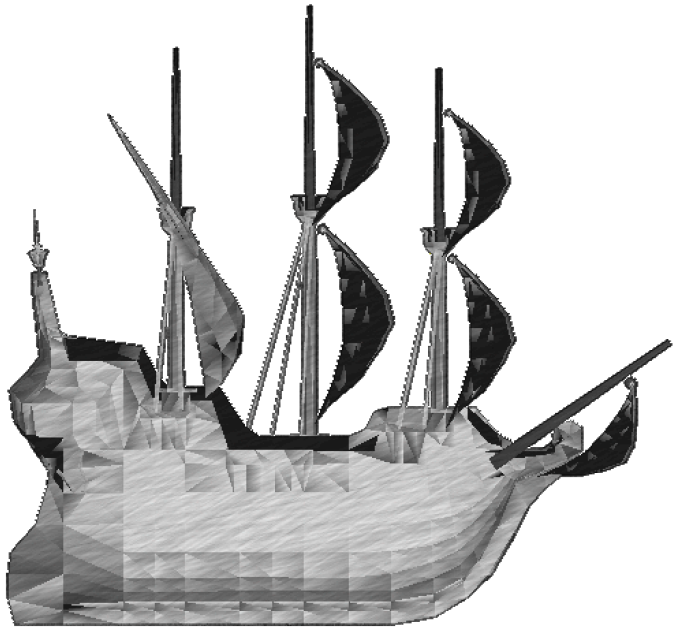
CAP 5705 - COMPUTER GRAPHICS



Project Report

REAL TIME PENCIL RENDERING

Submitted By,

Amlan Pradhan

Shashank Ranjan

Sarath Francis

Table of Contents

Abstract 2
Goals of the Project 3
Literature Review 4
System Overview 5
Implementation 7
System as a Whole 7
Pencil Textures 8
Curvature Calculation 8
Contour Detection 8
Mapping pencil texture to the contours 9
Interior Shading 9
Texture Rotation 9
3-way blending 10
Composing the contour image and the textured image 12
Gouraud Shading 13
Achieved goals and Shortcomings 14
Achieved goals 14
Shortcomings of the project 14
Learning Outcomes 15
Future Enhancements 15
Member Contributions 16
Sample Output Images 17
Acknowledgements 18
References 19
Literature: 19
Meshes: 19
Pencil Textures: 19
Online Resources: 20

Abstract

This project implements the technique for rendering 3D meshes in the pencil drawing style. The rendering process is fully implemented in the GPU using OpenGL. We implement contour drawing to imitate the human contour drawing. For interior shading, we perform a simple mapping of pencil textures on to the object surface. For improving the quality of pencil drawing, use Gouraud Shading to give realistic effect to the rendered images.

Goals of the Project

- **Multiple contour drawing:** When a person draws an object with a pencil, the object contours are usually drawn a few times with slightly different shapes. We implement an effective technique to handle this characteristic of pencil drawing.

• I**nterior shading:** We implement a shading technique that achieves realistic effects using overlapped textures. Our technique is fully implemented on a GPU and can incorporate paper effects into interior shading.

• **Pencil texture generation and mapping:** We implement techniques for generating and mapping pencil textures that reflect characteristics of graphite pencils.

• **GPU-based rendering framework:** Most of the components in our pencil rendering technique are implemented on a GPU. Consequently, the rendering speed is quite fast; 15 to 60frames per second for reasonably complicated meshes.

Literature Review

Non Photorealistic Rendering (NPR) is widely used in the field of Computer Graphics. In this era of computer games and animation movies, NPR is a hot topic. A variety of Non Photo-realistic rendering techniques have developed to provide software tools that simulates physical materials for drawing and painting.

A pencil is one of the most available and easy to handle tools for drawing. Pencil drawing is appealed as an interesting art form of its own. Pencil drawings can represent soft impressions of objects with pencil strokes, where the exact tones of strokes are determined by the properties of pencil.

There are numerous research papers published in this area and few of them are reviewed here.

Sousa and Buchman had worked on analyzing and simulating properties of various materials for pencil drawing. The simulation results showed nice effects and were comparable to real time pencil drawing. The main drawback of their work was the high computation for simulation. This high computation makes it unsuitable for the real time rendering.

With the advances in graphics hardware, real-time techniques for non-photo realistic rendering have been developed. **Majumder** and **Gopi** introduced a real-time technique for charcoal rendering of 3D objects, which utilizes graphics hardware.

In this project, we have followed the research paper on **"Real-Time Pencil Rendering"** by **Hyunjun Lee, Sungtae Kwon, Seungyong Lee** published in **NPAR '06 Proceeding of the 4th International Symposium on Non-photorealistic animation and rendering.**

System Overview

In a pencil drawing, the 2 important steps are drawing the contours (boundaries and silhouettes) and interior shading. The original implementation uses a 3D mesh of an object as the input and at the run time, the contour in the image is detected which will generate a contour image. The interior shading is performed in parallel. Then the output images of both the steps are combined to generate the final image.

There are 2 stages namely, Preprocessing stage and Run-time stage. In the preprocessing stage, we prepare data that are used in the run time.

