CIS 410/510: Advection (Part 4!)

Lecture #9
Hank Childs, University of Oregon
Outline

• Announcements
• Review
• Advection
Outline

- Announcements
- Review
- Advection
Announcements

• Weekly OH: Weds 1-2, Thurs: 11:30-12:30
• Sat OH are tough: fly at 1pm
• John & Jeff announcement

• Also: quiz on October 27th
Outline

• Announcements
• Review
• Advection
Particle advection is the foundation to many visualization algorithms.

Advection will require us to evaluate velocity at arbitrary locations.
LERPing vectors

- **LERP = Linear Interpolate**
- **Goal:** interpolate vector between A and B.
- **Consider** vector $X$, where $X = B - A$
- **$A + t*(B-A) = A + t*X$**
- **Takeaway:**
  - you can LERP the components individually
  - the above provides motivation for why this works
Formal definition for particle advection

• Output is an integral curve, $S$, which follows trajectory of the advection

• $S(t) =$ position of curve at time $t$
  
  – $S(t_0) = p_0$
    
    • $t_0$: initial time
    
    • $p_0$: initial position

  – $S'(t) = v(t, S(t))$
    
    • $v(t, p)$: velocity at time $t$ and position $p$
    
    • $S'(t)$: derivative of the integral curve at time $t$

This is an ordinary differential equation (ODE).
The integral curve for particle advection is calculated iteratively

\[ S(t_0) = p_0 \]

while (ShouldContinue())
{
    \[ S(t_i) = \text{AdvanceStep}(S(t_{i-1}) \]
AdvanceStep goal: to calculate $S_i$ from $S_{(i-1)}$

$S_1 = \text{AdvanceStep}(S_0)$
$S_2 = \text{AdvanceStep}(S_1)$
$S_3 = \text{AdvanceStep}(S_2)$
$S_4 = \text{AdvanceStep}(S_3)$
$S_5 = \text{AdvanceStep}(S_4)$

This picture is misleading: steps are typically much smaller.
AdvanceStep Overview

• Different numerical methods for implementing AdvanceStep:
  – Simplest version: Euler step
  – Most common: Runge-Kutta-4 (RK4)
  – Several others as well
Runge-Kutta Method (RK4)

- Most common method for solving an ODE
- **Definition:**
  - First, choose step size, h.
  - Second, `AdvanceStep(p_i, t_i)` returns:
  - **New position:** $p_{i+1} = p_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$
    
    - $k_1 = v(t_i, p_i)$
    - $k_2 = v(t_i + h/2, p_i + h/2*k_1)$
    - $k_3 = v(t_i + h/2, p_i + h/2*k_2)$
    - $k_4 = v(t_i + h, p_i + h*k_3)$
  - **New time:** $t_{i+1} = t_i + h$
Physical interpretation of RK4

• New position: \( p_{i+1} = p_i + (1/6)h(k_1 + 2k_2 + 2k_3 + k_4) \)
  
  \begin{align*}
  k_1 &= v(t_i, p_i) \\
  k_2 &= v(t_i + h/2, p_i + h/2k_1) \\
  k_3 &= v(t_i + h/2, p_i + h/2k_2) \\
  k_4 &= v(t_i + h, p_i + h/k_3)
  \end{align*}

Evaluate 4 velocities, use combination to calculate \( p_{i+1} \)
Steady versus Unsteady State

- Unsteady state: the velocity field evolves over time
- Steady state: the velocity field has reached steady state and remains unchanged as time evolves
Outline

• Announcements
• Review
• Advection
Dealing with steady state velocities

• Euler Method (unsteady):
  – New position: \( p_{i+1} = p_i + h \cdot v(t_i, p_i) \)
  – New time: \( t_{i+1} = t_i + h \)

• Euler Method (steady):
  – New position: \( p_{i+1} = p_i + h \cdot v(t_0, p_i) \)
  – New time: \( t_{i+1} = t_i + h \)
Unsteady vs Steady: RK4

