VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangam, Belagavi-590018

Project Report
on

GRAPHICAL SIMULATION OF DESKTOP AND ITS COMPONENTS

by

ADARSH REVANKAR — **4MT16CS003**

AKSHAYA M. — **4MT16CS007**

UNDER THE GUIDANCE OF

Mr. PRASHANTH B. S.
Asst. Professor
Department of CS & E,
M.I.T.E, Moodbidri

Mrs. SARANYA BABU
Asst. Professor
Department of CS & E,
M.I.T.E, Moodbidri

MITE
Invent Solutions

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
(NBA Accredited)

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING
(An ISO 9001:2015 Certified Institution)
BADAGA MIJAR, MOODBIDRI DK DIST-574225

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(An ISO 9001:2015 Certified Institution)

BADAGA MIJAR, MOODBIDRI, DK DIST-574225

MITE

Invent Solutions

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(NBA Accredited)

CERTIFICATE

This is to certify that the mini project entitled **"GRAPHICAL SIMULATION OF DESKTOP AND ITS COMPONENTS"** is a bonafide work carried out by **ADARSH KRISHNA REVANKAR (4MT16CS003)** in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi** during even semester of **2018 - 19** on **Computer Graphical and Visualization Lab (15CSL68)**. The project has been approved as it satisfies the academic requirements in respect of project work prescribed for Bachelor of Engineering degree.

.....	.....
Signature of Guides	Signature of HOD
1. Mr. PRASHANTH B. S.	**Dr. VENKATRAMANA P BHAT**
2. Mrs. SARANYA BABU	

Examinors	Signature with date
1.	.....
2.	.....

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(An ISO 9001:2015 Certified Institution)

BADAGA MIJAR, MOODBIDRI, DK DIST-574225

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(NBA Accredited)

CERTIFICATE

This is to certify that the mini project entitled **"GRAPHICAL SIMULATION OF DESKTOP AND ITS COMPONENTS"** is a bonafide work carried out by **AKSHAYA M. (4MT16CS007)** in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi** during even semester of **2018 - 19** on **Computer Graphical and Visualization Lab (15CSL68)**. The project has been approved as it satisfies the academic requirements in respect of project work prescribed for Bachelor of Engineering degree.

.....	.....
Signature of Guides **1. Mr. PRASHANTH B. S.** **2. Mrs. SARANYA BABU**	Signature of HOD **Dr. VENKATRAMANA P BHAT**
Examinors	Signature with date
1.	.....
2.	.....

ABSTRACT

Computer graphics is an art of drawing pictures, lines, charts, etc using computers with the help of programming. It is a powerful tool to visualize various concepts and ideas that is difficult to perceive by any other medium. OpenGL is one such platform which provides feature rich APIs that contain geometric primitives to render 2D and 3D vector graphics from which complex graphical objects can be built. OpenGL Utility Toolkit (GLUT) has been created to aid in the development of more complicated three-dimensional objects such as a sphere, a torus, and even a teapot. Using OpenGL a graphical simulation of desktop and its components is created. This system allows user to explore the different parts of a desktop and understand their functions. Thus it gives the real-life experience of working with a desktop.

ACKNOWLEDGMENT

We express our gratitude to our institution and management for providing us with good infrastructures, laboratory facilities, qualified and inspiring staff, whose guidance was of immense help in completion of project successfully.

We are extremely thankful to our dignified principle **Dr. G. L. Easwara Prasad** for his support and encouragement.

We thank **Dr. Venkatramana P Bhat**, Professor and Head of the Department, Computer Science and Engineering for his encouragement and favours.

We would like to thank **Mr. Prasanth B. S.** and **Mrs. Sharanya Babu**, project guides, who meticulously went away beyond the call of duty in rendering numerous details comments on technical matters and inspiring and elucidating guidance at all stages of development of this project.

We would also like to express our deep gratitude to all lecturers and lab staffs for their vital and timely help that have made our project a success.

We would also like to thank our friend who have contributed in all possible ways in completing this project successfully.

