**CHAPTER 1**

**INTRODUCTION**

**1.1 Computer Graphics**

“A picture is worth a thousand words” is a well-known saying and highlights the advantages and benefits of the visual presentation of our data. Computer graphics is concerned with all aspects of producing images using a computer. It concerns with the pictorial synthesis of real or imaginary objects from their computer-based models. Years of research and development were made to achieve the goals in the field of computer graphics. In 1950 the first computer driven display was used to generate only simple pictures. This display made use of a cathode ray tube similar to the one used in television sets. Although the term often refers to the study of three-dimensional computer graphics, it also encompasses two-dimensional graphics and image processing. Computer graphics studies the manipulation of visual and geometric information using computational techniques. It focuses on the mathematical and computational foundations of image generation and processing rather than purely aesthetic issues. Computer graphics is often differentiated from the field of visualization, although the two fields have many similarities.[1]

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 often abbreviated as CG, though sometimes erroneously referred to as computer-generated imagery (CGI).Some topics in computer graphics include [user interface design](https://en.wikipedia.org/wiki/User_interface_design), [sprite graphics](https://en.wikipedia.org/wiki/Sprite_(graphics)), [vector graphics](https://en.wikipedia.org/wiki/Vector_graphics), [3D modelling](https://en.wikipedia.org/wiki/3D_modeling), [shaders](https://en.wikipedia.org/wiki/Shader), [GPU](https://en.wikipedia.org/wiki/GPU) design, [implicit surface](https://en.wikipedia.org/wiki/Implicit_surface) visualization with [ray tracing](https://en.wikipedia.org/wiki/Ray_tracing_(graphics)), and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), among others. The overall methodology depends heavily on the underlying sciences of [geometry](https://en.wikipedia.org/wiki/Geometry), [optics](https://en.wikipedia.org/wiki/Optics), and [physics](https://en.wikipedia.org/wiki/Physics).

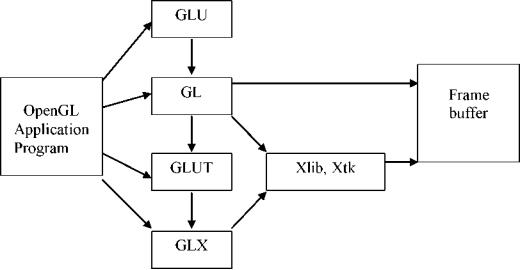
**1.2 Open GL**

Open GL emerged from Silicon Graphics Inc, in 1992, and has become a widely adopted graphics programming interface (API). It provides the actual drawing tools through a collection of functions that are called within an application. It is easy to install and learn, and its longevity as a standard API is being nurtured and overseen by the OpenGL Architecture Review Board (ARB), an industry consortium responsible for guiding its evolution.[3]

One aspect of OpenGL that suits it so well for use in computer graphics course is its device independence or portability. A student can develop and run a program on any available computer.

OpenGL offers rich and highly usable API for 2D graphics and image manipulation, but its real power emerges with 3D graphics.

Our project is based on the movement of the racing cars in the pit lane. They sort themselves based on the number on the cars. These are drawn and moved across the screen by using multiple GL functions like GL\_POLYGON, GL\_LINES, GL\_POINTS, etc. glutPostRedisplay is used to redraw the elements on the screen when they are translated or rotated.



**Figure 1.2.1 OpenGL Library Organization**

**1.3 Origami**

The main idea of origami is to step by step explain how to create a origami figure. In the demonstration window we have shown the steps of creating few types of origami in both 2D and 3D.

Origami  from ori meaning "folding", and kami meaning "paper" is the [art](https://en.wikipedia.org/wiki/Paper_art) of paper folding, which is often associated with Japanese culture. In modern usage, the word "origami" is used as an inclusive term for all folding practices, regardless of their culture of origin. The goal is to transform a flat square sheet of paper into a finished sculpture through folding and sculpting techniques. Modern origami practitioners generally discourage the use of cuts, glue, or markings on the paper.

This project will help people to create origami’s with ease because of the step by step procedure. Younger generation will learn to make origami’s thus making sure that this art won’t go extinct. This project implementation is done through OpenGl and graphics packages.

## **1.4 Applications of Origami.**

* Origami is an excellent creative outlet that anyone can use to help reduce stress generated by life challenges such as exhausting work schedules, or demanding family circumstances.
* It is a gentle activity that will shift our focus away from these challenges and help us focus on the present.
* It helps children to learn to create origami’s and thereby can be a distraction from the social media addiction.

**CHAPTER 2**

**REQUIREMENT ANALYSIS**

Requirement specification is focused specially on functioning of the system, functions to be carried out and performance levels to be obtained and corresponding interfaces to be established.

This report gives the description of the roles of users, the functional overviews of the project, input and output characteristics and also the hardware and software for the project.

**2.1 Functional Requirements**

For the execution of this project,the graphics has been written in C language and many simple user defined functions are used. The project requires access to OpenGL utility toolkit through the use of the header file “GL /glut.h”. This header file, in addition to the usual header files is needed for the working of the project. For running the program, any basic C running compatible version of Code Blocks or Linux (Ubuntu) based platform is sufficient.

**2.2 Non Functional Requirements**

The software should not accept wrong inputs and produce outputs. This should be meticulously taken care of. It should use memory as less as possible. For this, dynamic memory allocation is preferable to accomplish this task. The project needs to be memory efficient and be able to produce quality output at the same time.

