VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELAGAVI– 590018, KARNATAKA, INDIA

A PROJECT REPORT

on

“Brick Breaker Game”

Submitted in partial fulfilment of the requirements for the award of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING

Submitted By

Name	USN
Devadatta Bhat	4VP19CS024
Dhanush Ghate D	4VP19CS026

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING
VIVEKANANDA COLLEGE OF ENGINEERING & TECHNOLOGY
[A Unit of Vivekananda Vidyavardhaka Sangha Puttur (R)]
Affiliated to Visvesvaraya Technological University and Approved by AICTE New Delhi & Govt., of Karnataka
Nehru Nagar, Puttur - 574 203, DK, Karnataka, India.
JULY, 2022

VIVEKANANDA COLLEGE OF ENGINEERING & TECHNOLOGY

[A Unit of Vivekananda Vidyavardhaka Sangha Puttur (R)]

Affiliated to Visvesvaraya Technological University and Approved by AICTE New Delhi & Govt. of Karnataka

Nehru Nagar, Puttur - 574203, DK, Karnataka, India

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the project work entitled **"Brick Breaker Game"** is carried out by **DEVADATTA BHAT, DHANUSH GHATE D** bearing USNs **4VP19CS024, 4VP19CS026** respectively bonafide students of **Vivekananda College of Engineering & Technology, Puttur** in partial fulfilment for the award of **Bachelor of Engineering** in **Computer Science &Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2021-22. It is certified that all corrections/suggestions indicated during Internal Assessment have been incorporated in the report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature of the Guide	**Signature of the Guide**	**Signature of the HOD**
(Mr. Krishna Mohana A J)	**(Mr. Ajay Shastry C G)**	**Mr. Krishna Mohana A J**

EXTERNAL VIVA

Name of the Examiners   Signature with date

1)…………………………….…..………..   ……………………………….

2)…………………………….…..………..   ……………………………….

ACKNOWLEDGEMENT

We take this opportunity to express our deep heartfelt gratitude to all those people who have helped us in the successful completion of the project.

First and foremost, we would like to express our sincere gratitude to our guides, **Mr. Krishna Mohana A J**, for providing excellent guidance, encouragement and inspiration throughout the project work. Without their invaluable guidance, this work would never have been a successful one

We would like to express my sincere gratitude to our Head of the Department of Computer Science &Engineering, **Mr. Krishna Mohana A J** for his guidance and inspiration.

We would like to thank our Principal, **Dr. Mahesh Prasanna K** a for providing all the facilities and a proper environment to work in the college campus.

We are thankful to all the teaching and non-teaching staff members of Computer Science & Engineering Department for their help and needed support rendered throughout the project.

DECLARATION

We, **DEVADATTA BHAT (4VP19CS024), DHANUSH GHATE D (4VP19CS026)** students of sixth semester B. E. in Computer Science & Engineering, **Vivekananda College of Engineering & Technology**, Puttur, hereby declare that the project work entitled **“Brick Breaker Game”** has been carried out and duly executed by me at VCET, Puttur, under the guidance of **Mr. Krishna Mohana A J**, Assistant Professor, Department of Computer Science & Engineering, Vivekananda College of Engineering & Technology, Puttur, and submitted in partial fulfillment of the requirements for the award of degree in **Bachelor of Engineering in Computer Science & Engineering** by **Visvesvaraya Technological University**, Belagavi during the academic year 2021-2022

Devadatta Bhat	**4VP19CS024**	**signature**
Dhanush Ghate D	**4VP19CS026**	**signature**

Date:

Place: VCET, Puttur.

ABSTRACT

In this project titled as “Brick Breaker – the game”. We try to recreate the classic game of “breakout” which was developed Atari, Inc. in a simple way by using OpenGL library. This can be considered as the lite version of the original game. This was implemented to demonstrate the use of OpenGL library and its functions. The object of brick breaker is to break the bricks that are distributed around the top of the game screen. The bricks are broken after coming in contact with a ball that bounces around the screen. At the bottom is a paddle that in the classic game moves based on user input. The user has to make sure the ball bounces off the paddle without going off the bottom of the screen. It is a simple game where the user or player has to move the paddle located at the bottom of the window to not let the moving ball which is bouncing hit the bottom part of the window. The glut Special Function of mouse and keyboard is used for interactive inputs for playing the game. Where keyboard is used for controlling the paddle and mouse functions dictate the active state of the game. By using OpenGL reliable graphics package that provides the user with the basic working of the Brick breaker game with score calculation. The user-friendly interface allows the user to interact with it very effectively.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Introduction to Brick Breaker Game

Brick breaker Game is very easy to play. In the game, you play with a ball and a paddle. Your mission is to make the ball fly and touch the blocks in the above of the paddle. There are many bricks in specific order above your paddle. You have to click S letter to let the ball fly. The ball flies and touches the bricks above. When a brick is touched, it will fall off to the bottom of the screen. The ball bounces around your screen and falls too. When it falls down, you have to use the paddle to catch the ball so that it does not touch the bottom of the screen.

