CIS 441/541: Intro to Computer Graphics
Lecture 4: Interpolation
Announcements
Trying to Add the Class?

• Let’s talk after class...
Canvas is Operational

• Are you getting emails?
• Submit 1A to Canvas...
Questions About Final Projects
Windows vs Mac vs Linux
VTK 6 vs VTK 7...
Review
What Are We Rendering?

• Models made up of polygons
• Usually triangles
• Lighting tricks make surfaces look non-faceted
• More on this later...
Reminder: ray-tracing vs rasterization

• Two basic ideas for rendering: rasterization and ray-tracing
• Ray-tracing: cast a ray for every pixel and see what geometry it intersects.
  – O(nPixels)
    • (actually, additional computational complexity for geometry searches)
  – Allows for beautiful rendering effects (reflections, etc)
  – Will discuss at the end of the quarter
Reminder: ray-tracing vs rasterization

- Two basic ideas for rendering: rasterization and ray-tracing
- Rasterization: examine every triangle and see what pixels it covers.
  - \( O(n\text{Triangles}) \)
    - (actually, additional computational complexity for painting in pixels)
- GPUs do rasterization very quickly
- Our focus for the next 5 weeks
What color should we choose for each of these four pixels?
What color should we choose for each of these four pixels?
What color should we choose for each of these four pixels?
What color should we choose for each of these four pixels?
What color should we choose for each of these four pixels?
The middle and lower-left variants are half-pixel translations of the other.

We will use the lower-left convention for the projects.
Where we are...

• We haven’t talked about how to get triangles in position.
  – Arbitrary camera positions through linear algebra
• We haven’t talked about shading
• On Wednesday, we tackled this problem:
  How to deposit triangle colors onto an image?
Problem: how to deposit triangle colors onto an image?

• Let’s take an example:
  – 12x12 image
  – Red triangle
    • Vertex 1: (2.5, 1.5)
    • Vertex 2: (2.5, 10.5)
    • Vertex 3: (10.5, 1.5)
  • Vertex coordinates are with respect to pixel locations
Our desired output

How do we make this output? Efficiently?
Don’t need to consider any pixels outside these lines.
Scanline algorithm: consider all rows that can possibly overlap.

Don’t need to consider any Pixels outside these lines.
Scanline algorithm: consider all rows that can possibly overlap

We will extract a “scanline”, i.e. calculate the intersections for one row of pixels
– Red triangle
  • Vertex 1: (2.5, 1.5)
  • Vertex 2: (2.5, 10.5)
  • Vertex 3: (10.5, 1.5)
– Red triangle
  • Vertex 1: (2.5, 1.5)
  • Vertex 2: (2.5, 10.5)
  • Vertex 3: (10.5, 1.5)
– Red triangle
  • Vertex 1: (2.5, 1.5)
  • Vertex 2: (2.5, 10.5)
  • Vertex 3: (10.5, 1.5)

What are the end points?
- Red triangle
  - Vertex 1: (2.5, 1.5)
  - Vertex 2: (2.5, 10.5)
  - Vertex 3: (10.5, 1.5)

What are the end points?

(2.5, 5)
– Red triangle
  • Vertex 1: (2.5, 1.5)
  • Vertex 2: (2.5, 10.5)
  • Vertex 3: (10.5, 1.5)

What are the end points?

Y=5

(2.5, 5)
- Red triangle
  - Vertex 1: (2.5, 1.5)
  - Vertex 2: (2.5, 10.5)
  - Vertex 3: (10.5, 1.5)
- $Y = mx + b$
- $10.5 = m \times 2.5 + b$
- $1.5 = m \times 10.5 + b$
- $\rightarrow$
- $9 = -8m$
- $m = -1.125$
- $b = 13.3125$
- $5 = -1.125 \times x + 13.3125$
- $x = 7.3888$ (2.5, 5)

What are the end points?
Scanline algorithm: consider all rows that can possibly overlap.

Don’t need to consider any pixels outside these lines.