- Unsteady:
  - New position: \( p_{i+1} = p_i + (1/6) \cdot h \cdot (k_1 + 2k_2 + 2k_3 + k_4) \)
    - \( k_1 = v(t_i, p_i) \)
    - \( k_2 = v(t_i + h/2, p_i + h/2 \cdot k_1) \)
    - \( k_3 = v(t_i + h/2, p_i + h/2 \cdot k_2) \)
    - \( k_4 = v(t_i + h, p_i + h \cdot k_3) \)
  - New time: \( t_{i+1} = t_i + h \)
Unsteady vs Steady: RK4

- **Steady:**
  
  - New position: \( p_{i+1} = p_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4) \)

  - \( k_1 = v(t_0, p_i) \)
  - \( k_2 = v(t_0, p_i + \frac{h}{2}k_1) \)
  - \( k_3 = v(t_0, p_i + \frac{h}{2}k_2) \)
  - \( k_4 = v(t_0, p_i + hk_3) \)
  - New time: \( t_{i+1} = t_i + h \)
Project 4

• Assigned last weekend, prompt online
• Due Oct. 24th, midnight (→ October 25th, 6am)
• Worth 7% of your grade
• Provide:
  – Code skeleton online
  – Correct answers provided
• You send me:
  – source code
  – screenshot with:
    • text output on screen
    • image of results
Project 4 in a nutshell

- Do some vector LERPing
- Do particle advection with Euler steps
- Examine results
Project 4 in a nutshell

• Implement 3 methods:
  – EvaluateVectorFieldAtLocation
    • LERP vector field. (Reuse code from before, but now multiple component)
  – AdvectWithEulerStep
    • You know how to do this
  – CalculateArcLength
    • What is the total length of the resulting trajectory?
Project 5 in a nutshell

• Implement Runge-Kutta 4
• Assess the quality of RK4 vs Euler
• Open ended project
  – I don’t tell you how to do this assessment.
    • You will need to figure it out.
    • Multiple right answers
• Deliverable: short report (~1 page) describing your conclusions and methodology.
  – Pretend that your boss wants to know which method to use and you have to convince her which one is the best and why.
• Not everyone will receive full credit.
Additional Particle Advection Techniques

- This content courtesy of Christoph Garth, Kaiserslautern University.
Streamline, Pathline, Timeline, Streakline

- **Streamline**: steady state velocity, plot trajectory of a curve
- **Pathline**: unsteady state velocity, plot trajectory of a curve
- **Timeline**: start with a line, advect that line and plot the line’s position at some future time
• Advect a surface and see where it goes – "A sheet blowing in the wind"
Streamline, Pathline, Timeline, Streakline

• Streamline: steady state velocity, plot trajectory of a curve
• Pathline: unsteady state velocity, plot trajectory of a curve
• Timeline: start with a line, advect that line and plot the line’s position at some future time
• Streakline: unsteady state, introduce new particles at a location continuously
Streaklines

Will do an example on the board
Streaklines in real life
• Stream surface:
  – Start with a seeding curve
  – Advect the curve to form a surface

\[
\frac{d}{dt} S(s, t) = \vec{v}(t, S(s, t)) \quad \text{and} \quad S(s, 0) := C(s)
\]
• Stream surface:
  – Start with a seeding curve
  – Advect the curve to form a surface
Stream Surface Computation

- Skeleton from Integral Curves + Timelines
Stream Surface Computation

- Skeleton from Integral Curves + Timelines
- Triangulation

Stream Surface Example
Stream Surface Example
Stream Surface Example #2

Vortex system behind ellipsoid
Lagrangian Methods

• Visualize manifolds of maximal stretching in a flow, as indicated by dense particles

• Finite-Time Lyapunov Exponent (FTLE)
Lagrangian Methods

• Visualize manifolds of maximal stretching in a flow, as indicated by dense particles

  – forward in time: \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ indicates divergence

  – Backward in time: \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ \textbf{FTLE}^+ indicates convergence
Isosurfacing
Height field over a terrain
Height field over a terrain
Transparent grey plane at fixed elevation (height=20)
Rendering just the intersection of the plane and the height field.
Projecting the “height = 20” lines back to 2D space

These lines are called isolines.

Isolines represent a region where the field is constant.

