VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAVI, KARNATAKA -590 018

A Mini-Project Report on

"Origami 3D Animation"

Submitted in partial fulfillment for the Computer Graphics Laboratory with Mini-Project (18CSL67) course of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2021-22.

By

Team Code: **CGP2022-C07**

Gagan P	**4MH19CS028**
Manoj R M	**4MH19CS048**

: Under the Guidance of :

Harish H K

Assistant Professor

Department of CS&E

MIT Mysore

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, S.R. Patna Taluk, Mandya Dist-571477.

Accredited By:

2021-22

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

~~ CERTIFICATE ~~

*Certified that the mini-project work entitled "**Origami 3D Animation**" is a bonafide work carried out by **Gagan P** (4MH19CS028) & Manoj R M (4MH19CS048) for the Computer Graphics Laboratory with Mini-Project (18CSL67) of Sixth Semester in Computer Science & Engineering under Visvesvaraya Technological University, Belgavi during academic year 2021-22.*

It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

_____	_____
Signature of Guide	Signature of the HOD
Prof. Harish H K	**Dr. Shivamurthy R C**
Assistant Professor	Professor & HOD
Dept. of CS&E	Dept. of CS&E
MIT Mysore	MIT Mysore

External viva

Name of the Examiners	**Signature with date**

1)……

2) ……

~~~~ ACKNOWLEDGEMENT ~~~~

It is the time to acknowledge all those who have extended their guidance, inspiration, and their wholehearted co-operation all along our project work.

We are also grateful to **Dr. B G Naresh Kumar**, principal, MIT Mysore and **Dr. Shivamurthy R C**, HOD, CS&E, MIT Mysore for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

We wish to place a deep sense of gratitude to all Teaching and Non-Teaching staffs of Computer Science and Engineering Department for whole-hearted guidance and constant support without which Endeavour would not have been possible.

Our gratitude will not be complete without thanking our parents and our friends, who have been a constant source of support and aspirations

Gagan P

Manoj R M

~~~ ABSTRACT ~~~

The aim of this project is to develop a 3D animation using graphics package which supports basic operations and includes creating objects using geometric entities.

This project is implemented using OpenGL libraries underlying all basic concepts of computer graphics. The project shows the application of OpenGL in the paper boat animation at different sets of stages.

