

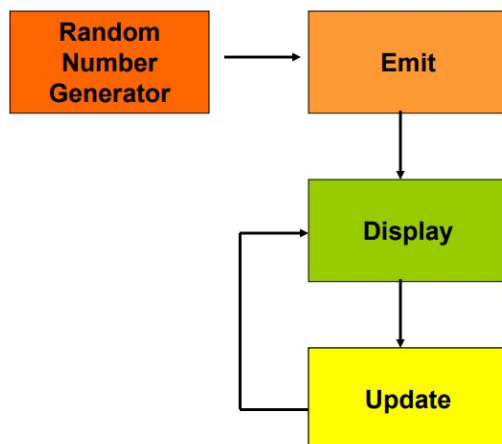
- Particle systems:

General idea

- Are used to simulate the appearance of particulate, hairy, or fuzzy phenomena.
- Involve the animation of large collections of (perhaps tiny) particles which have various graphics characteristics.
- Were originally developed by Pixar's Bill Reeves for the "Genesis Sequence" in the movie *Star Trek II: The Wrath of Khan*
- Have been used to create effects of fire, smoke, rain, snow, fireworks, disintegration, dust, sand, explosions, flow, waterfalls, stars, comets, plants, hair, fuzz. Surely many more.

Equations

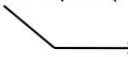
The basic process is this:



Data structures

The Emitter gives each particle a:

- Birth time
- Death time
- Start location
- Start velocity
- Start color
- Start size
- Start alpha (blending factor)


$$Color = (1 - \alpha)Color_0 + \alpha Color_1$$

Plus, any information about how these quantities change over time

Points

2D and 3D markers

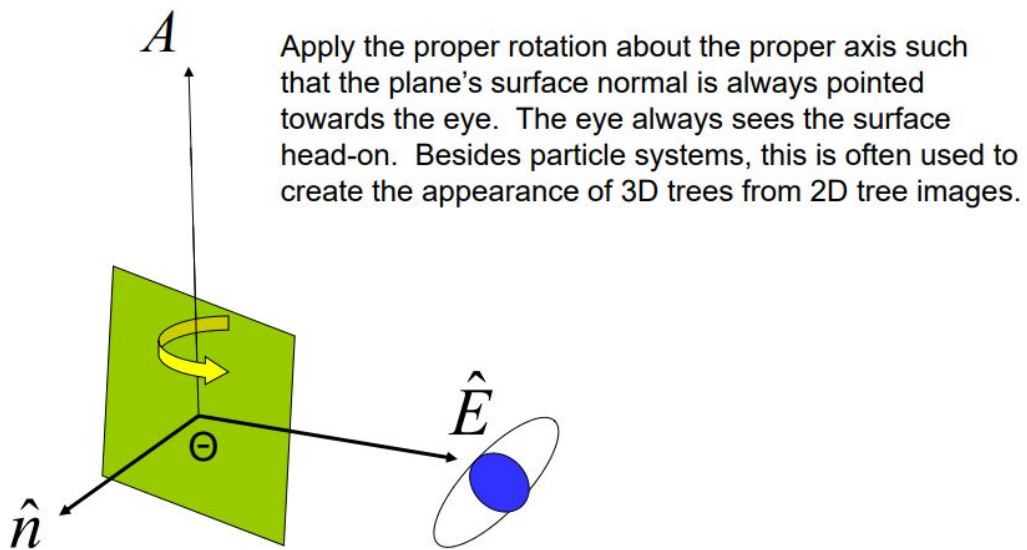
Sprites

A “sprite” is a 3D object pre-rendered to a flat 2D texture and “slipped” into a certain depth in the scene.



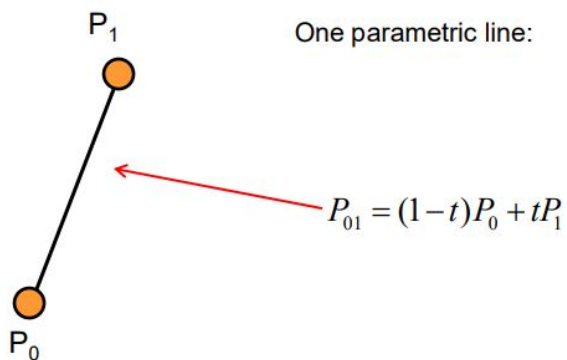
Billboarding

Billboarding



- Cubic Bézier Curves:

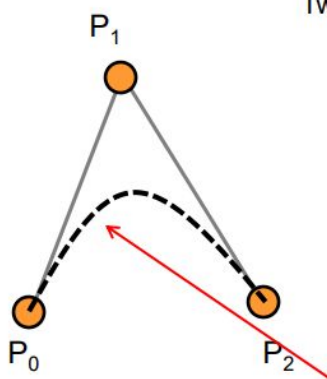
General Bézier formulation



1 1

How to derive the equation

Bézier Curves: the Derivation



Two parametric lines, blended:

$$P_{01} = (1-t)P_0 + tP_1$$

$$P_{12} = (1-t)P_1 + tP_2$$

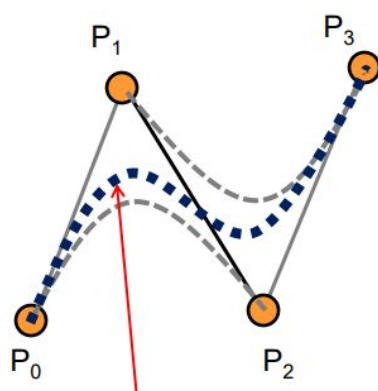
$$P_{012} = (1-t)P_{01} + tP_{12}$$

$$= (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

$$\begin{matrix} & 1 & & 1 \\ 1 & & 2 & & 1 \end{matrix}$$

2 end points + 2 control points.

Bézier Curves: the Derivation



Three parametric lines, blended:

$$P_{01} = (1-t)P_0 + tP_1$$

$$P_{12} = (1-t)P_1 + tP_2$$

$$P_{012} = (1-t)P_{01} + tP_{12}$$

$$= (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2$$

$$\begin{matrix} & & 1 & & 1 \\ & 1 & & 2 & & 1 \\ 1 & & 3 & & 3 & & 1 \end{matrix}$$