**2.3 Hardware Requirements**

The minimum/recommended hardware configuration required for developing the proposed software is given below:

* INTEL dual core and above compatible systems.
* 2GB RAM.
* Approximately 200MB free space in the hard disk.
* Hard disk access time must be less than 20milliseconds.

**2.4 Software Requirements**

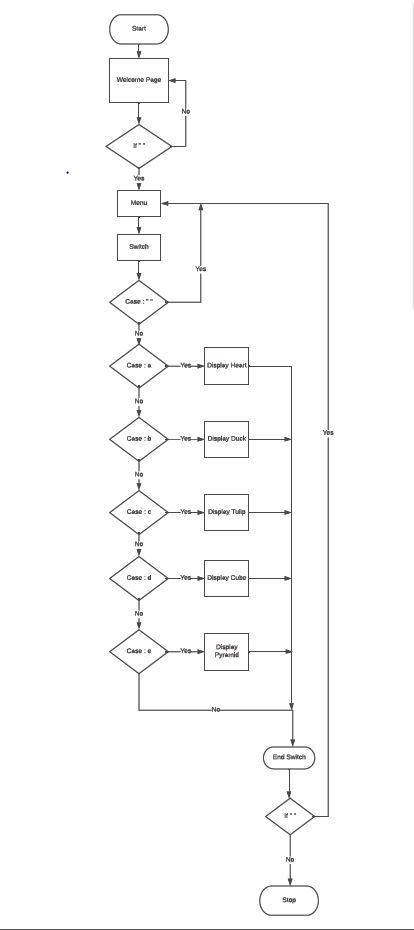
* OpenGL libraries.
* C language compiler.
* Windows
* NetBeans

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 Flowchart**

Flowchart is a common type of chart that represents an algorithm or process showing the steps as boxes of various kinds, their order by connecting these with arrows.



**Figure 3.1.1: Flowchart**

Figure 3.1.1 shows the flowchart of our project. When we run our project, we get the welcome page. When space bar is pressed the menu window appears where pressing key a will open window with the 2D animation of heart, pressing key b will open window with the 2D animation of duck, pressing key c will open window with the 2D animation of tulip, pressing key d will open the window with the 3D animation of cube and pressing key e will open the window with the 3D animation of pyramid. Pressing spacebar again will make menu window to appear.

