Chapter 1

Introduction

1.1 Computer Graphics

Computer Graphics are pictures and films created using computers. Usually, the term refers to computer-generated image data created with help from specialized graphical hardware and software. It is a vast and recent area in computer science. The phrase was coined in 1960, by computer graphics researchers Verne Hudson and William Fetter of Boeing. It is often abbreviated as CG, though sometimes erroneously referred to as CGI.

Important topics in computer graphics include user interface design, sprite graphics, vector graphics, 3D modeling, shaders, GPU design, implicit surface visualization with ray tracing, and computer vision, among others. The overall methodology depends heavily on the underlying sciences of geometry, optics, and physics.

Computer graphics is responsible for displaying art and image data effectively and meaningfully to the user. It is also used for processing image data received from the physical world. Computer graphic development has had a significant impact on many types of media and has revolutionized animation, movies, advertising, video games, and graphic design generally.

1.2 OpenGL

Most Widely Adopted Graphics Standard

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms. OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

High Visual Quality and Performance

Any visual computing application requiring maximum performance-from 3D animation to CAD to visual simulation-can exploit high-quality, high-performance OpenGL capabilities. These capabilities allow developers in diverse markets such as broadcasting, CAD/CAM/CAE, entertainment, medical imaging, and virtual reality to produce and display incredibly compelling 2D and 3D graphics.

Developer-Driven Advantages

- **Industry standard**

An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard.

- **Stable**

OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt changes. Backward compatibility requirements ensure that existing applications do not become obsolete.

- **Reliable and portable**

All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.

- **Evolving**

Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

- **Scalable**

OpenGL API-based applications can run on systems ranging from consumer electronics to PCs, workstations, and supercomputers. As a result, applications can scale to any class of machine that the developer chooses to target.

- **Easy to use**

OpenGL is well structured with an intuitive design and logical commands. Efficient OpenGL routines typically result in applications with fewer lines of code than those that make up programs generated using other graphics libraries or packages.

Chapter 2

System Specification

2.1 Software Requirements

- Visual studio VC++ 2010 or Visual Studio VC++ 2017(express editors or higher versions)
- OpenGL package: glut-3.7.6-bin(this consists of glut.h,glut32.lib,glut32.dll)
- .Net framework 2.0(for visual studio 2010)
- .Net framework 3.5(for visual studio 2017)

2.2 Hardware Requirements

- 512MB RAM and other as required for Visual Studio.
- 40GB Hard Disk.
- VGA graphics card
- Pentium 4 Processor
- The minimum hardware requirements for .Net 3.0