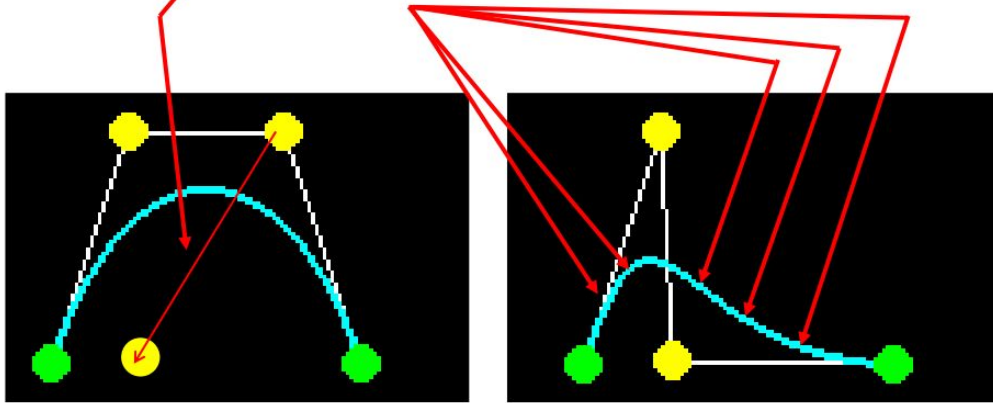
$$P_{123} = (1-t)P_{12} + tP_{23}$$

$$= (1-t)^2 P_1 + 2t(1-t)P_2 + t^2 P_3$$

$$P_{0123} = (1-t)P_{012} + tP_{123}$$

$$= (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

A Small Amount of Input Change Results in a Large Amount of Output Change



The General Form of Cubic Curves

$$P(t) = A + Bt + Ct^2 + Dt^3$$

In this form, you need to determine **4** quantities (A, B, C, D) in order to use the equation. That means you have to provide **4** pieces of information. In the **Bézier curve**, this happens by specifying the 4 points.

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3$$

Rearranging gives A, B, C, and D for the Bézier curve:

$$A = P_0$$

$$B = -3P_0 + 3P_1$$

$$C = 3P_0 - 6P_1 + 3P_2$$

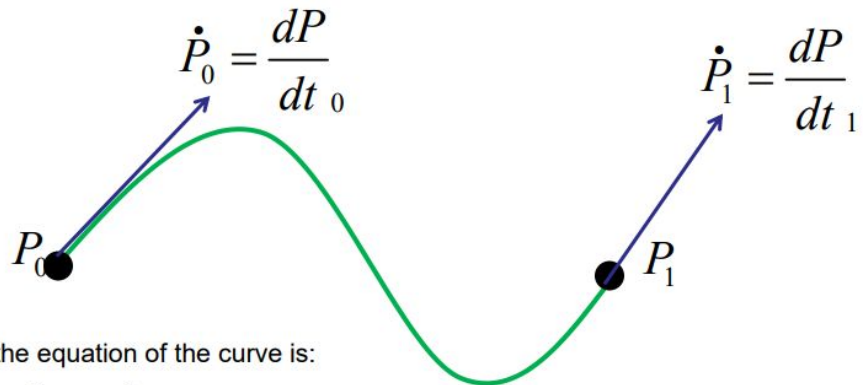
$$D = -P_0 + 3P_1 - 3P_2 + P_3$$

- Cubic Hermite/Coons curves:

2 end points + 2 slopes. (You do not need to know the equation.)

Another Approach: Coons (also called Hermite) Cubic Curves

Another approach to specifying the 4 pieces of information would be to give a start point, an end point, a start parametric slope, and an end parametric slope.



If we do this, then the equation of the curve is:

$$P = A + Bt + Ct^2 + Dt^3$$

where: $A = P_0$

$$B = \dot{P}_0$$

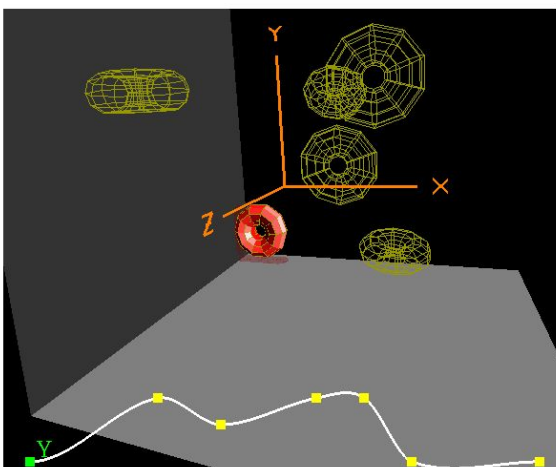
$$C = -3P_0 + 3P_1 - 2\dot{P}_0 - \dot{P}_1$$

$$D = 2P_0 - 2P_1 + \dot{P}_0 + \dot{P}_1$$

- Keyframe/Keytime Animation:

General idea

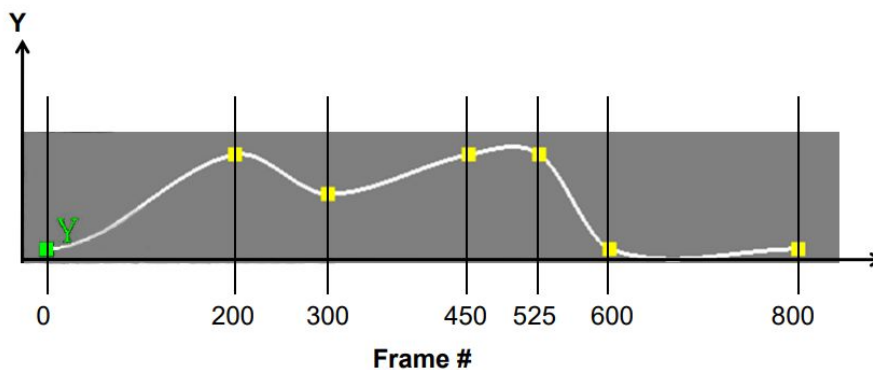
Now, Let's Apply this to the Y Translation of a Keyframe Animation



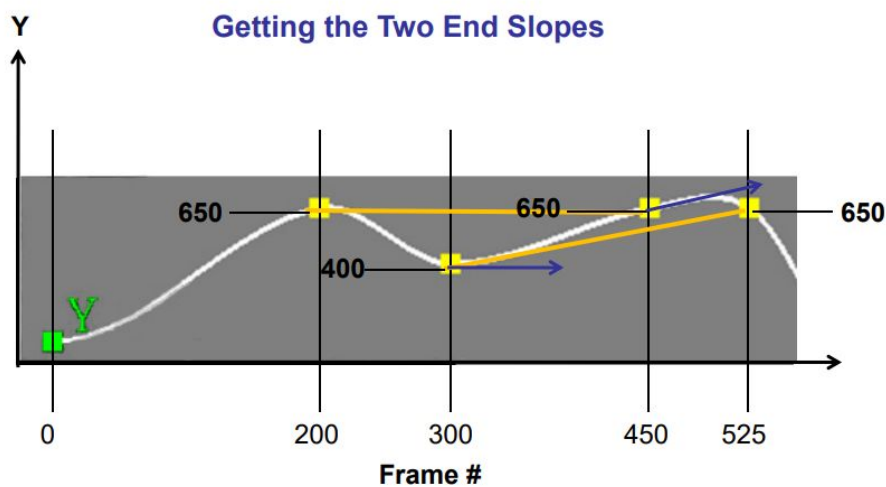
To make this simple to use, our goal is to just specify the keyframe values, not the slopes. We will let the computer compute the slopes for us, which will then result in being able to compute the in-between frames.

Piecewise cubic curves

The “Y vs. Frame” Curve Looks Like This



Determining derivatives (slopes) where the curves connect.