**CHAPTER 4**

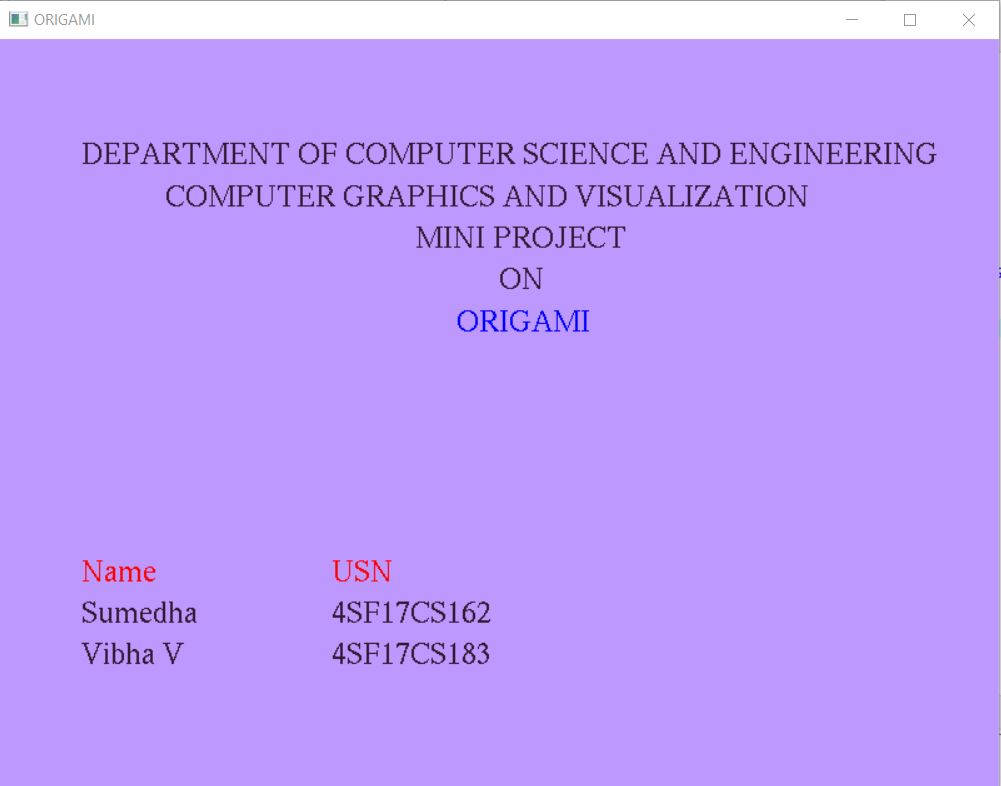
**IMPLEMENTATION**

**4.1 Implementation of OpenGL Built-In functions:**

* **glutCreateWindow(char \*name) –** Creates the window on the 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.
* **glutInitWindowSize(int width, int height) –** It specifies the initial height and width of the window in pixel.
* **glutInitWindowPosition(int x, int y) –** It specifies the initial position of the top left corner of the window.
* **glColor3 [b i f d ub ui] (TYPE r, TYPE g, TYPE b) –** sets the present RGB (or RGBA) colors. Valid types are byte(b), int(i), float(f), double(d), unsigned byte(ub), unsigned int(ui). The maximum and minimum values of the floating-points types are 1.0 and 0.0 respectively.
* **glutInit(&argc,argv) –** It takes arguments.
* **glutKeyBoardFunc(void \*(unsigned char key, int x, int y)func) -** It registers the keyboard callback function, takes the ASCII code of the key pressed and the position of the mouse.
* **glutMainLoop() –** It causes the program to enter tan event-processing loop, It should be the last statement in the main.
* **glutDisplayFunc(void(\*func)(void)) –** Registers the display function that is executed when the window needs to be redrawn.
* **glClear(GLbitfieldmask) –** Clears the specified buffers to their current clearing values. The mask argument is a bitwise logical OR combination of values i.e. GL\_COLOR\_BUFFER\_BIT,GL\_DEPTH\_BUFFER\_BIT.
* **glLoadIdentity() –** Replaces the current matrix with the identity matrix.
* **glTranslatef(GLfloat x,GLfloat y,GLfloat z) -** multiplies the current matrix by a translation matrix.
* **glRotatef(GLfloat angle,GLfloat x,GLfloat y,GLfloat z) -** multiplies the current matrix by a rotation matrix.
* **glClearColor(GLfloatR,GLfloatG,GLfloatB,GLfloatalpha) -** This clears the background color to current clearing values.
* **VoidgluOrtho2D(GLfloatleft,GLfloatright,GLfloattop,GLfloatbotom)-** Sets the ortho as specified values in the function argument.
* **glFlush() –** Forces all previously issued OpenGL commands to begin execution thus guarantying that they complete in finite time.
* **glVertex2f(GLfloat x,GLfloat y) –** specifies a vertex in 2D.
* **glFrustum(GLdouble left,GLdouble right,GLdouble bottom,GLdouble top,GLdouble zNear,GLdouble zFar) -**  function multiplies the current matrix by a perspective matrix.
* **glutTimerFunc(unsigned int msecs, void (\*func)(int value), value) -** registers a timer callback to be triggered in a specified number of milliseconds.
* **glutPostRedisplay() -** marks the current window as needing to be redisplayed.
* **glEnable(GLenum cap) -** functions enable OpenGL capabilities.
* **glDisable(GLenum cap) -**  functions disable OpenGL capabilities.
* **glutBitmapCharacter(void \*font int char) –** Renders the character with ASCII code char at the current raster position using the raster font given in the font.
* **glLineStipple**(**GLint factor, GLushort pattern) -** is used to draw the dotted line.
* **glCullFace(GLenum mode) -** function specifies whether front-facing or back-facing facets can be culled.
* **glBegin(GL\_LINE\_LOOP) –** Draws a hollow diagram. Takes the end points from the glvertex function.
* **glBegin(GL\_POLYGON) –** Draws a filled polygon diagram. Takes the end points from the glertex function.
* **glMatrixMode(GLenum mode) -** specifies which matrix will be affected by subsequent transformations. Mode can be GL\_MODELVIEW,GL\_PROJECTION.
* **glPushMatrix() -** It pushes the matrix stack corresponding to the current matrix mode.
* **glPopMatrix() –** It pops the matrix stack corresponding to the current matrix mode.
* **glEnd() –** Indicates the end of glBegin().
  1. **Implementation of OpenGL User-Defined Functions:**
* **void displayText() –** This function is used to display the project title and team members name and usn.
* **void update1(int value) -** This function is used to update the parameters for displaying cube.
* **void update3(int value) -** This function is used to update the parameters for displaying pyramid.
* **void init1() -** used initialize window for displaying cube and pyramid.
* **void printCharacters(float x, float y, float z, const unsigned char\* str):** To display text on the screen
* **void displayList() –** This function is used to display the menu.
* **void display\_heart() –** This function is used to display the 2D animation of creating a heart step by step.
* **void display\_duck()** - This function is used to display the 2D animation of creating a duck step by step.
* **void display\_tulip() -** This function is used to display the 2D animation of creating a tulip step by step.
* **void display\_cube() -** This function is used to display the 3D animation of creating a cube.
* **void display\_pyramid() -** This function is used to display the 3D animation of creating a pyramid.
* **void stateChange(unsigned char key, int x, int y) –** This function is used to take input from the keyboard.
* **void sequential() –** This function is to decide which window should be displayed.
* **int main(int argc,char \*argv[]) –** This is the main routine of the program.