For contour drawing, following are the list of actions performed in the preprocessing stage

- Contour texture generation
- Distorted plane generation

For interior shading, the preprocessing stage activities are:

- Pencil texture generation
- Curvature calculation

In the run-time stage, contour drawing related activities are

- Contour detection and contour shaking
- Multiple contour drawing

The activities for interior shading are:

- Brightness Adjustment
- Texture Rotation
- 3 – way blending

Composing the output images of contour drawing and the interior shading generates the final output image.

The below image depicts the whole process.

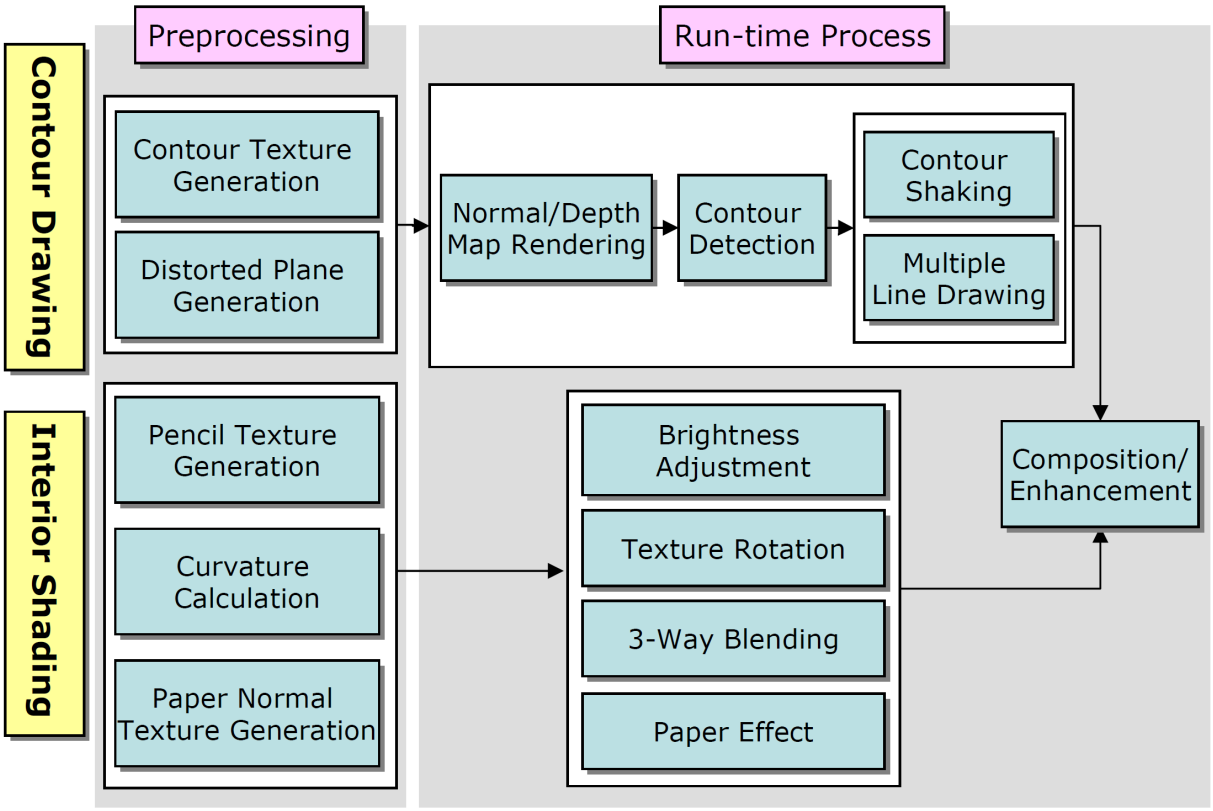


Fig. 1: System Overview (http://cg.postech.ac.kr/research/pencil_rendering/paper/npar.pdf)

