# TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

Computer graphics are the image creating by plotting pixels on the computer screen. Computer Graphics provides one of the most natural means of communicating with the computer, since our highly developed 2D and 3D pattern recognition allow us to perceive and pictorial data rapidly and efficiently.

Data encryption is a process to transforms information from a form that is readable to a form that is not. Decryption reverses this process. Ideally, it should not be possible to perform one or both of these operations without knowing some secret key, which generally takes the form of a string of 1's and 0's.

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. Decryption is the process of transforming encrypted information so that it is intelligible again. A cryptographic algorithm, also called a cipher, is a mathematical function used for encryption or decryption. In most cases, two related functions are employed, one for encryption and the other for decryption.

The important of this project is encryption makes information systems trustworthy in a variety of ways. First, encryption can protect information confidentiality. If information is encrypted before it is transmitted across a telephone network or stored in a database, eavesdroppers or hackers may capture the encrypted information, but they won't understand it. Second, encryption can be used for authentication, i.e. to verify the identity of other parties. For example, if only Ann holds the key to encrypt a message, then by performing this operation on an encrypted message, Ann can prove her identity. This is the basis of a digital

signature, which can be used to authorize payments or sign contracts. Third, encryption can be used to protect the integrity of information. Consider a case where only Bob can encrypt a message, but anyone can decrypt it. Bob records the message and an encrypted version of the message. If decrypting the latter still produces the former, then the message could not have been altered by any one except Bob.

# CHAPTER 2

# SYSTEM REQUIREMENTS

## HARDWARE REQUIREMENTS

Processor : Intel(R) Core(TM) i5-6200U

RAM : 1GB

ROM : 1GB

## SOFTWARE REQUIREMENTS

Operating system : WINDOWS 10

Programming language : C++

IDE Used : Dev C++

Libraries : OpenGL libraries(openGl32.lib glu32.lib glut32.lib)

# CHAPTER 3

# DESIGN

Software design is the process by which an agent creates a specification of a software artefact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems or the activity following requirements specification and before programming, as a stylized software engineering process.

## 3.1 FLOW CHART DIAGRAM

A flowchart is a diagram that depicts a process, system or computer algorithm.

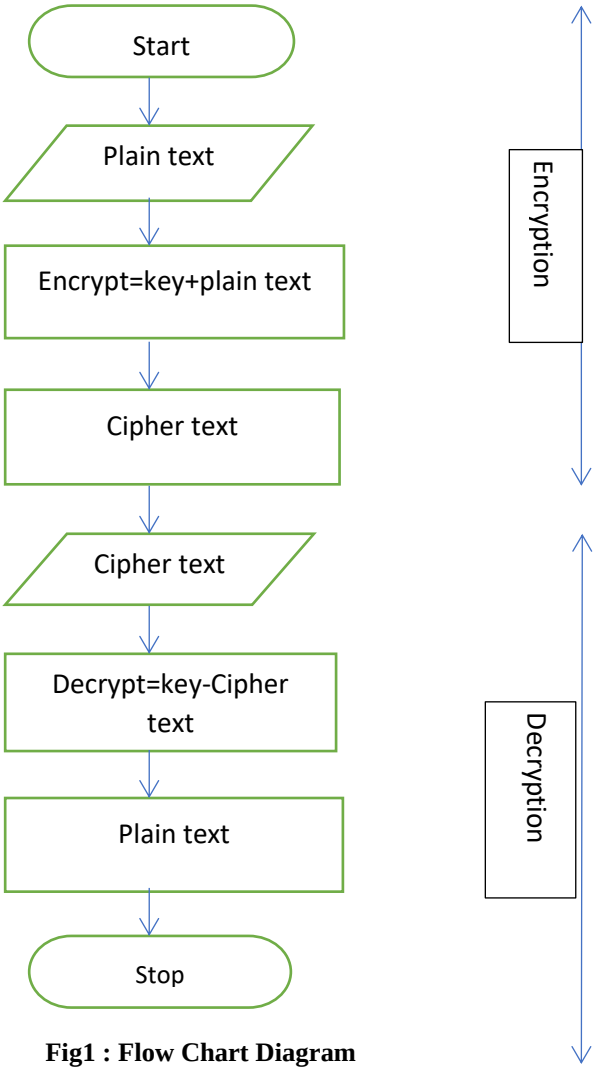


**Fig1 : Flow Chart Diagram**

# CHAPTER 4

# IMPLEMENTATION

Implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment.OpenGL is the standard programmer's interface both for writing application programs and developing high-level products for multiplatform applications. OpenGL supports application ranging from large scientific visualization to cell phone games. Applications are designed to access directly through functions in their libraries.

