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha*):
Sets the current clearing color for use in clearing color buffers in RGBA mode. The red, green, blue, and alpha values are clamped if necessary to the range [0,1]. The default clearing color is (0, 0, 0, 0), which is black.

void glPointSize(GLfloat size):
Sets the point size attribute in pixels.

void glFlush():
The glFlush function forces execution of OpenGL functions in finite time.

void glutInit(int *argc, char **argv):
initializes GLUT. The arguments from main are passed in and can be used by the application**.**

void glutInitDisplayMode(unsigned int mode):
Specifies the display mode, normally the bitwise OR-ing of GLUT display mode bit *masks.* This api specifies a display mode (such as RGBA or color-index, or single or double-buffered) for windows.

int glutCreateWindow(char *title):
The parameter *title* specifies any name for window and is enclosed in double quotes. This opens a window with the set characteristics like display mode, width, height, and so on. The string name will appear in the title bar of the window system . The value returned is a unique integer identifier for the window. This identifier can be used for controlling and rendering to multiple windows from the same application.

void glutInitWindowSize(int width, int height):
Specifies the initial width and height of the window in pixels.

void glutInitWindowPosition(int x,int y):
specifies the initial position of the top-left corner of the window in pixels.

void glViewport(int x, int y, GLsizei width, GLsizei height):
specifies a width X height viewport in pixels whose lower-left corner is at (x, y) measured from the origin of the window.

void glutMainLoop(void):
cause the program to enter an event processing loop. It should be the last statement in main.

void glutDisplayFunc(void (* func) (void)):
registers the display function func that is executed when the window needs to be redrawn.

void glutPostRedisplay():
requests that the display callback be executed after the current callback returns.

void glutSwapBuffers():
swaps the front and back buffers.

void glutReshapeFunc(void *f(int width, int height)):
Registers the reshape callback function f. The callback function returns the height and width of the new window. The reshape callback invokes a display callback.

void glutkeyboardFunc (void *f(int width, int height)):

registers the keyboard callback function f. The callback function returns the ASCII code of the key pressed and the position of the mouse.

void glutIdleFunc(void (*f) (void)):
registers the display callback function f that is executed whenwver there are no other events to be handed.

void glutCreateMenu(void (*f) (int value)):
returns an identifier for a top-level menu and registers the callback function f that eturns an integer value corresponding to the menu entry selected.

void glutAddMenuEntry(char *name, int value):
adds an entery with the string name displayed to the current menu. Value is returned to the menu callback when the entry is selected.

void glutAttachMenu(int button):
Attches the current menu to the specified mouse button.

void glMatrixMode():
It accepts three values GL_MODELVIEW, GL_PROJECTION and GL_TEXTURE. It specifies which matrix is the current matrix. Subsequent transformation commands affect the specified matrix.

void glLoadIdentity() :
It replaces the current matrix with the identity matrix.

void glPushMatrix() :
glPushMatrix pushes the current matrix stack down by one level, duplicating the current matrix.

void glPopMatrix() :
glPopMatrix pops the top matrix off the stack, destroying the contents of the popped matrix. Initially, each of the stacks contains one matrix, an identity matrix.

void glTranslate[fd](TYPE X, TYPE Y, TYPE Z):
alters the current matrix by a displacement of (X, Y, Z). TYPE is either GLfloat or GLdouble.

void glRotate[fd](TYPE angle, TYPE dx, TYPE dy, TYPE dz):
Alters the current matrix by a rotation of angle degrees about the axis (dx, dy, dz). TYPE is either GLfloat or GLdouble.

void glScale[fd](TYPE sx, TYPE sy, TYPE sz):
alters the current matrix by a scaling of (sx, sy, sz). TYPE is either Glfloat or GLdouble.

void gluOrtho2D(GLdouble left, Gldouble right, GLdouble bottom, GLdouble top):
defines a two-dimensional viewing rectangle in the plane z = 0.

void glutBitMapCharacter(*void *font, int character*):
Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.

void glRasterPos2f(GLfloat x, GLfloat y):

x Specifies the x-coordinate for the current raster position.

y Specifies the y-coordinate for the current raster position.

