1. INTRODUCTION

In the project " **2D GAME**", the project is programmed using c/c++ . Oops concepts are explored and the project involves the Bat and Ball on which basis game is built.In this game there will be two players . PC will be the First player and USER will be the second player. This program contains Functions to drag the ball, draw the bat etc.

This project includes the multiple windows, menus and submenus using which color of the bat & ball, screen color, ball size will be changed. These all actions are assigned to keyboard and mouse.

User-interface is provided by means of both Keyboard and Mouse. By using arrow keys bat can be moved. Mouse interaction is achieved by means of a menu which is operational only with the "right mouse button" through which bat,ball,screen color changing,speed settings are enabled.

1.1 Computer Graphics:

Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen.

Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D patter-recognition abilities allow us to perceive and process data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

1.2 OpenGL Interface:

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. **Main GL:** Library has names that begin with the letter gl and are stored in a library usually referred to as GL.
2. **OpenGL Utility Library (GLU):** This library uses only GL functions but contains code for creating common objects and simplifying viewing.
3. **OpenGL Utility Toolkit (GLUT):** This provides the minimum functionality that should be accepted in any modern windowing system.

1.3 OpenGL Overview:

- OpenGL(Open Graphics Library) is the interface between a graphic program and graphics hardware. ***It is streamlined.*** In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.

- OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.
- ***It is system-independent.*** It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
- ***It is a state machine.*** At any moment during the execution of a program there is a current model transformation
- ***It is a rendering pipeline.*** The rendering pipeline consists of the following steps:
    - Defines objects mathematically.
    - Arranges objects in space relative to a viewpoint.
    - Calculates the color of the objects.
    - Rasterizes the objects.

2. SYSTEM SPECIFICATION

2.1 HARDWARE REQUIREMENTS:

- Dual Core Processor
- 2GB RAM
- 40GB Hard disk
- DVD drive
- Mouse and other pointing devices
- Keyboard

2.2 SOFTWARE REQUIREMENTS:

- Programming language – C/C++ using OpenGL
- Operating system – Linux operating system
- Compiler – C Compiler
- Graphics library – GL/glut.h
- OpenGL 2.0

2.3 FUNCTIONAL REQUIREMENTS:

OpenGL APIs:

If we want to have a control on the flow of program and if we want to interact with the window system then we use OpenGL API'S. Vertices are represented in the same manner internally, whether they are specified as two-dimensional or three-dimensional entities, everything that we do are here will be equally valid in three dimensions. Although OpenGL is easy to learn, compared with other APIs, it is nevertheless powerful. It supports the simple three dimensional programs and also supports the advanced rendering techniques.

GL/glut.h:

We use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

The application program uses only GLUT functions and can be recompiled with the GLUT library for other window system. OpenGL makes a heavy use of macros to increase code readability and avoid the use of magic numbers. In most implementation, one of the include lines

3. ABOUT THE PROJECT

3.1 Overview:

Our game is a simple ball with bat game. The bat will be moved according to the movement of the mouse. And the ball will move randomly in the created window. When the ball hits the right, left, or top wall – we will refer to the window border as a wall - it will return back. When it hits the bottom wall it will not only return back but it will increase the score of the computer, but if the player can hold it by the bat, his score will be increased.

3.2 User Interface:

The interface is mainly concentrated on use of mouse and keyboard. Clicking right button of mouse displays a menu which has various options which helps in changing color of bat, ball and background..

By pressing P and R we can pause and restart the game. By pressing N we can move from the current window to next window.

3.3 Purpose:

The aim of this project is to develop a graphics package which supports basic operations which include building a 2D GAME using Open GL. The package must also has a user-friendly interface. The objective of developing this model was to design and apply the skills we learnt in class.

3.4 Scope:

It provides most of the features that a graphics model should have. It is developed in C language. It has been implemented on LINUX platform. The graphics package designed here provides an interface for the users for playing 2D GAME using bat and ball.

4. IMPLEMENTATION

4.1 FUNCTIONS IN OPEN GL

- **void glClear(glEnum mode);**

Clears the buffers namely color buffer and depth buffer. mode refers to GL_COLOR_BUFFER_BIT or DEPTH_BUFFER_BIT.

- **void glTranslate[fd](TYPE x, TYPE y, TYPE z);**

Alters the current matrix by displacement of (x, y, z), TYPE is either GLfloat or GLdouble.