Chapter 3

Analysis

Purpose

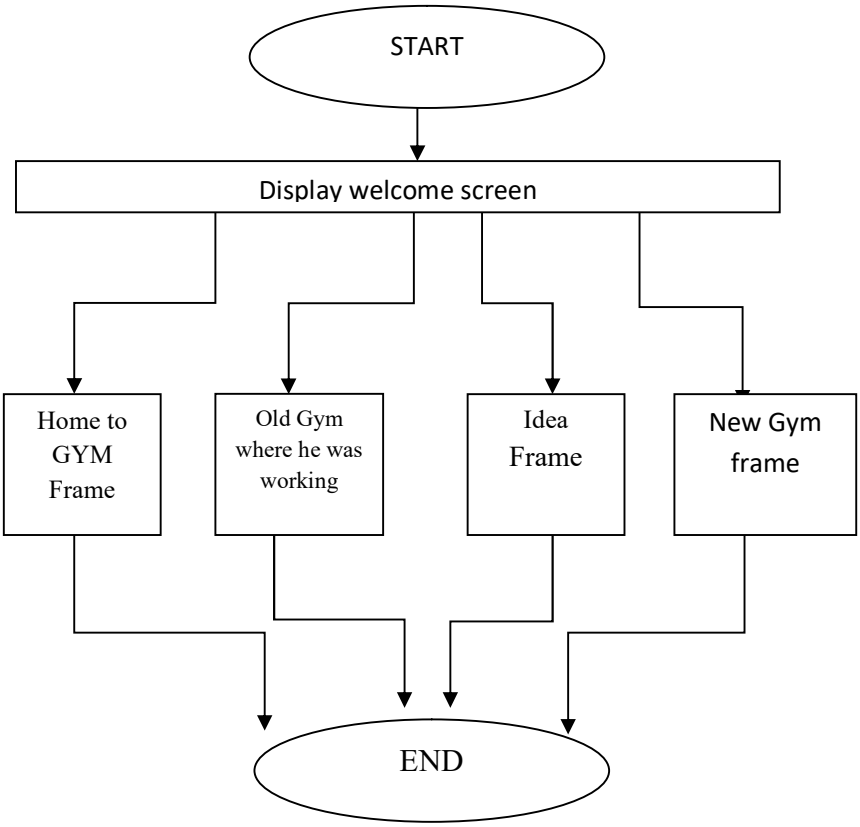
- This project was designed to illustrate and represent the Gym Life of an individual.
- This project also helped in providing a deeper understanding of the OpenGL programming language as well as event based programming which involves defining logic based on events.
- First the information to be portrayed was defined and the objects required to visually represent the scenario were developed.
- The required methods and properties on these objects were defined and developed. These methods are functions which define the object movement.
- The required processing and memory to define these scenarios were defined. Finally the objects were added into one single environment and synchronized to depict movement in the scenario.

Chapter 4

Design

4.1 Flow Diagram

Flow diagram is the visual representation of algorithms with the help of pre defined elements which represents the state of the algorithm at any given point during execution.



4.2 Description of Flow Diagram

The flow of this diagram is that 1st it displays the welcome screen. ('n' key represents the next frame). Then it goes to home page where the guy will be going from his house to Gym. Then in Gym keeps working out and gets bored of this gym because their won't be any entertainment or any instructors to guide him. So, while he was thinking he gets ab idea about his friend who was working in a different where they provide many facilities. Then the next day he goes to that gym and joins there. Later on, he feels very much happy for joining this gym and his life went on smoothly.

Chapter 5

Implementation

5.1 Built in Functions:

Since OpenGl is an API developed to be cross platform and supports graphic based programming, it relieves the programmer from determining self-specific content and helps him focus on only the logic.

The built in functions that were used in the project were

- **void glMatrixMode(GLenum,model) ;**
  Specifies whether the model view or projection matrix will be modified using the argument GL_MODELVIEW,GL_PROJECTIONS,GL_TEXTURE for mode.

- **void glClear(GLbitfield mask) ;**
  Clears the specified the buffers to their current clearing values
  .
- **void glFlush(void);**
  Forces previously issued OpenGL commands to begin execution, thus guaranteeing that they complete in finite time.

- **void glutMainLoop(void);**
  Enters the glut processing loop, never to return. Registered callback functions will be called.

- **void glColor3f(GLfloat red,GLfloat green,GLfloat blue);**
  Specifies the color to be stored in the frame buffer.

  void glBegin(GLenum mode);
  This mode specifies the geometric type that we want our vertices to define.

- **void glutInit(int* argc,char** argv[]);**
  Allows the user to pass command line arguments which provide the interaction between the windowing system and OpenGL.

- **void glutInitDisplayMode(unsigned int mode);**
  Specifies the display mode for windows created when glutCreateWindow() is called.

- **void glutInitWindowSize(int width,int height);**
  Specifies the width and height, in pixels, of our window.

- **void glutInitWindowPosition(int x,int y);**
  Specifies the initial x and y location for the upper left corner of the window in relation to the upper left corner of the monitor's screen.

- **int glutCreateWindow(char* name);**
  Creates a window with an OpenGl context using the previously set characteristics. The string name appears in the title bar.