OpenGL maintains a 3-D position in window coordinates. This position, called the raster position, is maintained with sub pixel accuracy. It is used to position pixel and bitmap write operations. The current raster position consists of three window coordinates (x, y, z), a clip coordinate w value, an eye coordinate distance, a valid bit, and associated color data and texture coordinates. The w coordinate is a clip coordinate, because w is not projected to window coordinates

Chapter-2

REQUIREMENTS SPECIFICATION

2.1 Hardware requirements:

- Pentium 4 or Equivalent Processor.
- 256 MB RAM
- 5 MB Disk Space
- Intel GMA 850 or better display adaptor
- Mouse and Keyboard input devices

2.2 Software requirements:

- Microsoft Windows XP / Linux 2.6.27 or later
- OpenGL and GLUT libraries
- CodeBlocks IDE

Platform and Tools:

- The project has been carried out in Windows platform.
- Built using Code Blocks IDE with glut libraries and OpenGL.
- Keyboard interface is required for this project.

Chapter-3

INTERFACE

KEYBOARD FUNCTIONS:

KEY(‘UP’):used to make the shooter up.

KEY(‘DOWN’):used to make the shooter down.

KEY(‘S’):used to shoot the bird.

KEY(‘R’):used to resume the game.

KEY(‘H’):used to know the rules to play the game.

User can always press RIGHT MOUSE BUTTON to play or quit the game.

```
void key(unsigned char k, int x, int y)
{

        if(k==' ')
             glutDisplayFunc(help);
        if(k=='r'||k=='R')
             glutDisplayFunc(background);
       if(k=='h'||k=='H')
             glutDisplayFunc(help);
       if(k=='s')
            {
              shotBullet();
              c+=1;
            }
      if(c>10)
             glutDisplayFunc(gameover);
}
```

Chapter-4

IMPLEMENTATION

Implementation: implementation of the project is the most important part of the project. The implement phase deals with the quality, performance, accuracy and debugging. The end derivable is the final game..

This game is implemented using keyboard interfaces which are as follows:

Keyboard Function :

void Key(unsigned char, int x, int y);

This function is used to make responds to the keys that are pressed from the keyboard. In this project, these keyboard keys are used to aim at the targets and shoot .

```
void key(unsigned char k, int x,int y)
{

if(k==' ')
    glutDisplayFunc(help);
if(k=='r'||k=='R')
    glutDisplayFunc(background);
if(k=='h'||k=='H')
    glutDisplayFunc(help);
if(k=='s')
{
   shotBullet();
   c+=1;
}
if(c>10)
   glutDisplayFunc(gameover);
}
```

Display Function :

void display();

In this function, rotation,translation and scaling is done by pushing and popping the matrix .

```
void background(void)
{
   glClearColor(0.0,0.749,1.0,1.0);
   glClear(GL_COLOR_BUFFER_BIT);
   glPointSize(5.0);
   glPushMatrix();
   glTranslatef(i,0.0,0.0);
   clouds();
   glPopMatrix();
   mountains();
   triangle(theta);
   sprintf(str,"POINTS:%d",point/20);
   text1(470,780,str);
   sprintf(str,"MISSED:%d",miss);
   text1(470,730,str);
   if(shot1==0)
   {
   glPushMatrix();
   glTranslatef(t1,150.0,0.0);
   target();
   glPopMatrix();
   }
   if(shot2==0)
   {
   glPushMatrix();
   glTranslatef(t1,0.0,0.0);
```

```
    target();
    glPopMatrix();
    }
    if(shot3==0)
    {
    glPushMatrix();
    glTranslatef(t1,-150.0,0.0);
    target();
    glPopMatrix();
    }
    drawBullets();
    drawtarget();
    glFlush();
    glutSwapBuffers();
}
```