- **void glutSwapBuffers();**

Swaps the front and back buffers.

- **void glMatrixMode(GLenum mode);**

Specifies which matrix will be affected by subsequent transformations, Mode can be GL_MODELVIEW or GL_PROJECTION.

- **void glLoadIdentity( );**

Sets the current transformation matrix to identity matrix.

- **void glEnable(GLenum feature);**

Enables an OpenGL feature. Feature can be GL_DEPTH_TEST (enables depth test for hidden surface removal), GL_LIGHTING (enables for lighting calculations), GL_LIGHTi (used to turn on the light for number i), etc.

- **void glPushMatrix();**
  void glPopMatrix();

Pushes to and pops from the matrix stack corresponding to the current matrix mode.

- **void glutBitmapCharacter(void *font, int character);**

Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font. The fonts used are GLUT_BITMAP_TIMES_ ROMAN_24, GLUT_BITMAP_ HELVETICA_18.

- **void glRasterPos[234][ifd](GLfloat x, GLfloat y, GLfloat z);**

This position, called the raster position, is used to position pixel and bitmap write operations.

- **void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);**

Defines an orthographic viewing volume with all the parameters measured from the centre of the projection plane.

- **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);**

Requests a display with the properties in the mode; the value of mode is determined by the logical OR of options including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE).

- **void glutCreateWindow(char *title);**

Creates a window on display; the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- **void glutMainLoop();**

Causes the program to enter an event-processing loop.

- **void glutDisplayFunc(void (*func)(void))**

Registers the display function func that is executed when the window needs to be redrawn.

- **void glutMouseFunc(void *f(int button, int state, int x, int y)**

Registers the mouse callback function f. The callback function returns the button (GLUT_LEFT_BUTTON,etc., the state of the button after the event (GLUT_DOWN), and the position of the mouse relative to the top-left corner of the window.

- **void glutKeyboardFunc(void *f(char key, 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 glClearColor(GLclampf r, GLclampf g, GLclamp b, Glclamp a)**

Sets the present RGBA clear color used when clearing the color buffer. Variables of type GLclampf are floating point numbers between 0.0 and 1.0.

- **void glViewport(int x ,int y, GLsizei width, GLsizei height)**

Specifies the width*height viewport in pixels whose lower left corner is at (x,y) measured from the origin of the window.

- **void glColor3[b I f d ub us ui](TYPE r, TYPE g, TYPE b)**

Sets the present RGB colors. Valid types are bytes(b), int(i), float(f), double(d), unsigned byte(ub), unsigned short(us), and unsigned int(ui). The maximum and minimum value for floating point types are 1.0 and 0.0 respectively, whereas the maximum and minimum values of the discrete types are those of the type, for eg, 255 and 0 for unsigned bytes.

- **void glutInitWindowSize(int width, int height);**

Specifies the initial height and width of the window in pixels.

- **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 the display callback.

- **void createMenu(void);**

This function is used to create menus which are used as options in program.

4.2 USER DEFINED FUNCTIONS:

- **void reshape ( ) function:**

It is used for defining the viewport volume using glViewport(int x ,int y,GLsizei width,Glsizei height).

- **int main ( ) function:**

The main function is used for creating the window for display of the model of the atom. Here, we create menu for ease of use for the user. The callback functions, i.e., mouse callback, keyboard callback, display callback, idle callback, are written in main. The callback functions registered in main ( ) are,

```
glutKeyboardFunc (mykeys);
glutDisplayFunc (display);
glutReshapeFunc (reshape);
glutMouseFunc (mouse);
```

- **void incsize(RECTA rect)**

This function is used to increase the size of the ball. It is done in game using arrow keys.

- **void decsize(RECTA rect)**

This function is used to decrease the size of the ball. It is done by using arrow keys.

- **void DrawBall(RECTA rect,double r,double g,double b)**

It is used to draw ball on the screen of game. RECTA struct is used to draw ball in rectangle shape.

- **void DrawBat(RECTA rect,double r,double g,double b)**

This function is used for drawing bat for playing game.Here we make use of user defined struct RECTA.

- **void Timer(int v)**

Using this function in the program the ball movements towards left, right, top, bottom is achieved.

- **void drawText(char* string,int x, int y)**

This is used to draw text used in frontscreen, startscreen etc which contains the information about projects team members, rules of game, instructions to games etc.

