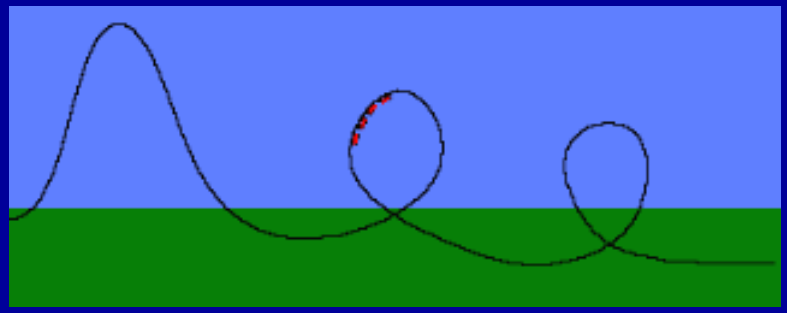
Physics of a Mass Point & Basics of Textures

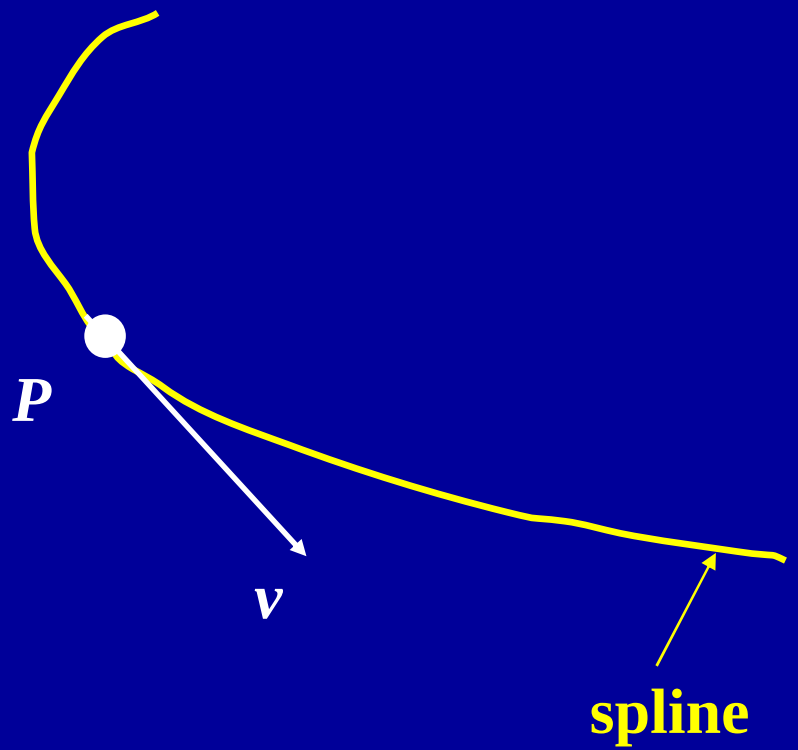
Point mass simulation
Basics of texture mapping in OpenGL

Roller coaster

- Next programming assignment involves creating a 3D roller coaster animation
- We must model the 3D curve describing the roller coaster, but how?
- How to make the simulation obey the laws of gravity?



Back to the physics of the roller-coaster: mass point moving on a spline



frictionless model, with gravity

- **Velocity vector always points in the tangential direction of the curve**

Mass point on a spline (contd.)
frictionless model, with gravity

- Our assumption is : no friction among the point and the spline
- Use the conservation of energy law to get the current velocity

chalkboard

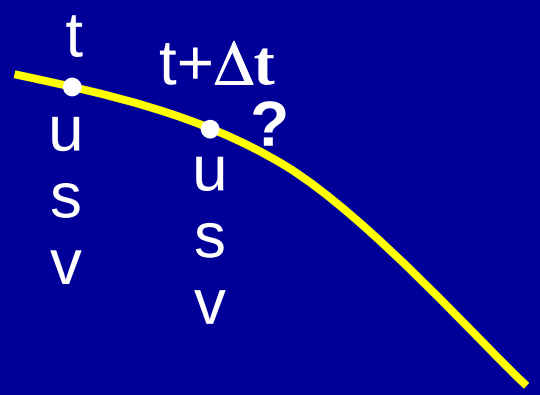
Mass point on a spline (contd.)
frictionless model, with gravity

- Given current h, we can always compute the corresponding |v|:

$$|v| = \sqrt{2g(h_{\max} - h)}$$

Simulating mass point on a spline

- **Time step Δt**
- **We have: $\Delta s = |v| * \Delta t$ and $s = s + \Delta s$.**
- **We want the new value of u, so that can compute new point location**
- **Therefore:**
  We know s, need to determine u
  Here we use the bisection routine to compute u=u(s).