Main Function :

int main(int argc, char *argv[]);

In this function, we are creating the window, enabling the mouse and keyboard functions, initializing the display mode and also enables the view.

```
int main(int argc,char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
    glutCreateWindow("STORM");
    glutInitWindowSize(1000,1000);
    glutInitWindowPosition(0,0);
    glutDisplayFunc(starter);
    glutKeyboardFunc(key);
    glutReshapeFunc(reshape);
    glutSpecialFunc(special);
```

```
    glutSpecialUpFunc(specialup);
    glutCreateMenu(menu);
    glutAddMenuEntry("PLAY",1);
    glutAddMenuEntry("QUIT",2);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    glutIdleFunc(idle);
    init();
    glutMainLoop();
}
```

Chapter-5

SNAPSHOTS

SAMPLE OUTPUT

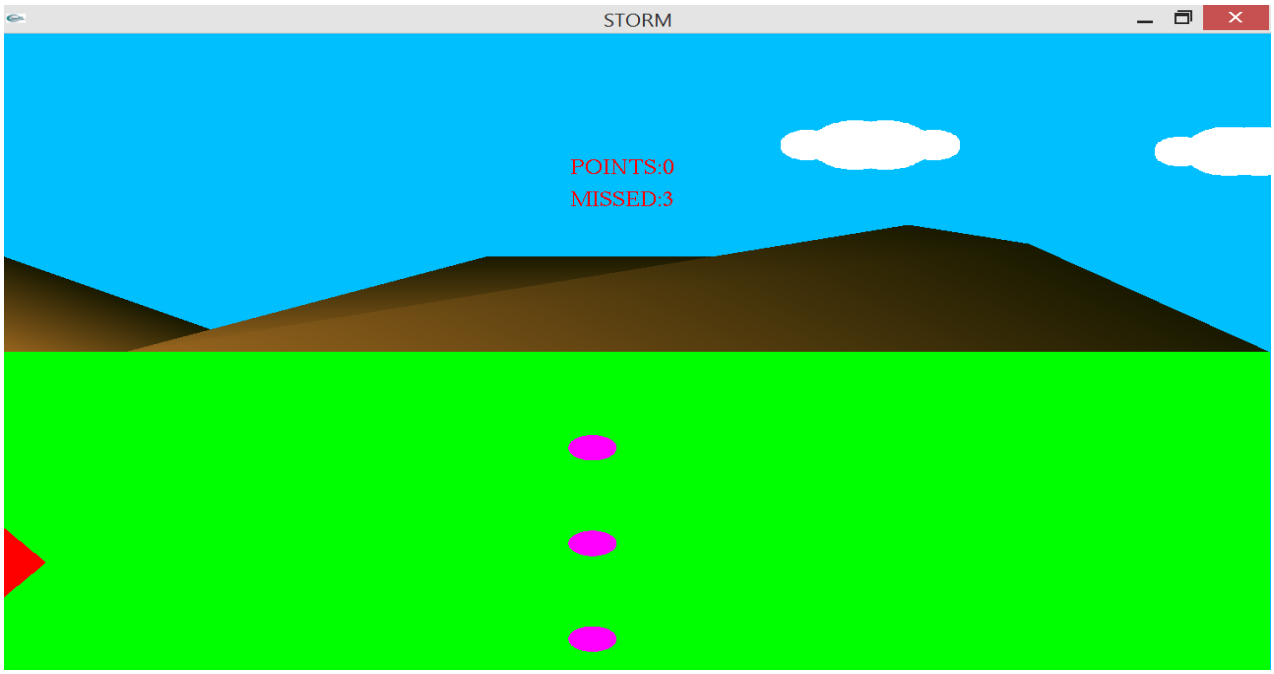


Fig5.1 View of Objects

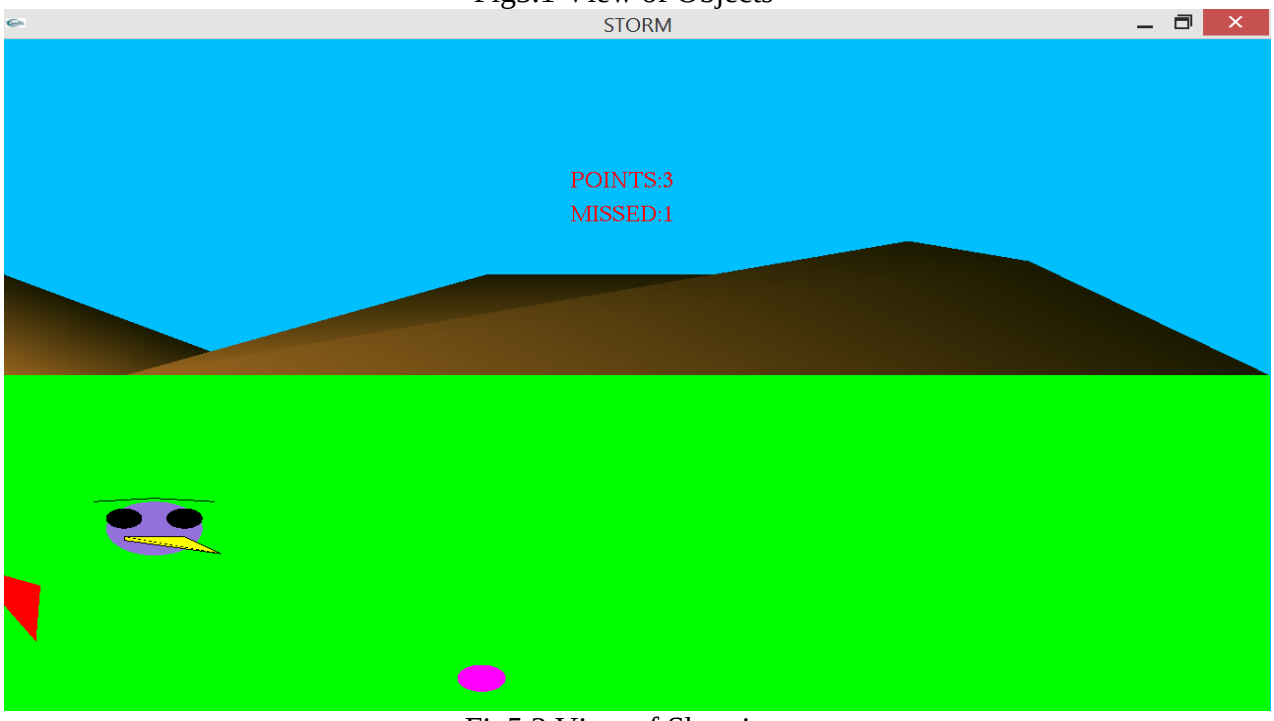


