VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM - 590014

A CG Mini-Project Report

On

"PRIMITIVE STICK FIGHTERS"

A Mini-project report submitted in partial fulfillment 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:

THEEKSHANA V(1DT18CS108)

YUKTI VIJAY (1DT19CS181)

Under the Guidance of:

Mrs.Apoorva (Asst.Prof.Dept of CSE)

Mrs.Mangala (Asst.Prof.Dept of CSE)

Department of Computer Science and Engineering

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY

AND MANAGEMENT

Kanakapura Road, Udayapura, Bangalore

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY ANDMANAGEMENT,

Kanakapura Road, Udayapura, Bangalore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Mini-Project on Computer Graphics (CG) entitled **"PRIMITIVE STICK FIGHTERS"** has been successfully carried out by **THEEKSHANA V (1DT18CS108) and YUKTI VIJAY (1DT19CS181)** a bonafide students of **Dayananda Sagar Academy of Technology and Management** in partial fulfillment 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 2021-22. 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.

Dr. C. Nandini
(Vice Principal & HOD, Dept. of CSE)

GUIDE:
Mrs. Apoorva Busad
(Asst. Prof. Dept of CSE)

Examiners: **Signature with date:**

1.

2.

ACKNOWLEDGEMENT

It gives us immense pleasure to present before you our project titled **“PRIMITIVE STICK FIGHTERS”.**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. Ravishankar M** for all their support.

We express our deepest gratitude and special thanks to **Dr. C. Nandini, Vice Principal and HOD of Department of Computer Science Engineering,** for all her guidance and encouragement.

We sincerely acknowledge the guidance and constant encouragement of our mini-project guide **Mrs. Apoorva Busad (Asst. Prof. Dept of CSE), Mrs. Mangala H S (Asst. Prof. Dept of CSE).**

THEEKSHANA V (1DT18CS108)
YUKTI VIJAY (1DT19CS173)

TABLE OF CONTENTS

Chapter #	Chapter Name	Page #
1	INTRODUCTION	1
	About Computer Graphics	1
	About OpenGL	2
	Features of OpenGL	2
	OpenGL Architecture	3
	About the project	3
2	REQUIREMENTS	4
	Software Requirements	4
	Hardware Requirements	4
3	DESIGN	5
	Abstract	5
4	IMPLEMENTATION	6
	Headers defined	6
	Implementation of OpenGL built-in functions used	6
	User defined functions used	8
	Standard library functions used	9
	Source code	11
5	SNAPSHOTS	19
6	CONCLUSION	21
	Conclusion	21
	References	22

LIST OF FIGURES

SL#	TOPIC	PAGE No
1	Graphic System	2
2	OpenGL Architecture	3
3	Design of the project	5
4	First screen	19
5	Game Starts	19
6	Fighting	20
7	Game Over	20

s

Chapter1

INTRODUCTION

About Computer Graphics

The Computer Graphics is one of the most effective and commonly used methods to communicate the processed information to the user. It displays the information in the form of graphics objects such as pictures, charts, graphs and diagram instead of simple text.

In computer graphics, pictures or graphics objects are presented as a collection of discrete picture elements called **pixels**. The pixel is the smallest addressable screen element

Computer graphics today is largely interactive: The user controls the contents structure, and appearance of objects and their displayed images by using input devices, such as a keyboard, mouse, or touch-sensitive panel on the screen.

Computer Graphics relies on an internal model of the scene, that is, mathematical representation suitable for graphical computations. The model describes the 3D shapes, layout and materials of the scene. This 3D representation then has to be projected to compute a 2D image from a given viewpoint, this is rendering step. Rendering involves projecting the objects, handling visibility (which parts of objects are hidden) and computing their appearance and lighting interactions. Finally, for animated sequence, the motion of objects has to be specified.
Computer graphics today is largely interactive: The user controls the contents structure, and appearance of objects and their displayed images by using input devices, such as a keyboard, mouse, or touch-sensitive panel on the screen.