The details of each step are described in the following sections

Implementation

System as a Whole

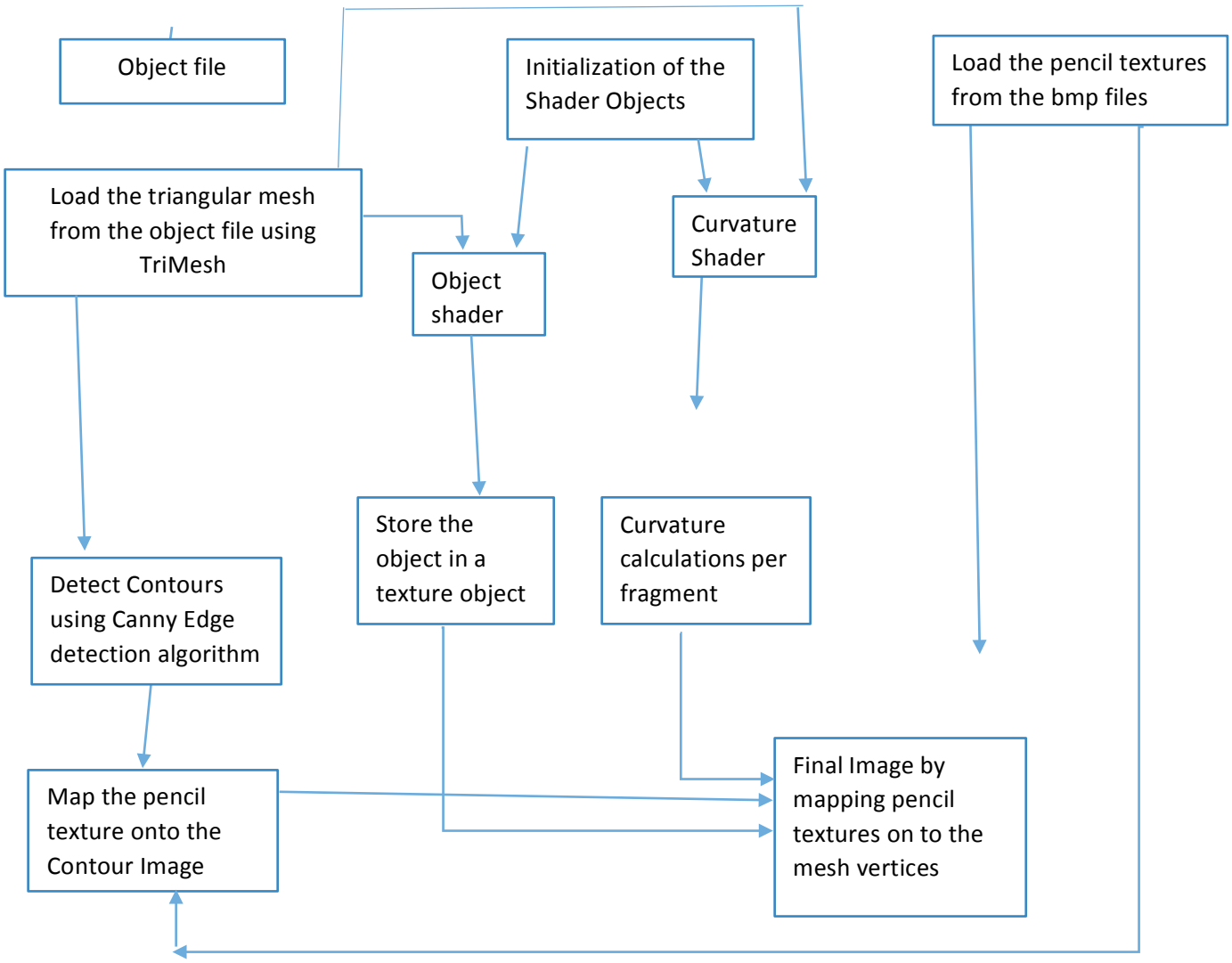


Fig. 2: Schematic Diagram of Rendering System

Pencil Textures

We have used pre-generated pencil textures.

Curvature Calculation

We used trimesh2 library to load the meshes, which internally implements and provides API's for curvature calculation at each vertex of the mesh

Contour Detection

We use **Canny edge detection algorithm** for detecting contours in the input 3D mesh. Canny algorithm is one of the simplest and widely used edge detection algorithms. The result of the contour detection is a black and white grey scale image with just the contours.

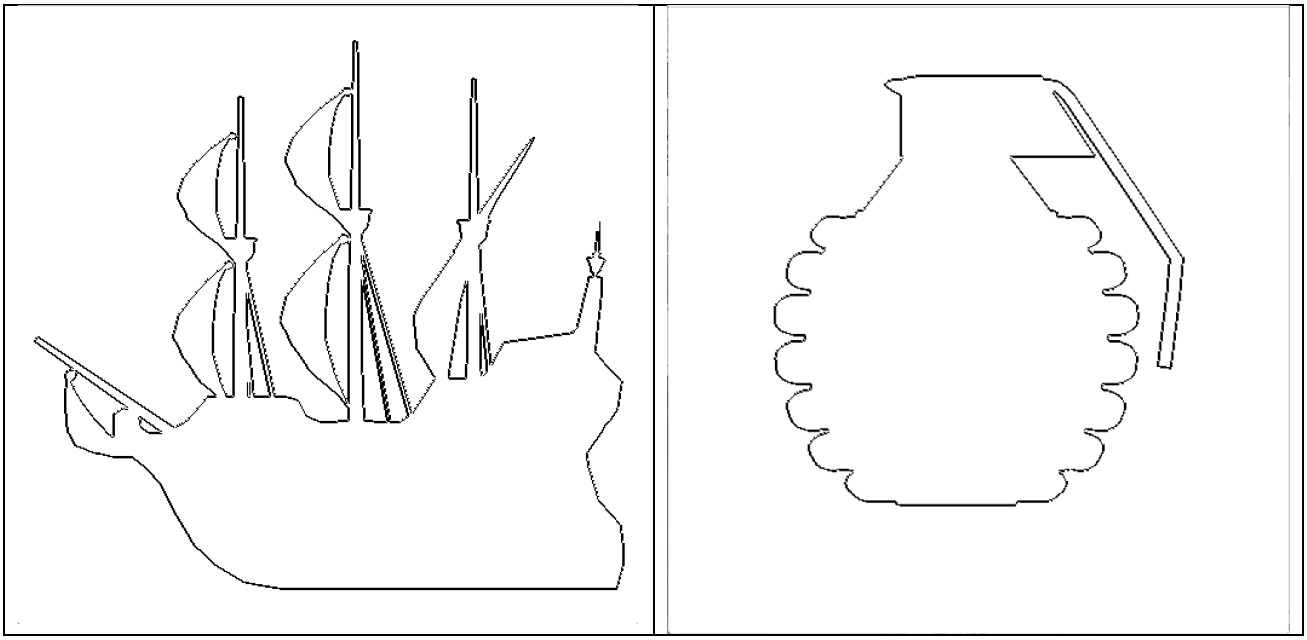


Fig. 3: Output of Canny Edge Detector. Galleon (left), Grenade (right)