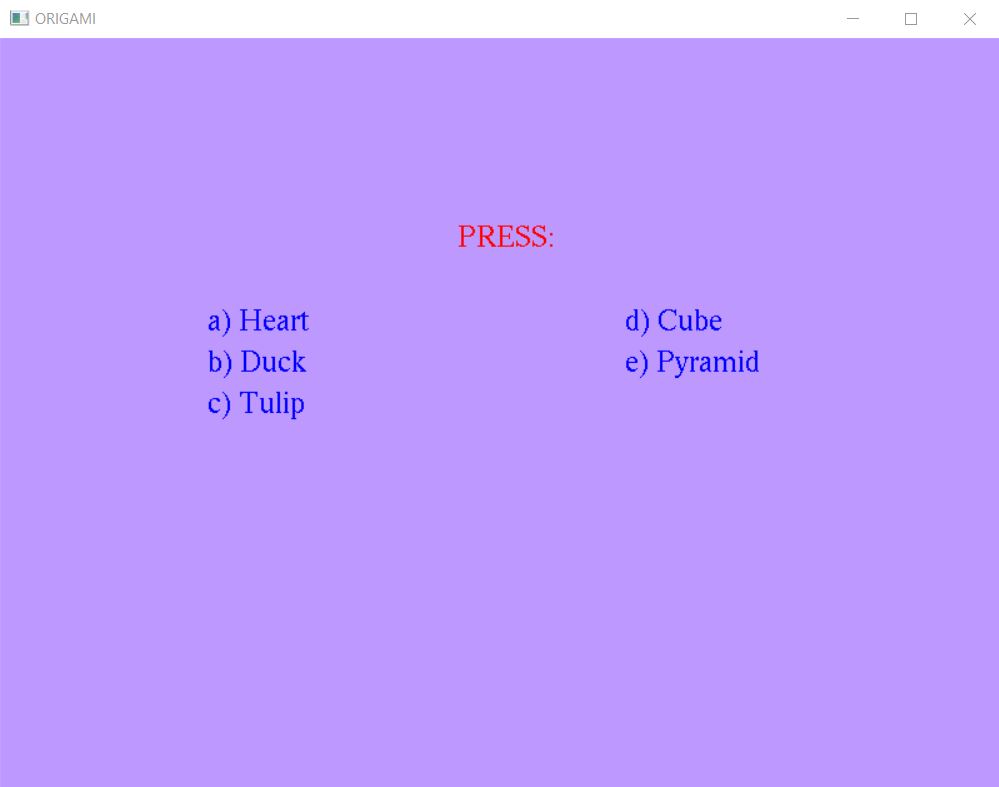
**CHAPTER 5**

**RESULTS**

****

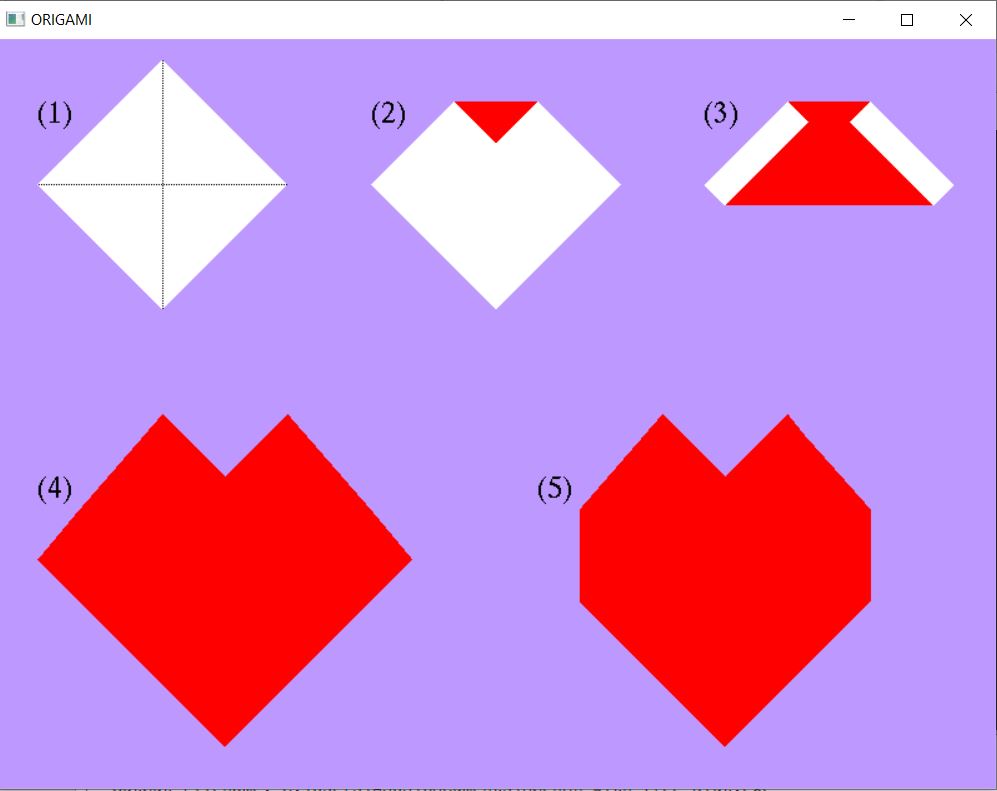
**Figure 5.1: Project title**

Figure 5.1 shows title of the project and team members name. spacebar is to be pressed to move to the next page.