- **int Test_Ball_Wall(RECTA ball , RECTA wall)**

To find out the collision between wall and ball and to assign the scores to
PC and player this function is used.

- **bool Test_Ball_Player(RECTA ball,RECTA player)**

According to the movements of bat,ball and collision of ball with wall and bat the scores are calculated in this function.

- **void MouseMotion(int x,int y)**

This function is used to move the bat on x-axis with the help of mouse.

5. TESTING

Testing process started with the testing of individual program units such as functions or objects. These were then integrated into sub-systems and systems, and interactions of these units were tested.

Testing involves verification and validation.

Validation: “Are we building right product?”

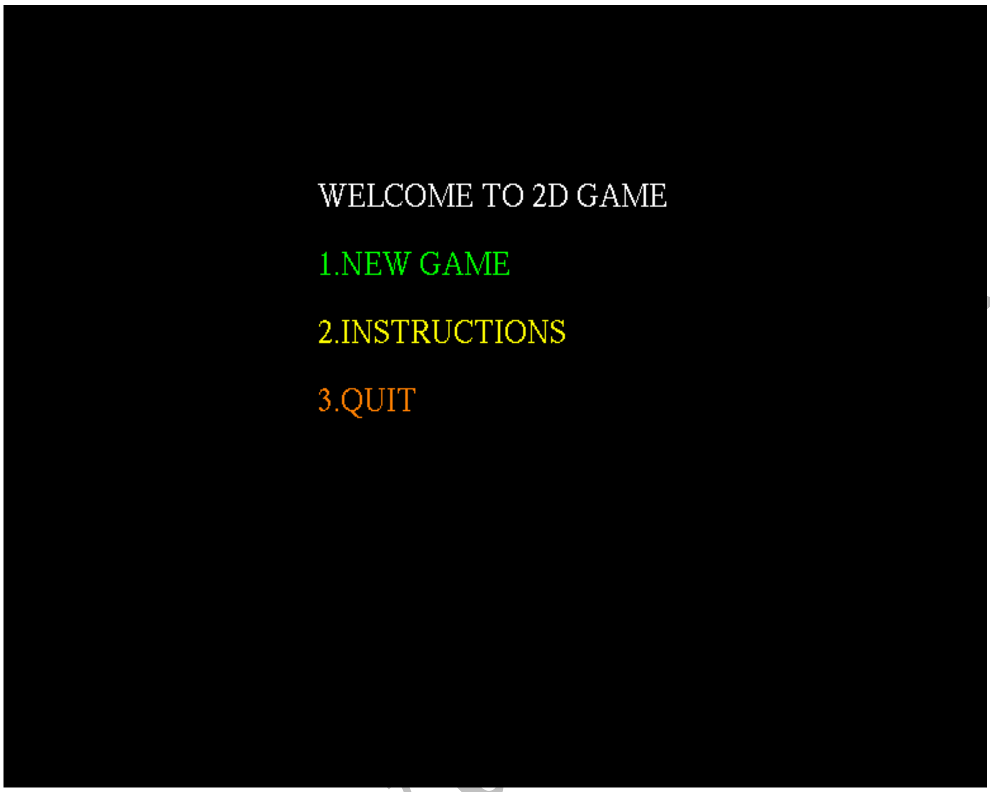
Verification: “Are we building the product right?”

