Lecture 1:

Course Intro:
Overview

Computer Graphics
CMU 15-462/662
TODAY: Overview Computer Graphics

- Two main objectives:
  - Try to understand broadly what computer graphics is about
  - “Implement” our 1st algorithm for making images of 3D shapes
Q: What is computer graphics?
Probably an image like this comes to mind:
Q: ...ok, but more fundamentally: What is computer graphics—and why do we need it?
Early computer (ENIAC), 1945

punch card (~120 bytes)
There must be a better way!
Sketchpad (Ivan Sutherland, 1963)
2021: 8k monitor
7680x4320 (~130MB)
Coming down the pipe...

2021 virtual reality headset: 2x 2160x2160 @ 90Hz => 2.3GB/s
Why visual information?

About 30% of brain dedicated to visual processing...

...eyes are highest-bandwidth port into the head!
What is computer graphics?

**computer graphics** /kəmˈpyʊdər ˈɡrafɪks/ *n.* The use of computers to synthesize visual information.
What is computer graphics?

computer graphics /kəmˈpyʊərəˌɡrɑːfɪks/ n.
The use of computers to synthesize visual information.

Why only visual?

computation

digital information

visual information

Graphics has evolved a lot since its early days... no longer just about turning on pixels!
Turning digital information into sensory stimuli

*computer graphics* /ˈkɒmˌpjuːdiər ˈɡræfɪks/ *n.*
The use of computers to synthesize and manipulate sensory information.

(...What about taste? Smell?!)

(sound)

(touch)
Turning digital information into physical matter
Definition of Graphics, Revisited

**computer graphics** /kəmˈpyʊdər ˈɡrafiks/ *n.*
The use of computation to turn **digital information** into sensory stimuli.
Even this definition is too narrow...
SIGGRAPH 2021 Technical Papers Trailer
Computer graphics is everywhere!
Entertainment (movies, games)
Entertainment

- Not just cartoons!
Art and design
Industrial design
Computer aided engineering (CAE)
Architecture
Scientific/mathematical visualization
Medical/anatomical visualization
Navigation
Communication
Foundations of computer graphics

- All these applications demand **sophisticated** theory & systems

- **Theory**
  - **basic representations** (how do you digitally encode shape, motion?)
  - **sampling & aliasing** (how do you acquire & reproduce a signal?)
  - **numerical methods** (how do you manipulate signals numerically?)
  - **radiometry & light transport** (how does light behave?)
  - **perception** (how does this all relate to humans?)
  - ...

- **Systems**
  - parallel, heterogeneous processing
  - graphics-specific programming languages
  - ...

ACTIVITY: modeling and drawing a cube

- Goal: generate a realistic drawing of a cube
- Key questions:
  - Modeling: how do we describe the cube?
  - Rendering: how do we then visualize this model?
ACTIVITY: modeling the cube

- Suppose our cube is...
  - centered at the origin (0,0,0)
  - has dimensions 2x2x2
  - edges are aligned with x/y/z axes

QUESTION: What are the coordinates of the cube vertices?

A: ( 1, 1, 1 )   E: ( 1, 1,-1 )
B: (-1, 1, 1 )   F: (-1, 1,-1 )
C: ( 1,-1, 1 )   G: ( 1,-1,-1 )
D: (-1,-1, 1 )   H: (-1,-1,-1 )

QUESTION: What about the edges?

AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH
ACTIVITY: drawing the cube

Now have a digital description of the cube:

**VERTICES**

A: (1, 1, 1)   E: (1, 1, -1)
B: (-1, 1, 1)   F: (-1, 1, -1)
C: (1, -1, 1)   G: (1, -1, -1)
D: (-1, -1, 1)   H: (-1, -1, -1)

**EDGES**

AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH

How do we draw this 3D cube as a 2D (flat) image?

Basic strategy:
1. map 3D vertices to 2D points in the image
2. connect 2D points with straight lines

...Ok, but how?
Perspective projection

- Objects look smaller as they get further away ("perspective")
- Why does this happen?
- Consider simple ("pinhole") model of a camera:
Perspective projection: side view

- Where exactly does a point $p = (x,y,z)$ end up on the image?
- Let’s call the image point $q = (u,v)$
Where exactly does a point \( p = (x, y, z) \) end up on the image?