This animation includes the square paper at the start divided into 16 similar triangle. The complete animation is shown step by step. The user can change the angle of view by using i, j and k keyboard keys.

~~~~~ CONTENTS ~~~~~

Chapter 1: Introduction

1.1 Aim of the Project:

The aim of this project is to develop a 3D animation using graphics package which supports basic operations and includes creating objects using geometric entities.
This project is implemented using OpenGL libraries underlying all basic concepts of computer graphics.

The project shows the application of OpenGL in the paper boat animation at different sets of stages. This animation includes the square paper at the start divided into 16 similar triangles. The complete animation is shown step by step. The user can change the angle of view by using i, j and k keyboard keys.

1.2. Objective of the Project:

The objective of this program is to transform a flat square sheet of paper into a finished sculpture through folding and sculpting techniques. Showing how to make the most well-known origami boat with this quick and easy step-by-step.

Origami comes from the Japanese. The word means the art of folding paper. "Ori" means "folding," and kami means "paper." In modern usage, the word "origami" is used as an inclusive term for all folding practices.

The goal is to transform a flat square sheet of paper into a finished sculpture through folding and sculpting techniques.

1.3 Outcome of the Project:

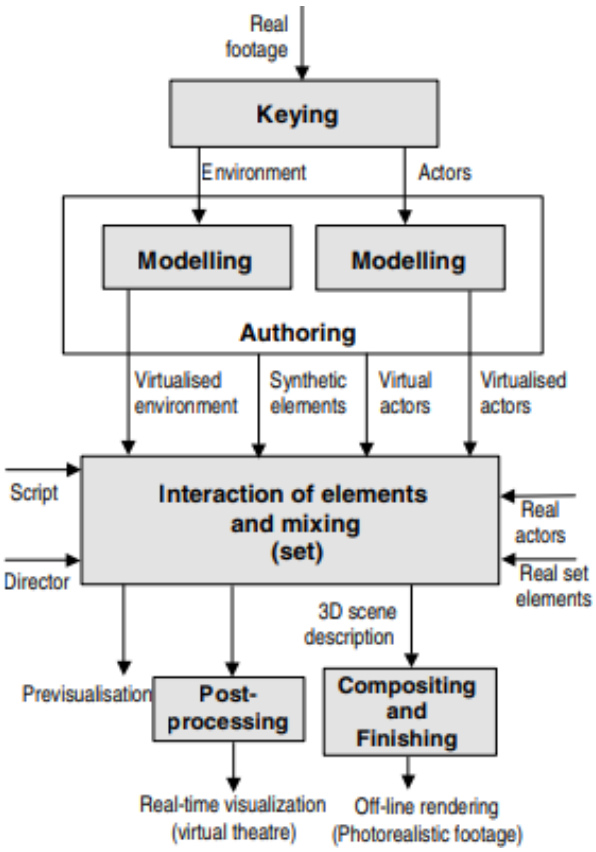
Origami, a geometric network of identic, regular, tridimensional origami, enjoys great favor. Origami, however, is an extremely popular art expression of creativity and phantasy; new shapes has been conceived with decorative purpose as well as reproduction of the reality and creative expression of taste.

Origami gives a positive impulse to the interactive language exchanges because children participate not only at a cognitive level, but also at a multi-sensoric and emotive one.

Another relevant strategy that origami is able to develop is work organizing: the art of origami itself is made of consecutive steps and requires a scheme of things that comes from the necessarily step-by-step procedure.

Chapter 2: Design And Implementation

2.1. Flowchart:



2.2. OpenGL API's Used with Description:

glutInit(&argc,argv)

glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.

Argc: A pointer to the program's unmodified argc variable from main. Upon return, the value pointed by argc will be updated, because glutInit extracts any command line options intended for the GLUT library.

display():

This function creates and translates all the objects in a specified location in a particular order and also rotates the objects in different axes.

glClear(GL_COLOR_BUFFER_BIT
); glFlush();

glMatrixMode ():

Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL_MODELVIEW, GL_PROJECTION, and GL_TEXTURE. The initial value is GL_MODELVIEW.Additionally, if the ARB_imaging extension is supported GL_COLOR is also accepted.

MainLoop():

This function whose execution will cause the program to begin an event processing loop.

PushMatrix():

Save the present values of attributes and matrices placing ,or pushing on the top of the stack.

PopMatrix():

We can recover them by removing them from stack.

Translated();

In translate function the variables are components of the displacement vector.

main():

The execution of the program starts from this function. It initializes the graphics system and includes many callback functions.

glLoadIdentity ():

glLoadIdentity replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix

1 0 0 0
0 1 0 0
0 0 1 0
0 0 0

Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0,0). width, height.Specify the width and height of the viewport. When a GL context is first attached to a window, width and height are set to the dimensions of that window.

gluPerspective ():

gluPerspective 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. For example, aspect = 2.0 means the viewer's angle of view is twice as wide in *x* as it is in *y*. If the viewport is twice as wide as it is tall, it displays the image without distortion.

The matrix generated by gluPerspective is multiplied by the current matrix, just as if glMultMatrix were called with the generated matrix. To load the perspective matrix onto the current matrix stack instead, precede the call to gluPerspective with a call to glLoadIdentity.

glBegin():

glBegin and glEnd delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies in which of ten ways the vertices are interpreted. Taking n as an integer count starting at one, and N as the total number of vertices specified.

GL_POINTS:

Treats each vertex as a single point. Vertex n defines point n. N points are drawn.

glRotate():

glRotate produces a rotation of angle degrees around the vector x y z . The current matrix (see glMatrixMode) is multiplied by a rotation matrix with the product replacing the current matrix, as if glMultMatrix were called .

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.An implicit glFlush is done by glutSwapBuffers before it returns. Subsequent OpenGL commands can be issued immediately after calling glutSwapBuffers, but are not executed until the buffer exchange is completed. If the layer in use is not double buffered, glutSwapBuffers has no effect.

glutPostRedisplay():

Mark the normal plane of *current window* as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane. Multiple calls to glutPostRedisplay before the next display callback opportunity generates only a single redisplay callback. glutPostRedisplay may be called within a window's display or overlay display callback to re-mark that window for redisplay.Logically, normal plane damage notification for a window is treated as a glutPostRedisplay on the damaged window. Unlike damage reported by the window system, glutPostRedisplay will *not* set to true the normal plane's damaged status.

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.

Because any GL program might be executed over a network, or on an accelerator that buffers commands, all programs should call glFlush whenever they count on having all of their previously issued commands completed. For example, call glFlush before waiting for user input that depends on the generated image.

glColor3f(GLfloat red,GLfloat blue,Glfloat green):

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

glutKeyboardFunc (void (*func) (unsigned char key, int x, int y)):

Sets the current window's callback function for when a keyboard key is pressed. The x and y values passed to the callback function are the mouse cursor's pixel coordinates at the time of the key press.

glutMainLoop():

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.

glutReshapeFunc (void (*func) (int width, int height)): Sets the resize callback function for the current window. The reshape callback is triggered when a window is reshaped. A reshape callback is also triggered immediately before a window's first display callback after a window is created or whenever an overlay for the window is established.

The width and height parameters of the callback specify the new window size in pixels. Before the callback, the current window is set to the window that has been reshaped. If a reshape callback is not registered for a window Or NULL is passed to glutReshapeFunc.

glutTimerFunc(unsigned int msecs, void (*func)(int value), value): It registers a timer callback to be triggered in a specified number of milliseconds. The value parameter to the timer callback will be the value of the value parameter to glutTimerFunc. Multiple timer callbacks at same or differing times may be registered simultaneously. The number of milliseconds is a lower bound on the time before the callback is generated. GLUT attempts to deliver the timer callback as soon as possible after the expiration of the callback's time interval.There is no support for canceling a registered callback. Instead, ignore a callback based on its value parameter when it is triggered.

2.3. Source Code:

Headers:

```
#include<windows.h>
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<GL/glut.h>
```

Timer Function:

```
void timer(int iunused)
{
   glutPostRedisplay();
   glutTimerFunc(30,timer,0);
}
```

Display Function:

```
void display()
{
   glClearColor(0.0,1.0,0.0,1.0);
   glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
      if (displayFrontScreen==1)
      { displayFrontScreen=0;
   }
   displayStage1();

      if(stage3_mode == 1)
      {
         rotateWorld(stage3_mode);
      }

  if(b0==0)
 {
      line_mid = (line_mid + 1);
      if(line_mid>179){
     b0=1;
     state1=1;
      }
```

```
        else{state3=1;}
    }
     if(b0==1 && b1==0)
    {
     line_perp = (line_perp + 1);
     if (line_perp >180)
        {
           b1=1;
           state2=1;
        }
     else {state3=1;}
    }
    if(state3>0)
    {
     if(b1==1 && b2==0)
     {
        turn1 = (turn1 + 1);
        if(turn1>180)
           {
              b2=1;
              stage=2;
           }
     }
    }
     if(stage==2)
     {
        if(b2==1 && b3==0 && x1>0)
        {

  //stage3_mode=0;
              z=z+0.2;
              x1=x1-0.2;
              x2=x2+0.2;
              line_perp = (line_perp + 3);
                 if (line_perp >170) {
                    state2 = 1;
```

```
                    //b3=1;
                }
                if (state1 &&state2)
                  { state3 = 1;
                }
        }
          else
        {
          b3=1;
        }
        Sleep(200);
        displayStage2(b3);
    }
    if(b3==1 && b4==0 && valy<=0.8)
    {
            valy +=.2;
            valz = computeZ(valy);
            transvalx += .2;
            transvaly+= .007;
            displayStage3();
            Sleep(200);
             valy +=.2;
            valz = computeZ(valy);
            transvalx += .2;
            transvaly+= .007;
            displayStage3();
            Sleep(200);
             valy +=.2;
            valz = computeZ(valy);
            transvalx += .2;
            transvaly+= .007;
            displayStage3();
            Sleep(200);
    }
    glutSwapBuffers();
}
```

Keyboard Function:

```
void keys(unsigned char key, int x, int y)
{
    switch(key){
          case 'i':
            stage3_mode = 1;
            world_angles_true[0] = 1;
            world_angles[0] += 1;
            glutPostRedisplay();
            break;
         case 'j':
                stage3_mode = 1;
            world_angles_true[1] = 1;
            world_angles[1] += 1;
            glutPostRedisplay();
            break;
         case 'k':
                stage3_mode = 1;
            world_angles_true[2] = 1;
            world_angles[2] += 1;
            glutPostRedisplay();
            break;
         case  'z':
         secondInit();
         glutPostRedisplay();
         glFlush();
         break;
          default: stage3_mode = 0;
            break;