Fig5.2 View of Shooting

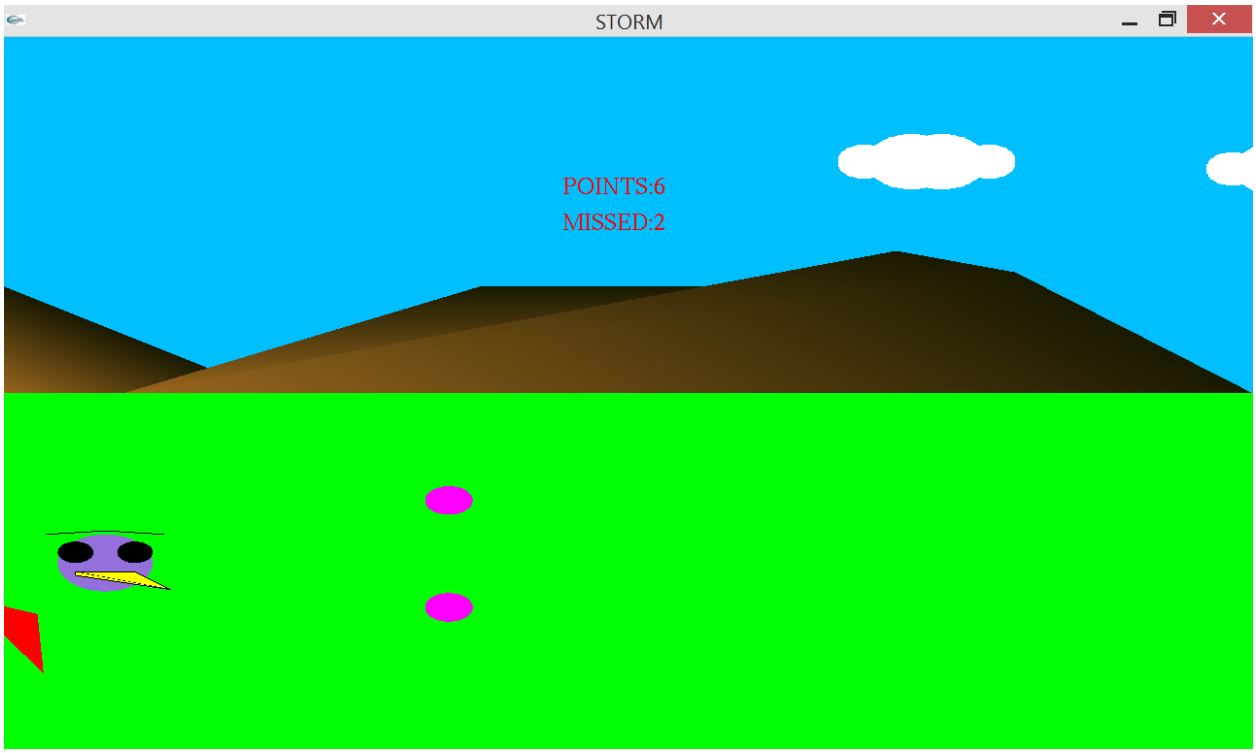


Fig5.3 Points gained after shooting

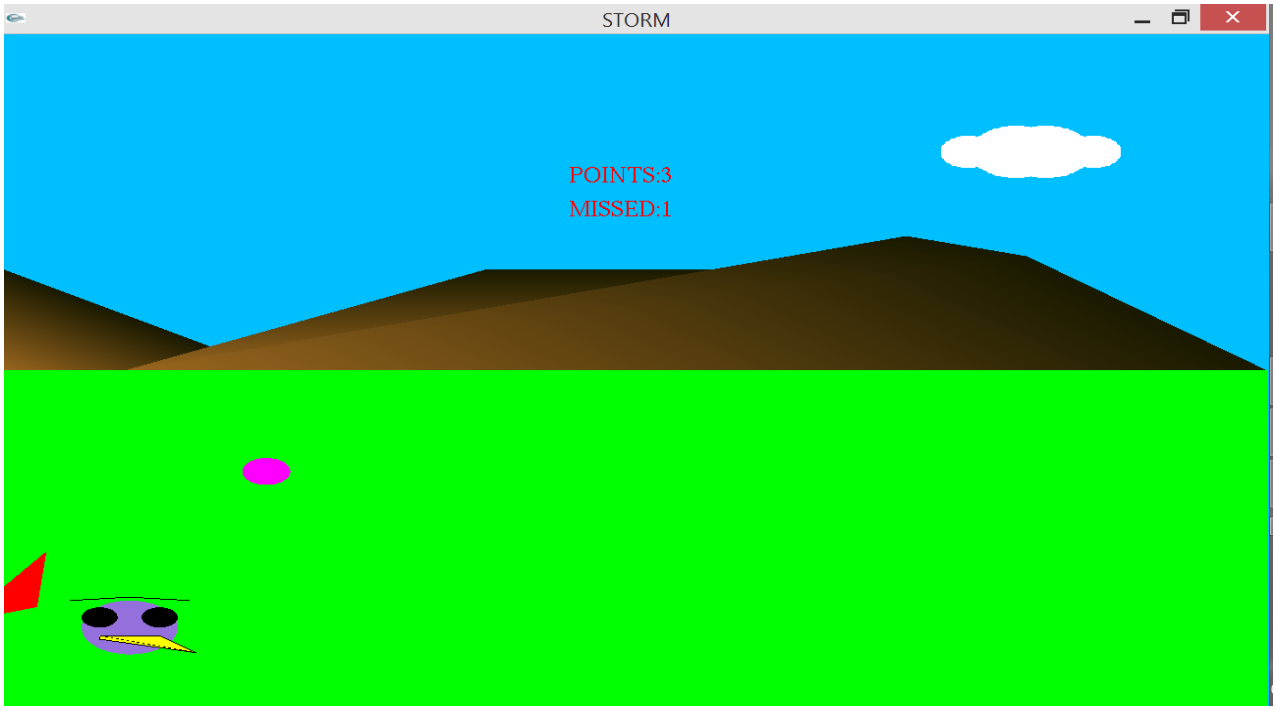


Fig5.4 Target left to shoot

Chapter-6

FUTURE ENHANCEMENT

1. Projectile motion can be implemented.
2. More number of designs can be implemented.
3. We can add number of other objects.
4. Can be modified further to make a better game.
5. Sound and music can be added.
6. Blocks can be destroyed.
7. Further enhancement to develop the game in 3D.

Chapter-7

CONCLUSION

We have been successful in our attempt at implementing “STORM”.

This project was developed in Windows 8 as it is a programmer friendly OS. It was also tested on Windows XP, Windows 7 Mac OS X and Ubuntu 10.04 which is a Linux based OS making it a truly cross-platform project