- **void glViewPort(Glint x,Glint y,GLsizei width,GLsizei height);**
  Defines a pixel rectangle in the window into which the final image is mapped.

- **void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);**
  Sets the current clearing color for use in clearing color buffers in RGBA mode.

- **void glBegin(GLenum mode);**
  Marks the beginning of a vertex data list that describes a geometric primitive.

  GL_POINTS: Individual points.
  GL_LINES: Pairs of vertices interpreted as individual line segments.
  GL_LINE_LOOP: Series of connected line segments with a segment added between last and first vertices.
  GL_POLYGON: Boundary of a simple convex polygon.

- **void glEnd(void);**
  Marks the end of a vertex data list.

- **void glPointSize(GLfloat size);**
  Set the width in pixels for rendered points.

- **void glLineWidth(GLfloat width);**
  Sets the width in pixels for rendered lines.

- **void glLoadIdentity(void);**
  Sets the currently modifiable matrix to the 4x4 identity matrix.

- **void glTranslate(type x,type y,type z);**
  Multiplies the current matrix by a matrix that moves an object by the given x,y and z values.

- **void glRotate(type angle,type x,type y,type z);**
  Multiplies the current matrix by a matrix that rotates an object in a counter clockwise direction about the ray from the origin through the point (x,y,z). The angle parameter specifies the angle of rotation in degrees.

- **void glOrtho2D(GLdouble left,GLdouble right,GLdouble bottom,GLdouble top);**
  Creates a matrix for projecting two dimensional coordinates onto the screen and multiplies the current project matrix by it.

- **void glutKeyboardFunction(void(*func)(unsigned char key,int x,int y));**
  Specifies the function func that is called when a key that generates an ASCII character is pressed. The key callback parameter is the generated ASCII character. The x and y callback parameters indicate the location of the mouse pointer when the key was pressed.

5.2 User Defined Functions:

These are the user defined functions that have been used to put together the main idea of the project.

```
void introduction()
{
	// Background
	glClearColor(0.05, 0.05, 0.05, 1.0);
	glClear(GL_COLOR_BUFFER_BIT);

	// Story Title
	drawText1("GYM ADVERTISING",
		1, 1, 1, title_fade, 400, 480, .3, .3, 2);

	drawText1("by Jyothishree S",
		1, 1, 1, title_fade, 650, 400, .15, .15, 1);
	drawText1("(1DS17CS410)",
		1, 1, 1, title_fade, 700, 370, .1, .1, 1);
	drawText1("&",
```

```
                1, 1, 1, title_fade, 750, 340, .1, .1, 1);
        drawText1("Sai Kumar K G",
                1, 1, 1, title_fade, 700, 310, .15, .15, 1);
        drawText1("(1DS17CS426)",
                1, 1, 1, title_fade, 700, 280, .1, .1, 1);

        drawText1("Press N to read the story...",
                1, 1, 1, intro_next_text_appear, 560, 10, .09, .09, 1);
}

void LifestyleTitleScreen() {
        // Background
        glClearColor(0.05, 0.05, 0.05, 1.0);
        glClear(GL_COLOR_BUFFER_BIT);

        // Story Title
        drawText1("CHAPTER 1",
                1, 1, 1, gy_chap_fade, 400 + trans_x_chap1, 500, .2, .2, 2);
        drawText1("MOTIVATED",
                1, 1, 1, gy_title_fade, 450 + trans_x_title1, 430, .3, .3, 2);

}

void drawSchoolBoy(GLfloat tx, GLfloat ty,
        GLfloat sx, GLfloat sy,
        GLfloat shirt_r, GLfloat shirt_g, GLfloat shirt_b) {
        glPushMatrix();

        glScalef(sx, sy, 0);
        glTranslatef(tx, ty, 0);

        // Shoes
        drawSemiCircle(1160, 120,
                1, 1,
                255, 255, 255,
                10,
                -90, 91);

        // Shoe Extension
        glBegin(GL_POLYGON);
        glColor3ub(255, 255, 255);
        glVertex2f(1160, 120);
        glVertex2f(1170, 120);
        glVertex2f(1170, 135);
        glVertex2f(1160, 135);
        glEnd();

        // Pants
        glBegin(GL_POLYGON);
        glColor3ub(0, 0, 200);
        glVertex2f(1158, 135);
        glVertex2f(1172, 135);
        glVertex2f(1170, 190);
        glVertex2f(1160, 190);
        glEnd();

        // Shirt
        glBegin(GL_POLYGON);
```