****

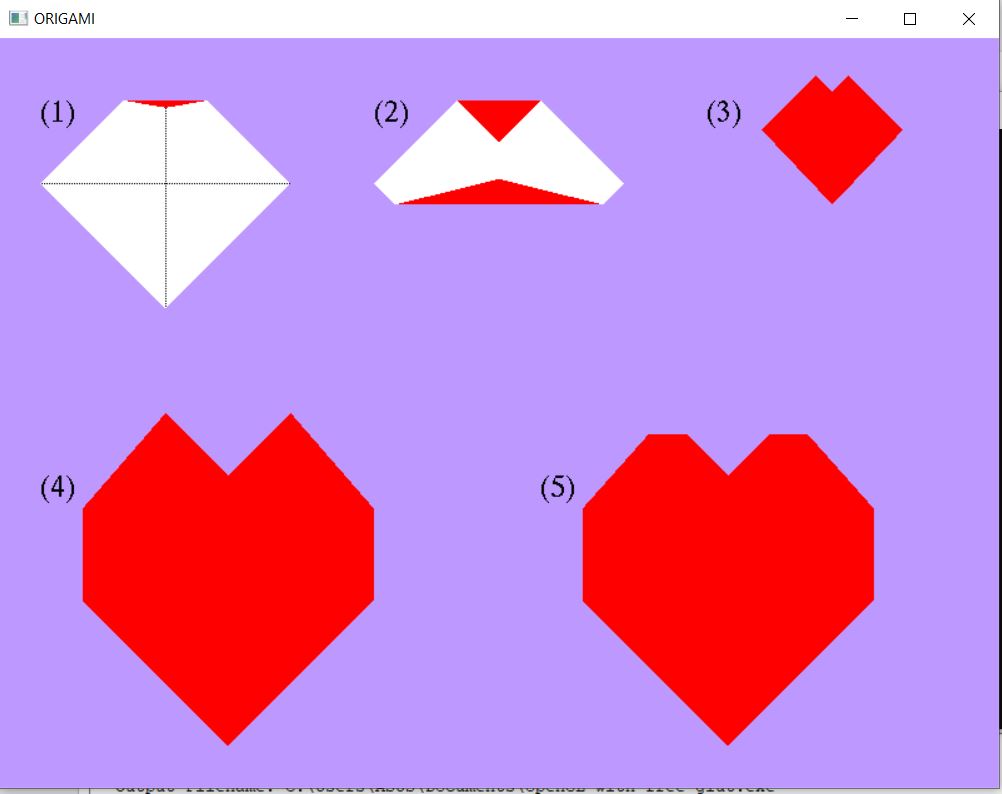
**Figure 5.2: Menu**

Figure 5.2 shows the menu page. Pressing the key will open a window with the corresponding origami.



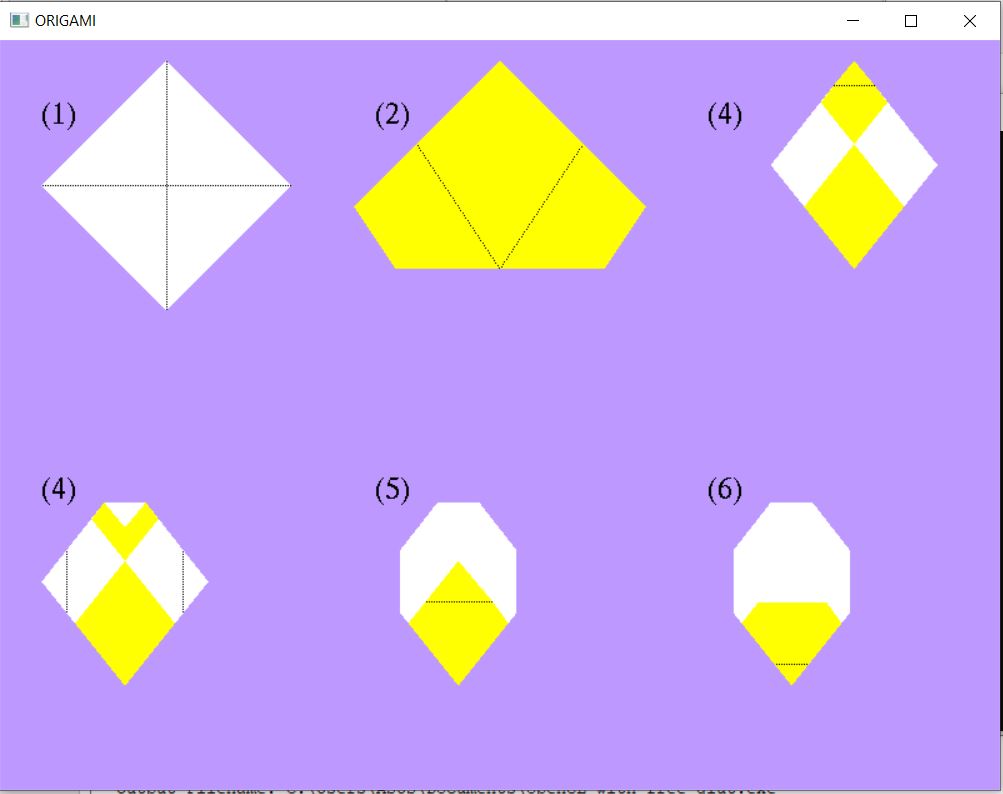
**Figure 5.3: Heart origami.**

Figure 5.3 shows the steps of making a heart origami. When a key is pressed the animation of the corresponding fold is shown.