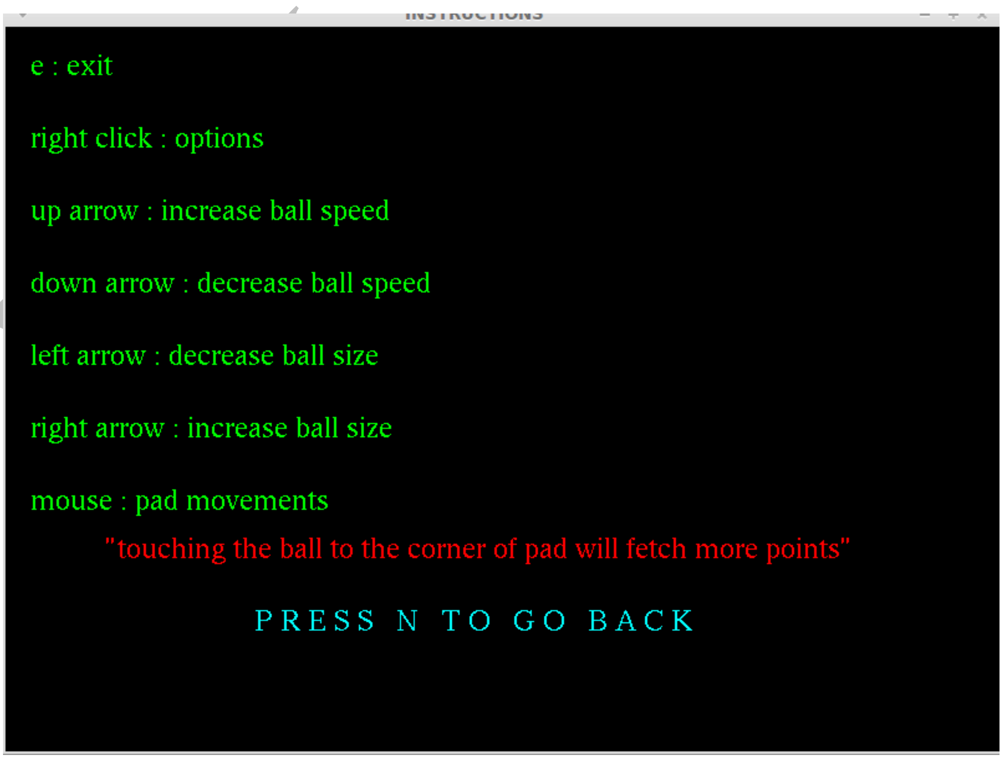
The ultimate goal of the verification and validation process is to establish confidence that the software system is ‘fit for purpose’. The level of required confidence depends on the system’s purpose, the expectations of the system users and the current marketing environment for the system.

With the verification and validation process, there are two complementary approaches to the system checking and analysis:

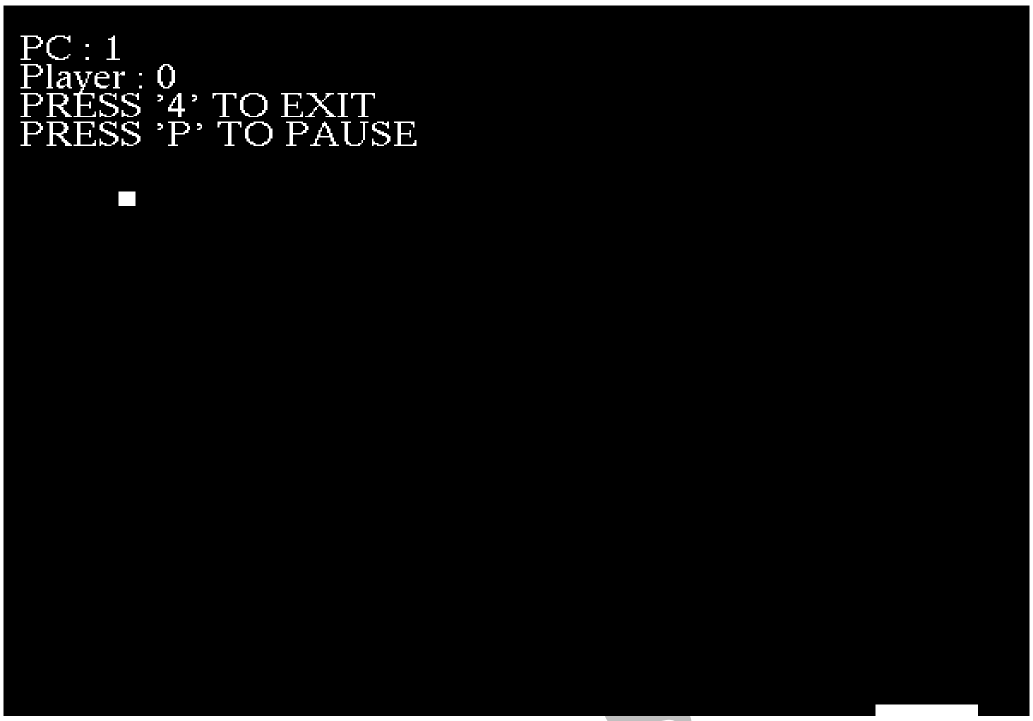
Software inspections or peer reviews analyses and check system representations such as the requirements document, design diagrams, and the program source code. Software testing involves running an implementation of the software with test data.