Naturally we take full responsibility for any lack of clarity, occasional erratum or inexactness that may occur.

Adarsh Krishna Revankar (4MT16CS003)
Akshaya M. (4MT16CS007)

TABLE OF CONTENTS

Sl. No.	CONTENTS	Page No.
Abstract		**i**
Acknowledgement		**ii**
1	**Introduction**	**1**
2	**Literature survey**	**2**
	2.1 Existing system	**2**
3	**Proposed system**	**3**
4	**Software Requirement Analysis**	**5**
5	**System Design**	**6**
6	**Implementation**	**9**
7	**Result and Snapshot**	**12**
8	**Conclusion and Future Enhancements**	**13**
References and Bibliography		**14**

CHAPTER - 1

INTRODUCTION

Computer has become a powerful tool for the rapid and economical production of pictures. Computer Graphics remains one the most exciting and rapidly growing fields. Old Chinese saying "*One picture is worth of thousand words*" can be modified in this computer era into "*One picture is worth of many kilobytes of data*". It is natural to expect that graphical communication will often be more convenient when computers are utilized for this purpose. Many people for different domain of applications use interactive graphics. There is virtually no area in which graphical displays cannot be used to some advantage, and so it is not surprising to find the use of computer graphics so widespread. Today, we find Computer Graphics used routinely in such diverse areas such as science, engineering, medicine, business, industry, government, art, entertainment, advertising, education, training, etc.

Computer graphics are the representation of image data by a computer specifically with help from specialized graphic hardware and software. Computer graphics is creation, manipulation and storage of geometric objects (modelling) and their images (Rendering) shown in *Figure 1.1*.

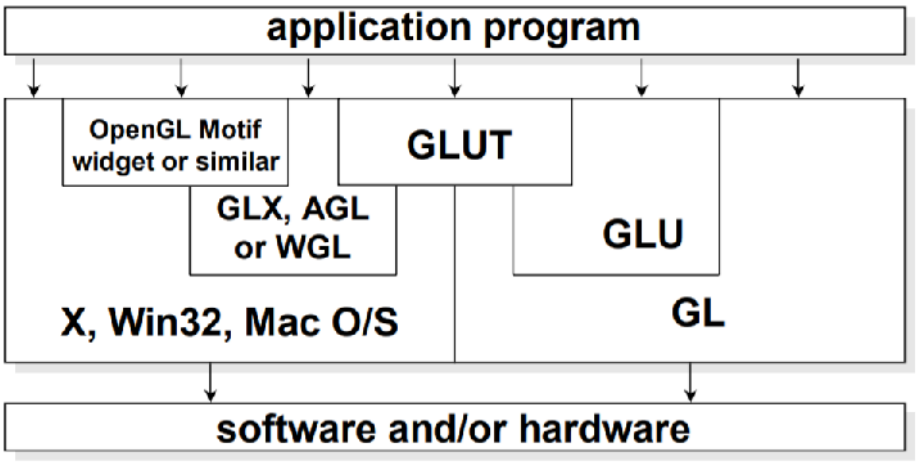

Figure 1.1 : OpenGL and Related APIs

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 the software interface to graphics hardware. OpenGL graphic rendering commands issued by your applications could be directed to the graphic hardware and accelerated.

OpenGL contains 3 sets of libraries, they are

1). Core OpenGL (GL)

Creation of models using set of geometric primitives such as points, line and polygon.

2). OpenGL Utility Library (GLU):

This library is built on top of Core OpenGL to provide important utilities (such as setting camera view and projection) and more building models (such as qradric surfaces).

3). OpenGL Utilites Toolkit (GLUT):

OpenGL is designed to be independent of windowing system or Operating System. GULT is needed to interact with Operating System and also provides more building models. This is built on top of platform-specific OpenGL extension (such as GLX - Windows System).

Summary: Introduction gives an overview of Computer Graphics, OpenGL and the 3 main libraries of OpenGL.

CHAPTER - 2
LITERATURE SURVEY

2.1 Existing System

Following modules have been obtained from existing systems and configured accordingly to graphically simulate desktop components.