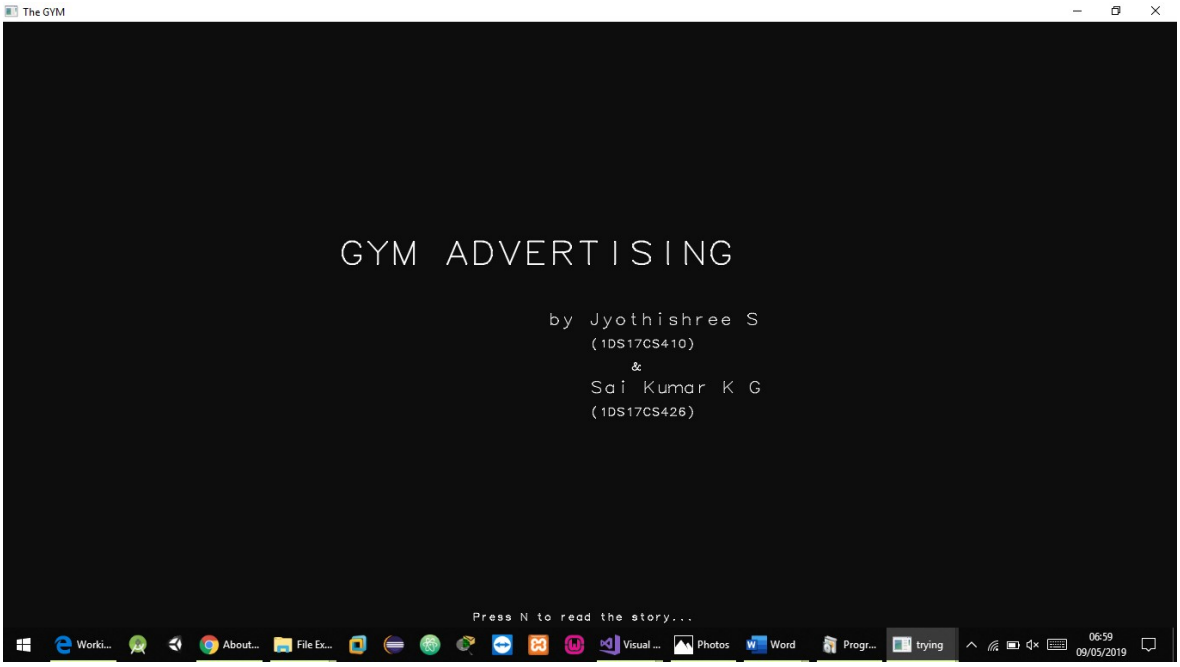
```
        glColor3ub(shirt_r, shirt_g, shirt_b);
        glVertex2f(1155, 190);
        glVertex2f(1170, 190);
        glVertex2f(1170, 260);
        glVertex2f(1160, 260);
        glEnd();

        // Bag
        glBegin(GL_POLYGON);
        glColor3ub(156, 86, 47);
        glVertex2f(1170, 250);
        glVertex2f(1180, 245);
        glVertex2f(1185, 200);
        glVertex2f(1170, 195);
        glEnd();

        // Head
        drawCircle(1164, 273,
               232, 190, 123,
               1, 1.4,
               12);

        // Hair
        drawSemiCircle(1167, 277,
               1, 1,
               0, 0, 0,
               14,
               -80, 180);

        // Nose
        glBegin(GL_TRIANGLES);
        glColor3ub(232, 190, 123);
        glVertex2f(1155, 270);
        glVertex2f(1152, 260);
        glVertex2f(1157, 262);
        glEnd();

        // Eye
        glPointSize(2);
        glBegin(GL_POINTS);
        glColor3ub(0, 0, 0);
        glVertex2f(1156, 270);
        glEnd();
        glPointSize(1);

        // Lips
        glPointSize(1);
        glBegin(GL_POINTS);
        glColor3ub(0, 0, 0);
        glVertex2f(1158, 256);
        glVertex2f(1159, 257);
        glVertex2f(1160, 258);
        glEnd();
        glPointSize(1);

        glPopMatrix();
}
```