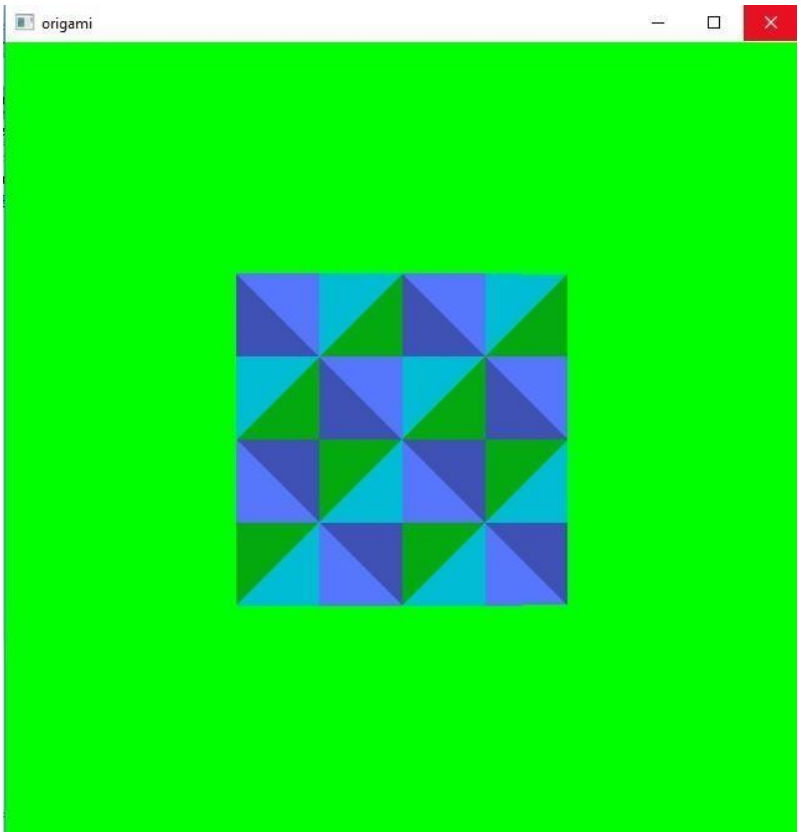
    }
}
```

Main Function:

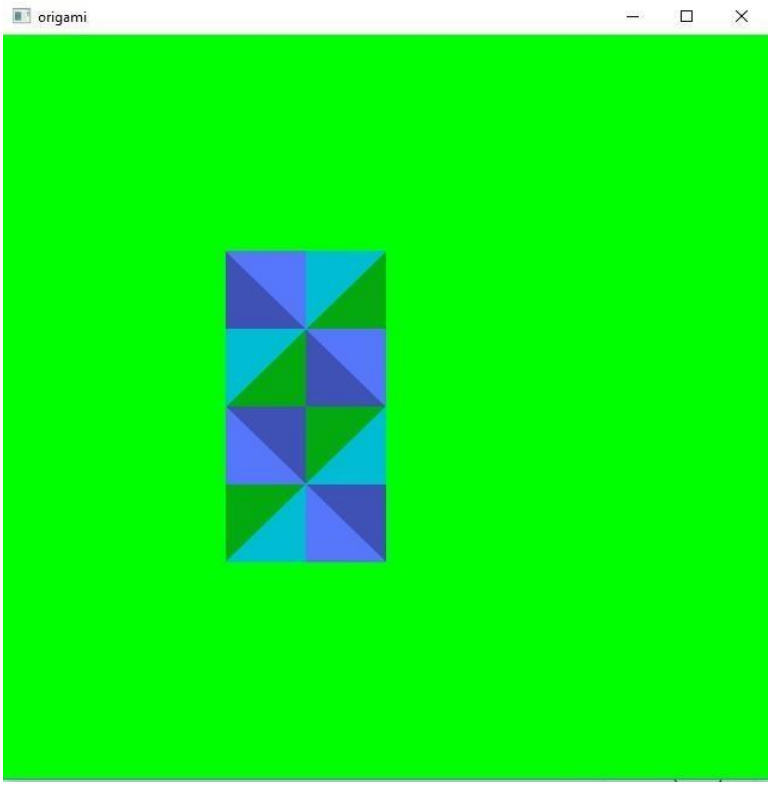
```
int main(int argc,char **argv)
{
  printf("\n\n\t\t\t     ********* ORIGAMI 3D ANIMATION
  *********\n"); printf("    \n\t\t\t\t ANIMATION OF A PAPER
  BOAT\n"); printf("\n\n\n\n\n\n\n\n\n\t\t\t\t MADE BY---\n");
  printf("\t\t\t\t\t GAGAN P(4MH19CS028) \n \t\t\t\t\t MANOJ R M(4MH19CS48)\n");
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_RGB|GLUT_DEPTH|GLUT_DOUBLE);
  glutInitWindowPosition(50,50);
  glutInitWindowSize(640,640);
  glutCreateWindow("origami");
  glEnable(GL_DEPTH_TEST);
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutKeyboardFunc(keys);
  timer(0);
  glutMainLoop();
}
```

Chapter 3: Result and Analysis

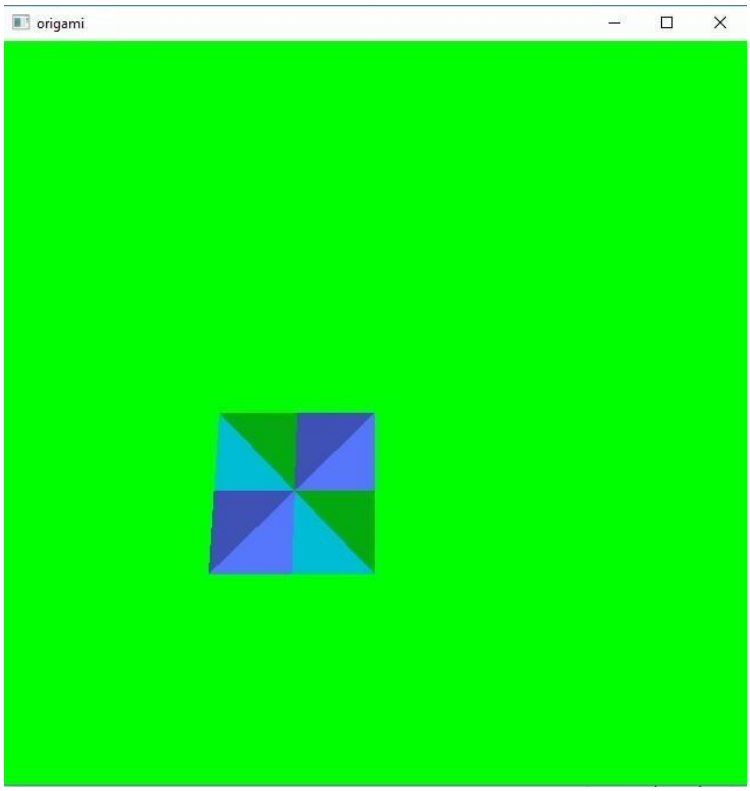
3.1 Snapshots:



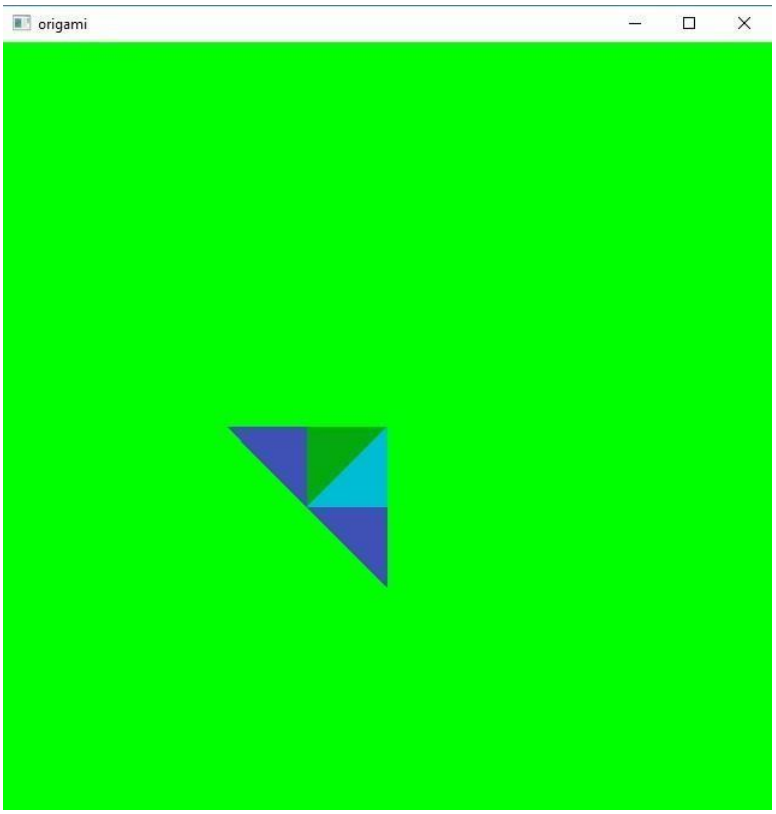
Initial Paper



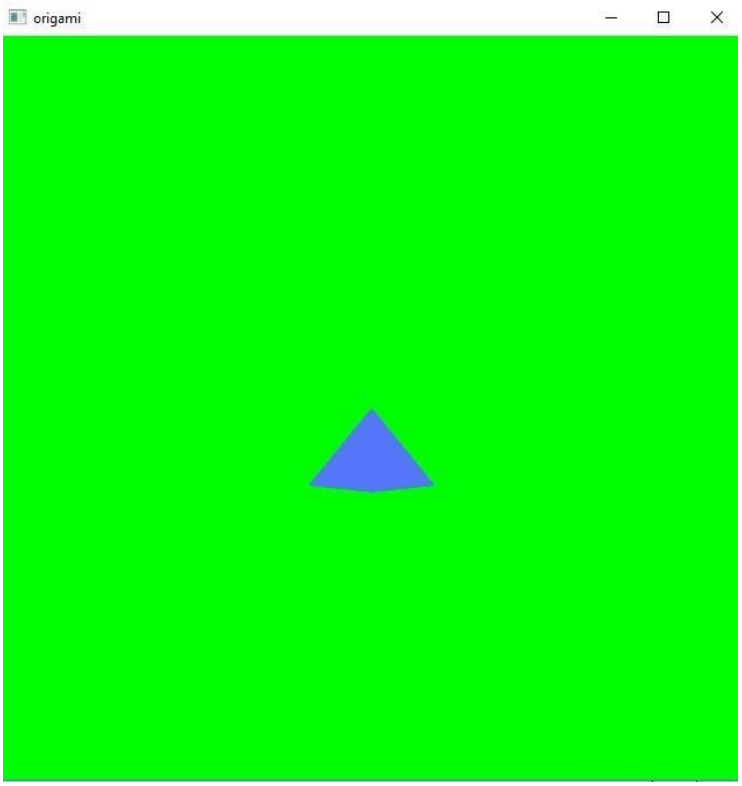
First Fold



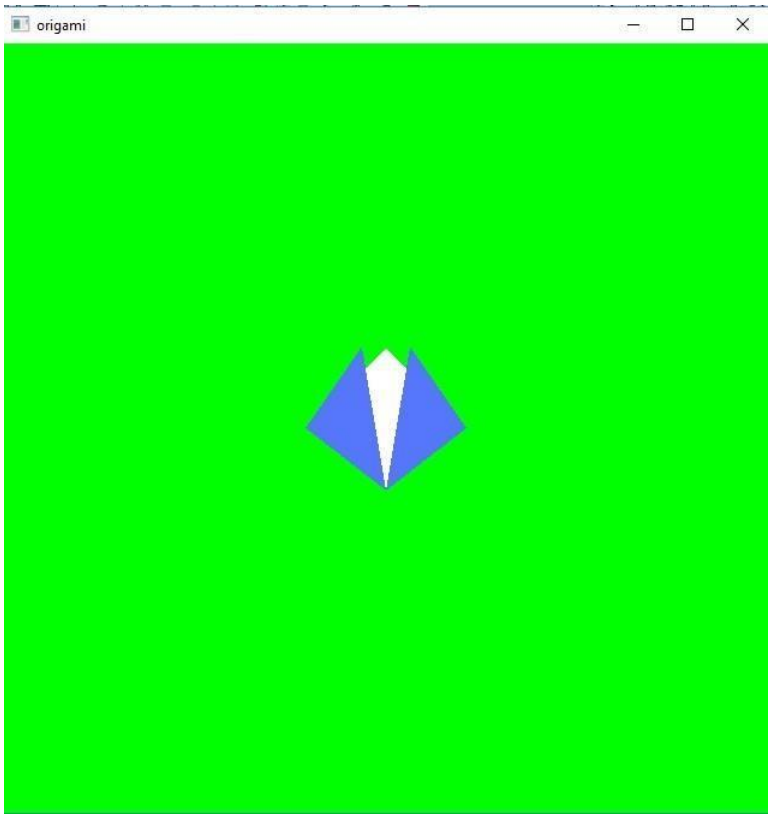
Second Fold