To get the slope at a keyframe point, draw a line between one keyframe back from that one and one keyframe ahead

- Dynamic Physics:

$$a = F/m$$

$$\Delta v = a * \Delta t$$

$$\Delta x = v * \Delta t$$

Discrete Dynamics

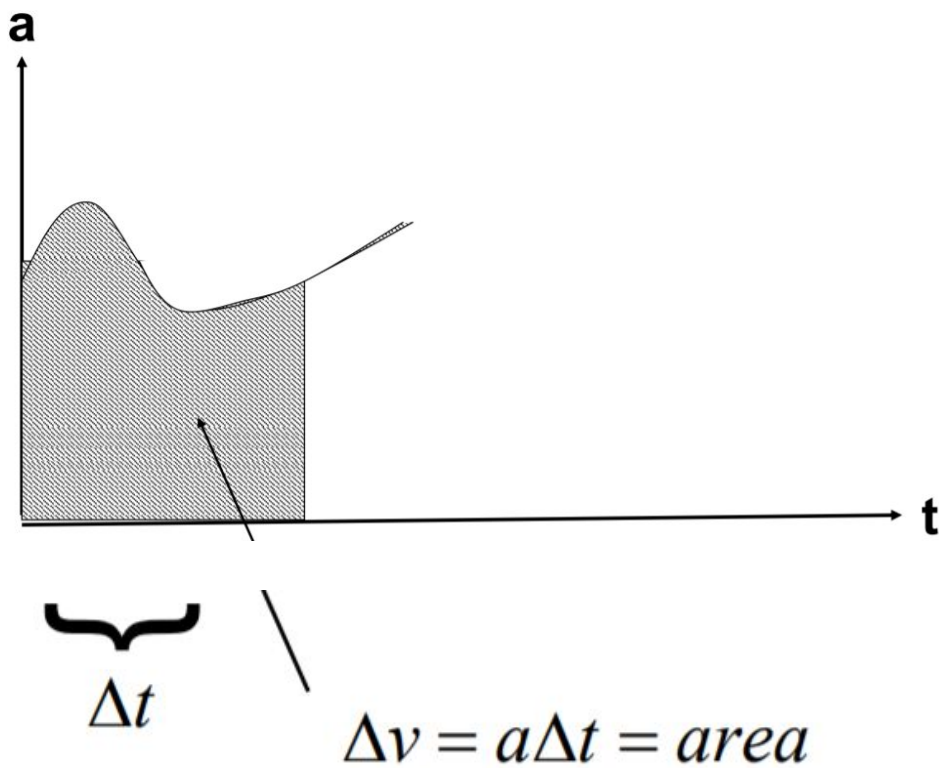
2. Force equals mass times acceleration ($F=ma$)

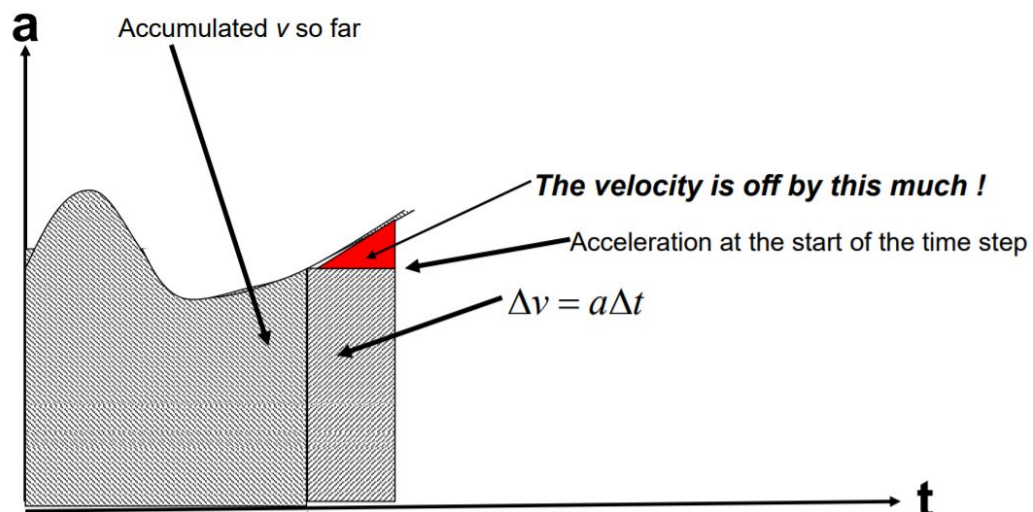
$$a = \frac{F}{m}$$

$$\frac{\Delta v}{\Delta t} = a \quad \rightarrow \quad \Delta v = a\Delta t \quad \rightarrow \quad v = \sum \Delta v = \sum a\Delta t$$

$$\frac{\Delta x}{\Delta t} = v \quad \rightarrow \quad \Delta x = v\Delta t \quad \rightarrow \quad x = \sum \Delta x = \sum v\Delta t$$

These can be thought of as summing areas under curves



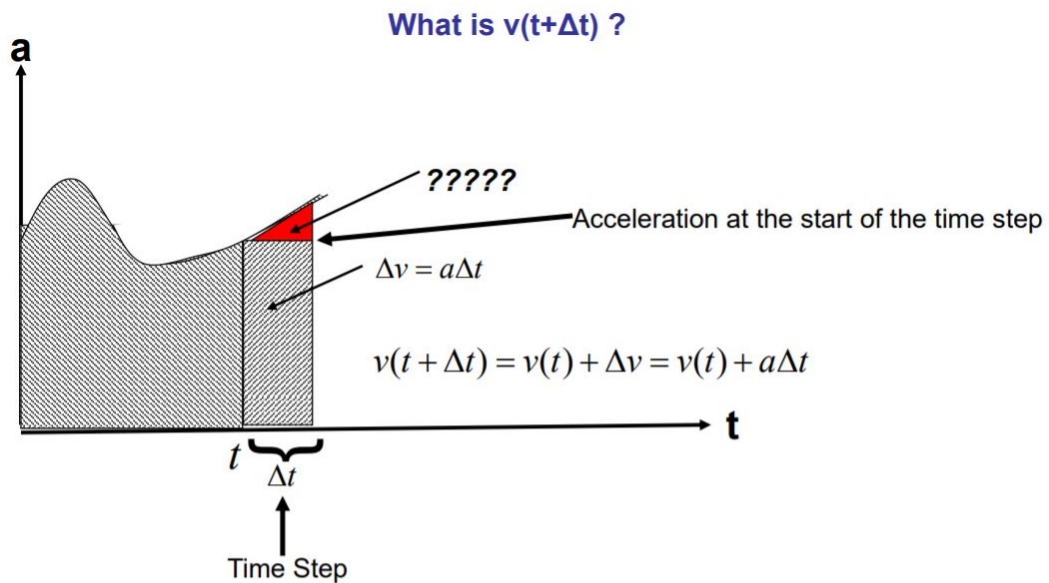


$$v(t + \Delta t) = v(t) + \Delta v = v(t) + a\Delta t$$

This is close, but is clearly not exactly right!