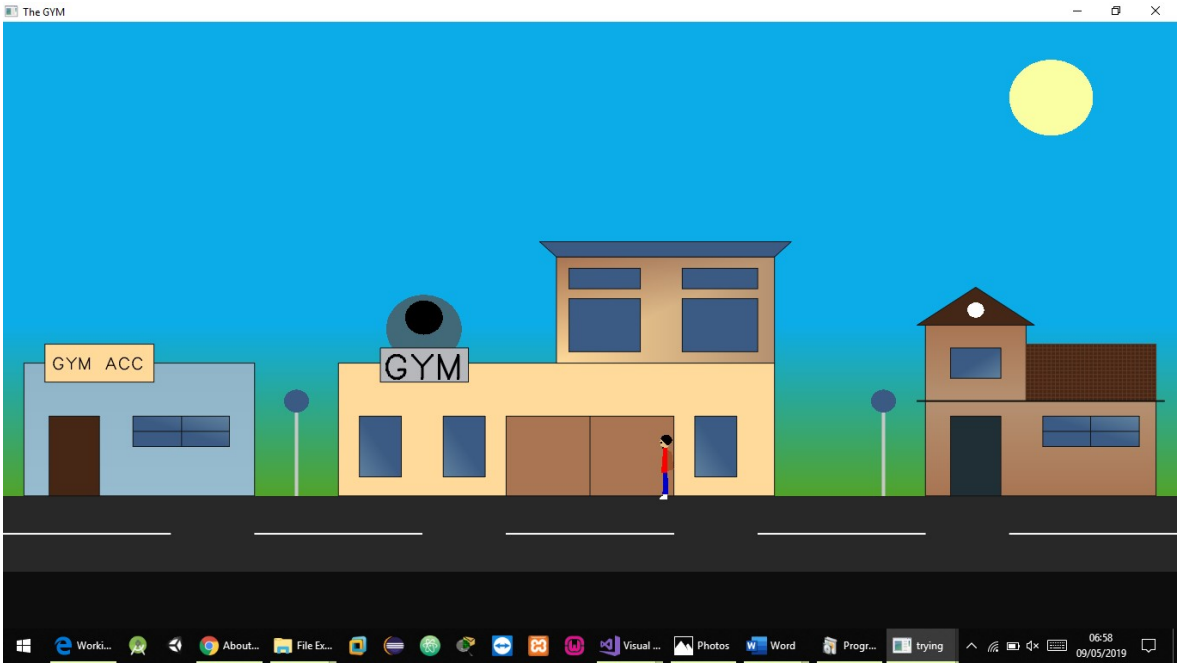
```
void renderScene() {
	// Switch to know which scene is playing
	switch (SCENE_ID) {
	case 0:
		introduction();
		break;
	case 1:
		LifestyleTitleScreen();
		break;
	case 2:
		highSchool();
		break;
	case 3:
		GYMENV();
		break;
	case 4:
		gymenvironment1();
		break;
	case 5:
		titlethree();
		break;

	case 6:
		gym();
		break;

	case 7:
		titlefour();
		break;

	case 8:
		gymenvironment();
		break;

	default:
		break;
	}
	glFlush();
}


void mouseClick(int button, int state, int x, int y) {
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
		std::cout << x << "\t" << 800 - y << "\n";
}

void keyPress(unsigned char key, int x, int y)
{
	switch (key) {
		// Go to Previous Scene
	case 'b':
	case 'B':
		if (SCENE_ID == 0)
			break;
		SCENE_ID--;
		break;
		// Go to Next Scene
```