Let’s call the image point \( q = (u, v) \)

Notice two similar triangles:

Assume camera has unit size, \textbf{origin} is at pinhole \( c \)

Then \( v/1 = y/z \), i.e., vertical coordinate is just the slope \( y/z \)

Likewise, horizontal coordinate is \( u = x/z \)
ACTIVITY: now draw it!

- Repeat the same simple algorithm 12 times
  - Once for each edge
  - Assume camera is at $c=(2,3,5)$
  - Convert $(X,Y,Z)$ of both endpoints to $(u,v)$:
    1. subtract camera $c$ from vertex $(X,Y,Z)$ to get $(x,y,z)$
    2. divide $(x,y)$ by $z$ to get $(u,v)$—write as a fraction
  - Draw line between $(u_1,v_1)$ and $(u_2,v_2)$

VERTICES

A: ( 1, 1, 1 )    E: ( 1, 1,-1 )
B: (-1, 1, 1 )    F: (-1, 1,-1 )
C: ( 1,-1, 1 )    G: ( 1,-1,-1 )
D: (-1,-1, 1 )    H: (-1,-1,-1 )

EDGES

AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH
Your Assignment for the next three minutes (write this down!)

- If your first (preferred) name begins with ....
  - A                           you have edge AB
  - B, C, or D                  you have edge CD
  - E, F, or G                  you have edge EF
  - H, or Ja                    you have edge GH
  - Je, Ji, Jo                  you have edge AC
  - Ju, K, Z                    you have edge BD
  - L, M                        you have edge EG
  - N, O, P                     you have edge FH
  - Q, R, Sa, Sc                you have edge AE
  - Se, Sh, T                   you have edge CG
  - V, W, X                     you have edge BF
  - Y                           you have edge DH
ACTIVITY: output on graph paper
ACTIVITY: How did you do?

2D coordinates:

A: 1/4, 1/2
B: 3/4, 1/2
C: 1/4, 1
D: 3/4, 1
E: 1/6, 1/3
F: 1/2, 1/3
G: 1/6, 2/3
H: 1/2, 2/3
ACTIVITY: Previous year’s result
Success! We turned purely digital information into purely visual information, using a completely algorithmic procedure.
But wait…

How do we draw lines on a computer?
Close up photo of pixels on a modern display

Lorem
Output for a raster display

- Common abstraction of a raster display:
  - Image represented as a 2D grid of “pixels” (picture elements)
  - Each pixel can take on a unique color value

** We will strongly challenge this notion of a pixel “as a little square” soon enough. But let’s go with it for now. ;-)
What pixels should we color in to depict a line?

“Rasterization”: process of converting a continuous object to a discrete representation on a raster grid (pixel grid)
What pixels should we color in to depict a line?

Light up all pixels intersected by the line?
What pixels should we color in to depict a line?

Diamond rule (used by modern GPUs):
light up pixel if line passes through associated diamond
What pixels should we color in to depict a line?

Is there a right answer?
(consider a drawing a “line” with thickness)
How do we find the pixels satisfying a chosen rasterization rule?

- Could check every single pixel in the image to see if it meets the condition...
  - $O(n^2)$ pixels in image vs. at most $O(n)$ “lit up” pixels
  - must be able to do better! (e.g., work proportional to number of pixels in the drawing of the line)
Incremental line rasterization

- Let’s say a line is represented with integer endpoints: \((u_1,v_1), (u_2,v_2)\)
- Slope of line: \(s = (v_2-v_1) / (u_2-u_1)\)
- Consider an easy special case:
  - \(u_1 < u_2, v_1 < v_2\) (line points toward upper-right)
  - \(0 < s < 1\) (more change in x than y)

\[
v = v_1;
\text{for}(u=u_1; u<u_2; u++)
{
    \text{draw}(u, \text{round}(v))
    v += s;
}
\text{draw}(u_2, \text{round}(v))
\]

Easy to implement... not how lines are drawn in modern software/hardware!
We now have our first complete graphics algorithm!