Time Step

First Order solution



The problem is that we are treating all of the quantities as if they always have the value that they had at the start of the Time Step, even though they don't.

This is known as a *First Order solution*.

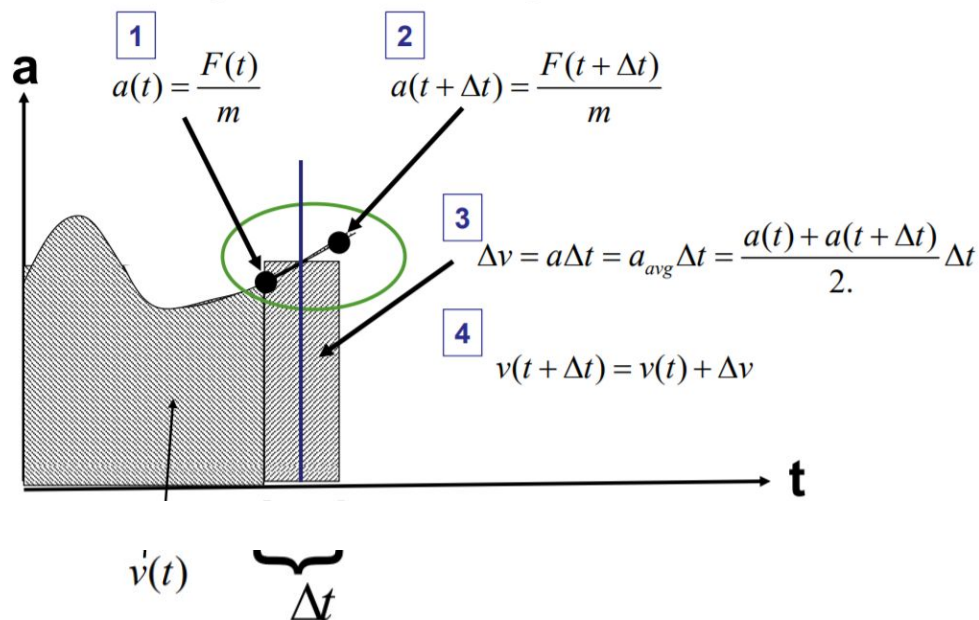
Why a First Order solution can go unstable.

see last pic

Second Order solution

What is $v(t+\Delta t)$?

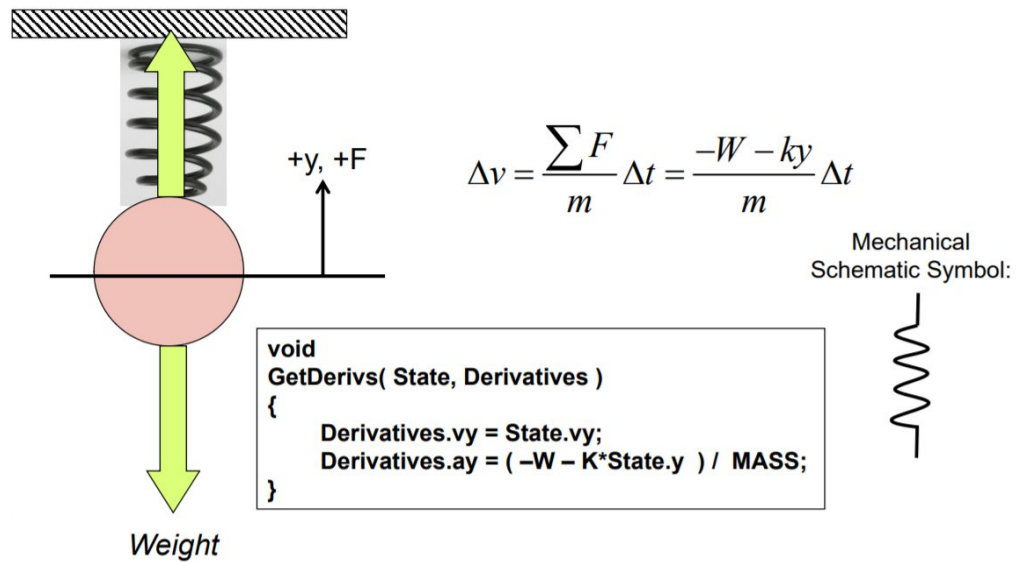
A *Second Order solution* is obtained by doing the First Order solution, determining all quantities at *time* = $t + \Delta t$ then averaging them with the quantities at *time* = t and then treating them as constant throughout the interval.



- Dynamic Physics: (Know these equations.)
springs

Solving Motion where there is a Spring

$$F_{spring} = -ky \leftarrow \text{This is known as Hooke's law}$$



fluid resistance (drag)

Air Resistance Force

$$F_{drag} = \frac{1}{2} \rho v_y^2 A C_d$$

Drag Coefficient
 Fluid density
 Y Velocity
 Cross-sectional area

+y
 Body Falling
 W

Air Resistance always acts in a direction opposite to the velocity of the object

Solving Motion where there is Air Resistance

$$\Delta v = \frac{\sum F}{m} \Delta t = \frac{-W - \text{Sign}(v_y) \frac{1}{2} \rho v_y^2 A C_d}{m} \Delta t$$

$$\rho_{air} = 1.293 \frac{kg}{m^3}$$

The **Sign()** function returns +1. or -1., depending on the sign of argument

Terminal Velocity

When a body is in free fall, it is being accelerated by the force of gravity. However, as it accelerates, it is encountering more and more air resistance force. At some velocity, these two forces balance each other out and the velocity becomes constant, that is, $\Delta v = 0$.

This is known as the **terminal velocity**.

$$\Delta v = \frac{-W + \frac{1}{2} \rho v_y^2 A C_d}{m} \Delta t$$

The velocity becomes constant when $\Delta v = 0$:

$$W - \frac{1}{2} \rho v_y^2 A C_d = 0$$

$$v_t = \sqrt{\frac{2W}{\rho A C_d}}$$

Human Terminal Velocity

Assume:

Weight = 200 pounds = 890 Newtons

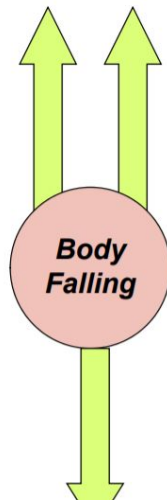
$C_d = 1.28$

$A = 6 \text{ ft}^2 = 0.558 \text{ m}^2$

$\rho_{air} = 1.293 \frac{\text{kg}}{\text{m}^3}$

$$v_t = \sqrt{\frac{2W}{\rho A C_d}} = 43.90 \frac{\text{m}}{\text{sec}} \approx 98 \text{ mph}$$

How about a Cliff Jumper on a Bungee Cord?



$$F_{spring} = -ky$$

$$F_{drag} = -\text{Sign}(v_y) \frac{1}{2} \rho v_y^2 C_d A$$

$$\Delta v = \frac{\sum F}{m} \Delta t = \frac{-W - ky - \text{Sign}(v_y) \frac{1}{2} \rho v_y^2 A C_d}{m} \Delta t$$