Here the keys like UP , DOWN , and S are used to perform certain operations like moving the shooter up, moving the shooter up, and to shoot respectively. Here we have used some of the function display(), mouse(), myinit(), myReshape(), keypress(). The functions come under the header file GL/glut.h…

This project makes use of various functions provided by OpenGL. Doing this project has helped us understand and implement the concepts of OpenGL. In this project we have successfully designed the STORM(BIRD FIRING ON TARGET).

REFERENCES

Books:

[1] Interactive Computer Graphics A Top-Down Approach with OpenGL -Edward Angel

[2] Computer Graphics using Open GL by F.S. Hill Jr. & Stephen M. Kelley Jr

Websites:

[1] http://www.opengl.org [Official Website]

[3] http://www.glprogramming.com/red/ [The Official Guide to Learning OpenGL]

[4] http://www.opengl.org/resources/libraries/glut/glut_downloads.php [GLUT]

[5] http://pyopengl.sourceforge.net/documentation/manual [Documentation]

[6] http://stackoverflow.com/questions/tagged/opengl [Questions]

APPENDIX

USER MANUAL

This game is about hitting a target. Here bird hits the target.

All the instruction will be displayed once you run the program.

KEY('UP'):used to make the shooter up.

KEY('DOWN'):used to make the shooter down.

KEY('S'):used to shoot the bird.

KEY('R'):used to resume the game.

KEY('H'):used to know the rules to play the game.

User can always press RIGHT MOUSE BUTTON to play or quit the game.

PERSONAL DETAILS

NAME: SURAJ MANDAL

TWINKLE AGARWAL

USN: 1DT14CS100

1DT14CS106

SEMESTER AND SECTION: 6TH SEMESTER AND SECTION 'B'

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

COLLEGE: DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

EMAIL ID: surajmandal83@gmail.com

twinklertg@gmail.com