1. Components shaped similar to ***cuboid, rectangle*** have been found in existing models [2] as shown in *Figure 2.1*. The code snippet has been obtained and modified according to the shape required (like Hard disk, Desktop Cabinet) and simple motions to enhance the visual feel has been added.

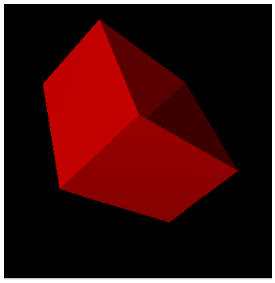

Figure 2.1: Cuboid / Rectangle

2. Components shaped similar to ***sphere*** have been found in existing models [3] as shown in *Figure 2.2*. The code snippet has been obtained and modified according to the shape required (like CPU Cooler, Screws) and simple motions to enhance the visual feel has been added.

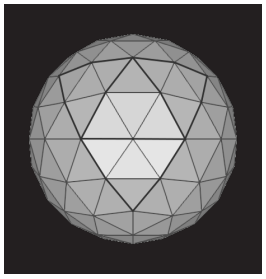

Figure 2.2: Sphere

3. Components shaped similar to ***Cylinder*** have been found in existing model [4] as shown in *Figure 2.3.* The code snippet has been obtained and modified according to the shape required (like Cables, Capacitors) and simple motions to enhance the visual feel has been added.

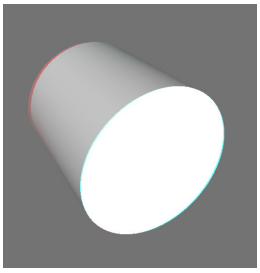

Figure 2.3: Cylinder

4. The component similar to CPU Fan has been found in existing model [5] as shown in *Figure 2.4.* The code snippet consist of the animations related to the motion of the fan, which could be implemented to the current system.

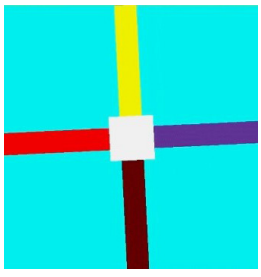

Figure 2.4: Fan Motion

Problem Statement

The disassembly of a real desktop is not an effective or cost-efficient way of studying the components of it. The existing manual way of disassembly of a desktop will not provide complete knowledge about the internal hardware and its components. A virtual way of dealing with this problem would result in better attention to individual needs. Hence the current system is proposed.

Summary: Literature Survey gives overview of the existing system and problems within it.

CHAPTER - 3

PROPOSED SYSTEM

This proposed system simulates the components of a desktop with which user can interact through some interface and information about the component. Thus, Proposed system contains of 3 modules - Object Module, Text Module and Transformation Module.

3.1. Object Module

Current System consists of a CPU Cabinet which accommodates smaller components like Motherboard, CPU, GPU, PSU, Head Sink, Cooling Fan, RAM Stick(s), Hard disk drive or and SSD etc

Figure 3.1: Object Module - CPU Cabinet consisting of all the components intact

3.2. Input Module

This module defines the acceptable and valid user interactions.