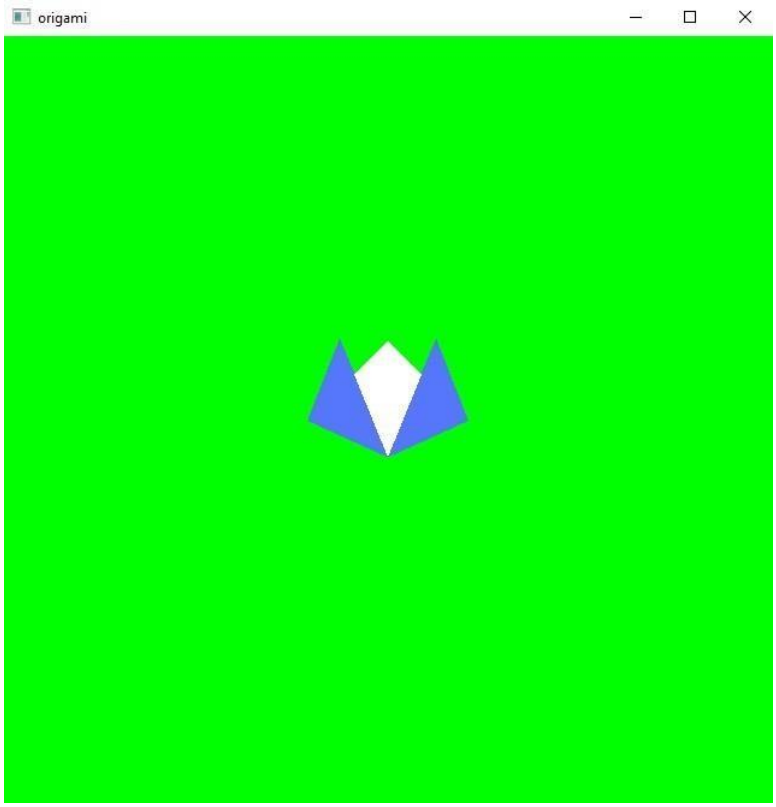
Third Fold



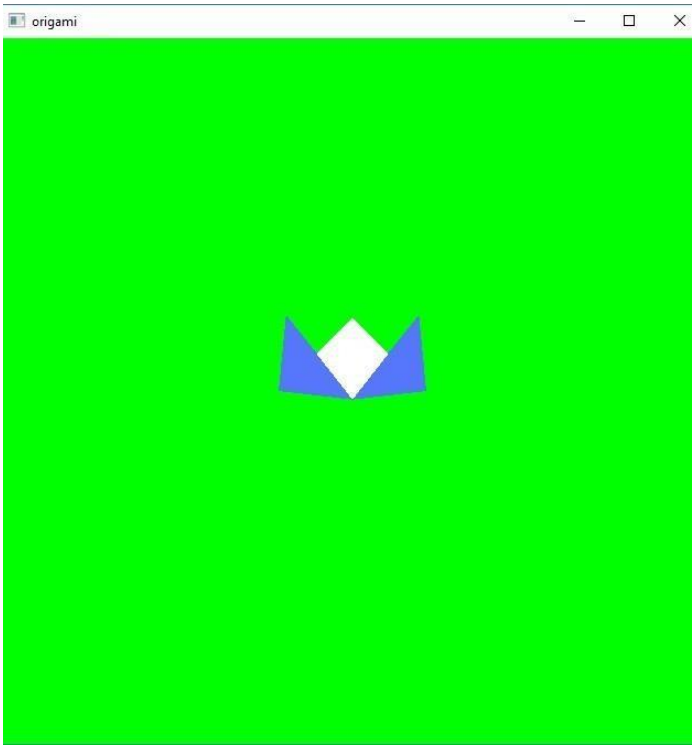
Transformatory Phase



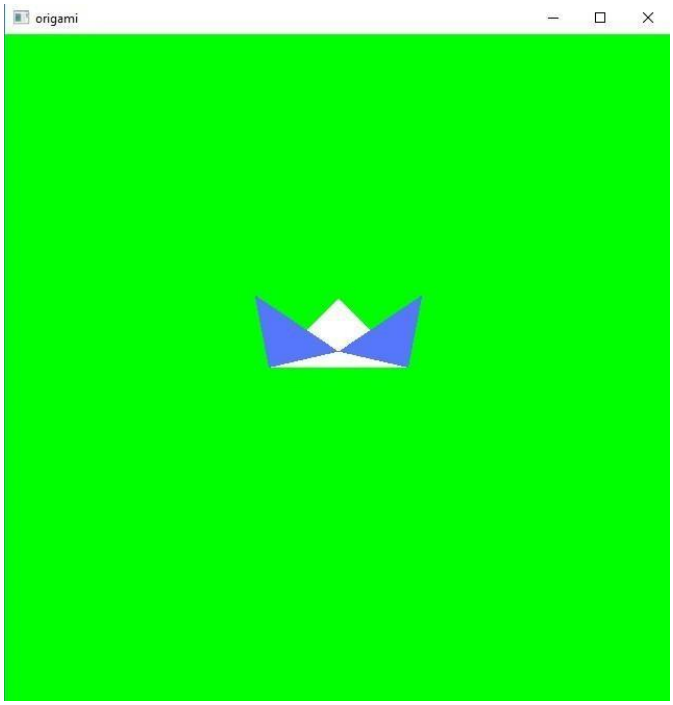
Boat Unfolding-Phase 1



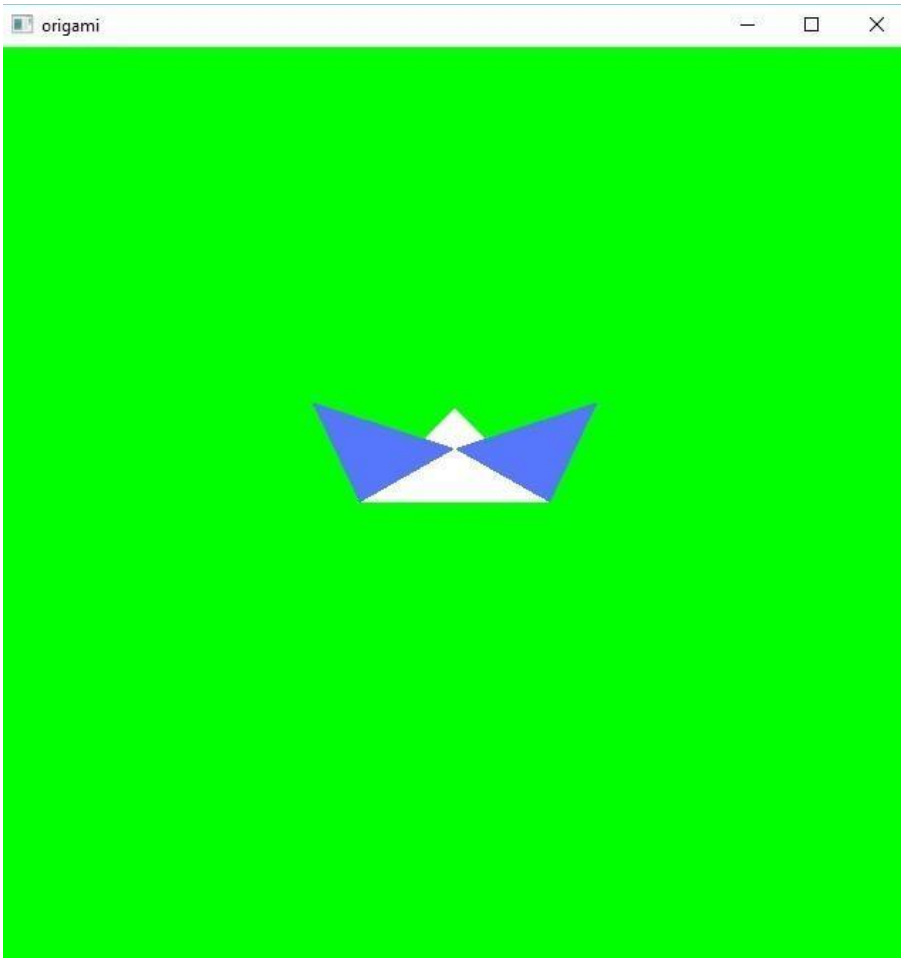
Boat Unfolding-Phase 2



Boat Unfolding-Phase 3



Boat Unfolding-Phase 4



Final Boat

3.2. Discussions:

Metric reconstruction of all the surfaces that describe the object's geometry, plus modelling of the reflectance map of such surfaces and of the illumination.

Among the methods under study are:

Volumetric techniques based on the evolution of multi-resolution level sets, driven by several sources of information, including texture mismatch, feature proximity and extremal boundary deformation.

The methods are topology-independent, robust and fast. They produce models of scalable complexity, and do not necessitate background segmentation.

Chapter 4: Conclusion and Future Work

4.1. Conclusion:

The ORIGAMI project exhibits a mix of scientifically challenging goals and extremely pragmatic application scenarios, with measurable performance in terms of cost effectiveness.

In fact, one peculiarity of this project is in the fact that the quality of the results is not a goal but a constraint. The goals of ORIGAMI led us to the choice of the project name.

In fact, origami is the art of paper folding. The word is Japanese, literally meaning "to fold" (oru) "paper" (kami). In fact, it is more than that, as it gives life to a flat world.

This project has helped me to understand the different concept used in Computer Graphics and Visualization lab.

The use of translation and rotation throughout the context of the program has helped in better understanding the Model View transformations and Matrix transformations in general.

4.2. Future Enhancement:

The practice and study of origami encapsulates several subjects of mathematical interest. For instance, the problem of flat-foldability (whether a crease pattern can be folded into a 2-dimensional model) has been a topic of considerable mathematical study.

A number of technological advances have come from insights obtained through paper folding. For example, techniques have been developed for the deployment of car airbags and stent implants from a folded position.

The problem of rigid origami ("if we replaced the paper with sheet metal and had hinges in place of the crease lines, could we still fold the model?") has great practical importance. For example, the Miura map fold is a rigid fold that has been used to deploy large solar panel arrays for space satellites.

Origami can be used to construct various geometrical designs not possible with compass and straightedge constructions. For instance paper folding may be used for angle trisection and doubling the cube.

ORIGAMI intends to develop advanced technology to perform a seamless mixing of real and virtual content coming from both live action and off-line footage. The interaction between real actors and virtual actors (computer animations that are often equipped with a certain intelligence and autonomy) constitutes the “live” content. All the elements of the environment that do not react to the live action are considered part of the off-line content. In fact, such elements are obtained through the “virtualisation” (digital modelling) of off-line footage of a real environment, which may be digitally altered and/or enriched with synthetic elements (extended set). The offline content is usually made of static or quasi-static elements, but it could also include passive “actors”, which are dynamic elements that are not expected to react to the live action.

Chapter 5: References

1. Interactive Computer Graphics A Top-Down Approach with OpenGL- Edward Angel, 5th Edition, Addison-Wesley, 2008

2. Computer Graphics Using OpenGL – F.S Hill, Jr 2nd Edition, Pearson Education, 2001.

3. Computer Graphics: Principles and Practice in C: by Addison Wesley.