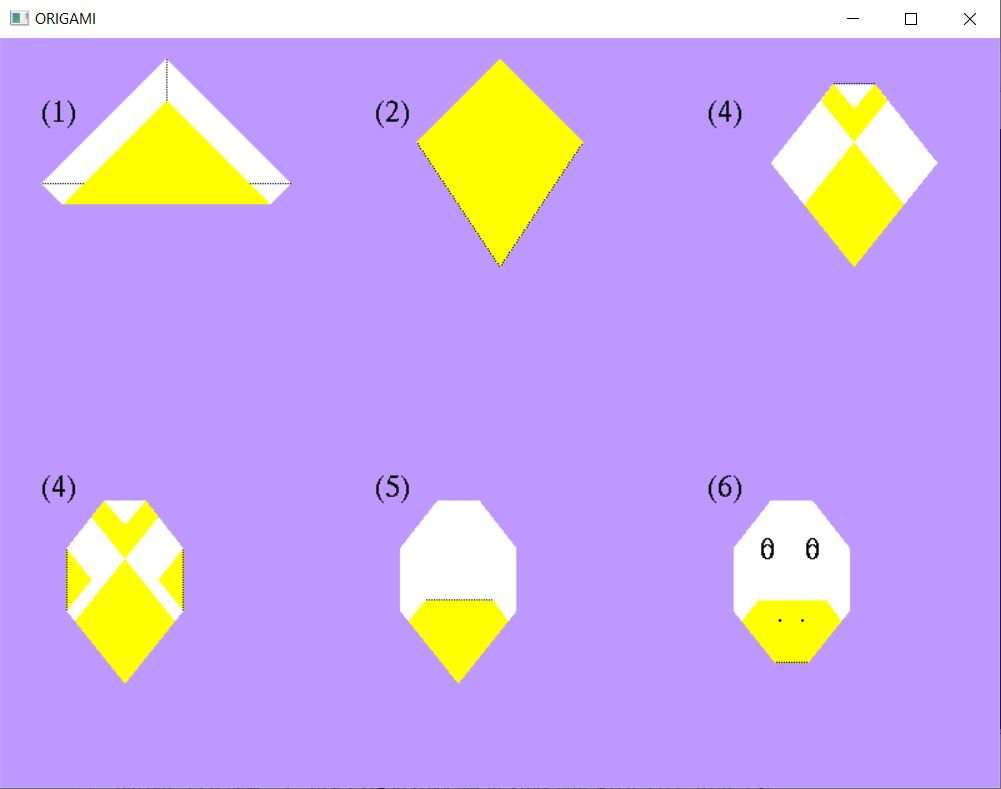
**Figure 5.4: Heart origami after the key press.**

Figure 5.4 shows the origami after the animation.

****

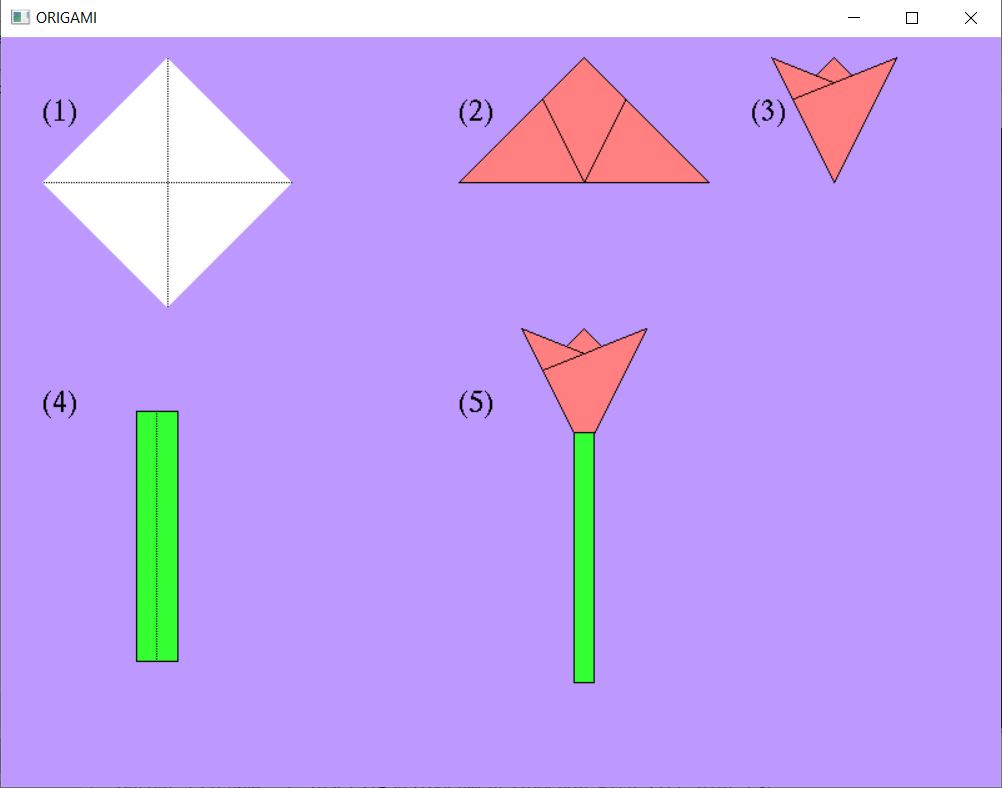
**Figure 5.5: Duck origami.**

Figure 5.5 shows the steps of creating a duck origami. When a key is pressed the animation of the corresponding fold is shown.