Coulomb Damping

This is very much like drag force, but it is the resistance of a fluid being squeezed through a small opening. The resisting force is proportional to the velocity:



$$F_{damping} = -cv$$

Damping
Coefficient

Velocity

lift

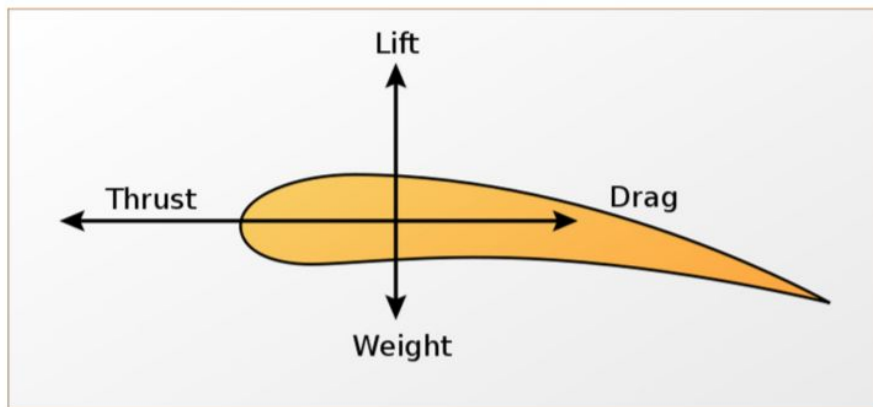


Lift – Another Good Force to Know About

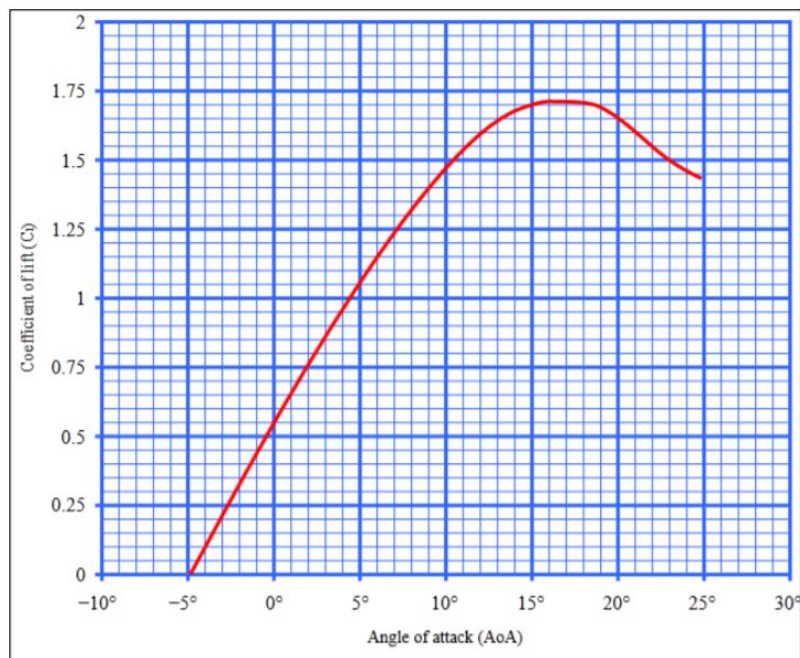
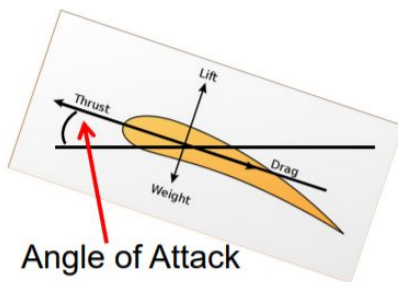
$$F_{lift} = \frac{1}{2} \rho v^2 A C_L$$

← Coefficient of Lift, for a given angle of attack

Air density Airspeed Platform area



http://en.wikipedia.org/wiki/Lift_%28force%29

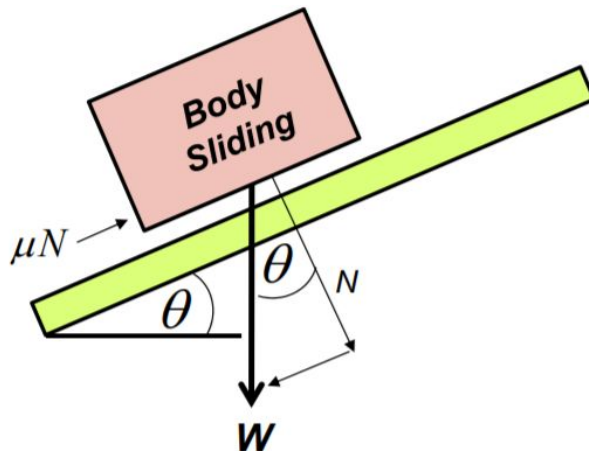


friction

$$F_{friction} = \mu N$$

Normal force (i.e., amount of force that is perpendicular to the surface)

Coefficient of Friction



buoyancy

Buoyancy – Another Good Force to Know About

Archimedes' Principle says that the buoyancy force on an object in a fluid is the weight of the fluid that is being displaced by the object.

$$\left. \begin{array}{l} \rho_{air} = 4.66 \times 10^{-5} \text{ pounds / in}^3 \\ \rho_{helium} = 0.65 \times 10^{-5} \text{ pounds / in}^3 \end{array} \right\} \text{Densities}$$

So, for a helium balloon that is one foot in diameter (i.e., radius=6 inches), it has its weight pulling it down and a buoyancy force pushing it up. The net force pushing it up because of the gas inside the balloon is:

$$F_{buoyancy} = V_{balloon} \rho_{helium} - V_{balloon} \rho_{air} = V_{balloon} (\rho_{helium} - \rho_{air})$$

$$V_{balloon} = \frac{4}{3} \pi r^3 = 904.78 \text{ in}^3$$

$$F_{buoyancy} = 904.78 \text{ in}^3 (4.01 \times 10^{-5} \text{ pounds / in}^3) = 0.036 \text{ pounds}$$

- Collision Avoidance:

Functional Animation

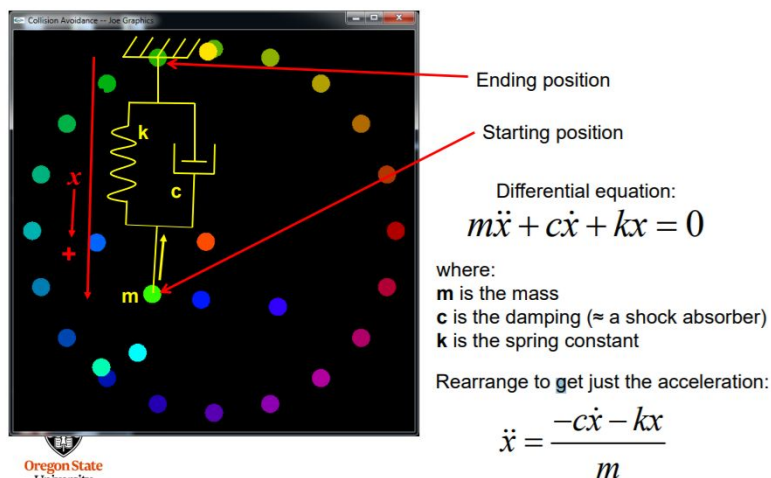
In Functional Animation, we setup a fake force system (with fake springs and other mechanical components) to make an object “want” to go to a certain place without us having to actually animate it to go there.