Mapping pencil texture to the contours

To mimic the realistic drawing of the borders or contours of the image, we texture the contour image obtained from the previous step using a dark pencil texture.

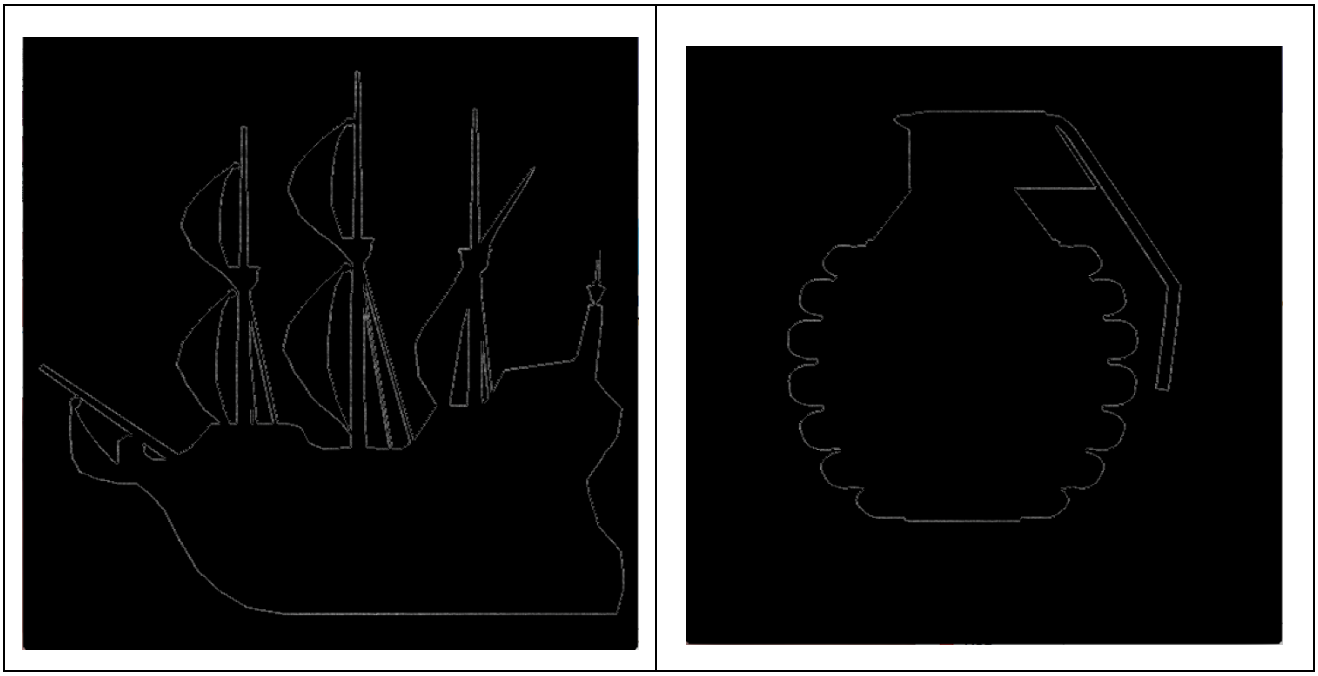


Fig. 4: Pencil texture mapped onto the contour image. Galleon (left), Grenade (right)

Interior Shading

Interior shading includes the series of steps required for shading the interior of the 3D mesh with the pencil textures. The major techniques that we implemented are described below.

Texture Rotation

In real pencil drawing, the stroke directions generally follow the principle curvature directions. For applying this rule for the interior shading, we compute and store the curvature directions at mesh vertices in the preprocessing stage. We use the tensor field computation technique for computing the curvature direction.

In the run- time rendering phase, the pencil textures are onto the mesh vertices and while mapping, texture is rotated along the minimum curvature direction at each vertices to achieve realistic feeling for the shading.

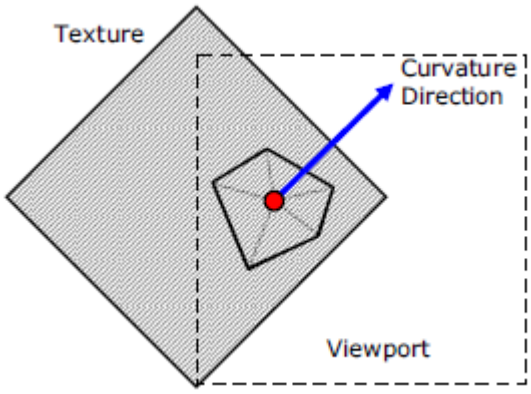


Fig. 5: Texture Rotation (http://cg.postech.ac.kr/research/pencil_rendering/paper/npar.pdf)

3-way blending

Each face of the object is assigned with 3 pencil textures as a result of the texture rotation. The actual mapping of three textures onto a face is implemented by multi- texturing with three different 3D texture coordinates at each vertex. The three rotated textures are blended together in the pixel- shader to get the final realistic texture.