## 4.1 Modular Description

### 4.1.1 BUILT IN METHODS

**glutAttachMenu()**;glutAttachMenu attaches a mouse button for the current window to the identifier of the current menu

**glutMainLoop()**: glutMainLoop enters the GLUT event processing loop.

**glutAddMenuEntry()**:glutAddMenuEntry adds a menu entry to the bottom of the current menu.

**glutCreateMenu()**:glutCreateMenu creates a new pop-up menu.

**glutDisplayFunc()**: glutDisplayFunc sets the display callback for the current window.

**glutKeyboardFunc()**:glutKeyboardFunc sets the keyboard callback for the current window.

**glutCreateWindow()**:glutCreateWindow creates a top-level window.

**glutInitDisplayMode()**: glutInitDisplayMode sets the initial display mode.

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

**glutInit()** : glutInit is used to initialize the GLUT library.

**glutDisplayFunc()**:glutDisplayFunc sets the display callback for the current window.

**glutSwapBuffers()**:Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer. The contents of the back buffer then become undefined. The update typically takes place during the vertical retrace of the monitor, rather than immediately after glutSwapBuffers is called.

**glFlush()**: Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

**glMatrixMode()**:Specify which matrix is the current matrix.

**glutWireCube()**: glutWireCube render a wireframe cube.

**glBegin()**:Delimit the vertices of a primitive or a group of like primitives.

**GlVertex()**: glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices.

**glCallList()**:glCallList or glCallLists to execute display lists that include only the preceding functions.

**glClear()**: glClear sets the bitplane area of the window to values previously selected by glClearColor, glClearIndex, glClearDepth, glClearStencil, and glClearAccum. Multiple color buffers can be cleared simultaneously by selecting more than one buffer at a time using glDrawBuffer.

**glRasterPos()** :The GL maintains a 3D position in window coordinates. This position, called the raster position, is used to position pixel and bitmap write operations. It is maintained with subpixel accuracy.

**glColor()** : The GL stores both a current single-valued color index and a current four-valued RGBA color. glColor sets a new four-valued RGBA color. glColor has two major variants: glColor3 and glColor4.glColor3 variants specify new red, green, and blue values explicitly and set the current alpha value to 1.0 (full intensity) implicitly. glColor4 variants specify all four color components explicitly.

**glutBitmapCharacter()** :Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font.

**glPushMatrix()** : There is a stack of matrices for each of the matrix modes. In GL_MODELVIEW mode,the stack depth is at least 32. In the other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is at least 2. The current matrix in any mode is the matrix on the top of the stack for that mode. glPushMatrix pushes the current matrix stack down by one, duplicating the current matrix. That is, after a glPushMatrix call, the matrix on top of the stack is identical to the one below it.

**glTranslatef()** : glTranslate produces a translation by x,y,z. The current matrix (see glMatrixMode) is multiplied by this translation matrix, with the product replacing the current matrix, as if glMultMatrix were called with the following matrix for its argument:

1 0 0 x

0 1 0 y

0 0 1 z

0 0 0 1

**glScaled()** : glScale produces a nonuniform scaling along the x, y, and z axes. The three parameters indicate the desired scale factor along each of the three axes.

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

### 4.1.2 USER DEFINED FUNCTIONS:

**Computer():**This method is used to draw the the sender and receiver computer.

```
void computer()
{
        glColor3f(1.0, 0.0, 0.0);
        glBegin(GL_QUADS);
        glVertex2f(55.0, 340.0);
        glVertex2f(145.0, 340.0);
        glVertex2f(150.0, 350.0);
        glVertex2f(60.0, 350.0);
        glEnd();
}
```

**Cipher():**This method will encrypt the plain text to cipher text.

```
void cipher()
{
        glColor3f(0.0f, 0.0f, 1.0f);
        glPushMatrix();
        glScalef(40, 20, .5);
        glTranslatef(3, 14, 0);
        glutWireCube(2);
        output(-1, 0,(char *)" CIPHER", 2);
        glPopMatrix();
}
```

**Message_data():**This method to pass the message to the block.

```
void message_data()
{
        glColor3f(1.0f, 1.0f, 0.0f);
        glPushMatrix();
        glScalef(40, 20, .5);
        glTranslatef(3, 14, 0);
        glutWireCube(2);
        output(-1, 0,(char *)" MESSAGE", 2);
        glPopMatrix();
}
```

**Message_key():**To attach the key to the message.

```
void message_key(void)
{
        glColor3f(0.0f, 1.0f, 0.0f);
        glPushMatrix();
        glScalef(20, 20, .5);
        glTranslatef(3, 14, 0);
        glutWireCube(2);
        output(-1, 0,(char *)"KEY", 2);
        glPopMatrix();
}
```