6. SNAPSHOTS

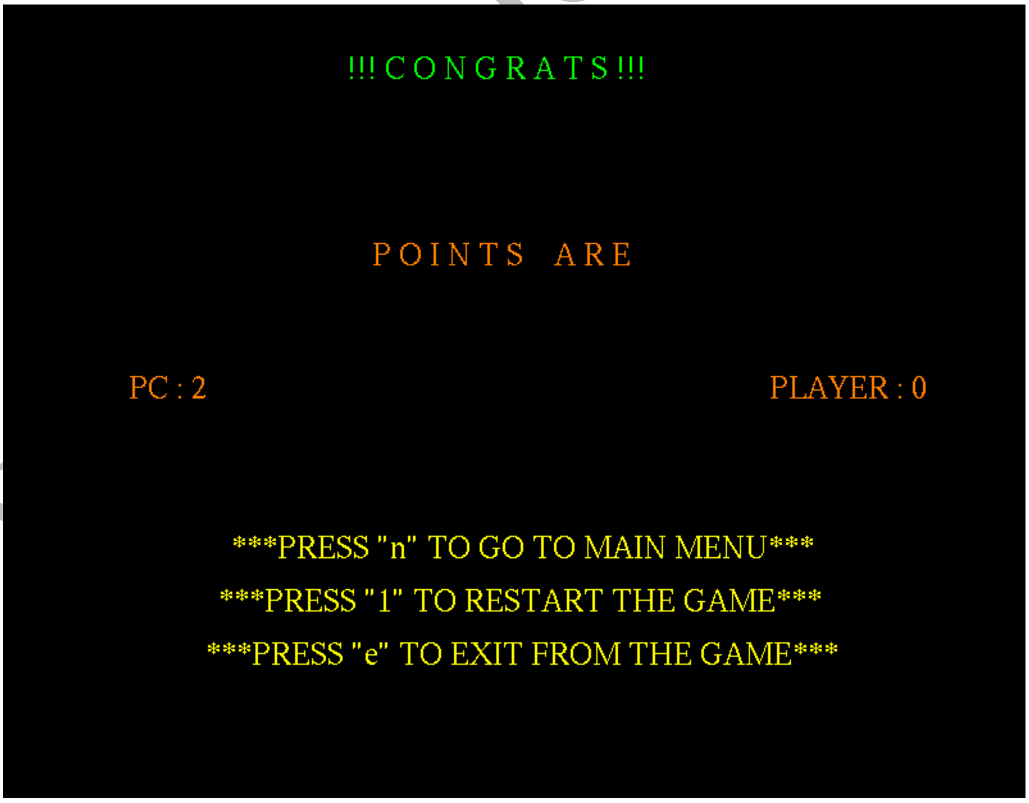




```
e : exit

right click : options

up arrow : increase ball speed

down arrow : decrease ball speed

left arrow : decrease ball size

right arrow : increase ball size

mouse : pad movements
        "touching the ball to the corner of pad will fetch more points"

                    P R E S S   N   T O   G O   B A C K
```



```
PC : 1
Player : 0
PRESS '4' TO EXIT
PRESS 'P' TO PAUSE
```



```
!!! C O N G R A T S !!!

P O I N T S   A R E

PC : 2                                  PLAYER : 0

***PRESS "n"  TO GO TO MAIN MENU***
***PRESS "1"  TO RESTART THE GAME***
***PRESS "e"  TO EXIT FROM THE GAME***
```

7. CONCLUSION AND ENHACEMENTS

The project was 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 forced with the tradeoff 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. It has an open source and no security features has been included. The user is free to alter the code for feature enhancement. Checking and verification of all possible types of the functions are taken care. Care was taken to avoid bugs. Bugs may be reported to creator as the need.

Further this project can be enhanced by adding few more options i,e menus in game. Using this we can design a 3D game which contains cube instead of single window and multiple number of balls which are randomly moving and all faces of cube is considered as wall.

8. BIBLIOGRAPHY

Reference Books and E-books

1. **EDWARD ANGEL**

Interactive Computer Graphics, 5th edition, universities of New Mexico.

2. **Websites for Reference**

- www.opengl.org/resources/code/samples/redbook.
- www.wikipedia.org