1 Traditional Animation: Drawing on Film

At my office hours (Thursday 3-4 or Friday 11-12) pick up one strip of clear 16mm film. Use permanent felt tip markers to draw an animation. Note that each frame is centred between the two sprocket holes on the side, a little rectangle about $10mm$ by $4mm$. Also recall that film goes by at 24 frames per second—so for example, if you want an image to stay on the screen for a second, you need to draw it a full 24 times. The film should start at the top of the strip with the holes on the left.

NOTE: please don't fold or crease your film—it won't go through the projector properly if you do. Keep it rolled up in a safe place where it will not get crushed or bent.

Hand the film back to me in class Monday September 12 or if you absolutely can't, in my mailbox later that day. No late submissions—I will need some time to splice them together to show in class. **Make sure to write your name at the TOP (start) of the strip so I can tell who handed back the film.**

2 Theory: Working with Splines

2.1 Background

Note: see section 3.1 and Appendix B.4 in the text for more on splines than what I have written below. There will also be a lecture or two on this.

A spline (if you haven't seen them before) is a function that consists of parts of polynomials cut and pasted together. For example, a spline $f(t)$ could be equal to $t^2 + 1$ on the interval $[0, 1]$, then equal to $3 - t$ on the interval $[1, 2]$, and then $-t^3 + 2$ on the interval $[2, 3]$. Splines are especially useful in graphics (and animation in particular) since they can very flexibly represent or at least approximate any sort of smooth curve, and are very simple and efficient to compute.

The points at which the spline switches from one polynomial to another are called "knots". We want the spline to interpolate (go through) or approximate (go near) a list of specified values at these knots—these are called "control points". Users control the shape of the spline curve by adjusting the control points, and the computer fills in the smooth polynomials between automatically. Sometimes the control points include not just the values that the spline should interpolate/approximate but also values for the spline's derivatives.

The two big issues for splines are smoothness and control. Smoothness means how many times you can differentiate the spline and get a continuous answer, and control refers to how easy it is to adjust the control points to get the shape you want.

Between knots, a spline is just a polynomial which is infinitely smooth (all its derivatives exist). However, at the knots when the spline switches from one polynomial to another, the limit from the left might not equal to the limit from the right. If they're not equal, the spline is discontinuous. Usually splines are constrained to be continuous, so the limits are equal: the two polynomials achieve the same value at the knot they share. We sometimes say the spline is C^0 if it is continuous. The simplest C^0 spline is a piecewise linear curve, which is composed of straight line segments connecting the control points.

Even if a spline is continuous, the first derivatives of two adjacent polynomials may disagree at their common knot, so if we differentiate the spline we would see a discontinuity there. This looks like a sharp corner in the curve. Many splines are constrained to have the first derivatives equal at the knots, which means the spline has a continuous first derivative. We sometimes say then that the spline is C^1.

Similarly, you can look at if the second derivatives of the polynomials match at the knots. If so, the spline is twice continuously differentiable, or C^2, which looks even smoother. This can keep going, but in practice, not much is gained by going beyond C^2 smoothness. In fact for animation, C^1 is often good enough, and in certain cases we need the spline to be only C^0.

Splines are computed to interpolate or approximate the given values at the control points. By adjusting the control points, the user can control what the spline looks like. (And for some spline algorithms, the user can control the first derivative of the spline at the knots too). The big question is how much of the spline curve changes when the user makes an adjustment. A

scheme has "local" control if only the nearby polynomials change, and the rest of the curve stays exactly the same. A scheme has "global" control if the entire curve changes. Local control is generally a much better quality than global control: working on one part of the curve doesn't mess up parts of the curve you already perfected. However, good global control schemes at least make sure that, far away from the control point you adjust, the changes to the curve are very small. Without that property, a spline is effectively useless.

2.2 The Problem

Construct a C^2 interpolating spline with local control. Give the formulas for your construction and explain in clear English your reasoning for how it works. The input should just be a sequence of values that the spline will pass through.

Hand this part of the assignment in to me Monday September 12, either in class, or under my office door or in my mailbox before I get in Tuesday morning.