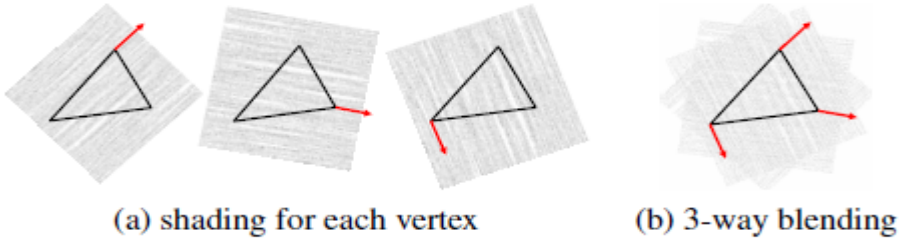


Fig. 6: 3-way blending (http://cg.postech.ac.kr/research/pencil_rendering/paper/npar.pdf)

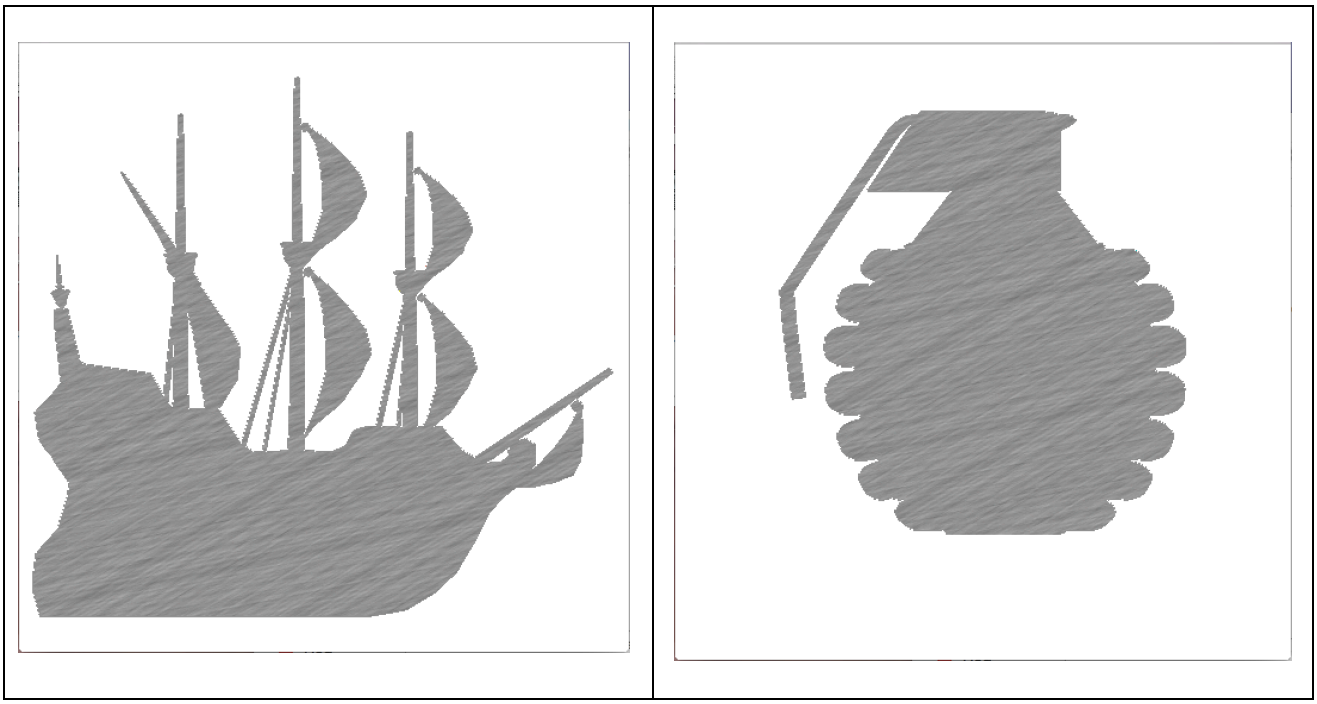


Fig. 7: Image with only pencil texture applied. Galleon (left), Grenade (right)

Composing the contour image and the textured image

We generated the contour image and interior image separately and stored as textures. Then these two textures are combined together to render the final image in the screen space. The composition is performed using a pixel shader.