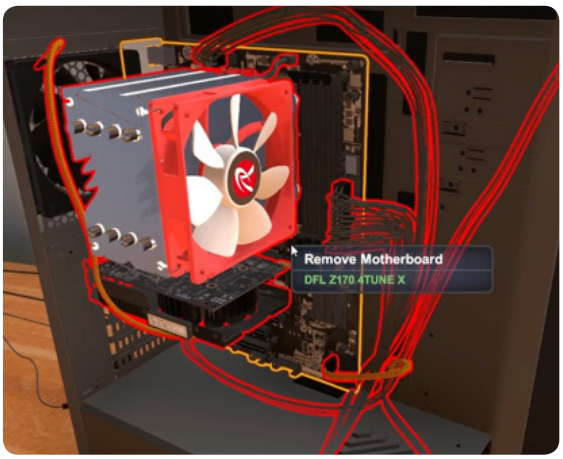

Figure 3.2

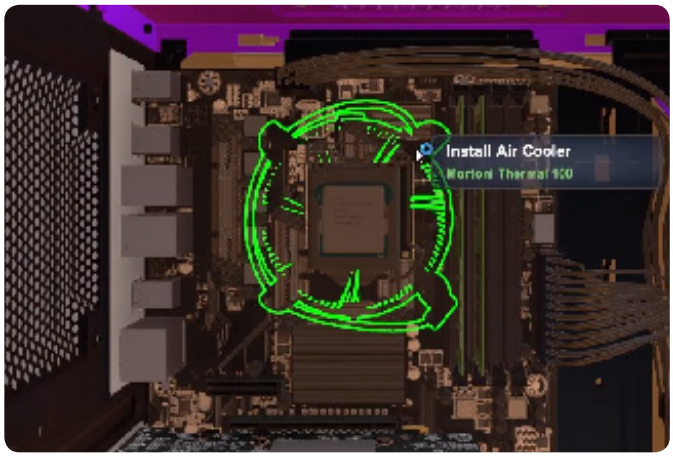

Figure 3.3

Figure 3.2 : Mother Board and sub parts are selected through the mouse input.
Figure 3.3 : CPU Cooler is select through mouse click (input)

3.3. Transformation Module

Transformation module changes the view from generic to specific components of the desktop.

The transformation occurs due to the action specified by the Input Module.

Figure 3.4

Figure 3.5

Figure 3.6

Figure 3.4, 3.5 and 3.6 : Represents the transformation of CPU Cooler, which it will be fitted on top of CPU. The transition is shown in small amount in the distance from Mother Board to CPU Cooler in serial pictures from left to right.

3.4. Text Module

Text module annotates information based on the action specified by the Input Module. It provides details about specific interacted components of the desktop.

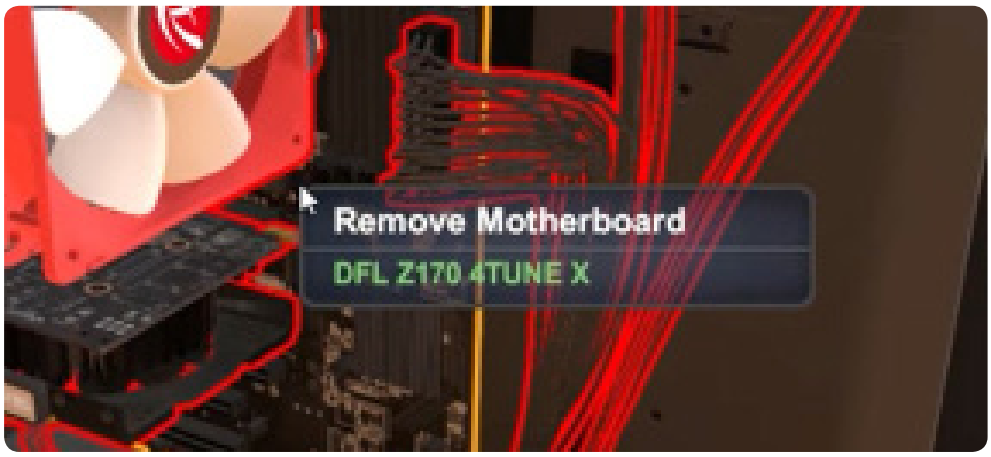

Figure 3.7 : Text Module - Represents the text based information over the window to give information about the system.

Summary: Propose system gives an overview of the solution provided to the problems in the existing system.

CHAPTER - 4

SOFTWARE REQUIREMENT ANALYSIS

Definition

A software requirements specification (SRS) is a description of a software system to be developed. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function (in a market-driven project, these roles may be played by the marketing and development divisions). Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. The SRS is divided into functional and non-functional requirements

4.1. Non-Functional Requirements

In system engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.