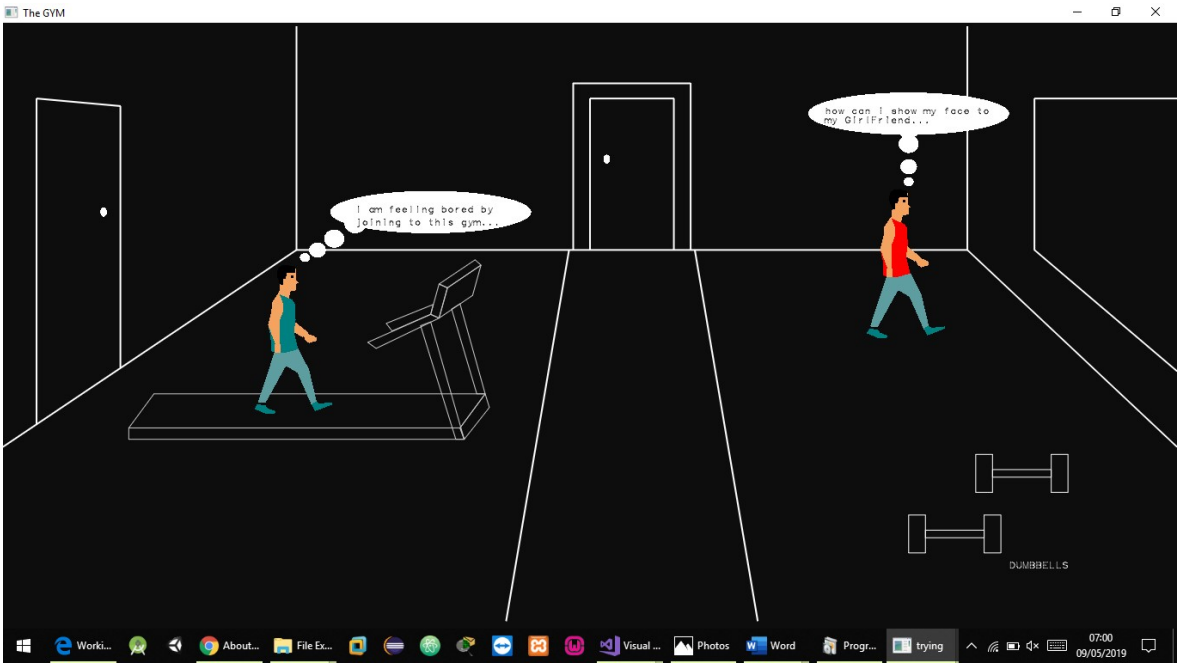
```
	case 'n':
	case 'N':
		if (SCENE_ID == 15)
			break;
		SCENE_ID++;
		break;
		// Quit Story
	case 'q':
	case 'Q':
		exit(0);
		break;
	default:
		break;
	}

	glutPostRedisplay();
}


void initializeScreen() {
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0, 1400, 0, 800);
}



int main(int argc, char **argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
	glutInitWindowPosition(0, 0);
	glutInitWindowSize(1400, 800);
	glutCreateWindow("The GYM");

	// Enables Transparency
	glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
	glEnable(GL_BLEND);

	// Enable Smoothening
	glEnable(GL_LINE_SMOOTH);
	glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
	// Display Function
	glutDisplayFunc(renderScene);
	// Input Functions
	glutKeyboardFunc(keyPress);
	glutMouseFunc(mouseClick);
	initializeScreen();
	// Timer Function
	// every 25 milliseconds, update function is called
	glutTimerFunc(25, update, 0);

	glutMainLoop();

	return 0;
}
```