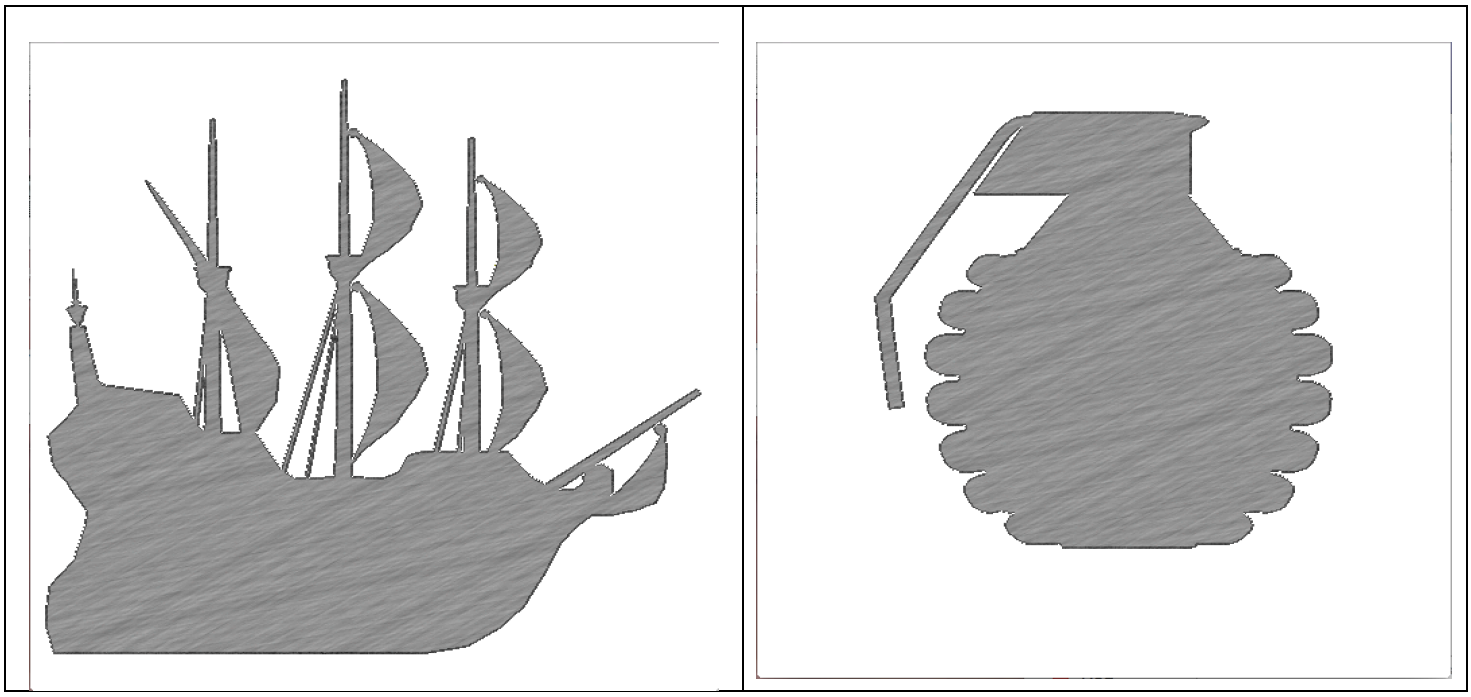


Fig. 8: Textured image and the contour image composed. Galleon (left), Grenade (right)

Gouraud Shading

We apply Gouraud shading to the final composed image to add more realism. The shading worked fine for within a fragment, but errors showed up while dealing with a mesh of fragments (neighboring ones).