4.2. Functional Requirements

The functional requirements are the requirements which are needed to develop the software application. The functional requirement are broadly classified into 2 categories, they are :

4.2.1 Software Requirements

The minimum requirements are

- **Operating System :** Windows 8.1 / 10
- **IDE** : Microsoft Visual Studio 2017
- **Libraries** : Freeglut (GLUT Libraries)

4.2.2 Hardware Requirements

The minimum requirements are

- **Processor** : Intel / AMD, clock Speed 1.5 GHz
- **RAM** : 2 GB DDR3
- **Hard Disk Space :** 500 MB
- **Screen:** Coloured Monitor with 1920 * 1080 resolution

CHAPTER - 5

SYSTEM DESIGN

The modular design will describe the various components used in this project. Figure 5.1 it shows the different modules present in the proposed graphical project.

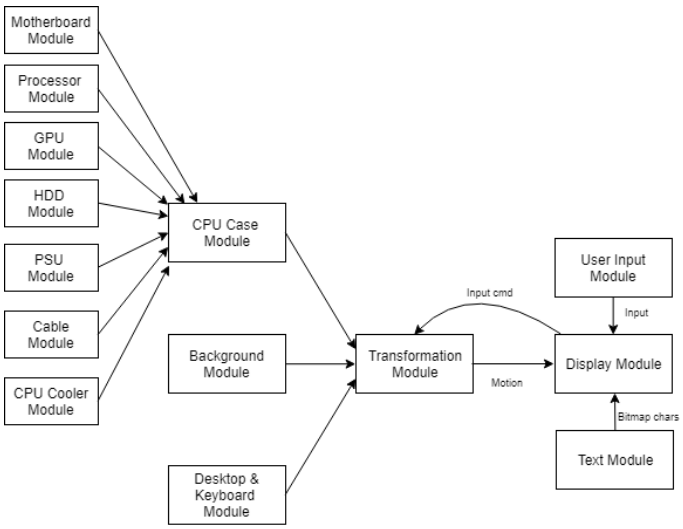

Figure 5.1 : Modular Design - Construction of system

5.1 CPU Case Module

This module consists of various sub-modules of the CPU module.

5.1.1 Motherboard Module *:* Contains the structure of motherboard (CPU Socket etc.)

5.1.2 Processor Module : Contains the processor structure.

5.1.3 GPU Module : Contains the Graphical Processing Unit structure.

5.1.4 HDD Module : Contains the Hard Disk Drive structure.

5.1.5 PSU Module : Contains the Power Supply Unit (SMPS Box) structure.

5.1.6 Cable Module : Connections between various modules is achieved through this module.

5.1.7 CPU Cooler Module : Contains the Cooling fan and Heat sink structure.

5.2 Background Module

This module contains various background objects to complete the picture of background. This module contains the structure of the house, sofa, table, desk etc.

5.3 Desktop & Keyboard Module

This module contains the desktop and keyboard structure of the computer.

5.4 Transformation Module

Transformation module performs the various types of transformation on the components. This module provides the camera free motion, rotation, translation and scale motions. All the other object modules undergo transformations due to this module, according to the user input provided at 5.6.

5.5 Display Module

This module produces the frame of the current scene under transformation. This module is supported by the text module which uses Bitmap to produce text to guide the user.

5.6 Input Module

User actions (Mouse click, Mouse hover, Key press) is recorded through this module and given to the transformation module for further action and for processing the inputs.

5.2. Hardware Design

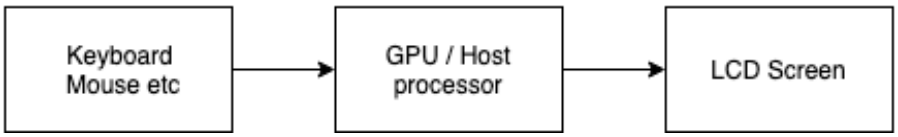

Figure 5.2 : Hardware Design

- **Keyboard and Mouse :** Keyboard is a device which contains the keys which returns the ASCII value when pressed. According to the user interaction the intended procedure is carried out. Similarly, mouse contains buttons which can obtain user input in the form of a click or hovering and carry out the intended procedure.
- **GPU / Host Processor :** GPU is Graphical Processing Unit which can handle parallel processing like drawing lines, points, polygons better than serial processor ie. CPU. GPU accelerates the creation of image in frame buffer intended for displaying device.
- **LCD Screen :** This contains individual Light Emitting Diode that contains either Red, Green, Blue light colour emitting diodes. These will be illuminated when stream of bytes (usually 32bit) of information comes to the display at 30 to 60 times per second. The diode emits light according to this information.