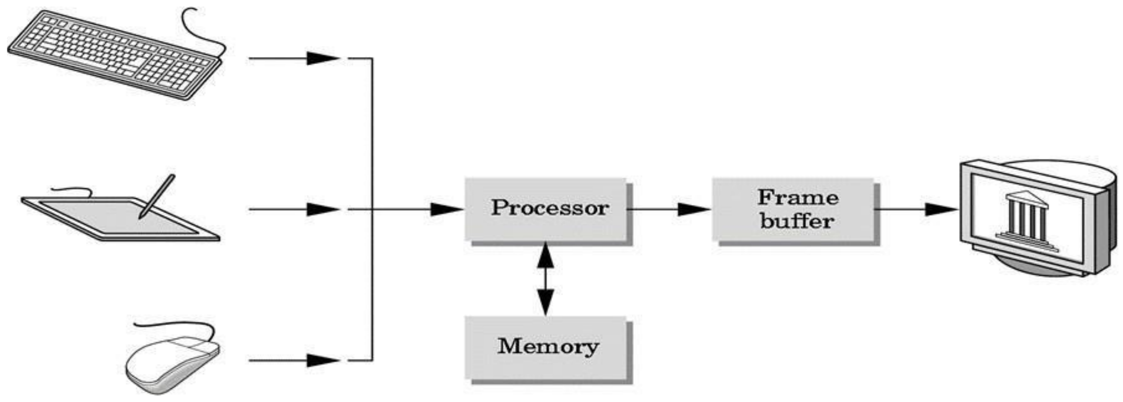


Fig 1.1: Graphic System

The image processing can be classified as

- Image enhancement.
- Pattern detection and recognition
- Scene analysis and computer vision.

The image enhancement deals with the improvement in the image quality by eliminating noise or by increasing image contrast. Pattern detection and recognition deals with the detection and clarification of standard patterns

And finding deviations from these patterns .The optical character recognition (OCR) technology is a practical example for pattern detection & recognition. Scene analysis deals with the recognition and reconstruction of 3D model of scene from several 2D images.

About OpenGL

OpenGL (**O**pen **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.

OpenGL's basic operation is to accept primitives such as points, lines and polygons, and convert them into pixels. This is done by a graphics pipeline known as the OpenGL state machine.

Features of OpenGL:

- Geometric Primitives allow you to construct mathematical descriptions of objects.
- Viewing and Modeling permits arranging objects in a 3-dimensional scene, move our camera around space and select the desired vantage point for viewing the scene to be rendered.
- Materials lighting OpenGL provides commands to compute the color of any point given the properties of the material and the sources of light in the room.
- Transformations: rotation ,scaling, translations, perspectives in 3D etc.

OpenGL Architecture

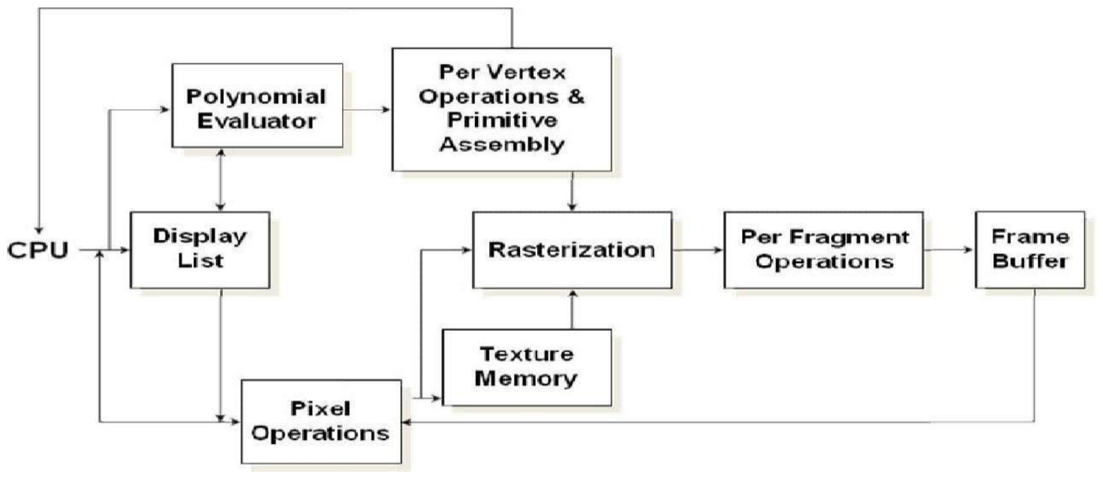


Fig 1.2: OpenGL Architecture

This is the most important diagram you will see today, representing the flow of graphical information, as it is processed d from CPU to the frame buffer.

There are two pipelines of data flow. The upper pipeline is for geometric, vertex-based primitives. The lower pipeline ne is for pixel-based, image primitives. Texturing combines the two types of primitives together.