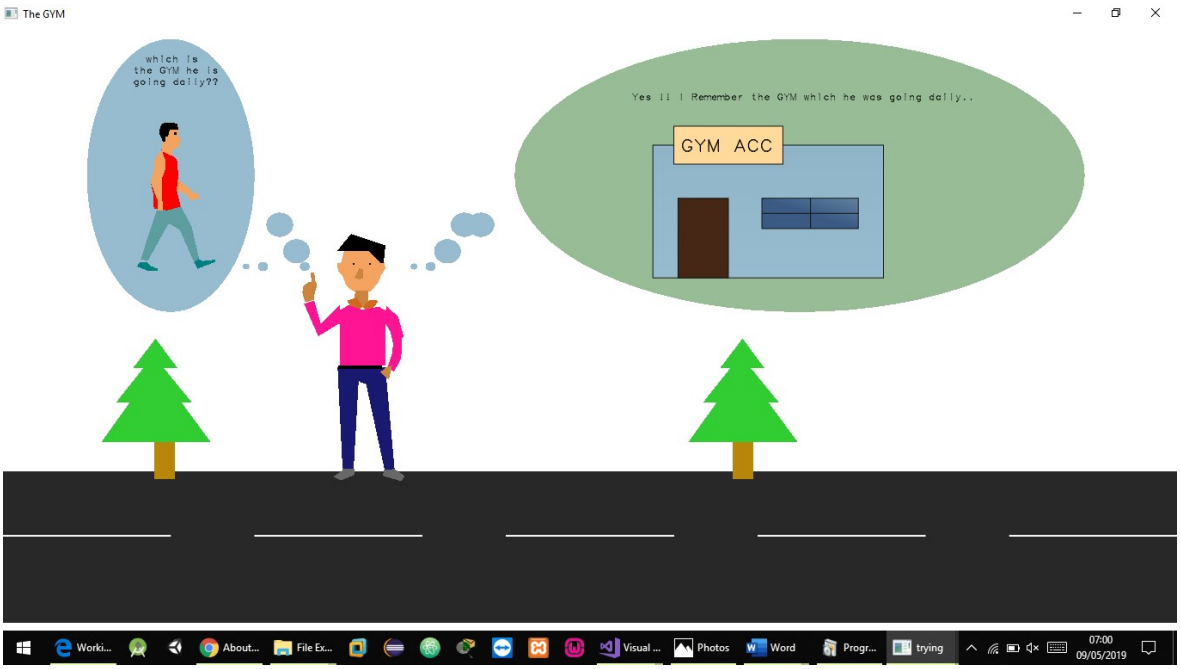
Chapter 6

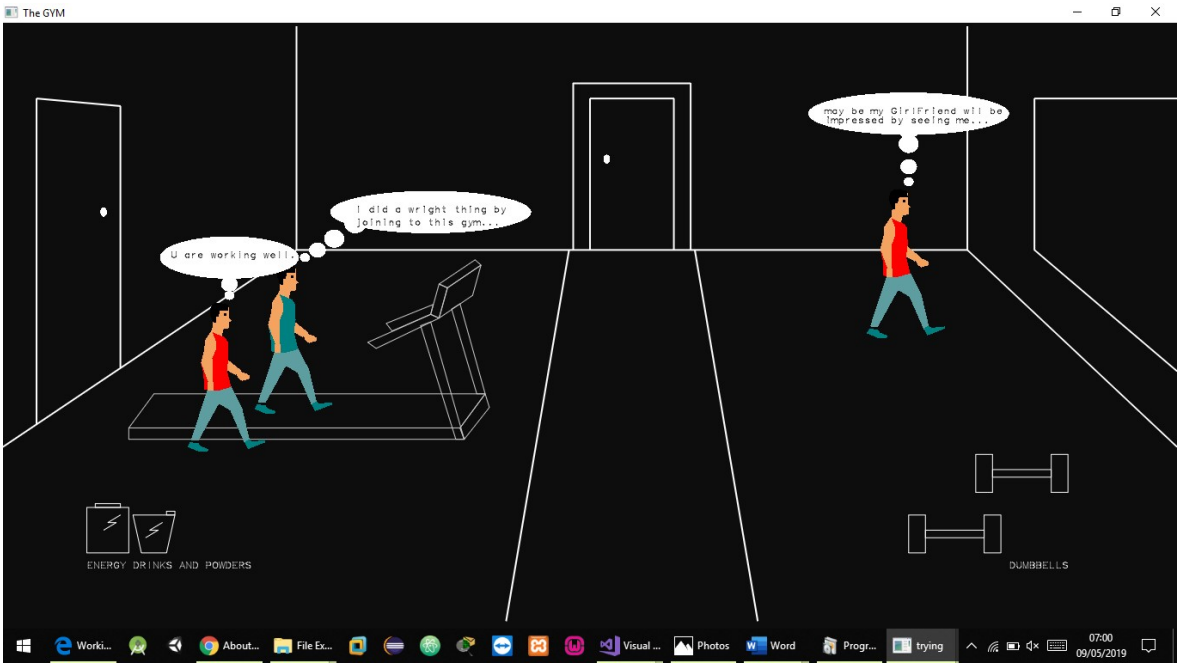
Snapshots






The GYM
I am feeling bored by joining to this gym...
how can I show my face to my GirlFriend...
DUMBBELLS


The GYM
which is the GYM he is going daily??
Yes !! I Remember the GYM which he was going daily..
GYM ACC


The GYM
U are working well!
I did a wright thing by joining to this gym...
may be my GirlFriend will be impressed by seeing me...
ENERGY DRINKS AND POWDERS
DUMBBELLS
Worki...
About...
File Ex...
Visual ...
Photos
Word
Progr...
trying
07:00
09/05/2019

Chapter 7

Conclusion and Future Enhancement

8.1 Conclusion

The source code developed is relatively simple, easy to comprehend and works effectively. A simple representation of how the person feels with the old Gym and how his life changes when he joins new gym.

Development of this computer graphics project has given us a better understanding of the topics and concepts learnt in theory. The project increased our knowledge in the field of computer graphics.

The hands on experience of developing a graphics project has helped us inculcate the good practices and principles of software engineering. This project has imbibed within us the spirit of teamwork.

8.2 Future Enhancement

We can Improve the project further by doing the following

- Adding More Animations to each screen
- Adding More frames to elaborate on the central theme
- Making the animations more interactive

Appendix

- Initially the user needs to install Microsoft Visual Studio C++ 17.0
- When he has to create a new project which is of Win32 Console Application Format, the Creation must be done as shown Below
- File -> New -> In the project menu select Windows 32 Console Application -> Give Appropriate name to the project -> Finish, This creates a new Project
- Then Insert a new class as Follows
- Insert -> New Class -> Give Appropriate name -> OK
- Select the file new and click on the source name file -> Source files -> Source filename.cpp and then paste the file into the workspace
- Compile the code by Build -> Compile
- Link the object file by selecting Build -> Build filename.exe
- Execute the filename.exe file by Build -> Execute filename.exe

Bibliography

Computer Graphics Principles and Practice – Foley, VanDam, Feiner, Huges

Graphics in C – Y. P. Kanetkar

Interactive Computer Graphics – Edward Angel

References:

- www.stackoverflow.com
- www.opengl.org
- http://www.aksiom.net/rgb.html
- http://en.wikipedia.org/wiki/List_of_colors:_A-F