Digital information

**VERTICES**
A: (1, 1, 1)
B: (-1, 1, 1)
C: (1, -1, 1)
D: (-1, -1, 1)
E: (1, 1, -1)
F: (-1, 1, -1)
G: (1, -1, -1)
H: (-1, -1, -1)

**EDGES**
AB, CD, EF, GH,
AC, BD, EG, FH,
AE, CG, BF, DH

**CAMERA**
C = (2, 3, 5)

Visual information

This is fundamentally what computer graphics is all about...
So far, just made a simple line drawing of a cube.

For more realistic pictures, will need a much richer model of the world:

- GEOMETRY
- MATERIALS
- LIGHTS
- CAMERAS
- MOTION
  
  ...  

Will see all of this (and more!) as our course progresses.
Learn by making/doing!

- Build up “Scotty3D” package for modeling/rendering/animation

Scotty3D

Broken up into four major assignments...
Assignment 1: Rasterization
Motivation: display images like these!

Computer Graphics

[sources: Charles Williams, @xlavoc on codepen.io]
Assignment 2: Geometric Modeling
Motivation: create models like these!

[sources: Richard Yot, 3D-Ace, contrafibbularities, 3ddd.ru]
Assignment 3: Photorealistic Rendering
Motivation: render images like these!

WALL-E (Pixar 2009)

Lucas Lira (2020)

Moana (Disney 2016)
Assignment 4: Animation

(cribbed from Alec Jacobson)
Motivation: make animations like these!

Stephen Candell / Sony Pictures Imageworks (2017)

Yans Media (2015)

Pixar (2016)

Autonomous Systems Lab (2016)
Gallery of past student projects

https://www.youtube.com/watch?v=yJ5eY3ElmxA
A little bit more detail..
Meet our staff!

- Course web page: [http://15462.courses.cs.cmu.edu/spring2022/](http://15462.courses.cs.cmu.edu/spring2022/)
- Piazza page: [https://piazza.com/class/kydbag6zstjxr](https://piazza.com/class/kydbag6zstjxr)