About the Project

Computer graphics involves the designing of objects in different forms which are regular and irregular in shape. The mini project named “ICEBERG COLLISION” creates 2D design of a river view in which the impact of an iceberg is represented. This project is designed and implemented using OpenGL interactive application that basically deals with providing the graphical interface between the user and the system. Throughkeyboard events the user can start or abort the display has been implemented, which also includes day and night mode of river view.

Chapter 2

REQUIREMENT SPECIFICATION

The requirement specification is a comprehensive description of the software and the hardware requirements required to run the project successfully.

Software Requirements

Operating System: Windows XP, Windows 2000 Language

Tools: OpenGL

Compiler: GNU GCC compiler/C++ compiler

Documentation Tool: CodeBlocks

Hardware Requirements

Processor: Pentium 3 and above RAM: 1GB with 256MHz

Hard disk: 2MB

Keyboard: Standard 101 key keyboard

Chapter3

DESIGN

Design is the planning that lays the basics for the making of every objects or systems. This chapter involves designing of various aspects and different stages of project. When program is made to execute, the output window is displayed first. The flow of operation from output window is shown in figure.

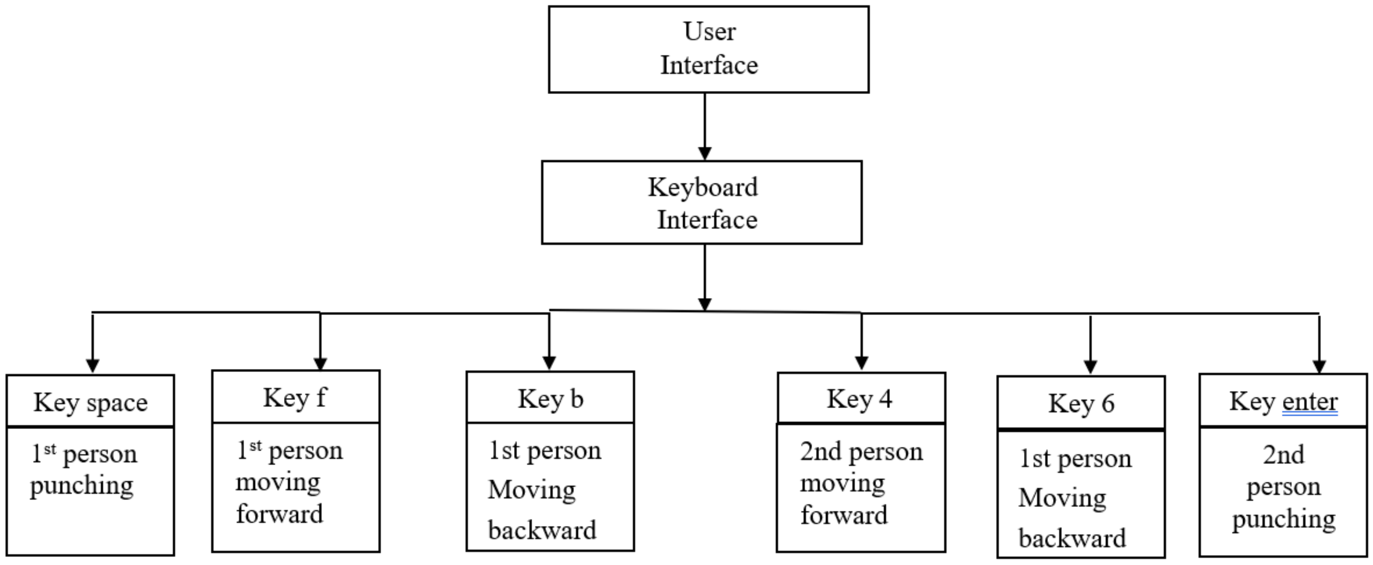


Fig 3.1 Design of the project

Chapter 4

IMPLEMENTATION

The implementation stage of this model involves the following phases.

- Implementation of OpenGL built in functions.
- User defined function Implementation.

Headers defined

The inbuilt functions are defined in the OpenGL library. Some of the headers that are usedare as follows.

- **#include<stdio.h>:** To take input from standard input and write it to standard output.
- **#include<GL/glut>:** To include glut library functions.
- **#include<math.h>:** To define common mathematical functions.

Implementation of OpenGL Built In Functions used:

1. glutInit():

glutInit is used to initialize the GLUT library.

Usage: void glutInit(int *argcp, char **argv);

Description: glutInit will initialize the GLUT library and negotiate a session with the windowsystem.