If the ball touches the bottom of the screen, you will lose a life. You have only one lives for each time you play this game. Therefore, you must always be attentive and mustn't let your ball touch the bottom of the screen. For each brick the ball touches, you will receive points. The game counts the points you have score to claim your record. Try your best to make the ball touch the bricks. However, if the ball touches the bricks, the bricks will explode and make the ball's flying way change, which makes it harder to catch the ball when it falls.

You have only one turn to play in each time you play, while the game has so many modes Therefore, you have to try very hard to finish/win the game. It is easy to play this game, but it is not easy to win this game.

1.2 Introduction to Computer Graphics

Computer Graphics in today's world is one of the most widely used powerful and interesting features of the computer which gives us the power to handle the graphical data very efficiently and also process them rapidly and effectively. It is concerned with all the aspects of producing pictures or images using a computer.

Graphics are defined as any sketch or a drawing or a special network that pictorially represents some meaningful information. Computer Graphics is used where a set of images needs to be manipulated or the creation of the image in the form of pixels and is drawn on the computer. Computer Graphics can be used in digital photography, film, entertainment, electronic gadgets, and all other core technologies which are required. It is a vast subject and area in the field of computer science. Computer Graphics can be used in UI design, rendering, geometric objects, animation, and many more. In most areas, computer graphics is an abbreviation of CG. There are several tools used for the implementation of Computer Graphics.

1.3 Types of Computer Graphics

1.3.1 Raster Graphics:

In raster, graphics pixels are used for an image to be drawn. It is also known as a bitmap image in which a sequence of images is into smaller pixels. Basically, a bitmap indicates a large number of pixels together.

1.3.2 Vector Graphics:

In vector graphics, paths made from mathematical formulae are used to draw different types of shapes, lines, objects, and colour.

1.4 Introduction to OpenGL

OpenGL (Open Graphics Library) is a cross-platform, hardware-accelerated, language-independent, industrial standard API for producing 3D (including 2D) graphics. Modern computers have dedicated GPU (Graphics Processing Unit) with its own memory to speed up graphics rendering. OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn’t the computer that runs your graphics program. OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. OpenGL doesn’t provide high-level commands for describing models of three-dimensional objects.

1.4.1 OpenGL Architecture

This is a diagram representing the flow of graphical information, as it is processed from CPU to the frame buffer, in OpenGL. There are two pipelines of data flow. The upper pipeline is for geometric, vertex-based primitives. The lower pipeline is for pixel-based, image primitives. Texturing combines the two types of primitives together.

