Interpolation

Elements of Graphics
CS324e
So far we’ve focused on how to construct static images.

One way to animate a scene would be to define a sequence of static images that form the final animation.

This is a very involved, very laborious process.

How can we simplify this process?
Tweening

- In-betweening
- Used in both traditional and digital animation
- Define distinct “keyframes” then automatically interpolate between them

(Chu and Lee, 2009)
Linear Interpolation

- Changes over time give the appearance of an animation
- Given a starting and ending target, we can change by a fixed value at each time step
- The value is (theoretically) the same from frame to frame
- This way, change happens at a linear rate
Translation over Time

- Consider a point at starting position \((x_0, y_0)\)
- Its target position is \((x_1, y_1)\)
- At each time step, its position \((x, y)\) will be on the blue line

(Wikipedia)
Moving Objects in Processing

- A simple way to change values over time is via the `draw()` function
- `draw()` is called at a (supposedly) fixed frame rate
- This frame rate determines the size of our time step
  - 60 fps = 16ms time step
  - 30 fps = 33ms time step
- Unexpected behavior can happen when the frame rate fluctuates though...
float x = 0.0;
float y = 250.0;
float dx = 5.0;

void draw() {
  background(210);
  ellipse(x, y, 30, 30);
  if (x < 250) {
    x += dx;
  }
}

void setup () {
  size(500, 500);
}

Question

- What if we want to know an object’s current (x, y) position at a particular time step?
Lerp Operation

- Processing has `lerp()` function
- Lerp is a contraction of "linear interpolation"
- Lerp determination the position between any two values, v0 and v1, based on parameter t:
  \[ v(t) = v0(1 - t) + v1(t) \]
- Parameter t can be any value between 0 and 1
- Note: Does not apply the lerp over time -- you must do that directly!
Question

If a point starts at $x_0 = 6$ and ends at $x_1 = 25$ moving along a line, what is its position on the x-axis at $t = 0.4$?
Lerp Uses

Color and texture

(ShaderForge)

Position over time

Data sets

(Wikipedia)
Cosine Interpolation

- Linear interpolation can have sharp discontinuities at each point
- Cosine interpolation smooths these transitions without requiring additional points:

\[ t2 = \frac{1 - \cos(\pi t)}{2} \]

\[ v(t) = (1 - t2)v_0 + (t2)v_1 \]
Cosine versus Linear Interpolation

Linear interpolation

Cosine interpolation

(http://paulbourke.net)
Hands-on: Linear Interpolation

Today’s activities:

1. Experiment with Processing’s `lerp()` method using different values for \( t \) (between 0 and 1)

2. Experiment with the general linear interpolation formula to recreate those results using the previous values for \( t \)

3. Create a method `move`, which takes a start position, an end position, and a step size. This method, called by `draw()`, will update the shape’s position at the given rate (step size)

4. If time allows, reimplement this method using cosine interpolation