****

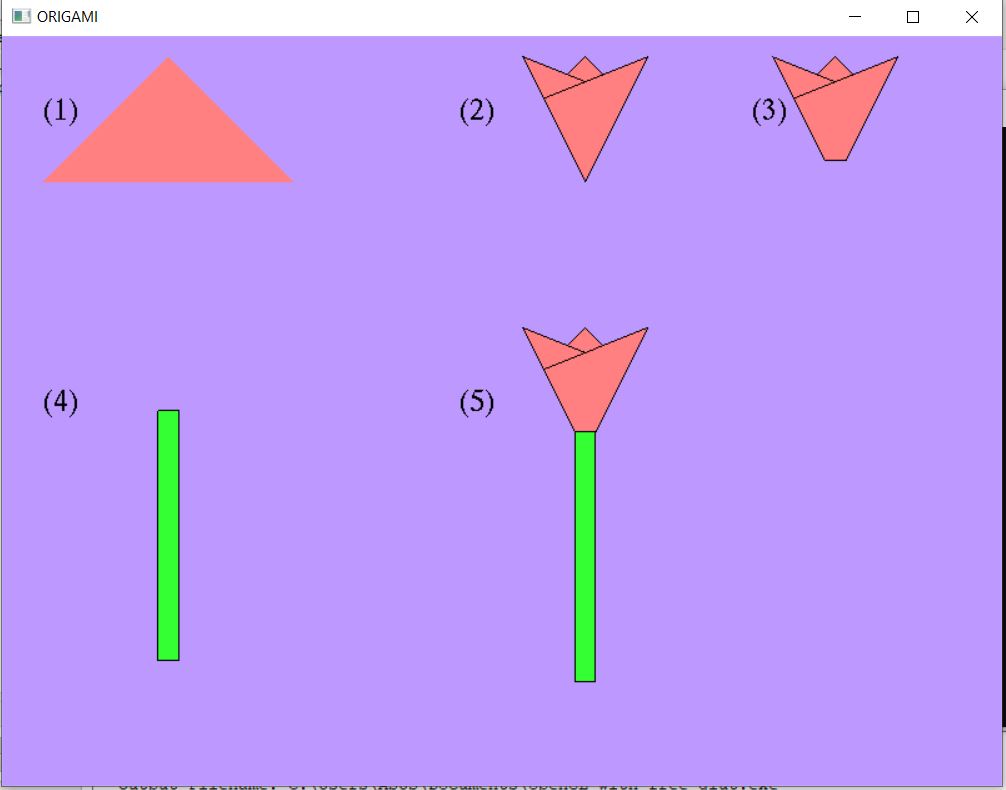
**Figure 5.6: Duck origami after the key press**

Figure 5.6 shows the duck origami after the animation.

****

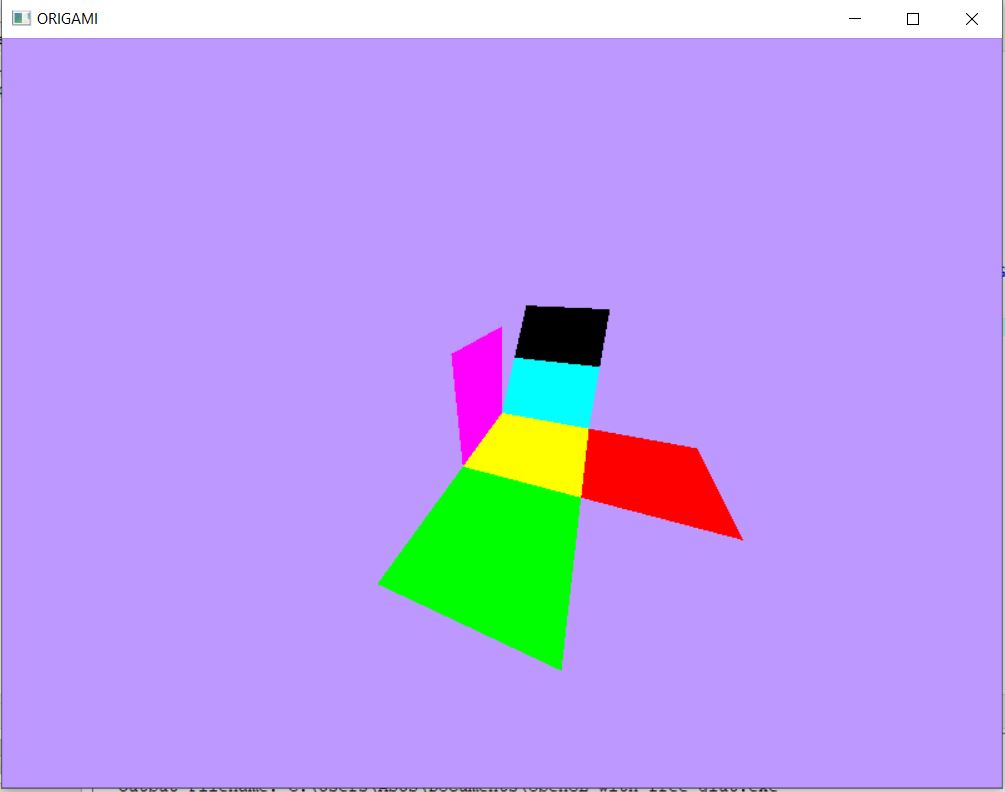
**Figure 5.7: Tulip origami**

Figure 5.7 shows the steps of creating a duck origami. When a key is pressed the animation of the corresponding fold is shown.