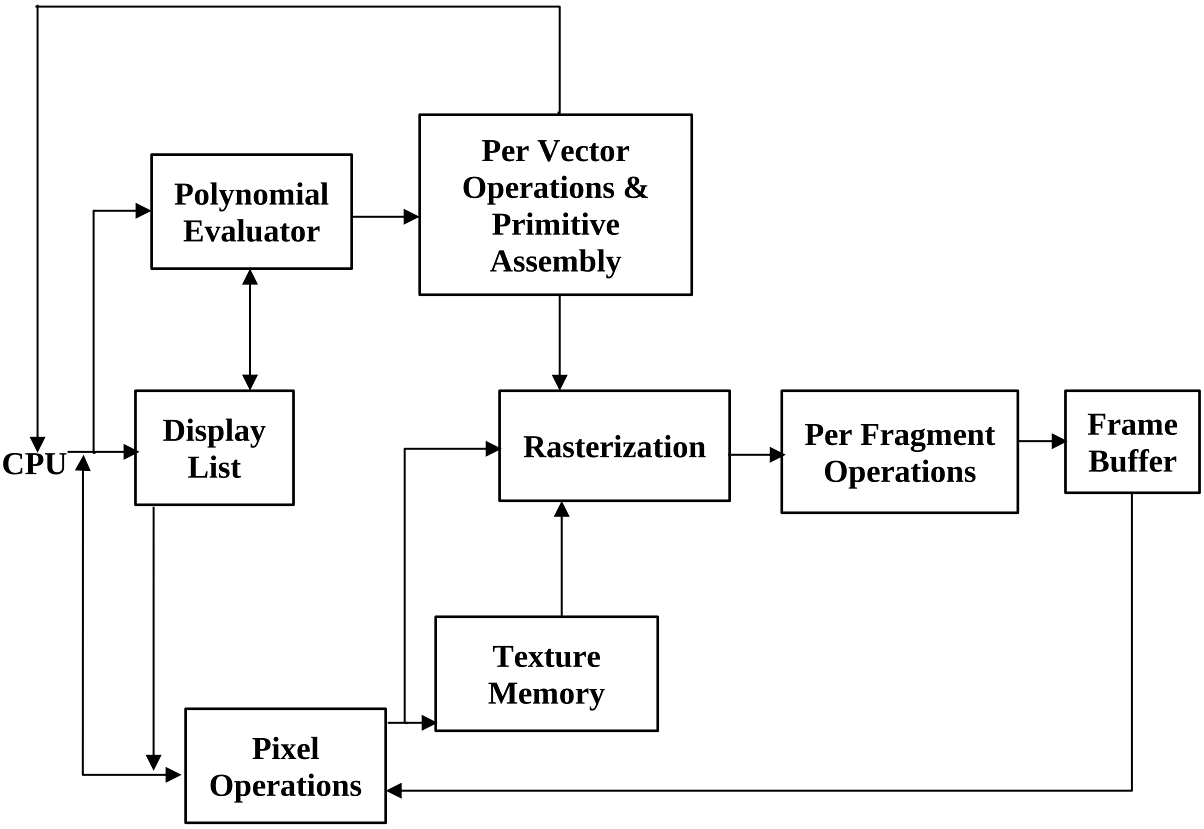


Fig: 1.4 Library Organization.

There are several APIs related to OpenGL:

- AGL, WGL, GLX	: OpenGL & windowing systems interfaces
- GLU	: 2D & 3D Geometry, tessellations, etc.
- GLUT	: Portable windowing API.

CHAPTER 2

ANALYSIS AND REQUIREMENT SPECIFICATION

2.1 System Requirements

The package is designed such that users with a computer having minimum configuration can also use it, which does not require complex graphics packages. The package requires simple in-built functions fount in <GL/glut.h> header file along with a few users defined functions.

2.1.1 Hardware specification

Processor	: Intel core i3 and above or the equivalent models.
Monitor resolution	:1024 x 768 display and above resolution.
Keyboard and Mouse.	
RAM 2 GB	
Hardware Space	: Minimum 2GB

2.1.2 Software specification

Operating system	: UBUNTU or Windows.
Tool used	: OpenGL.
Libraries	: freeglut3
Language used	: C/C++ Language.

CHAPTER 3

DESIGN

3.1 Algorithm

Algorithm is a well-defined, step-by-step, logical procedure for solving a problem. It takes a set of inputs and produces the desired output. In this project we use the algorithm to remove the bricks when a ball hits.

3.1.1 Algorithm for the generation of Bricks

Brick generation has to done at the top of screen row and column wise.

```
Start (Brick_Generation)
    Set two Variables i & j
            if(start == 0) then i is for rows and other one for column.
            To generate the bricks:
                for row ( brick_array[i][j].x = j variable increment);
                for column (brick_array[i][j].y = i variable increment);
end;
```

For both rows and column, the variables can be passed together to the brick generation function.

3.1.2 Function to handle the case when the ball strikes the bricks

```
Start (hit function)
    if( ball_y coordinate >= inside brick_array[i][j].y
                                            &&
                                            ball_x coordinate >= inside brick_array[i][j].x)
        brick_array[i][j].y = value set as zero;
        brick_array[i][j].x = valueset as zero;
        score gets incremented by 1;
end;
```

When the ball comes inside the vicinity of the generated brick array, that particular array row and column values are set to zero to remove that brick.

The score is incremented on each iteration of removal of brick.

Once all the array values become zero game ends and displays winning message.