5.3. Software Design

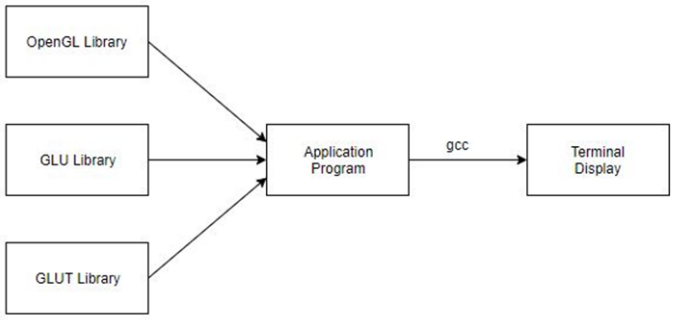

Figure 5.3 : Software Design

- **Application Program :** Executable is created by compiling and linking several *OpenGL* libraries. These instructions are loaded into memory which is already in executable format. Therefore the loading time will be less.
- **Terminal Display :** The window which provides the view-port for displaying the scene.

5.3. Flow Chart

- Flowchart is a graphical representation of a computer program in relation to its sequence of functions. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. Flowcharts are used in analysing, designing, documenting or managing a process or program in various fields.
- *Figure 5.3* gives a visual representation of the sequence of steps and decisions needed to perform a process in the proposed project.

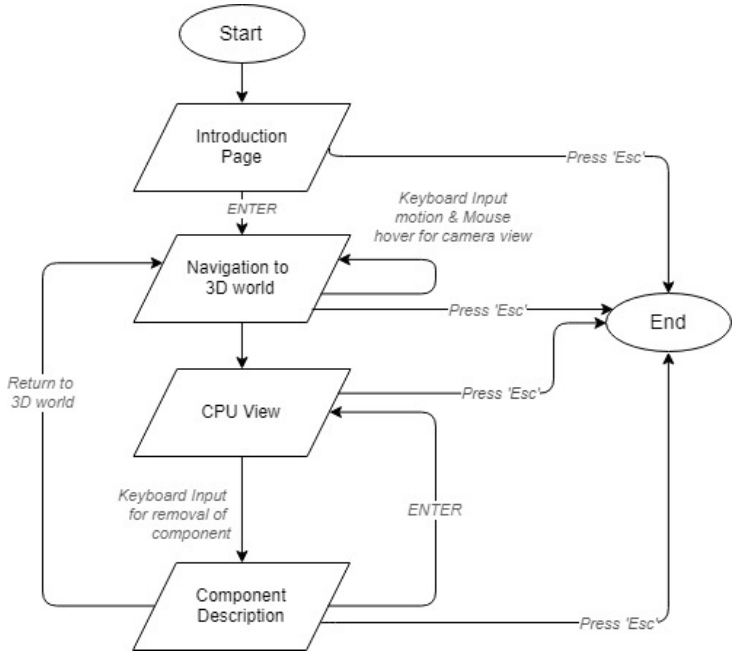

Figure 5.3 : Flow Chart of System - overall flow of system

Summary: This chapter gives brief overview on the architecture of the system.

CHAPTER - 6

IMPLEMENTATION

Implementation is the core step in software development life cycle. Implementation gives the detailed view of the project and describes the pseudo code and various important functions in the project. It will also give insight about various inbuilt modules and functions in OpenGL.

