Computer Graphics

Introduction

Selected Comments from last years TCE

- What did you like
  - I enjoyed applying the concepts learned in class through the programming projects.
  - I enjoyed learning the theory behind creating graphics on the computer.
  - The professor did not talk about coding much.
  - The projects were very interesting and seemed applicable.
  - Great course!
  - Good projects. They were always challenging and always instructive, without being unrealistically difficult.
  - Great projects. They were always challenging and always instructive, without being unrealistically difficult.
  - Prof Efrat explains things in detail so that they become clear. I like the way he discusses the Mathematics.
  - We had very interesting assignments and covered a wide variety of topics.

Comments from last year TCE

- Didn’t like / comments
  - ...also, the 3rd and 4th assignments were basically impossible without knowledge of OpenGL.
  - Talk more about OpenGL.
  - Teach more about the actual graphics pipeline (e.g. how to actually „use” OpenGL, how to actually „do” stuff with graphics), not splines, BREP, or unused/outrated graphics algorithms.
  - More code examples, more demonstration, less abstraction, more explanation of the platform and the techniques used with it.
  - More emphasis should be given on graphics programming than math. If we have a sufficient understanding of what should be done, its better the projects let us work on.
  - the actual intricacies of graphics programming, instead of spending too much precious time on solving a math problem
  - Overall great class!
  - Nice guy, though.

Syllabus

- Introduction
  - OpenGL
- Geometry & Transformations
- Scan Conversion
- Clipping
- Polygon Fill
- Hidden Surface Removal
- Geometric Data Structures
- Geometric Modeling
  - B-splines, Bezier Curves, linear Bilinear patches
  - surface of revolution (check the slides for “modeling” – if you really dislike calculus, this chapter might not appeal to you)
- Shading
- Shadows
- Texture Mapping
- Ray Tracing
- Antialiasing
- GPU – fragments programming / shaders
- Global Illumination and Radiosity.
- (need to know how to solve linear system of equations)

Computer Graphics

Synthesis of static/dynamic 2D images from 3D geometry using computers.

3D Graphics

- Geometric model
3D Graphics

- Rendering

Generating 3D Geometry

Explicit:
\[ z = \sqrt{x^2 + y^2} \]
Implicit:
\[ z = \sqrt{x^2 + y^2} \]

Generating 3D Geometry

Digitization

Generating 3D Geometry

3D model from CT images

Rendering

- Material Properties
  - Fog
  - Texture
  - Reflectivity
  - Refraction

Image Processing and Computer Vision

- Image enhancement
- Feature extraction
- Pattern recognition
- 3D model extraction
Computer Graphics

Introduction

Applications
- Geometric Modeling
- Mechanical Design

Applications
- Medical

Applications
- Special Effects

Applications
- Computer Games
- Images
  - Design
  - Advertising
  - Art
Computer Graphics

Applications

• Movies

Literature

• Advanced Animation and Rendering Techniques
• OpenGL Programming Guide

Hidden Surface Removal

Shadows

Texture Mapping

Geometric Modeling
Computer Graphics

Introduction

Selected Comments from last years TCE

- What did you like
  - I enjoyed applying the concepts learned in class through the programming projects.
  - I enjoyed learning the theory behind creating graphics on the computer.
  - The professor did not talk about coding much
  - The projects were very interesting and seemed applicable
  - Great course!
  - Good projects. The were always challenging and always instructive, without being unrealistically difficult.
  - Good projects. The were always challenging and always instructive, without being unrealistically difficult.
  - Prof Efrat explains things in detail so that they become clear. I like the way he discusses the Mathematics.
  - We had very interesting assignments and covered a wide variety of topics

Comments from last year TCE

- Didn’t like / comments
  - ...also, the 3rd and 4th assignments were basically impossible without knowledge of OpenGL.
  - Talk more about OpenGL.
  - Teach more about the actual graphics pipeline (e.g. how to actually "use" OpenGL, how to actually "do" stuff with graphics), not splines, BRDF, or unused/outdated graphics algorithms
  - more code examples, more demonstration, less abstraction, more explanation of the platform and the techniques used with it.
  - More emphasis should be given on graphics programming than math. If we have a sufficient understanding of what should be done, its better the projects let us work on
  - the actual intricacies of graphics programming, instead of spending too much precious time on solving a math problem
  - Overall great class!
  - Nice guy, though.