If this was all we were going to do, then keyframe animation would get the object there just as well.

But, the big advantage of Functional Animation is that we can add other fake forces to make the objects behave in more complex ways, such as avoiding each other.

basic ideas

First Goal – Make the Free Body Move Towards its Final Position



purpose of the coefficients in our collision avoidance example (ie, what effect each of them has).

m is mass

c is the damping

k is the spring constant

A larger k value:

- Gets the object to its goal faster
- Possibly overshoots

A larger c value:

- Gets the object to its goal slower
- Decreases spurious wiggles

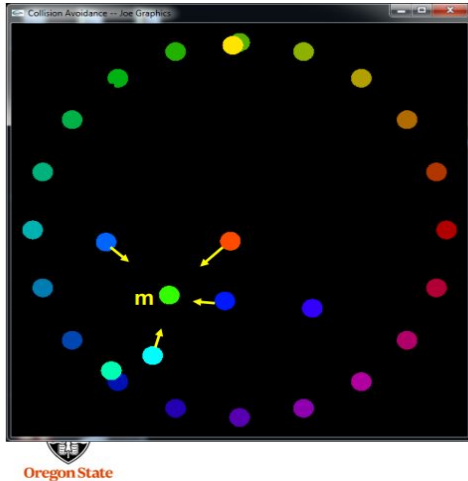
A larger Repulsion Coefficient:

- Objects give each other a wider berth
- It can get unnecessarily wide

A larger Repulsion Exponent:

- Influence waits to start until the objects are closer
- Influence increases quickly as the objects get closer
- Sometimes the influence increases too quickly and the objects do less avoiding and more bouncing

Second Goal – Make the Free Body Want to Move Away From All the Other Bodies



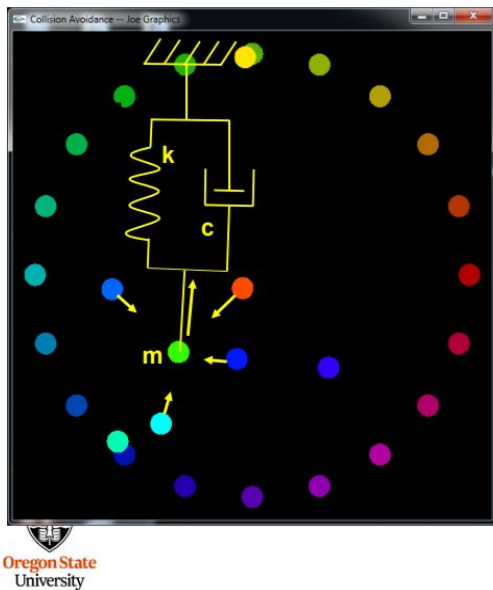
Differential equation:

$$m\ddot{x} = \sum F$$

Rearrange to get just the acceleration:

$$\ddot{x} = \frac{\sum F}{m}$$

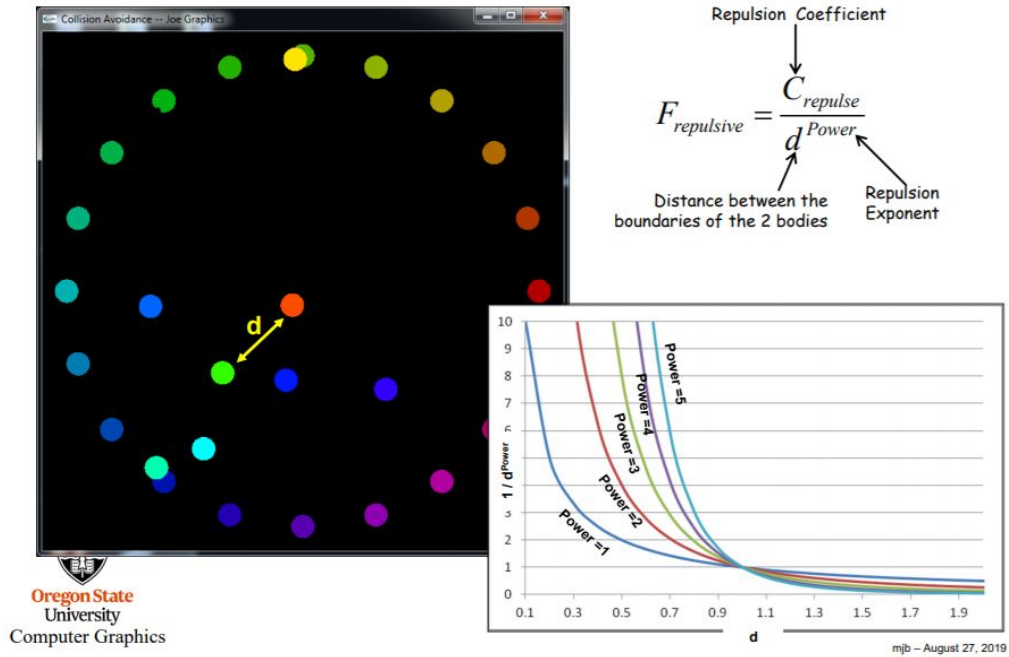
Total Goal – Make the Free Body Move Towards its Final Position While Being Repelled by the Other Bodies



$$m\ddot{x} + c\dot{x} + kx = \sum F$$

$$\ddot{x} = \frac{\sum F - c\dot{x} - kx}{m}$$

Repulsive Force



Accelerating an Object Towards a Target, Under the Influence of Outside Forces

$$m\ddot{x} + c\dot{x} + kx = \sum F$$

Fundamental equation of a second order system, if anchored at the origin

$$m\ddot{x} + c\dot{x} + k(x - x_T) = \sum F$$

Fundamental equation of a second order system, if anchored at a target position. (We're assuming that the target final velocity wants to be 0.)

$$\ddot{x} + c\dot{x} + k(x - x_T) = \sum F$$

We're not doing a real physics simulation, just going for an effect. Therefore, we can unitize the mass, and scale c , k , and the external forces appropriately

$$\ddot{x} = -c\dot{x} - k(x - x_T) - \sum F$$

Solve for the acceleration that moving towards the target needs and the outside forces influence

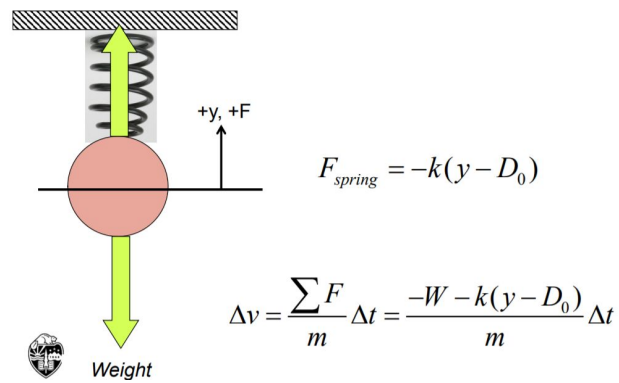
\iint Use that acceleration to compute the next position and velocity (1st order, 2nd order, 4th order, etc.)