6.1. Code Snippet : Bitmap Loader

The text module has been displayed in 3D environment, by the code snippet below. Here each character is encoded to display in view-coordinates.

```
void* times10 = GLUT_BITMAP_TIMES_ROMAN_10;
void* helv18 = GLUT_BITMAP_HELVETICA_18;
void* helv12 = GLUT_BITMAP_HELVETICA_12;

float prog;

void renderBitmapString(
    float x,
    float y,
    float z,
    void* font,
    char* string) {

    char* c;
    glRasterPos3f(x, y, z);
    for (c = string; *c != '\0'; c++) {
        glutBitmapCharacter(font, *c);
    }
}
```

6.2. Code Snippet : Texture Map Loader

In order to map the bitmap images to onto the 3D objects they must be read in specific way from file. Then using this array we are able to generate textures.

```
BmpLoader::BmpLoader(const char* filename)
{
    fopen_s(&file, filename, "rb");
    if (file == NULL) {
        printf("ERROR : BITMAP LOAD - File not found : File = %s\n", filename);
        exit(0);
    }

    /// Read Image height & width.
    getImageSize();

    /// Read Image content
    data = (unsigned char*)malloc(iWidth * iHeight * 3);

    fread(data, iWidth * iHeight * 3, 1, file);

    fclose(file);

    /// Swap the R <-> G
    for (int i = 0; i < iHeight * iWidth; ++i)
    {
        int index = i * 3;
        unsigned char B, R;
        B = data[index];
        R = data[index + 2];

        data[index] = R;
        data[index + 2] = B;
    }
```

6.3. Code Snippet : drawCPU

To assemble all the complex components defined in each of their class, the drawCPU method is used. Here all the objects are instantiated and the render( ) method of each object is called.

```
void drawCPU() {
    fan_.render();  //Renders Fan
    motherboard_.render();
    case_.render();
    ram_.render(8., 4.845, -4.296);
    ram_.render(8., 4.845, -4.268);
    ram_.render(8., 4.845, -4.235);
    chipset_.render();
    gpu_.render();
    psu_.render();
    harddisk_.render();
    env_.render();
    envTable_.render();

    //syncMove();
}
```

6.4. Code Snippet : Motherboard Surface Creation + Apply Texture Map

In the code snippet given below, activation of texture mapping & creation of surface and mapping to it's surface with the required indexed texture is shown. At the end texture mapping is disabled.

```
    // Motherboard front
    glBindTexture(GL_TEXTURE_2D, textures[MOTHERBOARD_FRONT]);
    glBegin(GL_QUADS);
        glTexCoord2f(0., 0.);   glVertex3f(-1., -1., 0.);
        glTexCoord2f(0., 1.); glVertex3f(-1., 1., 0.);
        glTexCoord2f(1., 1.); glVertex3f(1., 1., 0.);
        glTexCoord2f(1., 0.); glVertex3f(1., -1., 0.);
    glEnd();

    // Motherboard back
    glBindTexture(GL_TEXTURE_2D, textures[MOTHERBOARD_BACK]);
    glBegin(GL_QUADS);
        glTexCoord2f(0., 0.);   glVertex3f(-1., -1., -boardThickness);
        glTexCoord2f(0., 1.); glVertex3f(-1., 1., -boardThickness);
        glTexCoord2f(1., 1.); glVertex3f(1., 1., -boardThickness);
        glTexCoord2f(1., 0.); glVertex3f(1., -1., -boardThickness);
    glEnd();
    glDisable(GL_TEXTURE_2D);
```

- In the above code glTexCoord2f( ) specifies the texture's coordinate mapped to the respective vertex of the polygon using glVertex3f( ).
- MOTHERBOARD_BACK is a macro definition used to point the texture's index from the generated list. Each index is associated with a .bmp file location.
- glEnable(GL_TEXTURE_2D) is used to enable texture mapping. To disable the same we use glDisable(GL_TEXTUER_2D).

List of Built - in functions used

The following are list of OpenGL basic functions are used,

- ***void glColor3f(GLfloat red,GLfloat green,GLfloat blue):*** This function sets the present RGB colors. Different color is given to the object using the color parameters such as red,green,blue.

- ***void glClear(GL_COLOR_BUFFER_BIT)***:Clears all buffer whose bits are set in mask. The mask is formed by logical OR of values defined in gl.h GL_COLOR_BUFFER-BIT refers to color buffers.