2.5

Y=5

7.3888
Scanline algorithm: consider all rows that can possibly overlap

Don’t need to consider any pixels outside these lines

Color is deposited at (3,5), (4,5), (5,5), (6,5), (7,5)
Scanline algorithm

• Determine rows of pixels triangles can possibly intersect
  – Call them rowMin to rowMax
    • rowMin: ceiling of smallest Y value
    • rowMax: floor of biggest Y value

• For r in [rowMin → rowMax] ; do
  – Find end points of r intersected with triangle
    • Call them leftEnd and rightEnd
  – For c in [ceiling(leftEnd) → floor(rightEnd) ] ; do
    • ImageColor(r, c) ← triangle color
Scanline algorithm

• Determine rows of pixels triangles can possibly intersect

• For \( r \) in \([\text{rowMin} \rightarrow \text{rowMax}]\); do
  – Find end points of \( r \) intersected with triangle
    • Call them \( \text{leftEnd} \) and \( \text{rightEnd} \)
  – For \( c \) in \([\text{ceiling}(\text{leftEnd}) \rightarrow \text{floor}(\text{rightEnd})]\); do
    • \( \text{ImageColor}(r, c) \leftarrow \text{triangle color} \)

For \( r = 5 \), we call \( \text{ImageColor} \) with \((5,3), (5,4), (5,5), (5,6), (5,7)\)

Y values from 1.5 to 10.5 mean rows 2 through 10

For \( r = 5 \), \( \text{leftEnd} = 2.5 \), \( \text{rightEnd} = 7.3888 \)
Arbitrary Triangles

- The description of the scanline algorithm in the preceding slides is general.
- But the implementation for these three triangles vary:
Supersampling: use the scanline algorithm a bunch of times to converge on the “average” picture.
Project #1B

• Goal: apply the scanline algorithm to “flat bottom” triangles and output an image.
• File “project1B.cxx” has triangles defined in it.
• Due: Weds, Oct. 5th
• % of grade: 6%
Project 1B

- Cout/cerr can be misleading:

```c
fawcett:Downloads child$ cat t2.C
#include <iostream.h>
#include <iomanip>

int main()
{
    double X=188;
    X-=1e-12;
    cerr << X << endl;
    cerr << std::setprecision(16) << X << endl;
}

fawcett:Downloads child$ ./a.out
188
187.999999999999
```
Project 1B

- The limited accuracy of cerr/cout can cause other functions to be appear to be wrong:

```cpp
#include <iostream.h>
#include <iomanip>
#include <math.h>

int main()
{
    double X=188;
    X-=1e-12;
    cerr << "The floor of " << X << " is " << floor(X) << endl;
}
```

The floor of 188 is 187
Project 1B

• Floating point precision is an approximation of the problem you are trying to solve

• Tiny errors are introduced in nearly every operation you perform
  – Exceptions for integers and denominators that are a power of two

• Fundamental problem:
  – Changing the sequence of these operations leads to *different* errors.
  – Example: \((A+B)+C \neq A+(B+C)\)
For project 1B, we are making a binary decision for each pixel: should it be colored or not?

Consider when a triangle vertex coincides with the bottom left of a pixel:

We all do different variations on how to solve for the endpoints of a line, so we all get slightly different errors.
Project 1B

- Our algorithm incorporates floor and ceiling functions.
  - This is the right place to bypass the precision problem.
  - I have included “floor441” and “ceil441” in project prompt. You need to use them, or you will get one pixel differences.
Project 1B: other thoughts

• You will be building on this project ...
  – think about magic numbers (e.g. screen size of 1000)
  – add safeguards against cases that haven’t shown up yet
  – Assume nothing!
Where we are...

- We haven’t talked about how to get triangles into position.
  - Arbitrary camera positions through linear algebra
- We haven’t talked about shading
- On Friday, we tackled this problem:
  How to deposit triangle colors onto an image?

Still don’t know how to:
1) Vary colors (easy)
2) Deal with triangles that overlap

Today’s lecture will go over the key operation to do these two. Friday’s lecture will tell us how to do it.
What is a field?

Example field (2D): temperature over the United States
How much data is needed to make this picture?