- Modeling the World as a Mesh of Springs: general idea, lumped masses, springs, damping, $a=F/m$, 1D, 2D, 3D, physical barriers, instability, chain, cloth, jello.

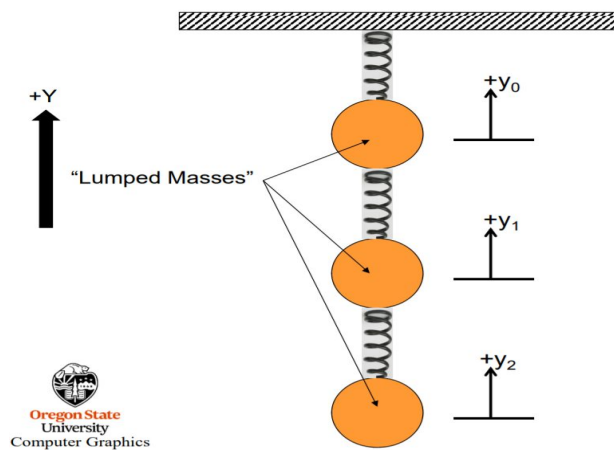
general idea

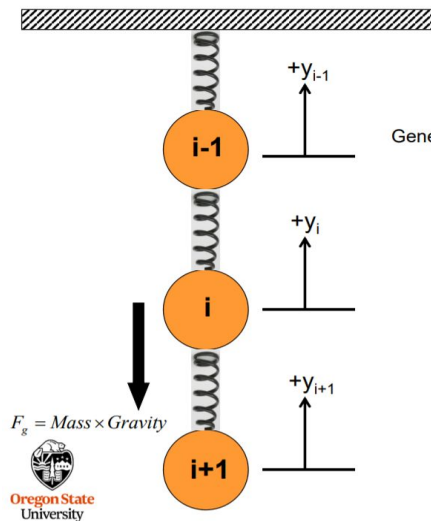
Solving for Motion where there is a Spring

2



lumped masses





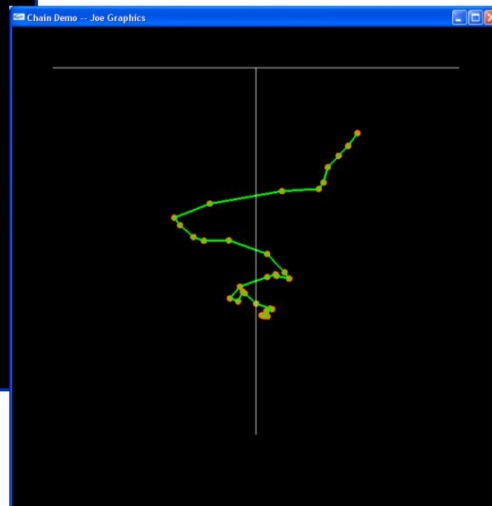
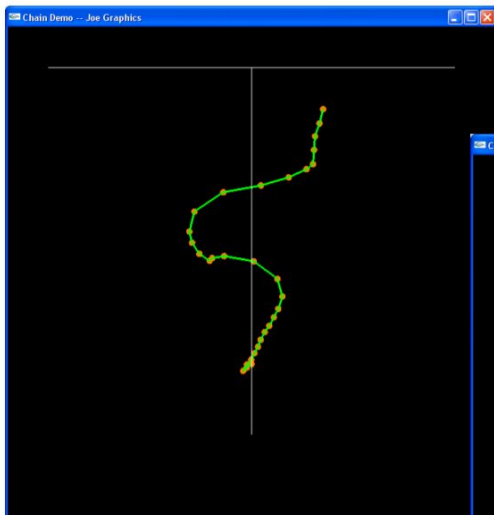
$$F_{i-1 \rightarrow i} = k(Y_{i-1} - Y_i - D_0)$$

