Tribhuwan University

Institute of Engineering

Pulchowk Campus



**THE DRAGON OF PHAROS**

A Project Report

On

Computer Graphics

Date: 2071-12-12

**Submitted By: Submitted to:**

Department of Electronics and Computer Engineering

Bishal Heuju: 069BEX409

Bishal Lakha: 069BEX410

Dipkamal Bhusal: 069BEX414

**Acknowledgement**

We would like to express deepest gratitude to the Department of Electronics and Communication Engineering, IOE Pulchowk campus for providing us an opportunity to work on a graphics project, which is likely to be very helpful for shaping our knowledge and career.

We would also like to express our deepest thanks to our subject teacher Dr. Basanta Joshifor his suggestions and influences.

**Contents**

1. Objectives
2. Introduction
3. Methodology
4. Implementation
5. Block diagram
6. Results
7. Problem Faced and Solutions
8. Limitations
9. Conclusion
10. References

**Objectives**

1. To understand the basic principles needed to design, use and understand computer graphic system
2. To implement the graphics algorithms in various applications
3. To understand and implement the algorithms studied in computer graphics course this semester
4. To create 3D graphics model of a dragon and a lighthouse.
5. To create a 3D environment.
6. To color and create lighting effects for the objects.
7. To render the 3D models on a computer screen.

.

**Introduction**

Computer graphics have become a powerful tool for the rapid and economical production of pictures. The use of computer graphics is so widespread that there is virtually no area in which graphical displays cannot be used. Its major applications are:

1. Computer aided design: a major use of computer graphics is in design process particularly in engineering and architectural system but now almost all products are computer designed.
2. Presentation graphics: Computer graphics are also used to produce illustrations for reports or generate slides or transparencies for projectors.
3. Computer art: Computer graphics are widely used in both fine and commercial art applications.
4. Education and training: Computer generated models of physics, finance and economics are often used a s educational aids.
5. Visualizations: Computer graphics are used to analyze large amounts of information to study the behavior of certain processes.
6. Image processing: Computer graphics are used on image processing to modify or interpret existing pictures.
7. Graphical user interfaces:

The Dragon of Pharos is thus application of all the graphics algorithms and processes we studied in the syllabus. This project contains three independent parts: a dragon, a light house and the environment displayed as one to match the title. All the objects are created in 3D plane using object oriented approach.

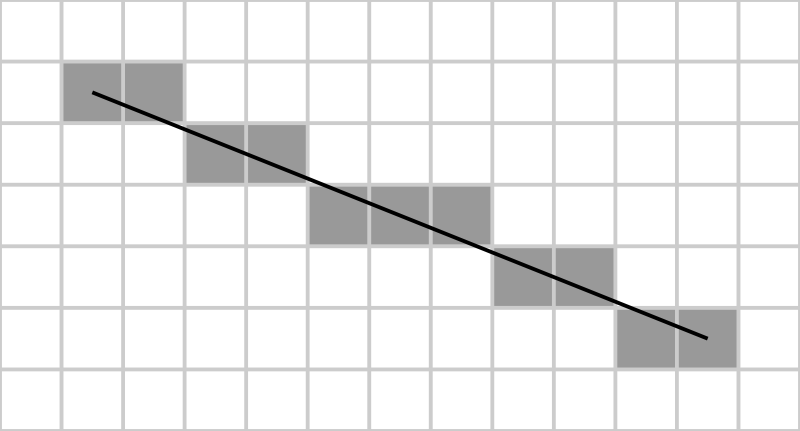
**Methodology**

The graphics image is created using C++ which allows us to use Object Oriented Programming (OOP) and SDL2.0 for graphics purpose.

Following were the algorithms used in the project:

1. **Bresenham’s line algorithm for wireframe:**

Bresenham’s line algorithm is an algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It is commonly used to draw lines on a computer screen as it uses only integer addition, subtraction and bit shifting. , all of which are very cheap operation in standard computer architecture.