# CHAPTER 5

# SCREENSHOTS

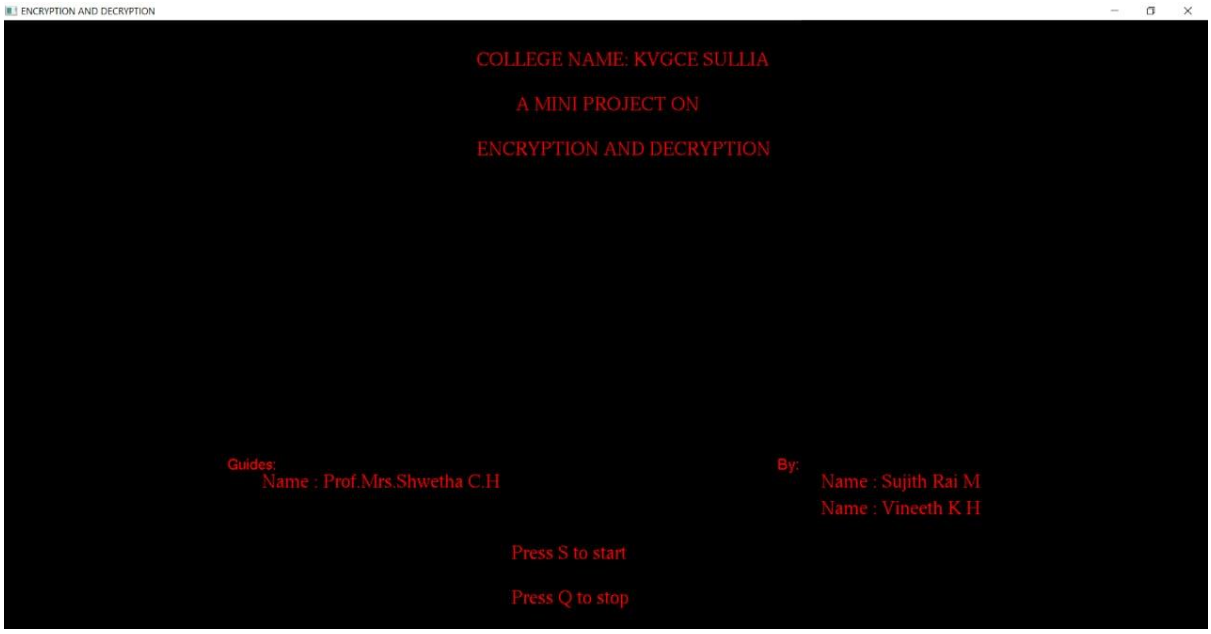


**Fig 3 :Home Page**

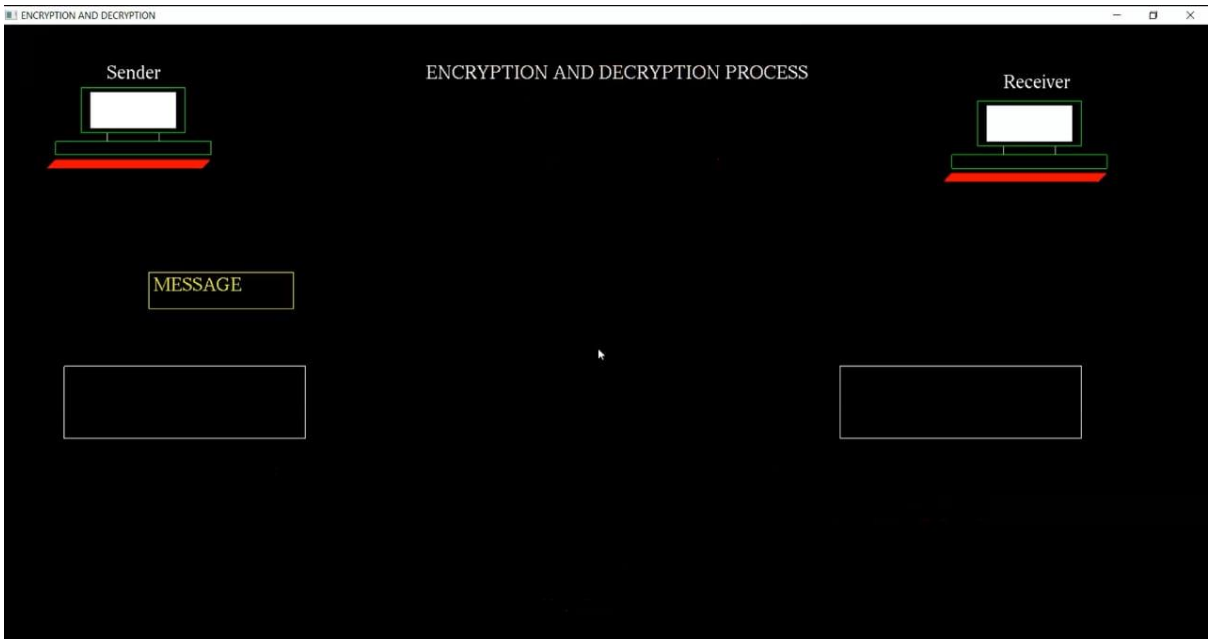


**Fig 4 : Sender Side**

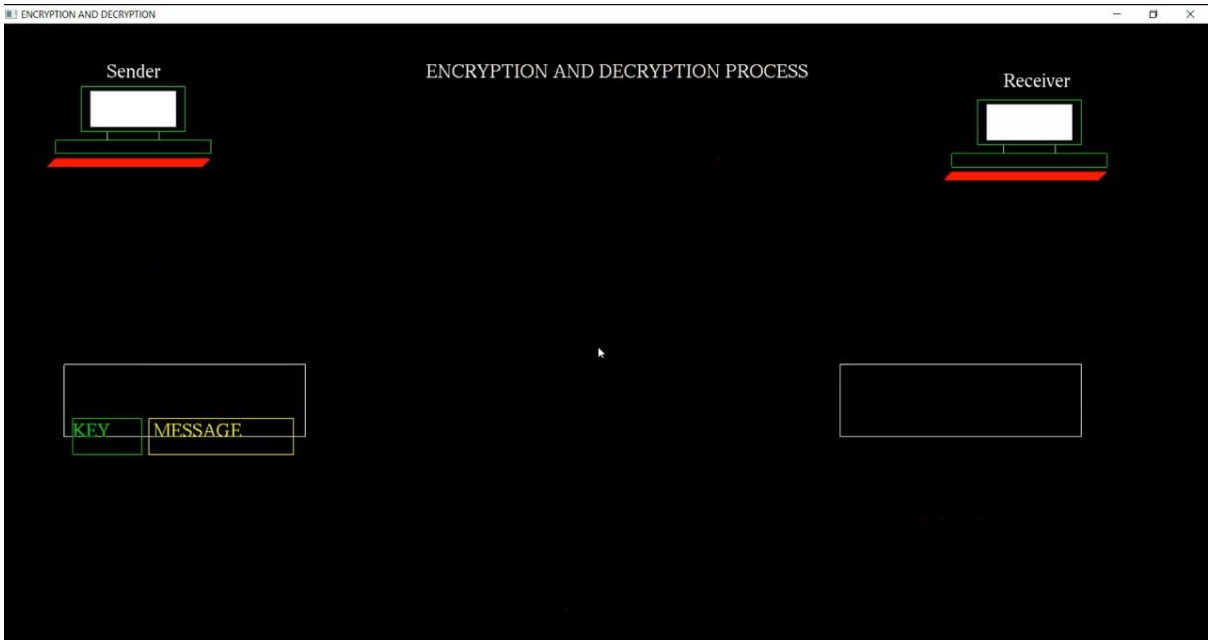


**Fig 5 :Key Generation at sender side**


**Fig 6 :Cipher Text**


**Fig 7 :Key generation at receiver side**


**Fig 8:Key elimination at receiver side**


**Fig 9: receiver side**

# CHAPTER 6

# USER MANUAL

This project contains 2 panes.First one is home window(fig2),second one is encryption and decryption process window.To simulate the encryption and decryption operation,follow the following steps;

**Step 1 :**Click on the Execute tab on menu bar and select Run on Dev C++ IDE to execute the program.

We can also press F10 button on the keyboard.

**Step 2 :**When we hit the Run button,it will show the welcome window.Press 'S' to start.When we Press 'S' it will display two computer for sender and receiver. Transmission is shown with movement of boxes. A message is passed by sender which is added with some key. This new message with secret key became the cipher. The Cipher travel via network and reaches receiver. At this point receiver decrypt the message to plain text by removing the secret key.

**Step 3 :**To stop the process Press 'Q',which will close the output window.We can also right click the mouse which will show the exit option,click on the that which will close the operation.To restart repeat the above step starting from step 1.

# CONCLUSION

Data encryption and decryption systems are used to improve information security to secure data that, thereby providing enhanced level of assurance such that the data that are encrypted cannot be viewed by unauthorized parties in the event of theft, loss or interception. This system replaces the existing data encryption and decryption system by adding some functionality such as digital signature.

Future works could be devoted to scaling the system to be able to encrypt and decrypt other types of files, including audio, video, image, to mention but three.

# REFERENCE

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

2. Edward Angel: Interactive computer graphics- A Top Down approach with OpenGL,5th edition. Pearson Education, 2011

3. M M Raikar, Computer Graphics using OpenGL, Fillip Learning / Elsevier,Bangalore / New Delhi (2013)

4. http://www.opengl-tutorial.org/beginners-tutorials/tutorial-1-opening-a-window/