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

- ***void glBegin()*** and void ***glEnd():***The glBegin and glEnd functions delimit the vertices of a primitive group of like primitives. Syntax of glBegin is void glBegin(GLenum mode);

- ***void glFlush(void):*** The glFlush function forces execution of OpenGL functions in finite time. This function has no parameters. This function does not return a value.

- ***void glMatrixMode(GL_PROJECTION):*** Applies subsequent matrix operations to the projection matrix stack.

- ***void glutMainLoop(void):*** 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.

- ***void glutKeyboardFunc(void (*func)(unsigned char key,int x, int y)):*** glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback.

- ***GLint gluBuild2DMipmaps(GLenum target,GLint internalFormat,GLsizei width,GLsizei height,GLenum format,GLenum type,const void * data) :*** builds a series of prefiltered two-dimensional texture maps of decreasing resolutions called a mipmap. This is used for the antialiasing of texture-mapped primitives.

- ***void gluPerspective(GLdouble fovy,GLdouble aspect, GLdouble zNear, GLdouble zFar) :*** specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in gluPerspective should match the aspect ratio of the associated viewport.

Summary: This chapter gives glance of few of the functions used for the development of the objects.

CHAPTER - 7

RESULT AND SNAPSHOTS

Figure 7.1 : Introduction Page

7.1 Snapshot

Introduction Page

Figure 7.1 shows the initial view of the project. This also contains information about the contributors and direction to the next page is given.

Figure 7.2 : Desktop View

7.2 Snapshot

Desktop View

Figure 7.2 shows the components of a desktop which is supposed to be disassembled.

Figure 7.3 : CPU View

7.3 Snapshot

CPU View

Figure 7.3 shows the CPU's internal components intact.

Figure 7.4 : Disassembled View

7.4 Snapshot

Disassembled View

Figure 7.4 shows all the components present in the CPU in detail.

Summary: This chapter contains the implemention view of the project. It includes snapshots of it.

CHAPTER - 8

CONCLUSION AND FUTURE ENHANCEMENT

7.1. CONCLUSION

This project tries to give a complete overview on the various components of the computer. OpenGL provides various functions with which the user interaction is made easier to achieve the aforementioned. Due to the utilization of texture mapping which is a components of OpenGL we can give the user a real life experience of working with a desktop.

7.2. FUTURE ENHANCEMENT

In the future enhancement of the project, the following features can be added :

- More user friendly interactions with the components.
- For each component, different variations can be given so that the user can choose from it during the assembly procedure.
- Each configuration of the assembled desktop by the user can be Benchmarked.

REFERENCES AND BIBLIOGRAPHY

[1]. Donald Hearn & Pauline Baker: Computer Graphics-OpenGL Version,3rd Edition, Pearson Education,2011

[2]. Cuboid Component : https://henry416.wordpress.com/2013/11/09/open-gl-3d-cuboid-transformation-example/

[3]. Spherical Component : http://www.songho.ca/opengl/gl_sphere.html

[4]. Cylinder Component : http://quiescentspark.blogspot.com/2011/05/rendering-3d-anaglyph-in-opengl.html

[5]. Fan Animation Code : https://vharesh4.wordpress.com/2015/09/09/animation-fan-c-program/

[6]. Wikipedia - OpenGL : https://en.wikipedia.org/wiki/OpenGL

[7]. OpenGL function description : https://www.khronos.org/

PHASE-1: SYNOPSIS

PROJECT TITLE:

”GRAPHICAL SIMULATION OF DESKTOP AND ITS COMPONENTS”

SUBMITTED BY:

ADARSH KRISHNA REVANKAR	**4MT16CS003**
AKSHAYA M.	**4MT16CS007**

Submitted to: 1. Mr. PRASHANTH B S, Asst. Professor, Dept. of CSE
2. Mrs. SARANYA BABU, Asst. Professor, Dept. of CSE

Submitted Date:

Name of the Guide	Signature of the Guide
Mr. Prashanth B S	**…….**
Mrs. SaranyaBabu	**…….**