2. glutInitDisplayMode():

glutInitDisplayMode sets the initial display mode.

Usage: void glutInitDisplayMode (unsigned int mode);

Mode-Display mode, normally the bitwise OR-ing of GLUT display mode bit masks.

Description: The initial display mode is used when creating top-level windows, sub windows, and overlays to determine theOpenGL display mode for the to-be-created window or overlay.

3. glutCreateWindow():

glutCreateWindow creates a top-level window.

Usage: int glutCreateWindow(char *name);

Name - ASCII character string for use as window name.

Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window. Each created window has a unique associated OpenGL context.

4. glutDisplayFunc():

glutDisplayFunc sets the display callback for the current window.

Usage: void glutDisplayFunc(void (*func)(void));Func- Thenew display callback function.

Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback forthe window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

5. glutMainLoop():

glutMainLoop enters the GLUT event processing loop.

Usage: void glutMainLoop(void);

Description: glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

6. glMatrixMode( ):

The two most important matrices are the model-view and projection matrix. At any time, the stateincludes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

7. glutTimerFunc():

glutTimerFunc registers a timer callback to be triggered in a specified number of milliseconds.

Usage: voidglutTimerFunc(unsigned int msecs,void (*func)(int value), value);

Description: glutTimerFunc registers the timer callback func to be triggered in at least msecs milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc. Multiple timer callbacks at same or differing times may be registered simultaneously. The number of milliseconds isa lower bound on the time before the callback is generated. GLUT attempts to deliver the timer callback as soon as possible after the expiration of

the callback's time interval. There is no support for canceling a registered callback. Instead, ignore callbackbased on its value parameter when it is triggered.

User defined functions used in the project

- **keyuppress( ) :** This function is called when the keyboard button is released.
- **keydownpress ( ) :** This function is called when the keyboard button is pressed.
- **drawString ( ) :** This function is used to draw a text on the screen.
- **drawStickman( ) :** This function is used to draw both stick man figures
- **updategame ( ) :** This function is used to position the players and are updated
- **display( ) :** This function is used to show all the possible display along with a timer and shows player1 and player2’s health point.
- **reshape ( ) :** This function is called when the opengl window is resized.

Standard library functions used in the project:

- **glutInit (int *argc, char **argv):** glutInit is used to initialize the GLUT library.
- **glutInitDisplayMode(unsigned int mode):**glutInitDisplayMode sets the initial display mode.
- **glutInitWindowPosition(intx,int y):** Specifies the initial position of the top-left cornerof the window in pixels.
- **glutInitWindowSize(intwidth,int height):** Specifies the initial height and width of the window in pixels.
- **glutKeyboardFunc (void *f(char key,intwidth,int height):**KeyboardFunc sets the keyboard callback for the current window.
- **glClear():**The clear function clears buffers to preset values.
- **glClearColor(GLclampf r, GLclampf g, GLclampf b, GLclampf a):**The glClearColor function specifies clear values for the color buffers.
- **glMatrixMode(GLenum mode):**This function specifies which matrix is the currentmatrix.
- **glLoadIdentity():** Set the current transformation matrix to an identity matrix.
- **glPushMatrix(),glPopMatrix():** This pushes to and pops from the matrix stack corresponding to the current matrix mode.
- **glPointSize():**The **glPointSize** function specifies the diameter of rasterized points.
- **glTranslate[fd](TYPE x,TYPEy,TYPE z):** This function multiplies the current matrix by a translation matrix.
- **gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top):**The gluOrtho2D function defines a 2-D orthographic projection matrix.

- **glBegin (glEnum mode):** It initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS, GL_LINES, GL_LINE_STRIP, and GL_POLYGON.
- **glEnd():** It terminates a list of vertices.
- **glFlush ():**It forces any buffered openGL commands to execute.
- **glutMainLoop() :**Itcauses the program to enter an event processing loop.
- **glutDisplayFunc(void (*func)(void)):**It registers the display function ‘func’ that isexecuted when the window needs to be redrawn.