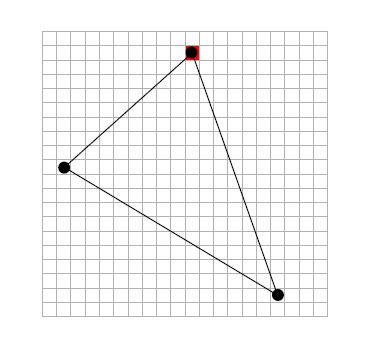
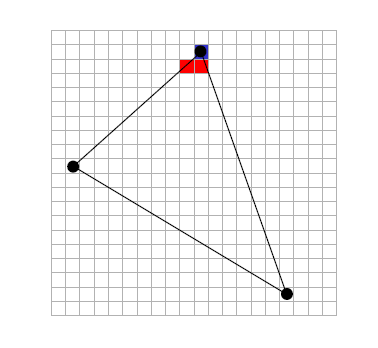
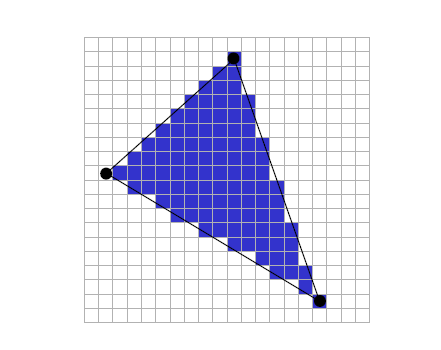
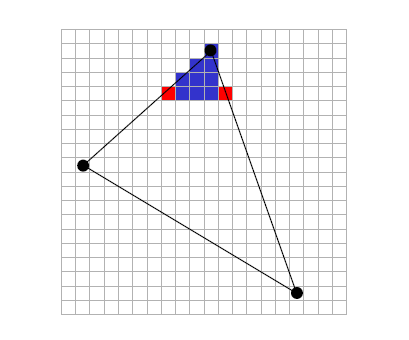
**ALGORITHM:**

1. Input the two endpoints and store the left endpoint in (X0, Y0)
2. Load (X0, Y0) into the frame buffer that is plot the forst point.
3. Calculate the constants del(x), del(y), and 2del(y)-2del(x) and obtain the starting value for the decision parameter as: p0=2del(y)-del(x)
4. At each x, along the line starting at k=0, perform the following tests. If pK< 0, the next point is (xK+1,yK) and

pK+1=pK+2del(y)

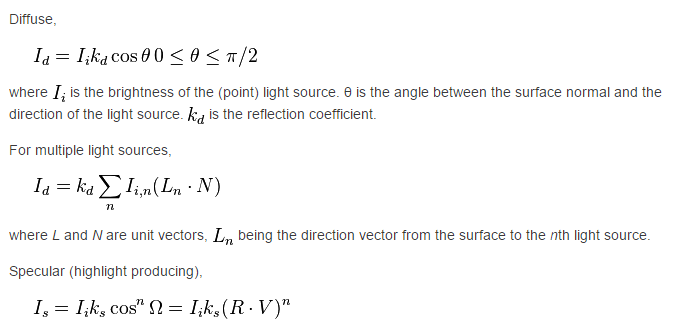
Otherwise the next point to plot is (xK+1,yK+1) and pK+1=pK+2del(y)-2del(x)

1. Repeat step 4 del(x) times.
2. **Scan line algorithm for filling-Interpolation method:**
3. Walk along edges one scan-line at a time
4. Rasterize spans between edges

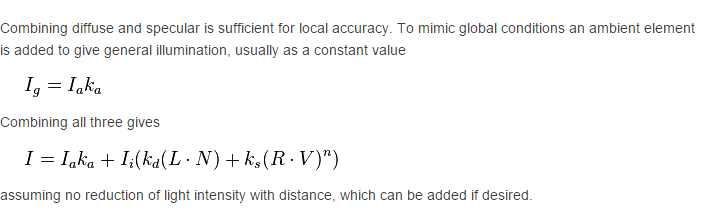


1. **Phong’s shading and illumination model:**

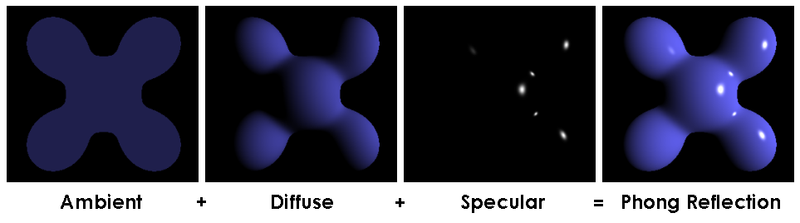
Phong reflection is a local illumination model devised by Bui Tuong Phong and can produce a certain degree of realism in three-dimensional objects by combining three elements-diffuse, specular and ambient light for each considered point on a surface. It has several assumptions-all light arepoints, only surface geometry, only local modeling of diffuse and specular exists, specular color is the same as light color and ambient is a global constant.



Where n indicates the surface reflectivity infinity would indicate a perfect mirror.



This is an empirical model which is not based on physics description of light interaction but instead on physical observation.



(Shi

1. **Visible surface detection-Z buffer method:**

In computer graphics, Z buffering also known as depth buffering is the management of image depth coordinates in three-dimensional graphics. It is one solution to the visibility problem which is the problem of deciding which elements of a rendered scene are visible and which are hidden. When an object is rendered the depth of a generated pixel (z-coordinate) is stored in a buffer (the z buffer or depth buffer). This buffer s usually arranged as two dimensional arrays (x-y) with one element for each screen pixel. If another object of the scene must be rendered in the same pixel, the method compares the two depths and overrides the current pixel if the object is closer to the observer. The chosen depth is then saved to the z-buffer, replacing the old one. in the end the z buffer will allow the method to correctly reproduce the usual depth perception; a close object hides the farther one.