3.2 Block Diagram

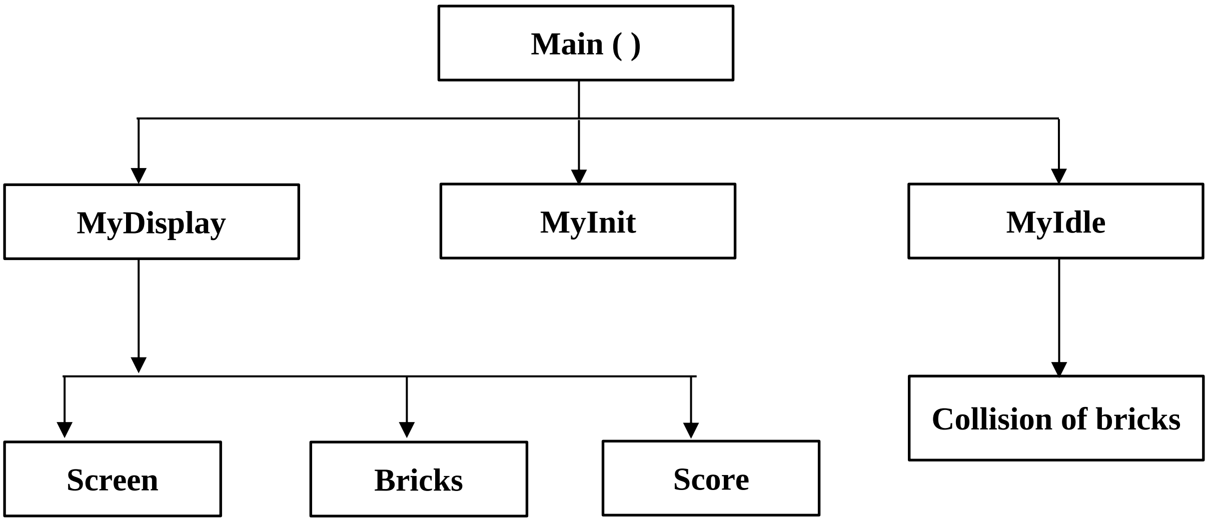


Fig: 3.2. Brick Breaker Game Block Diagram.

3.3 User Manual

By using the keyboard interface, we perform the required action.

The functionality of various keyboard key is:

s | S key is used to start the Game

a | A & d | D keys are used to move the paddle on the screen

Mouse movement can also be used to move the paddle.

Left Click gives the dropdown lists.

CHAPTER 4

IMPLEMENTATION

4.1 User-defined Functions

4.1.1 void display(void)

This function is used to display the actual logic of code in the display window.

glColor3f(): It can be used to give each vertex its own color.

glBegin():The glBegin function accepts a single argument that specifies which of ten primitives the vertices compose.

glVertex3f(): It is used locate a vertex with 3 floating point coordinates.

glEnd(): This function is used to specify end of a shapes that is defined in glBegin().

glutSwapBuffers(): glutSwapBuffers swaps the buffers of the current window if double buffered.

4.1.2 void user_obj(int x_value)

This function is used to display user object and based on x_value we can set the position of user object.

glFlush():The glFlush function empties all these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine.

4.1.3 void ground_line(void)

This function is used to display ground line in the screen.

glClear(GL_COLOR_BUFFER_BIT): Indicates the buffers currently enabled for color writing.

4.1.4 void rect_obstacle(float x_cord,float time_to_start)

This function is used to set the location of rectangular object that is falling from top of the screen. We set location coordinate of object by using x_cord value.

4.1.5 void timer(int)

This function is used to repeat the code at the rate of 60 frames per second.

glutTimerFunc():glutTimerFunc registers the timer callback function to be triggered in at least msecs milliseconds.

glutPostRedisplay():glutPostRedisplay marks the current window as needing to be redisplayed.

4.1.6 void keyboard_keys(int key, int x, int y)

This function is used to take user input from keyboard.

4.1.7 void draw ball()

This function is used to draw the spherical ball.

4.1.8 void handle_menu(int action)

This function is used to change the settings of the different actions present in menu bar.

4.1.9 void score(int score_value)

This function is used to display score of user in display window. //Function to handle the case when the ball strikes the bricks

4.1.10 void hit()

This function is used to handle the case when the ball strikes the bricks.

4.1.11 void idle()

The idle function. Handles the motion of the ball along with rebounding from various surfaces

4.2 OpenGL In-built Functions

4.2.1 GL_COLOR_BUFFER_BIT

This function call the constant indicates the buffers currently enabled for color writing Syntax: GL_COLOR_BUFFER_BIT

Example: glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

4.2.2 glClearColor ()

This function call sets the present RGB clear color used when clearing the color buffer.

General Syntax: Void glClearColor (GLclampfr, GLclampfg, GLclampfb, GLclamppfa);

Variables of type GLclampf are floating type number between 0.0 and 1.0.

Eg: glClearColor(1.0,0.0,0.0,1.0);

4.2.3 glFlush ()

This function call forces any buffered OpenGL command to execute. It empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine.

General Syntax: Void glFlush();

4.2.4 glutInit ()

This function call initializes glut library. The arguments from main are passed in and can be used by an application.

General Syntax: Void glutInit (int*argc, char**argv);

Example: glutInit (&argc, argv);

4.2.5 glutCreateWindow ()

This function call creates a window on the display. The string title can be used to label the window.

General Syntax: Void glutCreateWindow (char*title);

Example: glutCreateWindow (“CLIPPING PLANES”);

4.2.6 glEnable ()

This function can be enable the z-buffer algorithm either in main or initialization function.