Example field (2D): temperature over the United States
Linear Interpolation for Scalar Field $F$
Linear Interpolation for Scalar Field $F$

- General equation to interpolate:
  - $F(X) = F(A) + t \cdot (F(B) - F(A))$
- $t$ is proportion of $X$ between $A$ and $B$
  - $t = (X - A) / (B - A)$
Quiz Time #4

• F(3) = 5, F(6) = 11
• What is F(4)? = 5 + (4-3)/(6-3)*(11-5) = 7

• General equation to interpolate:
  – F(X) = F(A) + t*(F(B)-F(A))
• t is proportion of X between A and B
  – t = (X-A)/(B-A)
Consider a single scalar field defined on a triangle.
Consider a single scalar field defined on a triangle.

F(V1) = 10
F(V2) = 2
F(V3) = -2
What is $F(V4)$?
What is $F(V4)$?
• Steps to follow:
  – Calculate V5, the left intercept for Y=0.25
  – Calculate V6, the right intercept for Y=0.25
  – Calculate V4, which is between V5 and V6
What is the X-location of V5?

F(V1) = A → F(0) = 0
F(v2) = B → F(1) = 1
F(v) = A + ((v-v1)/(v2-v1))*(B-A):

F(v) = 0.25, find v

0.25 = 0 + ((v-0)/(1-0))*(1-0)
   v = 0.25
What is the F-value of V5?

F(v1) = A → F(0) = 10
F(v2) = B → F(1) = 2
F(v) = A + ((v-v1)/(v2-v1))*(B-A):

v = 0.25, find F(v)

F(v) = 10 + ((0.25-0)/(1-0))*(2-10)
= 10 + 0.25*(-8) = 10 - 2 = 8
What is the X-location of V6?

F(v1) = A \rightarrow F(1) = 1
F(v2) = B \rightarrow F(2) = 0
F(v) = A + ((v-v1)/(v2-v1))*(B-A):

F(v) = 0.25, find v

0.25 = 1 + ((v-1)/(2-1))*(0-1)
   = 1 + (v-1)*(-1)
0.25 = 2 - v
v = 1.75
What is the F-value of V6?

\[
F(v1) = A \quad \rightarrow \quad F(1) = 2 \\
F(v2) = B \quad \rightarrow \quad F(2) = -2 \\
F(v) = A + \frac{(v-v1)/(v2-v1)}{(B-A)}*(B-A): \\
\]

\[
v = 1.75, \text{ find } F(v) \\
F(v) = 2 + \frac{(1.75-1)/(2-1)*(-2 - +2)}{(-4)} \\
= 2 + (.75)*(-4) \\
= 2 - 3 \\
= -1
\]
What is the F-value of V5?
What is the F-value of V5?

\[ F(v1) = A \implies F(0.25) = 8 \]
\[ F(v2) = B \implies F(1.75) = -1 \]
\[ F(v) = A + ((v-v1)/(v2-v1))*(B-A) : \]

\[ v = 0.5, \text{ find } F(v) \]

\[ F(v) = 8 + ((0.5-0.25)/(1.75-0.25))*(-1-8) \]
\[ = 8 + (0.25/1.5)*9 = 8-1.5 = 6.5 \]
Visualization of $F$

How do you think this picture was made?
Outline

• Project 1B
• Scanline review
• Interpolation along a triangle
• Project 1C
Arbitrary Triangles

How do we handle arbitrary triangles?

• But the implementation for these three triangles vary:
Arbitrary Triangles

How do we handle arbitrary triangles?

• But the implementation for these three triangles vary:

Solve for location of this point and then solve two “base cases”.

Diagram showing three types of triangles with arrows indicating solutions.
Arbitrary Triangles

• Project #1B: implement the scanline algorithm for triangles with “flat bottoms”
• Project #1C: arbitrary triangles
Project #1C (3%), Due (soon)

- Goal: apply the scanline algorithm to arbitrary triangles and output an image.
- Extend your project1B code
- File proj1c_geometry.vtk available on web (80MB)
- File “reader.cxx” has code to read triangles from file.
- No Cmake, project1c.cxx