SOURCE CODE (FIRST 100 LINES)

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

    int player1x, player1y,
    player2x,player2y;
    int player1walkaction,
    player1walkcounter,
    player2walkaction,
    player2walkcounter;
    bool
    player1punch,player2punch,player
    1waspunching,
    player2waspunching;
    int player1punchcounter,
    player2punchcounter;
    int player1moveback,
    player2moveback;

    int player1hp, player2hp; //health
    point

bool keystates[256];
keys pressed either true or false
char buffer[100];
bool pause;
```

```
void initgame()
{
        player1x = 240;
        player1y = 300;
        player2x = 400;
        player2y = 300;
        player1hp = 10;
        player2hp = 10;
        player1moveback = 0;
        player2moveback = 0;
        player1walkaction = 0;
        player2walkaction = 0;
        player1punch = 0;
        player2punch = 0;
}

void KeyUpPress(unsigned char key,
int x, int y ){
    switch(key){
        case 'w':
      keystates['w'] = false;

        break;

        case 's':
      keystates['s'] = false;
                break;

        case 'f':
          keystates['f'] = false;
                break;
        case 'b':
      keystates['b'] = false;
                break;

        case '8':
      keystates['8'] = false;

        break;

        case '2':
      keystates['2'] = false;
                break;

        case '4':
          keystates['4'] = false;
                break;

        case '6':
      keystates['6'] = false;
                break;


        case ' ':
                keystates[' '] = false;
                break;

        case 13:
                keystates[13] = false;
                break;

        case 'c':
                break;
          case 27:
                  exit(0);
           }
}
void KeyDownPress(unsigned char
key, int x, int y ){
   switch(key){
        case 'w':
                keystates['w'] = true;
        break;

        case 's':
                keystates['s'] = true;
                break;

        case 'f':
                keystates['f'] = true;
                break;

        case 'b':
                keystates['b'] = true;
                break;

        case '8':
                keystates['8'] = true;
        break;

        case '2':
                keystates['2'] = true;
                break;

        case '4':
                  keystates['4'] = true;
                  break;

          case '6':
                  keystates['6'] = true;
                  break;

          case ' ':
                  keystates[' '] = true;
                  break;

          case 13:
                  keystates[13] = true;
                  break;

          case 'y':
                  if(pause == true)
                  {
                          pause = false;
                          initgame();
                  }
                  break;

          case 'n':
                  if(pause == true)
                  exit(0);
                  break;
          case 27:
                  exit(0);
}
     }
     void drawString (void * font, char
*s, float x, float y){
    unsigned int i;
    glRasterPos2f(x, y);
    for (i = 0; i < strlen (s); i++)
        glutBitmapCharacter (font,
s[i]);
}
void drawStickMan(int x, int y, bool mirror, int walkaction, bool punch)
{
    glLoadIdentity();
    glTranslatef(x, y, 0);
    if(mirror == true)
            glScalef(-1, 1, 1);

    glLineWidth(2);
    //head
    glBegin(GL_LINE_LOOP);
        glColor3f(0, 0, 0);
        glVertex2f(0 - 20, - 20 - 20);
        glVertex2f(0 - 20, - 20 + 20);
        glVertex2f(0 + 20, - 20 + 20);
        glVertex2f(0 + 20, - 20 - 20);
    glEnd();

    //body
    glBegin(GL_LINES);
        glColor3f(0, 0, 0);
        glVertex2f(0, 0);
        glVertex2f(0,  30);
    glEnd();

    if(walkaction == 1)
    {
        //feets
        glBegin(GL_LINE_STRIP);
            glColor3f(0, 0, 0);
            glVertex2f(0,  30);
            glVertex2f(0,  60);
            glVertex2f(0 + 10,  60);
        glEnd();
        glBegin(GL_LINE_STRIP);
            glColor3f(0, 0, 0);
            glVertex2f(0,  30);
            glVertex2f(0 + 15,  45);
            glVertex2f(0 - 3,  48);
            glVertex2f(0 + 2,  57);
        glEnd();
    }
    //stick man is walking
    else if (walkaction == 2)
    {
        //feets
        glBegin(GL_LINE_STRIP);
            glColor3f(0, 0, 0);
            glVertex2f(0,  30);
            glVertex2f(0 - 12,  55);
            glVertex2f(0 - 3,  60);
        glEnd();
        glBegin(GL_LINE_STRIP);
            glColor3f(0, 0, 0);
            glVertex2f(0,  30);
            glVertex2f(0 + 10,  60);
            glVertex2f(0 + 20,  55);
        glEnd();
    }
    //stick man is standing
    else
    {
        //feet
        glBegin(GL_LINE_STRIP);
            glColor3f(0, 0, 0);
            glVertex2f(0,  30);
            glVertex2f(0,  60);
            glVertex2f(0 + 10,  60);
        glEnd();
    }
    //hands of stick man if punch is true then draw the punch action
    if(punch == true)
    {
            glBegin(GL_LINE_STRIP);
                        glColor3f(0, 0, 0);
                        glVertex2f(0,  20);
                        glVertex2f(0 + 5,  30);
                        glVertex2f(0 + 10,  15);
            glEnd();
            glBegin(GL_LINES);
                        glColor3f(0, 0, 0);
                        glVertex2f(0,  20);
                        glVertex2f(0 + 40,  20);
            glEnd();
    }
    //else no punch action just standing
    else
```