Mass point simulation

- **Assume we have a 32-piece spline, with a general parameterization of $u \in [0,31]$**

chalkboard

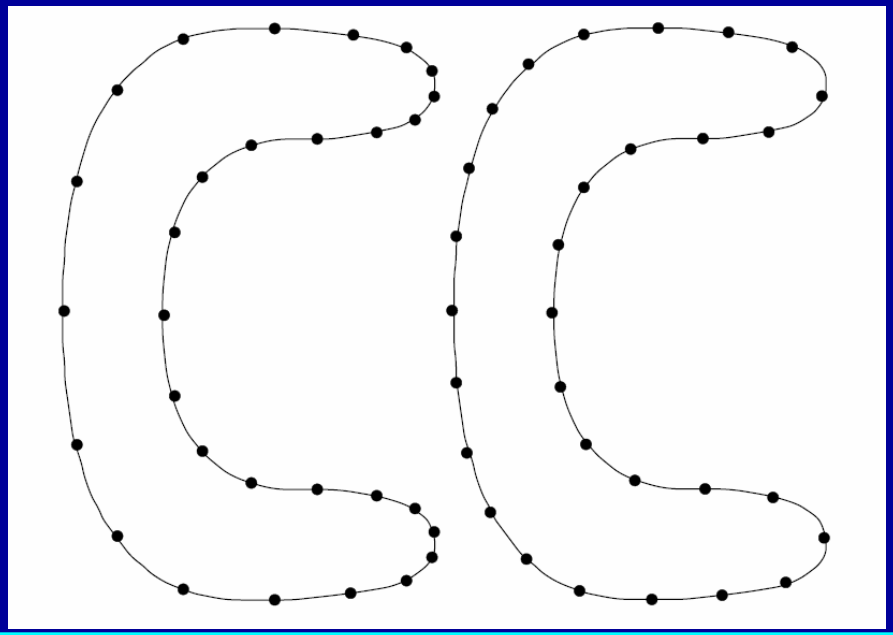
Mass point simulation

- **Assume we have a 32-piece spline, with a general parameterization of $u \in [0,31]$**

```
MassPoint(tmax) // tmax = final time
/* assume initially, we have t=0 and point is located at
u=0 */
    u = 0;
    s = 0;
    t = 0;
    While t < tmax
    {
        Assert u < 31;  // if not, end of spline reached
        Determine current velocity |v| using physics;
        s = s + |v| * Δt; // compute new arclength
        u = Bisection(u,u + delta,s); // solve for t
        p = p(u); // p = new mass point location
        Do some stuff with p, i.e. render point location, etc.
        t = t + Δt; // proceed to next time step
    }
```

Arclength Parametrization

- There are an infinite number of parameterizations of a given curve. Slow, fast, speed continuous or discontinuous, clockwise (CW) or CCW…
- A special one: arc-length-parameterization: u=s is arc length. We care about these for animation.



Arclength Parametrization

chalkboard

Arclength Parametrization Summary

- Arclength parameter s=s(u) is the distance from p(0) to p(u) along the curve

$$s(u) = \int_0^u \sqrt{x'(v)^2 + y'(v)^2 + z'(v)^2}\, dv$$

- The integral for s(u) usually cannot be evaluated analytically
- Has to evaluate the integral numerically
- Simpson's integration rule

$$\int_a^b f(x)dx = \sum_{k=1}^{(n-1)/2} \frac{h}{3}[f(x_{2k-1}) + 4f(x_{2k}) + f(x_{2k+1})] + O(h^5)$$

- Use bisection (next slide) to compute universe: u=u(s)

Computing inverse u=u(s)

- **Must have initial guess for the interval containing u**

```
Bisection(umin,umax,s)
/* umin = min value of u
   umax = max value of u; umin <= u <= umax
    s = target value */
    Forever // but not really forever
    {
        u = (umin + umax) / 2; // u = candidate for solution
        If |s(u)-s| < epsilon
            Return u;
        If s(u) > s  // u too big, jump into left interval
            umax = u;
        Else  // t too small, jump into right interval
            umin = u;
    }
```