Example: glEnable (GL_DEPTH_TEST);

4.2.7 glutInitDisplayMode ()

This function call request the display with a properties that are specified in a mode. The value of a mode is determined by the logical OR operation of options including the color model.

General Syntax: Void glutInitDisplayMode (unsigned int mode);

Example: glutInitDisplayMode (GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH);

4.2.8 glutInitWindowSize ()

This function call specifies the initial height and width of window in pixels.

General syntax: Void glutInitWindowSize (int width, int height);

Example: glutInitWindowSize (500,500);

4.2.9 glutInitWindowPosition ()

This function call specifies the initial position of top left corner of the window in pixels.

General syntax: Void glutInitWindowPosition (int x, int y);

Example: glutInitWindowPosition (100,100);

4.2.10 glutDisplayFunc ()

This function call registers the display function *func i.e. executed when the window needs to be redrawn.

General syntax: Void glutDisplayFunc (void (*func)(void))

Example: glutDisplayFunc (display);

4.2.11 glutPostRedisplay ()

This function call request the display call back be executed after the current call back returns.

General syntax: Void glutPostRedisplay();

Example: glutPostRedisplay();

4.2.12 glViewPort ()

This specifies a width*height view port in pixels whose lower left corner is at (x,y) measure from the origin of the window.

General syntax: Void glViewPort(intx, inty, GLSizeiwidth, GLSizeiheight);

Example: glVeiwPort(0,0,w,h);

4.2.13 glMatrixMode ()

This function call specifies which matrix will be specified by subsequent transformation.

General syntax: Void glMatrixMode(Glenum mode);

Example: glMatrixMode(GL_PROJECTION);

4.2.14 glLoadIdentity ()

This function call sets current transformation matrix to an identity matrix.

General syntax: Void glLoadIdentity ();

Example: glLoadIdentity();

4.2.15 glPlushMatrix ()

This function call pushes to the matrix stack corresponding to the current matrix mode.

General syntax: Void glPushMatrix();

Example: glPlushMatrix;

4.2.16 glPopMatrix ()

This function call pops from the matrix stack corresponding to the current matrix mode.

General syntax: Void glPopMatrix();

Example: glPopMatrix();

4.2.17 glutLookAt ()

This function call post multiples the current matrix by a matrix determined by a viewer at the i-point looking at that with the specified up direction.

General syntax:

Void gluLookAT(GLdoubleeyex,GLdoubleeyey,GLdoubleeyez,GLdoubleatx,GLdoubleaty, GLdoubleatz,GLdoubleupx,GLdoubleupy,GLdoubleupz);

Example: gluLookAt (0.0,0.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);

4.2.18 gluPerspective ()

This function call defines a perspective viewing volume using the y-direction field of view for measured in degrees, the aspect ratio of the front clipping plane and the near and far distances.

General syntax: Void gluPerspective(GLdoublefar,GLdoublespect,GLdoublenear,GLdouble far);

Example: gluPespective(45.0,(GLfloat)w/(GLfloat)h,1.0,300.0);

4.2.19 glutMouseFunc ()

The function call registers the mouse call back function *f. The call back function returns the function (GL_LEFT_BUTTON,GLUT_MIDDLE_BUTTON,GLUT_RIGHT_ BUTTON), the state of the button after the event will be either up or down(GLUT_UP,GLUT_DOWN) and position of the mouse relative to top left corner of window.

General syntax: Void glutMouseFunc(void*(intbutton,intstate,int x,int y);

Eg : glutMouseFunc(mousefunc);

4.2.20 glutReshapeFunc ()

This function call registers the reshape call back function f. The call back function returns the height and width of the new window. The Reshape call back invokes a display call back.

General syntax: Void glutReshapeFunc(void*f(intwidth,int height));

Eg: glutReshapeFunc(myreshape);

4.2.21 glutSwapBuffers ()

This function call swaps the front and back buffers.

General syntax: Void glutSwapBuffers();

4.2.22 glRotate ()

This function call alter the current matrix by a rotation of angle degrees about the axis (dx,dy,dz). Type can be either GLfloat and GLdouble.

General Syntax: Void glRotate[fd](TYPE angle, TYPEdx,TYPEdy,TYPEdz);

Eg: glRotate(0.0,1.0,0.0);

4.2.23 glTranslate ()

This function call, alerts the current matrix by a displacement of (x,y,z).

General Syntax: Void glTranslate[fd](TYPEx, TYPEy, TYPEz);

Example: glTranslate (0.0, 1.0, 0.0);

4.2.24 glClear ()

This function call used to clear the window. As the algorithm stores information in the depth buffer, it is necessary to clear the buffer whenever we wish to redraw the display.