****

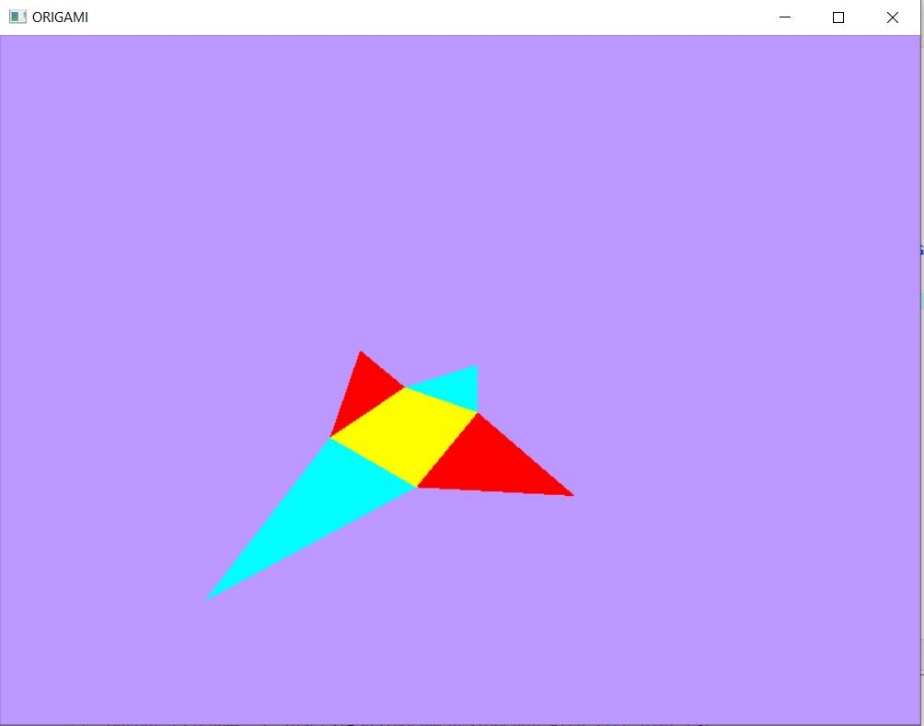
**Figure 5.8: Tulip origami after the key press.**

Figure 5.8 shows the tulip origami after the animation.

****

**Figure 5.9: Cube.**

Figure 5.9 shows the 3d animation of creating a cube.

****

**Figure 5.10: Pyramid**

Figure 5.10 shows the 3D animation of creating a pyramid.

**CHAPTER 6**

**CONCLUSION**

Computer Graphics plays a major role in today’s world where visualization takes the upper hand as compared to text based interactions. It is also significantly proven that people remember the things which they see rather than read or listen. This is largely true as we can see user interfaces more attractive. Computer graphics is a powerful tool in this era of mass media.

Designing and implementing project in graphics is a great experience.“ORIGAMI” is an intermediate level project. The OpenGL is used for all the graphics display, coordinate transformations, viewport transformations and orthogonal projections. Since OpenGL is a platform independent and has more or less like a C runtime open graphics library, so it is very efficient in displaying the graphics easily and clearly.

The combination of computers, networks and the complex human visual system, through computer graphics has led to new ways of displaying information, seeing virtual worlds, and communicating with people and machines.

**REFERENCES**

[1] Edward Angle:*Interactive Computer graphics A Top-Down Approach with OpenGL,* 5th Edition,Addition – Wesley,2008.

[2] James D Foley, andries Van Dam, Steven K Feiner, John F Huges: *Computer Graphics,* Pearson Education.

[3] Donald Hearn and Pauline Baker: *Computer Graphics-OpenGL Version,*2nd Edition, Pearson Education

* **glRasterpos3f(TYPE \* coordinates) –** Specifies the raster position. Parameters are same as glVertex.
* **glutBitmapCharacter(void \*font int char) –** Renders the character with ASCII code char at the current raster position using the raster font given in the font.
* **glBegin(GL\_LINE\_LOOP) –** Draws a hollow diagram. Takes the end points from the glertex function.
* **glMatrixMode(GLenum mode) -** specifies which matrix will be affected by subsequent transformations. Mode can be GL\_MODELVIEW,GL\_PROJECTION.
* **glPushMatrix() -** It pushes the matrix stack corresponding to the current matrix mode.6
* **glPopMatrix() –** It pops the matrix stack corresponding to the current matrix mode.
* **glBegin(GLenum mode) –** It initiates a new primitive of type mode and starts the collection of vertices. Values of mode includes GL\_POINTS, GL\_POLYGON, GL\_TRIANGLE, GL\_POINTS.
* **glEnd() –** Indicates the end of glBegin().

## **4.2 Functions used in “Tale of Lord Ganesha”**

* **void display(void) –** This function is used to display the program.
* **int main(int argc,char \*argv[]) –** This is the main routine of the program.

* **Voidkeys(**unsigned **char,int,int) –** This is the keyboard function to take input.
* **void display1()-**This function is used to display the scene one.
* **void display2()-**This function is used to display scene two with Lord Ganesh and Goddess Parvathi in the scene.
* **void display3()-**This function is used to display the scene in which Lord Shiva cuts Lord Ganesh human head.
* **void display4()-**This function is used to display the scene in which Lord Shiva gets elephants head and inserts it to Lord Ganesh’s body.

**CHAPTER 5**

**RESULTS**

**CHAPTER 6**

**CONCLUSION**

Computer Graphics plays a major role in today’s world where visualization takes the upper hand as compared to text based interactions. It is also significantly proven that people remember the things which they see rather than read or listen. This is largely true as we can see user interfaces more attractive. Computer graphics is a powerful tool in this era of mass media.

Designing and implementing project in graphics is a great experience. “Tale of Lord Ganesha” is an intermediate level project. The OpenGL is used for all the graphics display, coordinate transformations, viewport transformations and orthogonal projections. Since OpenGL is a platform independent and has more or less like a C runtime open graphics library, so it is very efficient in displaying the graphics easily and clearly.

The combination of computers, networks and the complex human visual system, through computer graphics has led to new ways of displaying information, seeing virtual worlds, and communicating with people and machines.

**REFERENCES**

[1] Edward Angle:*Interactive Computer graphics A Top-Down Approach with OpenGL,* 5th Edition,Addition – Wesley,2008.

[2] James D Foley, andries Van Dam, Steven K Feiner, John F Huges: *Computer Graphics,* Pearson Education.

[3] Donald Hearn and Pauline Baker: *Computer Graphics-OpenGL Version,*2nd Edition, Pearson Education