**ALGORITHM:**

1. Initialize the depth buffer and refresh buffer so that for all buffer positions (x,y),

depth(x,y)=0, refresh(x,y)=I(background)

1. For each position on each polygon surface compare depth values to previously stored values in the depth buffer to determine visibility.

Calculate the depth z for each (x,y) position on the polygon

If z>depth (x,y) then set

Depth(x,y)=z, refresh(x,y)=Isurf(x,y)

Where, I(background) is the value for the background intensity

Isurf(x,y) is the projected intensity values for the surface at pixel position (x,y)

**Implementations:**

1. **System Block Diagram:**

Initialize

Do Physics

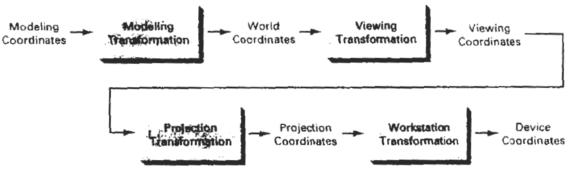
Render and Display

Clear Memory and Exit

Handle Inputs

**Commits:**

1. **General pipelining flowchart:**

****

1. **Classes in the project with their specifications:**

**Basic shape:**

1. Consists of predefined standard objects
2. Objects consists of vertices, faces and face normal

**Camera:**

1. A singleton class
2. Can return view matrix and initialize cameraposition and cameratarget.

**DoP class:**

1. A singleton class
2. Handles all processes

**Dragon class:**

1. Consists everything about dragon
2. Model space of each parts
3. Wireframe

**Environment class:**

1. Consists everything about environment

**Graphics class:**

1. A singleton class
2. Handle graphics
3. Line drawing
4. Pixel plotting
5. Can switch between OPENGl and SDL rendering by defining mode in DoP class; default OPENGL

**IlluminationHandler class:**

1. Handles all the processes related to the light source

**InputHandler class:**

1. A Singleton class
2. Handles all the inputs

**Matrix4 class:**

1. 4D matrix
2. Handles matrix operations
3. Translation, rotation and scaling
4. Generates special matrices (model, view, projection)