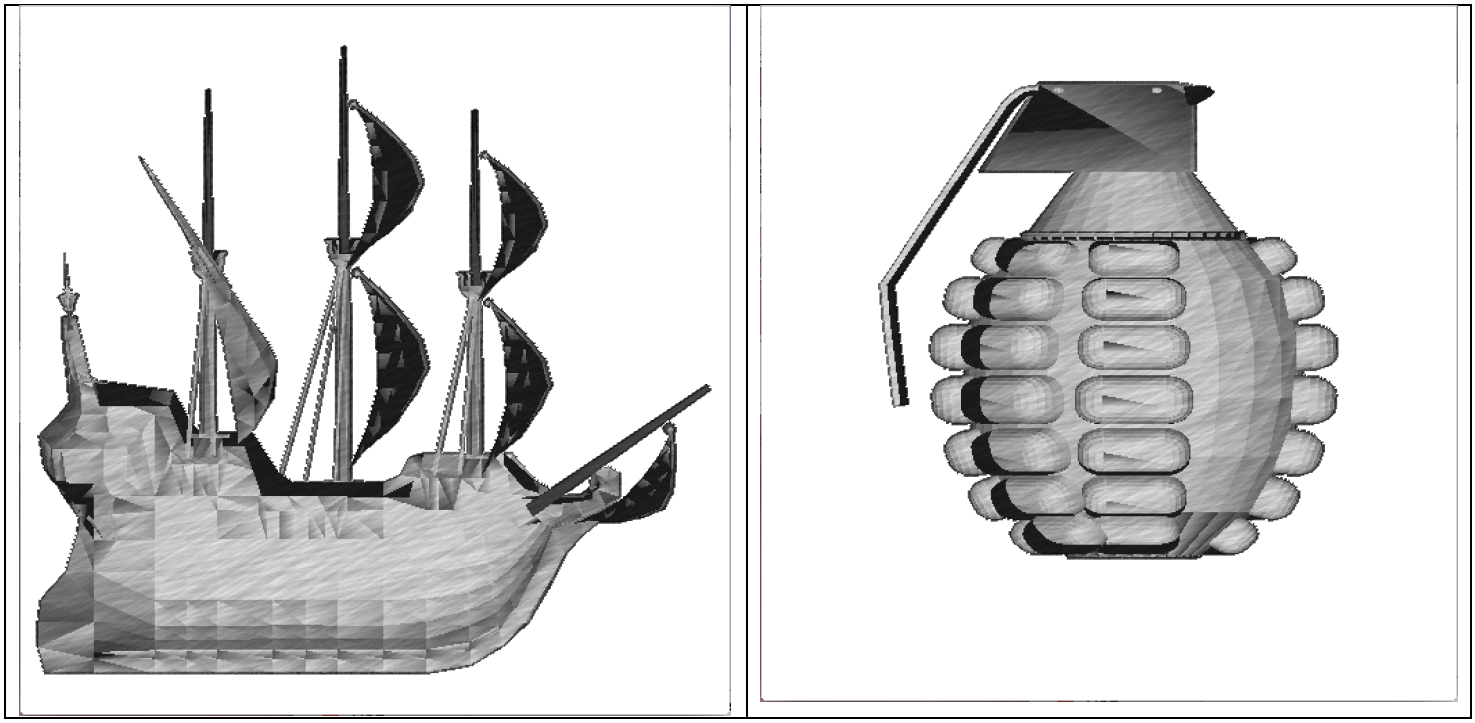


Fig. 9: Gouraud shading applied to the composed image. Galleon (left), Grenade (right)

Achieved goals and Shortcomings

Achieved goals

- Implementation of GLSL shader

  We used GLSL vertex and fragment shaders for shading the object.

- Off screen rendering using Frame buffer

  Frame Buffers are used for the off screen rendering of image

- Multi texturing

  The curvature calculations, the contour image and the final composed image are stored in different textures and in the final stage all the textures are composed into a single image using multi texturing

- Gouraud Shading

  We were able to implement the per fragment Gouraud shading, with some errors while shading across fragments

- 3-way texture blending

  We implemented the 3-way texture blending at each of the mesh vertices. This mimics the human method of pencil drawing

- Contour detection

  We used Canny edge detection algorithm for detecting the edges.

- Mapping pencil texture onto the contours

  The previously stored pencil texture is mapped onto the detected contours for realistically showing the pencil drawn contours

Shortcomings of the project

- Texture Rotation

  Due to the errors in curvature calculations, we were not able to implement the rotation of textures along the curvature directions

- Smooth Shading

  We were not able to get a smooth shading effect across fragments.

- We used pre-generated pencil textures rather than generating our own pencil textures.

- Edge detection using the image space approach

  We were able to implement Normal Map and Depth Map but the discontinuity calculation in the Normal map was giving errors, so we used canny edge detection algorithm for detecting the contours. As a result, we were able to identify the border and silhouette edge, but not the crease edges.

- Contour shaking and multiple contour drawing

  We were not able to implement distorted image plane. As a result contour shaking and multiple contour drawing couldn't be implemented.

Learning Outcomes

- Implementation of GLSL Shaders
- Texture mapping in OpenGL
- Off-screen rendering using Framebuffers.
- Implementing multi-texturing in Shaders
- Edge Detection in meshes using Canny Edge Detector
- Generation of Normal and Depth Maps using Pixel Shaders

Future Enhancements

- Contour detection for detecting creases
- Orienting the pencil textures along the curvature directions.
- Use multiple pencil textures with varying intensities
- Bump texture mapping for implementing the paper effect.

Member Contributions

- Amlan Pradhan :
  - Texture Blending
  - Interior Shading
  - Off Screen rendering to frame buffers
  - Multiple texturing
  - Texture Rotation (did not work)

- Sarath Francis
  - Contour Shaking
  - Multiple Contour Drawing
  - Project Report

- Shashank Ranjan
  - Edge Detection using Canny Edge Detection Algorithm
  - Trimesh2 library for curvature calculation
  - Edge Detection using Image Space (did not work)