Eg: glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

4.2.25 glColor3f ()

This function will set the current drawing color. This function given as the 3 stands means we need to send three values to this function, they are Red, Green and Blue respectively. The “f” means it needs the floats, for example 0, 34332421.0 is full intensity so glcolor3f (1.0, 1.0, 1.0) would produce full red, green and blue

Example; glColor (1.0,0.0,0.0);

4.2.26 glutMainLoop ()

This function call causes the program to enter an event processing loop. It should be the last statement in main.

General syntax: Void glutMainLoop();

Example: glutMainLoop();

4.3 Standard header and library functions

4.3.1 Stdio.h

This header file is used for standard input and output and stream manipulation function.

4.3.2 Stdlib.h

Stdlib.h is the header of the general purpose standard library of C programming language which include functions involving memory allocation, process control, conversions and others.

4.3.3 GL/glut.h

Library utility in the OpenGL program.

4.3.4 math.h

The math.h header defines various mathematical functions to perform various mathematical operations.

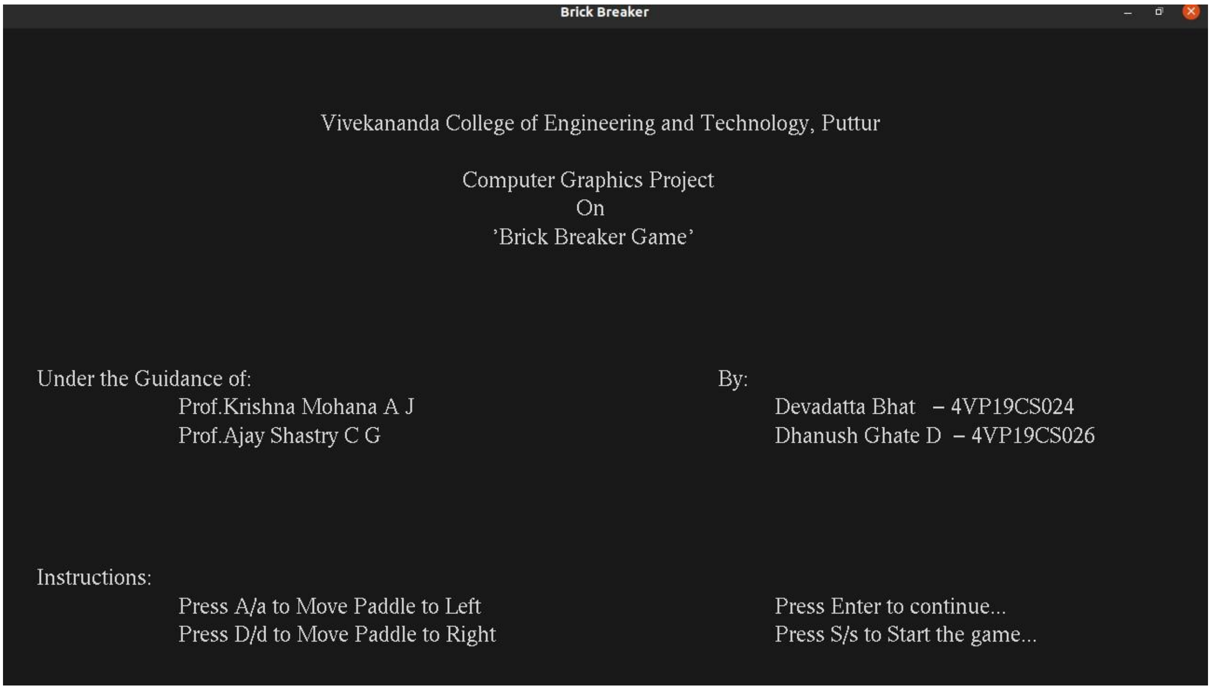
4.3.5 string.h

This header file declares a set of functions to work strings.

CHAPTER 5

SNAPSHOTS

The below figure shows the front page of the project containing instructions to play the game



Vivekananda College of Engineering and Technology, Puttur

Computer Graphics Project
On
'Brick Breaker Game'