**Object class:**

1. Defines object with points

**Pharos class:**

1. Consists everything related to lighthouse

**Vector2D class:**

1. 2D vector co-ordinate system

**Vector 3D class:**

1. 3D vector co-ordinate system

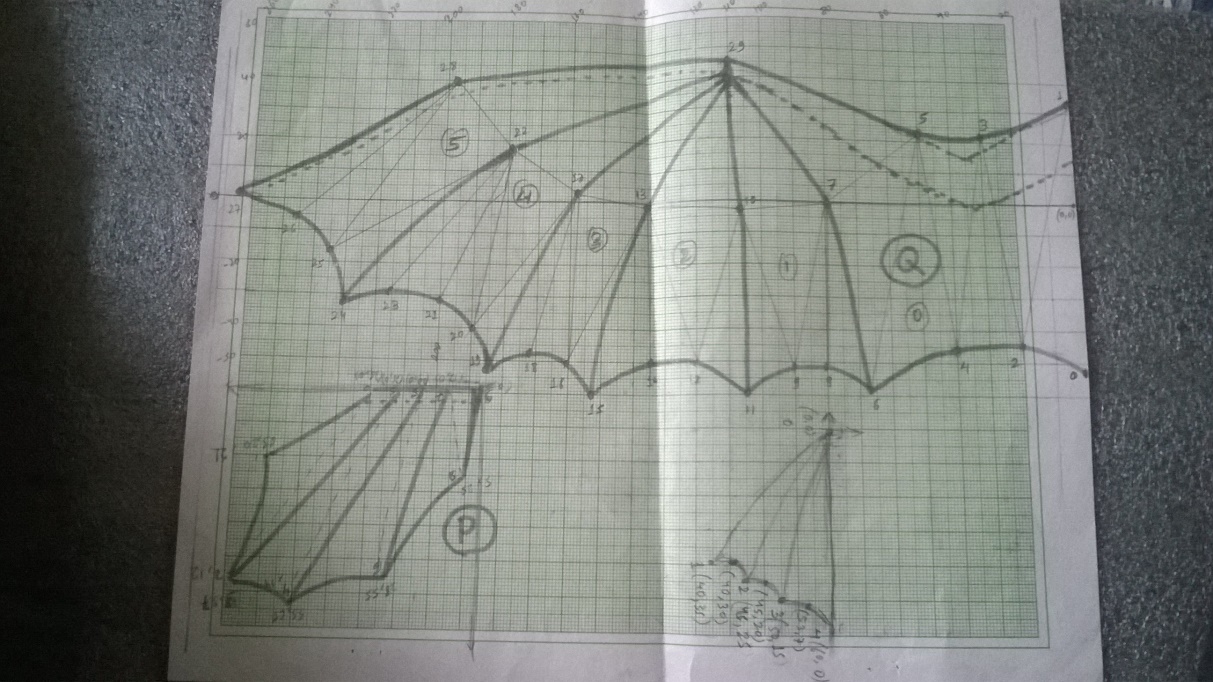
**Vector4D class:**

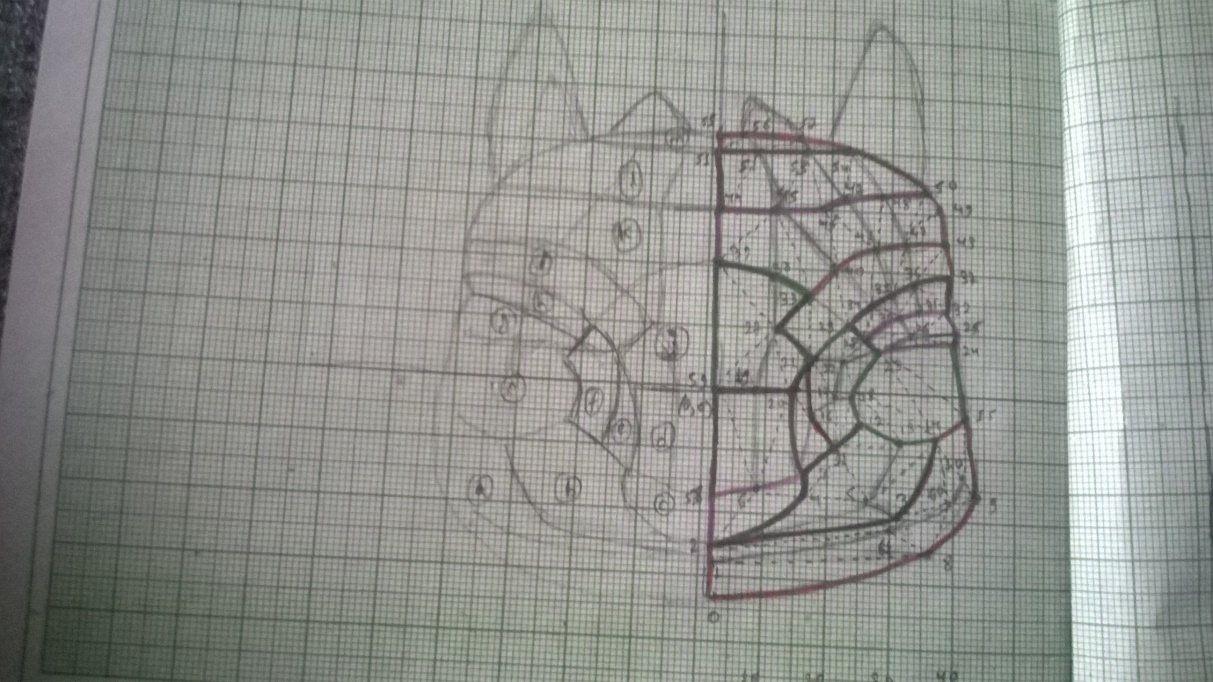
1. 4D vector co-ordinate system
2. Homogeneous 3D with w coordinates

**Results:**

**Problems Faced and Solutions:**

After the onset of project, we have had a few of problems and setbacks. The foremost hurdle was the inadequate resources for developing a 3D environment with certain elements in it. For example: for all the elements of the system (dragon of pharos), we initially drew graph to find out its vertices. It worked well for the ground (a cuboid structure) and pharos (consisting of hemispheres, cylinder and cubes) but it was too difficult to find out the vertices of the entire dragon.





But this was too cumbersome so we loaded an object file of dragon in blender to find out its co-ordinates.

With the start of programming, we also had a lot of errors while implementing the algorithms of graphics in C++ and SDL but with online resources and efficient debugging, we managed to develop a rigid approach to the algorithm implementation.Discussing with our project members was always beneficial for our system development. We could generate ideas and implement it later on.

**Limitations:**

The major limitation of our system is its slow rendering. Although we tried to fix this issue, our program is still slower respect to other implementation of algorithms that could have been possible.

* Excessive CPU load
* Slow rendering
* No proper lighting effects
* No Shadows

**Conclusion:**

This project has been an effective method to implement few of the graphics algorithms we studied in our course. The implementation of those algorithms helped us to realize the graphic system that is used in the world today.

**References:**

* C-Man (Previous C++ Project)
* <http://www.sitepoint.com/series/write-a-3d-soft-engine-from-scratch/>
* SDL Development programming