The isovalue for this plot is 20.
Have you ever seen a plot like this before? Where?
Elevation Map of a Shield Volcano
Neat Project for Understanding Isolines
Neat Project for Understanding Isolines

Activity 3. Topographic Profile of a Shield Volcano

classroomcompulsion.blogspot.com
Isolines vs Isosurfaces

• Isolines:
  – Input: scalar field over 2D space
  – Output: lines

• Isosurfaces:
  – Input: scalar field over 3D space
  – Output: surface

• Commonalities:
  – Reduce topological dimension by 1
  – Produce output where scalar field is constant
Isosurface of temperature from star explosion simulation (ignore blue haze)
Iterating Over Cells

- For isosurface/isoline calculation, we can iterate over the cells.
- At the end, we take the results from each cell and put them into a single scene.
First isoline calculation

**Goal:** calculate isoline for field=0.5.

**Quiz:** do you have all the information you need?

**Quiz:** draw this graph and sketch the isoline.
10x10 sampling of the field

Colors:
>0.5: red
<0.5: blue

What observations can we make from this sampling?
100x100 sampling of the field

Colors:
>0.5: red
<0.5: blue
0.5: white

What observations can we make from this sampling?
Per pixel sampling of the field

Colors:
>0.5: red
<0.5: blue
0.5: white

What observations can we make from this sampling?
Isolines

• Correct isoline appears to be close to a quarter circle around (0,0) of radius 0.5.

• Quiz: How would we represent this quarter circle?

Colors:
>0.5: red
<0.5: blue
0.5: white
Reality Check: context for isolines

• We have millions of cells
• If one cell can produce tens or hundreds of lines, then the isolines could take much more memory than the input data set
Quiz

• You want to understand which will take more memory: the isolines or the input data set.

• What facts do you need to know?
Quiz answer

• You want to understand which will take more memory: the isolines or the input data set.
• What facts do you need to know?
• Need to know:
  – How much data per cell
  – How much data per isoline in a cell
  – How many cells contain isolines
How many cells contain isolines?

This cell contains isolines if the isovalue is between 0 and 1. Otherwise, it does not.

This question is data dependent & then depends on the isovalue.
How much data per isoline in a cell?

Straight lines are easy to represent.
The memory for the correct answer is variable.
Big picture: what do we want from our isosurface?

• Tractable computation
  – Can’t create 100s or 1000s of line segments per cell for super-accurate representation

• Continuous surface
  – Triangles (or lines for isolines) need to connect up ... no gaps.
Big idea #1: approximate the isolines / isosurface

- Isolines: represent them with a minimal # of segments
- Isosurface: represent them with a minimal # of triangles
Quiz: how to approximate our “quarter circle”?
Big picture: what do we want from our isosurface?

• Tractable computation
  - Can’t create 100s or 1000s of line segments per cell for super-accurate representation

• Quiz: did we accomplish this?
  - Yes: very few per cell

• Continuous surface
  - Triangles (or lines for isolines) need to connect up ... no gaps.

• Quiz: did we accomplish this?

Answer: we got the answer exactly right at the edge of the cell ... hence no gaps. 😊
Effect of different isovalues

What are the similarities between these pictures?

Assume this cell layout
X:0→1, Y:0→1, F(0,0) = 0, F(1,0) = F(1,1) = F(0,1) = 1

Quiz: write pseudocode to calculate the isoline for any V, 0 < V < 1
Assume this cell layout
\[ X:0 \rightarrow 1, \ Y:0 \rightarrow 1, \ F(0,0) = 0, \ F(1,0) = F(1,1) = F(0,1) = 1 \]
Quiz: write pseudocode to calculate the isoline for any \( V, \ 0 < V < 1 \)

Answer: \{ return ((V, 0), (0, V)); \}
Consider arbitrary layout

\[ F(0,1) = C \quad F(1,1) = D \]

\[ A < V \quad V < B \quad V < C \quad V < D \]

(V == isovalue)

Where is isoline?

Note that the mesh coordinates are pretty simple ... you will need to take real coordinates into account.
Consider arbitrary layout

F(0,0) = A  F(1,0) = B
F(0,1) = C  F(1,1) = D

A < V
V < B
V < C
V < D
(V == isovalue)

Where is isoline?