Under the Guidance of:
Prof.Krishna Mohana A J
Prof.Ajay Shastry C G

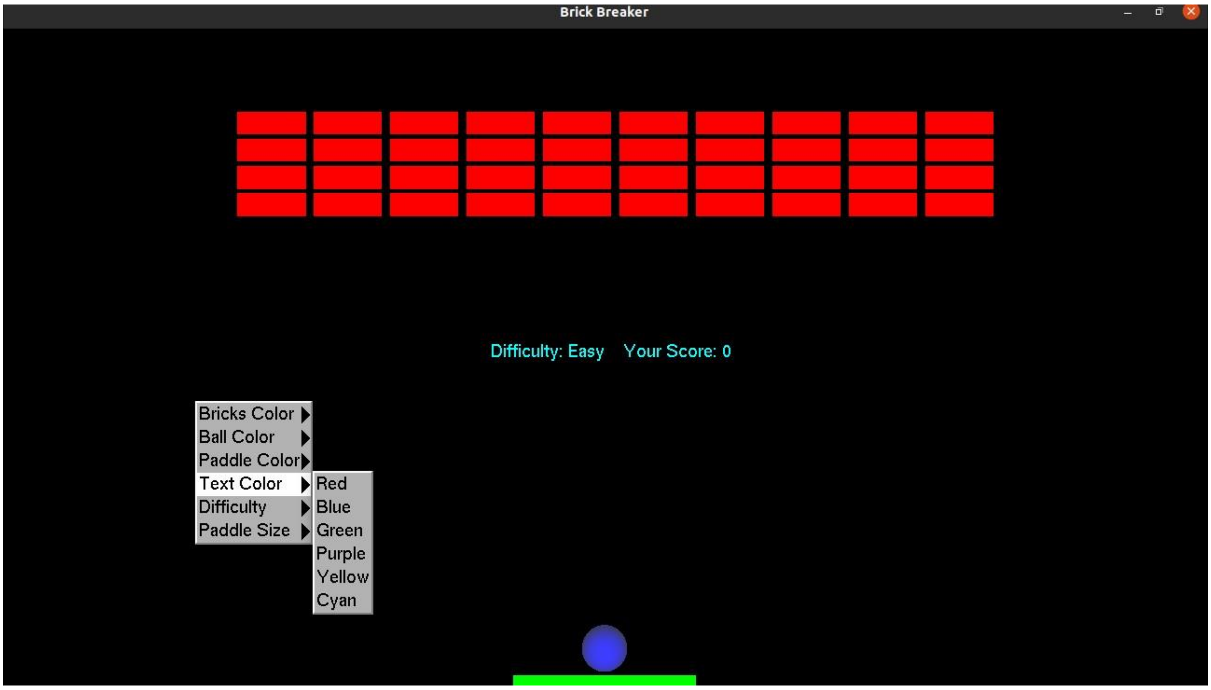
By:
Devadatta Bhat – 4VP19CS024
Dhanush Ghate D – 4VP19CS026

Instructions:
Press A/a to Move Paddle to Left
Press D/d to Move Paddle to Right

Press Enter to continue...
Press S/s to Start the game...

Fig 5.1 Project Front page

The below image contains the gameplay page of the project along with drop down list for changing game environment