```
    {
            glBegin(GL_LINE_STRIP);
                    glColor3f(0, 0, 0);
                    glVertex2f(0,  20);
                    glVertex2f(0 + 10,  30);
                    glVertex2f(0 + 20,  15);
            glEnd();
            glBegin(GL_LINE_STRIP);
                    glColor3f(0, 0, 0);
                    glVertex2f(0,  20);
                    glVertex2f(0 + 15,  30);
                    glVertex2f(0 + 25,  15);
            glEnd();
    }
    glLoadIdentity();
 }
```

Chapter 5

SNAPSHOTS

A snapshot is the state of the system at a particular point in time. It can refer to the actualcopy of a state of a system or to a capability provide by systems.

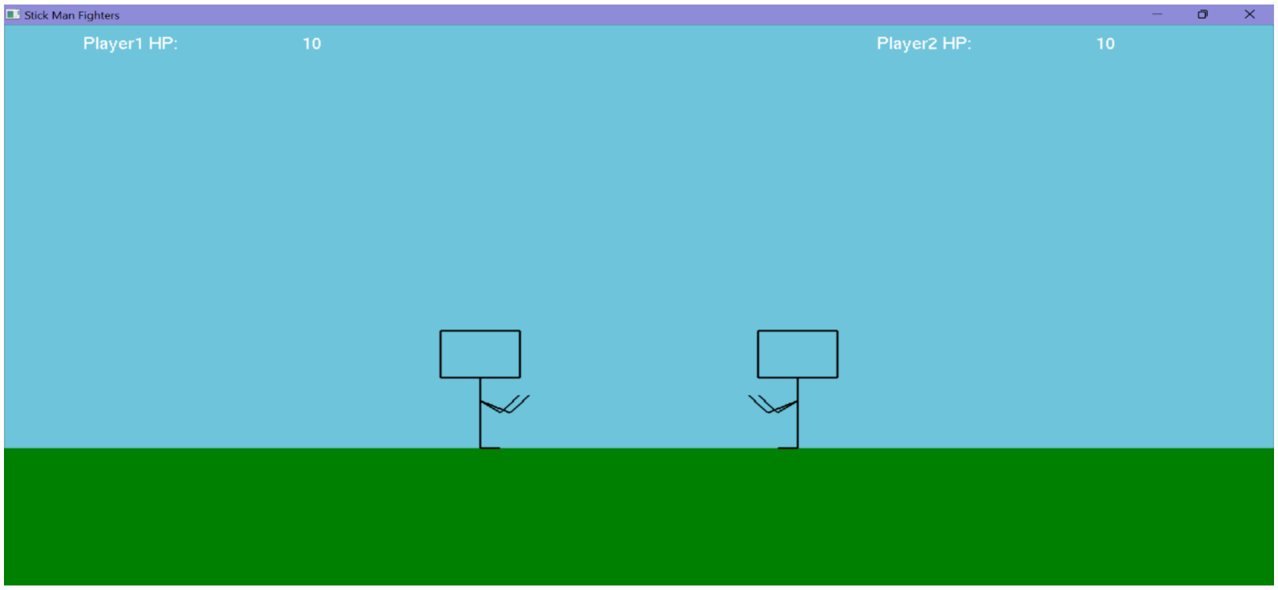


Fig 5.1: First screen

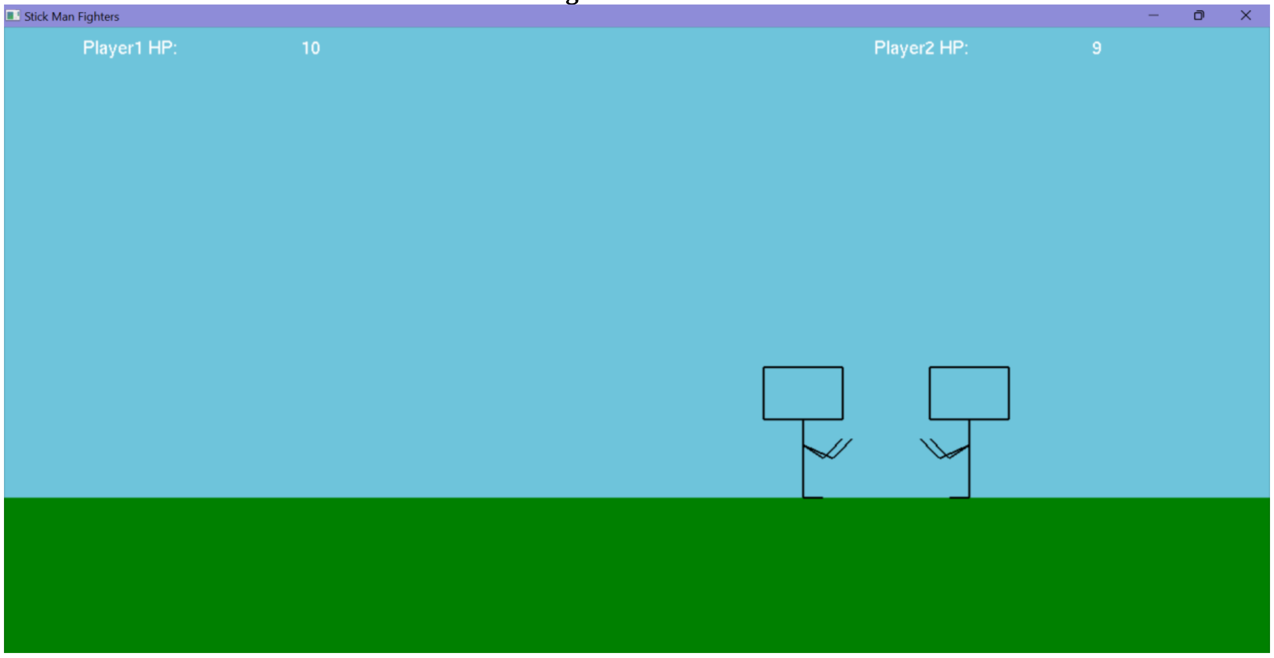


Fig 5.2: Game starts

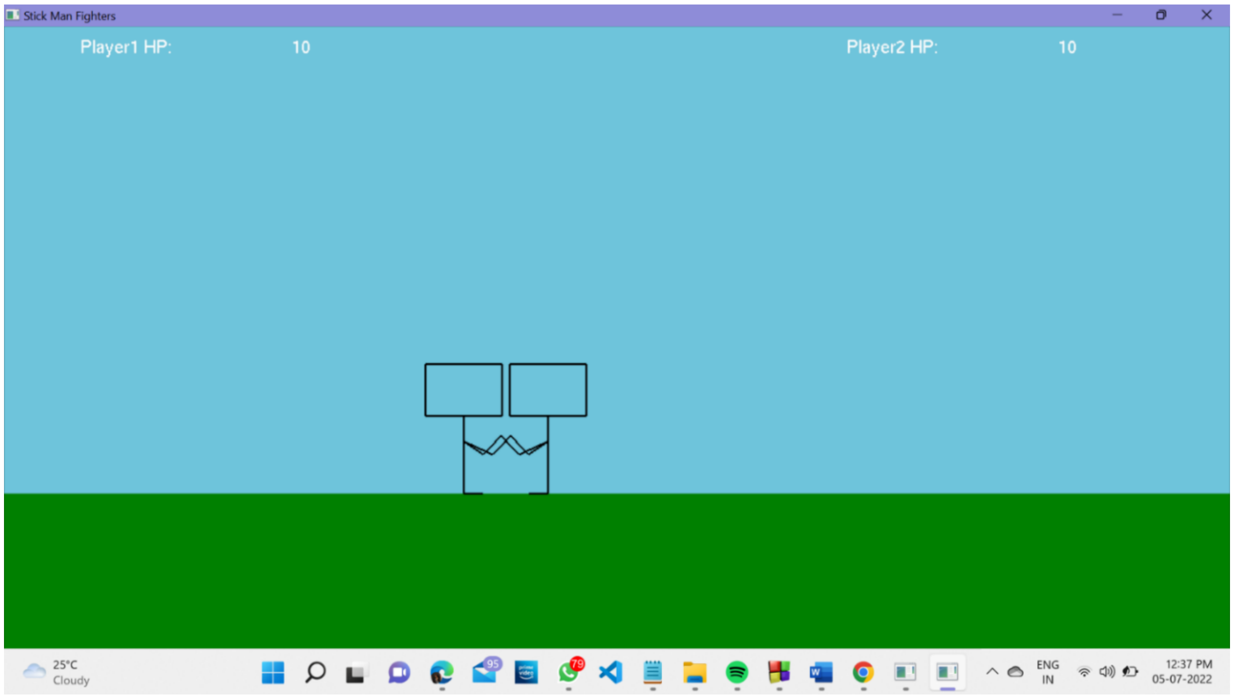
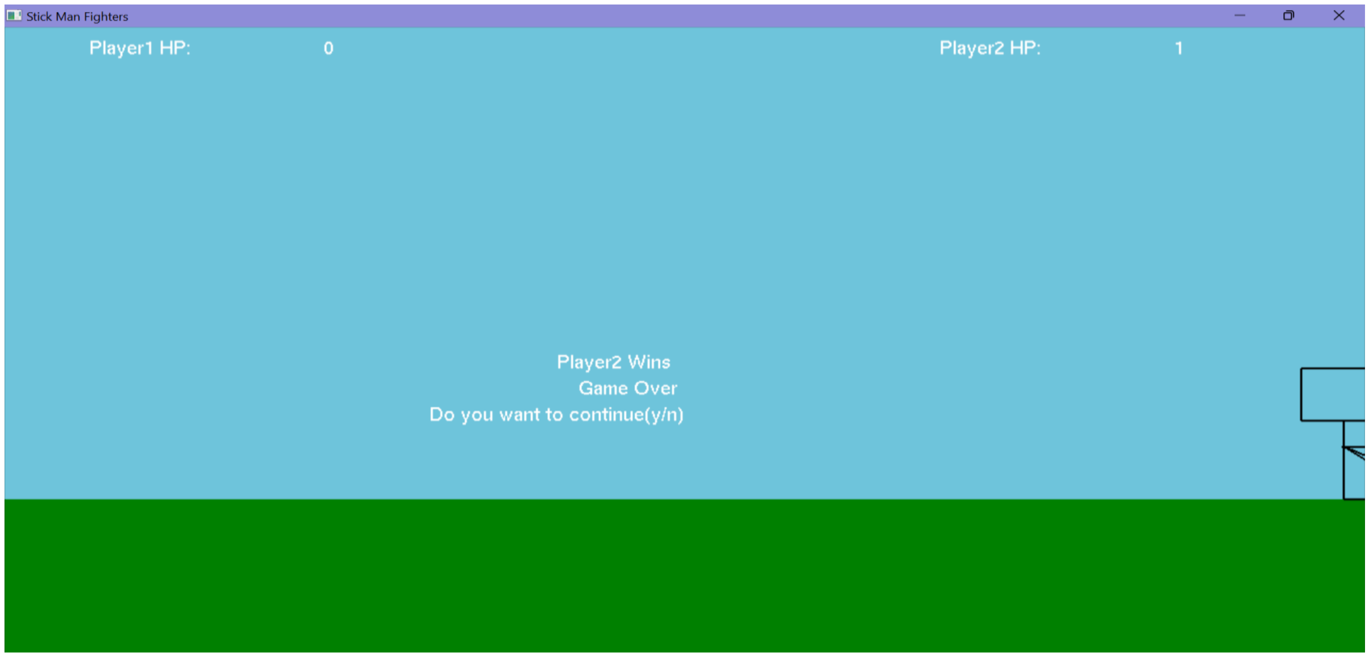


Fig 5.3: Fighting



```
Player2 Wins
Game Over
Do you want to continue(y/n)
```

Fig 5.4: Game over

Chapter 6

CONCLUSION

We have attempted to design and implement “Primitive Stickman Fighters”.OpenGl supports enormous flexibility in the design and the use of OpenGl graphics programs.The presence of many builtin classes methods takecare of much functionality and reduce the job of coding as well as makes the implementation simpler .We have implemented the project making it user-friendly and the error free as possible.

REFERENCES

BOOKS

[1] Edward Angel: Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, Pearson Education, 2008.

[2] Donald Hearn and Pauline Baker: Computer Graphics- OpenGL Version, 3rd Edition, Pearson Education, 2008.

[3] F.S Hill Jr.: Computer Graphics Using OpenGL, 3rd Edition, PHI, 2009.

WEBSITES

[1] http://www.opengl.org/resources/libraries/glut/

[2] http://www.opengl.ord/

[3]http://en.wikipedia.org/wiki/opengl