Sample Output Images

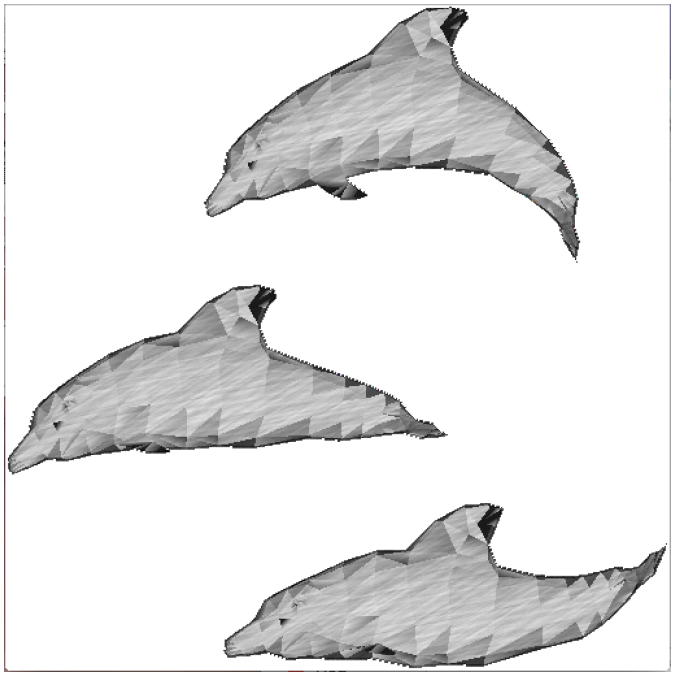


Fig. 10: Dolphins

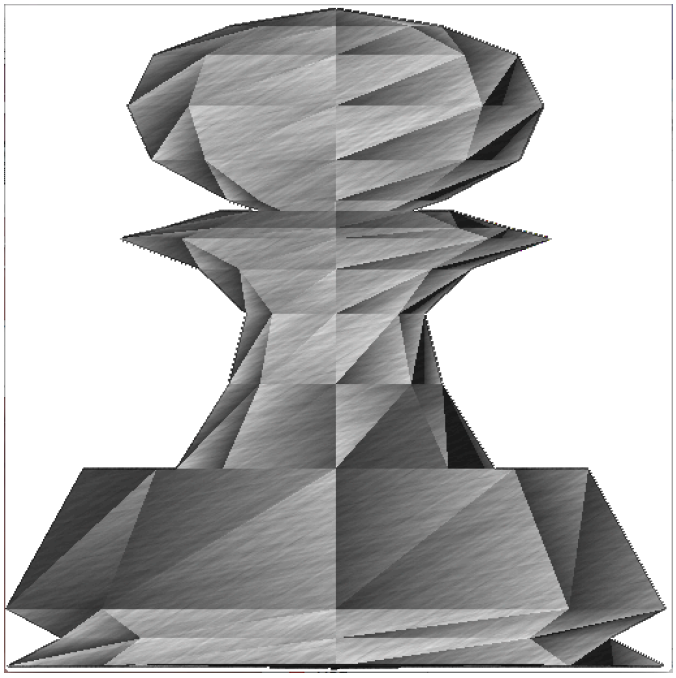


Fig. 10: Pawn

Acknowledgements

We especially wish to sincerely thank our project guide **Prof. Alireza Entezari**, Department of Computer & Information Science & Engineering, UF without whose help and support, we would not have been able to successfully complete this project.

References

Literature:

LEE, H., KWON, S., AND LEE, S. 2006. Real-time pencil rendering. In NPAR, 37–45.

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LEVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Transactions on Graphics 22*, 3, 485–493.

CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D. H. 1997. Computer-generated water- color. In *Proc. ACM SIGGRAPH 1997*, 421–430.

MAJUMDER, A., AND GOPI, M. 2002. Hardware accelerated real time charcoal rendering. In *Proc. NPAR 2002*, 59–66.

MAO, X., NAGASAKA, Y., AND IMAMIYA, A. 2001. Automatic generation of pencil drawing from 2D images using line integral convolution. In *Proc. 7th Int'l Conference on Computer Aided Design and Computer Graphics*, 240–248.

NIENHAUS, M., AND DOELLNER, J. 2003. Edge-enhancement an algorithm for real-time non-photorealistic rendering. *Int'l Winter School of Computer Graphics, Journal of WSCG 11*, 2, 346–353.

SAITO, T., AND TAKAHASHI, T. 1990. Comprehensible rendering of 3D shapes. In *Computer Graphics (Proc. ACM SIGGRAPH 1990)*, 197–206.

SALISBURY, M. P., WONG, M., HUGHES, J. F., AND SALESIN, D. H. 1997. Orientable textures for image-based pen-and-ink illustration. In *Proc. ACM SIGGRAPH 1997*, 401–406.

SOUSA, M. C., AND BUCHANAN, J. W. 2000. Observational models of graphite pencil materials. *Computer Graphics Forum 19*, 1, 27–49.

WEBB, M., PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2002. Fine tone control in hardware hatching. In *Proc. NPAR 2002*, 53–58.

WINKENBACH, G., AND SALESIN, D. H. 1994. Computer- generated pen-and-ink illustration. In *Proc. ACM SIGGRAPH 1994*, 91–100.

Meshes:

ROBINS, NATE. Smooth Normal Generation with Preservation of Edges.
https://user.xmission.com/~nate/smooth.html

Pencil Textures:

SHEN, TIANWEI, AND LEE, H.
https://github.com/hlzz/Pencil-Rendering/tree/master/Materials/texture

Online Resources:

KUNTZ, NOAH. CANNY TUTORIAL. http://www.pages.drexel.edu/~nk752/cannyTut2.html

LIGHTHOUSE3D.COM. GLSL 1.2 Tutorial. http://www.lighthouse3d.com/tutorials/glsl-tutorial/

AHN, SONG HO. OpenGL Frame Buffer Object (FBO) tutorial. http://www.songho.ca/opengl/gl_fbo.html

TREIB, MARC. Real – Time Pencil Rendering. http://www.slideserve.com/keon/real-time-pencil-rendering