- **Staff**
  - Alan Lee
    - [soohyun3 at andrew](#)
    - **Office hours:** Friday 5-7pm
    - **Location:** See Piazza
  - Mia Tang
    - [xinrant at andrew](#)
    - **Office hours:** Tuesday 5-7
    - **Location:** See Piazza
  - Joyce Zhang
    - [yunyizha at andrew](#)
    - **Office hours:** Tuesday 5-7
    - **Location:** See Piazza
  - Sanjay Salem
    - [svsalem at andrew](#)
    - **Office hours:** Thursday 6-8
    - **Location:** See Piazza
  - Daniel Zeng
    - [dlzeng at andrew](#)
    - **Office hours:** Wednesday 3-5
    - **Location:** See Piazza
## Spring 2022 Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 19</td>
<td>Course Introduction</td>
<td>Mar 9</td>
</tr>
<tr>
<td>Jan 24</td>
<td>Math Review Part I (Linear Algebra)</td>
<td>Mar 14</td>
</tr>
<tr>
<td></td>
<td>Assignment 0.0 OUT</td>
<td>Mar 16</td>
</tr>
<tr>
<td>Jan 26</td>
<td>Math Review Part II (Vector Calculus)</td>
<td>Mar 21</td>
</tr>
<tr>
<td></td>
<td>Assignment 0.0 DUE</td>
<td>Mar 23</td>
</tr>
<tr>
<td></td>
<td>Assignment 0.5 OUT</td>
<td>Mar 28</td>
</tr>
<tr>
<td>Jan 31</td>
<td>Drawing a Triangle</td>
<td>Mar 30</td>
</tr>
<tr>
<td></td>
<td>Assignment 0.5 DUE</td>
<td>Apr 4</td>
</tr>
<tr>
<td></td>
<td>Assignment 1.0 OUT</td>
<td>Apr 6</td>
</tr>
<tr>
<td>Feb 2</td>
<td>Coordinate Spaces and Transformations</td>
<td>Apr 11</td>
</tr>
<tr>
<td>Feb 7</td>
<td>3D Rotations and Complex Representations</td>
<td>Apr 13</td>
</tr>
<tr>
<td>Feb 9</td>
<td>Perspective Projection and Texture Mapping</td>
<td>Apr 18</td>
</tr>
<tr>
<td></td>
<td>Assignment 1.0 DUE</td>
<td>Apr 20</td>
</tr>
<tr>
<td></td>
<td>Assignment 1.5 OUT</td>
<td>Apr 25</td>
</tr>
<tr>
<td>Feb 14</td>
<td>Depth and Transparency</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Feb 16</td>
<td>Intro to Geometry / Manifolds / Local Operations on Manifolds</td>
<td>Apr 27</td>
</tr>
<tr>
<td></td>
<td>Assignment 1.5 DUE</td>
<td>Apr 27</td>
</tr>
<tr>
<td></td>
<td>Assignment 2.0 OUT</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Feb 21</td>
<td>Midterm Review</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Feb 23</td>
<td>MIDTERM EXAM</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Feb 28</td>
<td>Midterm Revisited / Subdivision</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Mar 2</td>
<td>Geometric Queries</td>
<td>Apr 27</td>
</tr>
<tr>
<td></td>
<td>Assignment 2.0 DUE</td>
<td>Apr 27</td>
</tr>
<tr>
<td></td>
<td>Assignment 2.5 OUT</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Mar 7</td>
<td>SPRING BREAK</td>
<td>TBD</td>
</tr>
<tr>
<td>Mar 9</td>
<td>SPRING BREAK</td>
<td>TBD</td>
</tr>
<tr>
<td>Mar 14</td>
<td>Spatial Data Structures</td>
<td>TBD</td>
</tr>
<tr>
<td>Mar 16</td>
<td>Color / Radiometry</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td>Assignment 2.5 DUE</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td>Assignment 3.0 OUT</td>
<td>TBD</td>
</tr>
<tr>
<td>Mar 21</td>
<td>The Rendering Equation</td>
<td>TBD</td>
</tr>
<tr>
<td>Mar 23</td>
<td>A3 PathTracer Specifics</td>
<td>TBD</td>
</tr>
<tr>
<td>Mar 28</td>
<td>Numerical Integration</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td>Assignment 3.0 DUE</td>
<td>TBD</td>
</tr>
<tr>
<td></td>
<td>Assignment 3.5 OUT</td>
<td>TBD</td>
</tr>
<tr>
<td>Mar 30</td>
<td>Monte Carlo Ray Tracing</td>
<td>TBD</td>
</tr>
<tr>
<td>Apr 4</td>
<td>Variance Reduction</td>
<td>TBD</td>
</tr>
<tr>
<td>Apr 6</td>
<td>More Variance Reduction</td>
<td>TBD</td>
</tr>
<tr>
<td>Apr 11</td>
<td>Introduction to Animation</td>
<td>TBD</td>
</tr>
<tr>
<td>Apr 13</td>
<td>Dynamics and Time Integration</td>
<td>TBD</td>
</tr>
<tr>
<td>Apr 18</td>
<td>Introduction to Optimization</td>
<td>TBD</td>
</tr>
<tr>
<td>Apr 20</td>
<td>Physically-Based Animation and PDEs</td>
<td>Apr 27</td>
</tr>
<tr>
<td></td>
<td>Assignment 4.0 DUE</td>
<td>Apr 27</td>
</tr>
<tr>
<td></td>
<td>Assignment 4.5 OUT</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Apr 25</td>
<td>Final Review</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Apr 27</td>
<td>Course WrapUp</td>
<td>Apr 27</td>
</tr>
<tr>
<td>Apr 27</td>
<td>Assignment 4.5 DUE (April 29)</td>
<td>Apr 27</td>
</tr>
<tr>
<td>APR 27</td>
<td>FINAL EXAM</td>
<td>Apr 27</td>
</tr>
</tbody>
</table>
Grading and Collaboration Policy

See the course info website —
http://15462.courses.cs.cmu.edu/spring2022/courseinfo
What is a Mini Homework?

- Written assignments to supplement lectures and projects
- Roughly once per week
- Can be done in groups of two or three
- Will typically be released on Monday and due the following Monday
- The first one will be out soon!
Before diving in, we’ll do a math review & preview

- Linear algebra, vector calculus
- Help make the rest of the course easier!