Difficulty: Easy Your Score: 0

Fig 5.2 Project Gameplay page

The below figure shows the gameplay in action

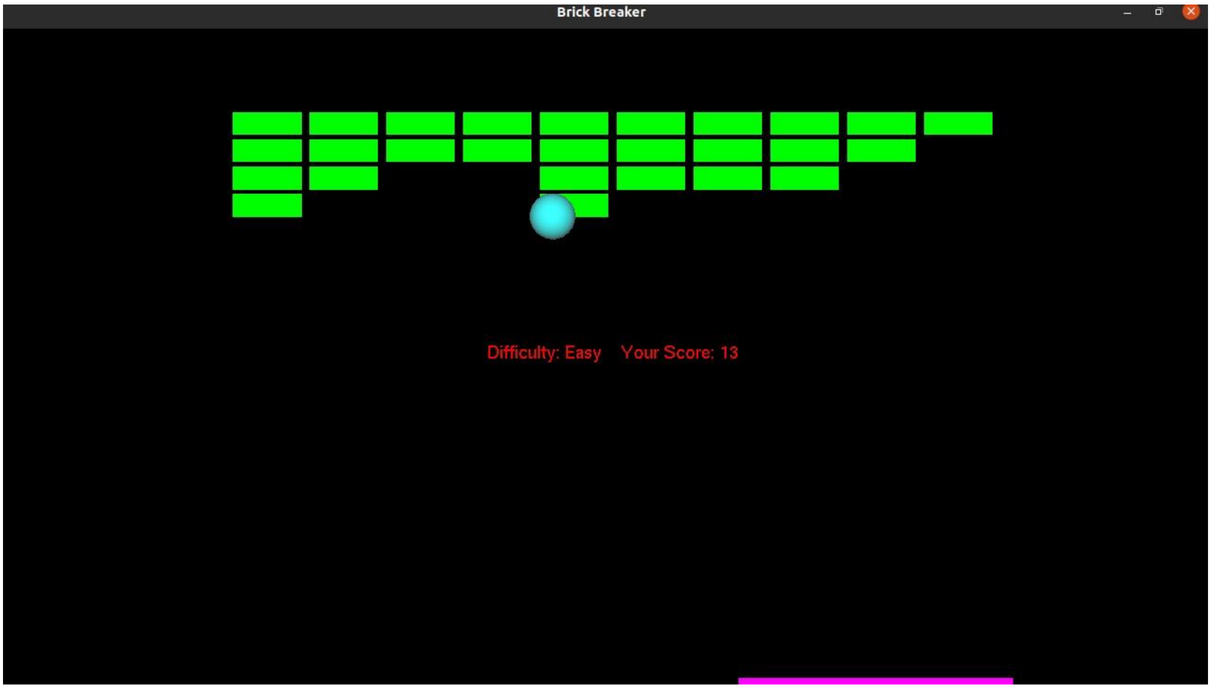


Fig 5.3 Gameplay in action

The below figure shows the congratulatory message after completing the game

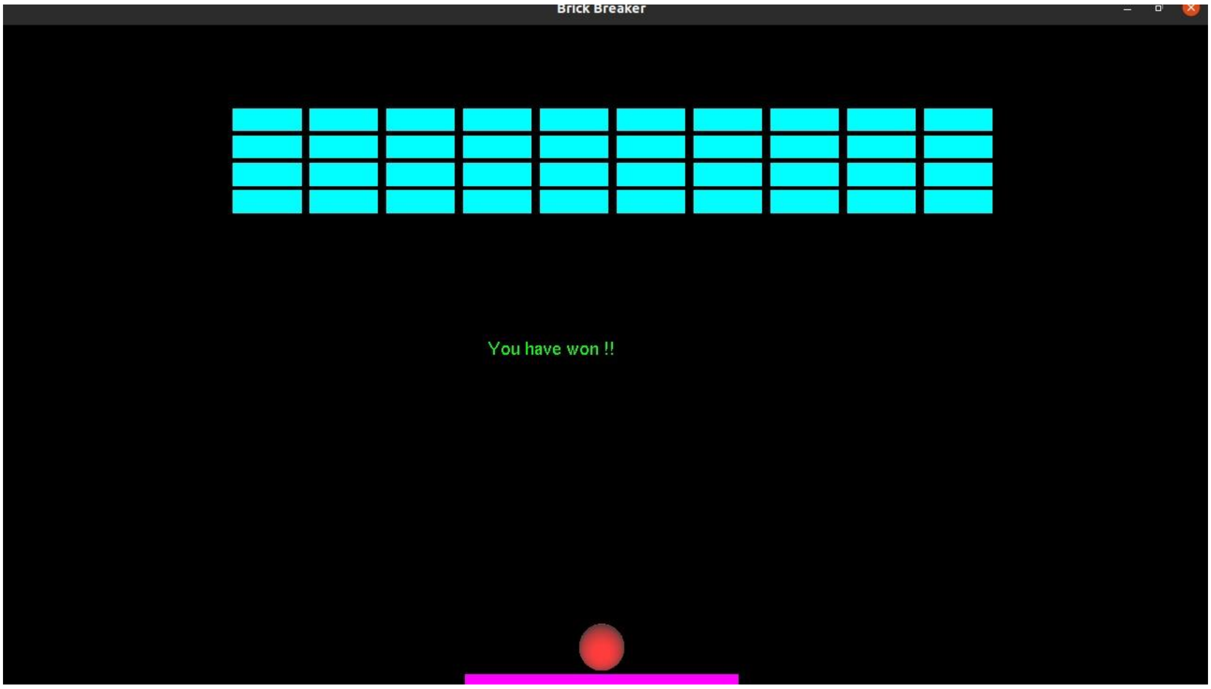


Fig 5.4 "You have won!!" Message

CHAPTER 6

CONCLUSION

We had a wonderful experience developing this project. By working on this project, we gained hands-on experience of using OpenGL software using which we experimented this project. We have made use of hardware devices such as mouse and keyboard driven interface. Thus, to reduce the complexity and make it user friendly.

The project helped in understanding the working of computer graphics using OpenGL and various concepts, functions and methodologies for the development of a graphics packages. We hope our project will serve its purpose without any hassles.

REFERENCES

[1] Computer Graphics Using OpenGL – F.S. Hill Jr. 2nd Edition, Pearson Education (2001)

[2] Computer Graphics – James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes Edition – Wesley (1997)

[3] Computer Graphics – OpenGL Version – Donald Hearn and Pauline Baker, 2nd Edition, Pearson Education (2003)

[4] https://learnopengl.com/

[5] https://open.gl/

[6] https://stackoverflow.com/questions/7968748/understanding-opengl