CMSC 335
COMPUTER GRAPHICS

LECTURE 1

• INTRODUCTION TO COURSE
• GRAPHICS OVERVIEW
WELCOME

• Questions?
SYLLABUS

• Questions?
WHAT IS COMPUTER GRAPHICS?

• Your thoughts?

• Any use of computers to create and manipulate images
  • Can be two- or three-dimensional, though we will heavily focus on 3D
  • Algorithms – e.g., physical simulation of water or fire
  • Hardware – e.g., GPUs or touch devices
  • Interaction – e.g., GUIs or touch devices

• Where do you see computer graphics?
AREAS OF COMPUTER GRAPHICS

• **Modeling** – mathematical specification of a shape and its appearance
• **Rendering** – creation of the images from the shape
• **Animation** – creating the illusion of motion

• Related areas
  • User interaction
  • Virtual Reality
  • Visualization
  • Image Processing
  • 3D Scanning
  • Computational Photography

• What interests you the most?
MAJOR APPLICATIONS

• With a partner/trio: name a computer graphics application area (other than video games)
• Describe how graphics is used within it
MAJOR APPLICATIONS

- Video Games
MAJOR APPLICATIONS

• Computer Generated Imaging
• Animated Films
• Cartoons
• Art
MAJOR APPLICATIONS

• Computer Aided Design (CAD)
• Education and training tools
MAJOR APPLICATIONS

- Research
- Simulation
- Data visualization
MAJOR APPLICATIONS

• Medical Imaging
• 3D Reconstruction
• 3D Printing
NUMERICAL ACCURACY

• In many applications numerical accuracy matters (affects quality)

• With a partner: Compare the needs of art vs physics.

• General guidelines
  • Use float by default (graphics hardware only uses floats)
  • Use double for general scientific applications
  • Use multi-precision for the most accuracy
EFFICIENCY

• With a partner: Compare the efficiency needs of video games vs Hollywood CGI.

• We will often consider efficiency and parallelism. Always keep these in mind
  • Integers are more efficient than floating point numbers
  • GPUs have many cores to utilize

• Avoid pre-mature optimization. Use a data-driven approach instead.
DEBUGGING

• Debugging is hard!

• Only can reason about the pixels of an image (or lack there of)

• Be creative, e.g., display blue if computation gives result you expect so you can test as you go.
BASIC GRAPHICS PIPELINE (VIEWING PIPELINE)

Model Coordinates → World Coordinates → Camera Coordinates → Projection Coordinates → Device Coordinates

(Model Coordinates) (World Coordinates) (Camera Coordinates) (Projection Coordinates) (Device Coordinates)
RENDERING METHODS

• **Raytracing** – shoot rays into a scene computing a color per pixel
  • Generally expensive and accurate

• **Rasterization** – finding pixels in an image occupied by a geometric primitive
  • Generally efficient and inaccurate

• Which applications would use each type?
MODELING

• Models are typically polygonal meshes
  • Vertex data
    • Position
    • Normal
    • Texture coordinate
    • Etc
  • Face data (triangles)

• Note a "Vertex" refers to ALL of its data, so in the example, each corner is possibly 3 separate vertices
WHY TRIANGLES

• Why triangles?
• Simplest geometric object
  • Defines a plane
  • Many other properties to exploit
• A triangle mesh is a collection of triangles
  • Want our models to be "watertight"
  • Triangles defined with a counter-clockwise rotation
LIGHTING

• It is important to model interactions of lighting with and between objects (shadowing) as it provides realism.

• A normal represents the direction that a vertex faces on a model. Lighting is biased by this normal.
TEXTURING

• Synonymous to gluing an image on an object (triangle)
• Gives an object its base color
• Each vertex has a texture coordinate which refers to a location in a texture
  • Texture coordinates are always in \([0,1]^2\) and are not pixel coordinates
  • Also called UV coordinates
• Can be used for other properties as well!
RASTER IMAGES

- A **pixel** is a picture element whose data is typically at least a color (but can be more, e.g., depth information).
- The **framebuffer** is a special location in memory of pixel data for the monitor to display.
- Monitor technology (e.g., CRT) used to be built upon the concept of a **scan line**, i.e., a row of pixels, and many algorithms still rely on this.
- The **refresh rate** is the frequency that an image can be redrawn.
COLORS

• Colors are usually expressed in Red-Green-Blue (RGB) format
  • 3 8-bit integers (each a value 0-255) or
    3 floating-point numbers (each a value 0-1) representing intensity. 0 is no intensity or black

• Often colors add in an alpha channel representing transparency. 0 is fully transparent. 255u or 1.f is fully opaque.
ANIMATION

• Animation is created by rapidly showing and updating a 2D image repeatedly