P1 = (x,0)
P2 = (0,y)

Quiz:
What are x and y?

t = (V-A)/(B-A)
x = 0+t*(1-0)
t = (V-A)/(C-A)
y = 0+t*(1-0)
Claim: we understand one case

- Case “1”:
  - \( F(P_0) < V \)
  - \( V < F(P_1) \)
  - \( V < F(P_2) \)
  - \( V < F(P_3) \)

- Quiz: how many cases are there?
- Answer:
  - 2 possible states for each point \( P \): \( F(P) < V \) or \( V < F(P) \)
    - (note we ignore \( F(P) = V \) for today)
  - 4 vertices, so ... \( 2^4 = 16 \) cases
    - Some cases are similar to each other
The 16 cases
We explored case 14
Quiz: write down cases #s that are similar to case 14
Quiz: how many different groupings are there?
Quiz answer:
There are 4 groupings
Problem case: ambiguity!!

Solution: just pick one and go with it.
Physical interpretation of ambiguity

One way connects them up, the other separates them. What’s right?
Big idea #2: pre-computation of all cases

If you knew which case you had, then you would know how to proceed

- Pre-compute correct answers for all 16 cases and store them in a lookup table
- For each cell, identify which case it is in
- Then use corresponding lookup table to generate isosurface
Big idea #2: pre-computation of all cases

- Pre-compute correct answers for all 16 cases and store them in a lookup table
- For each cell, identify which case it is in
- Then use corresponding lookup table to generate isosurface

If you knew which case you had, then you would know how to proceed
Pre-compute correct answers for all 16 cases and store them in a lookup table

• Observations about correct answers for a case:
  – It contains one or two line segments
  – The ends of the line segments are always along edges.
Pre-compute correct answers for all 16 cases and store them in a lookup table

• The ends of the line segments are always along edges.
  – We will need to number the edges
Pre-compute correct answers for all 16 cases and store them in a lookup table

• Correct answer for this case:
  – There is one line segment
  – That line segment has end points on edge 0 and edge 3
Big idea #2: pre-computation of all cases

If you knew which case you had, then you would know how to proceed

• Pre-compute correct answers for all 16 cases and store them in a lookup table
• For each cell, identify which case it is in
• Then use corresponding lookup table to generate isosurface
For each cell, identify which case it is in

- 4 vertices
- Each has one of 2 possible classification values
  - Lower than isovalue
    - Call this “0”
  - Higher than isovalue
    - Call this “1”
  - (ignore equality case)

Quiz: write down classification value for each vertex
For each cell, identify which case it is in

- **Goal:** turn classification values into a number
  - Number should be between 0 and 15
- **Idea:** use binary numbers
  - $V_3V_2V_1V_0 \rightarrow$
    - $1 \ 1 \ 1 \ 0 \rightarrow 14$

This is case 14
The 16 cases
Big idea #2: pre-computation of all cases

If you knew which case you had, then you would know how to proceed

- Pre-compute correct answers for all 16 cases and store them in a lookup table
- For each cell, identify which case it is in
- Then use corresponding lookup table to generate isosurface
Then use corresponding lookup table to generate isosurface

```c
int numSegments[16];
numSegments[0] = 0;
...
numSegments[6] = 2;
...
numSegments[14] = 1;
numSegments[15] = 0;
```
Then use corresponding lookup table to generate isosurface

```c
int lup[16][4]; // lup == lookup
lup[0][0] = lup[0][1] = lup[0][2] = lup[0][3] = -1;
...
lup[6][0] = 0; lup[6][1] = 1; lup[6][2]=2; lup[6][3] = 3;
...
lup[14][0] = 0; lup[14][1] = 3; lup[14][2] = lup[14][3] = -1;
```
Then use corresponding lookup table to generate isosurface

```cpp
int icase = IdentifyCase(cell); // case is a reserved word in C++
int nsegments = numSegments[icase];
for (int i = 0 ; i < nsegments ; i++)
{
    int edge1 = lup[icase][2*i];
    float pt1[2] = // Interpolate position along edge1
    int edge2 = lup[icase][2*i+1];
    float pt2[2] = // Interpolate position along edge2
    AddLineSegmentToOutput(pt1, pt2);
}
```
Next Time

- Review 2D
- Extensions for 3D
- More discussion of isosurface calculation
- Project 6
- Quiz on particle advection