$$F_{i+1 \rightarrow i} = k(Y_{i+1} - Y_i - D_0)$$

String

Simulating a String

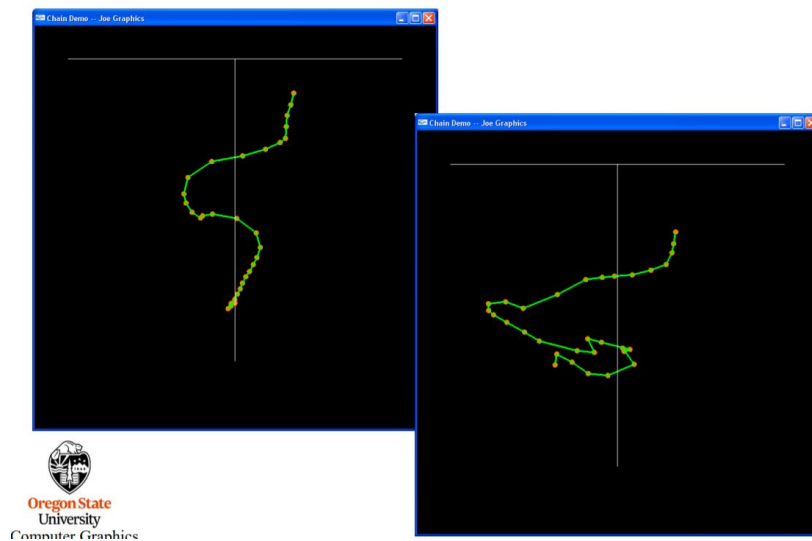
13



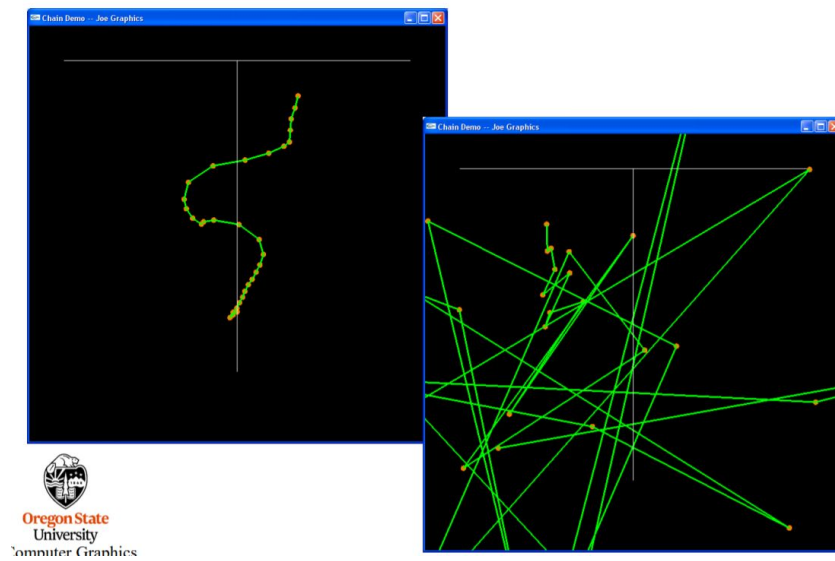
damping

Less Damping

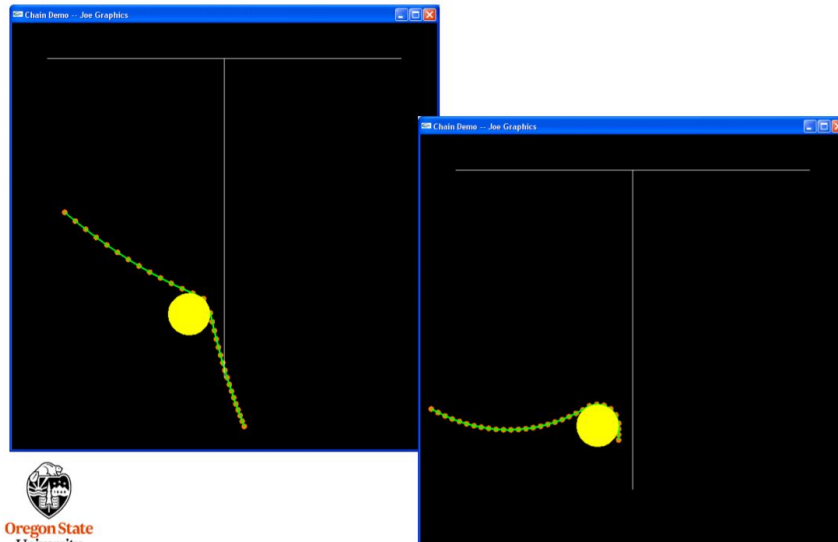
14



First Order Instability



Placing a Physical Barrier in the Scene



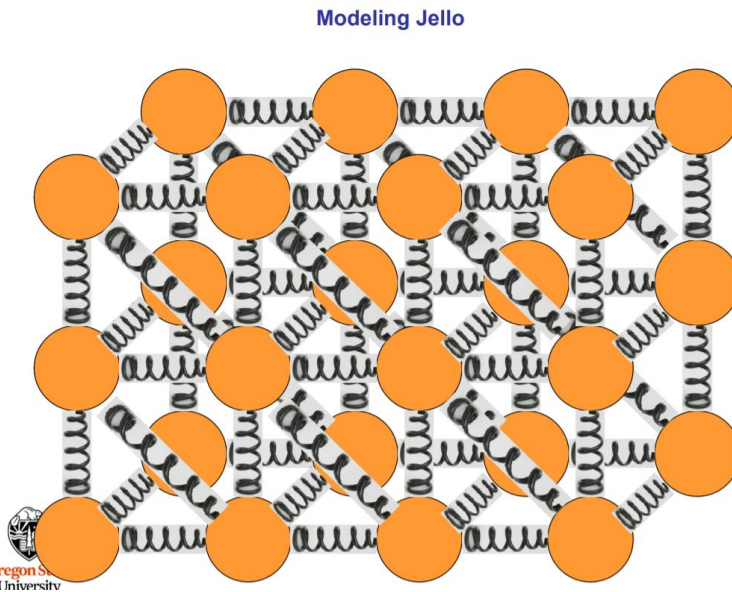
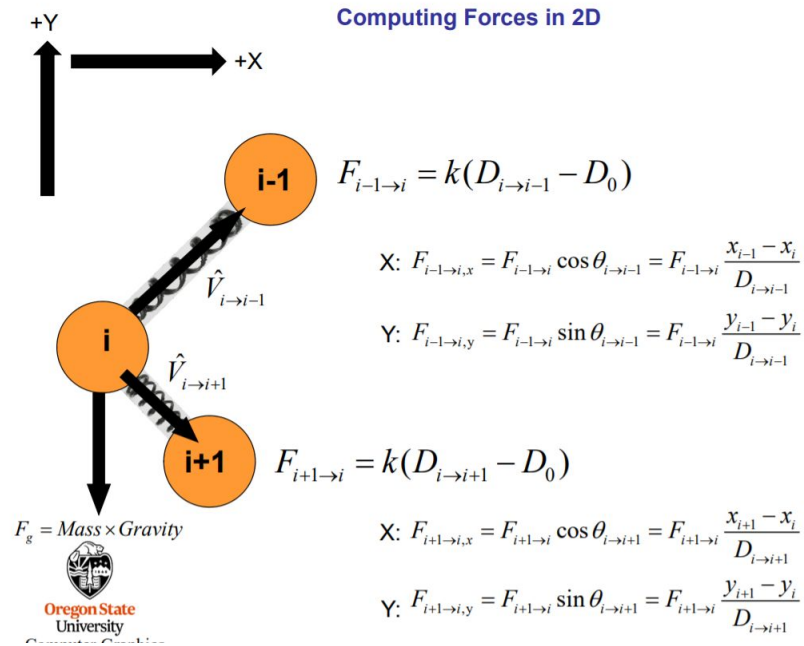
Placing a Physical Barrier in the Scene

```

if( DoCircle )
{
    for( int i = 0; i < NUMLINKS; i++ )
    {
        float dx = Links[i].x - CIRCX;
        float dy = Links[i].y - CIRCY;
        float rsqd = dx*dx + dy*dy;
        if( rsqd < CIRCRC*CIRCRC )
        {
            float r = sqrt( rsqd );
            dx /= r;
            dy /= r;
            Links[i].x = CIRCX + CIRCRC * dx;
            Links[i].y = CIRCY + CIRCRC * dy;
            Links[i].vx *= dy;
            Links[i].vy *= -dx;
        }
    }
}

```

Vector from circle center to the lumped mass
 If the lumped mass is inside the circle ...
 Unit vector from circle center to the lumped mass.
 $dx = \cos\theta$
 $dy = \sin\theta$
 Push the lumped mass from inside the circle to the circle's surface
 Keep just the tangential velocity

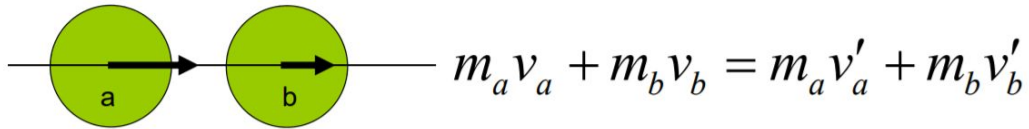


- Collision Physics:
Momentum

The **momentum** of an object is defined as its mass multiplied by its velocity:

$$\text{Momentum} = mv$$

In a collision, the total momentum after the impact is equal to the total momentum before the impact. Always.



where the primes ' refer to velocities after the impact

This is referred to as the **Conservation of Momentum Law**

Momentum is always conserved through **any** collision

energy

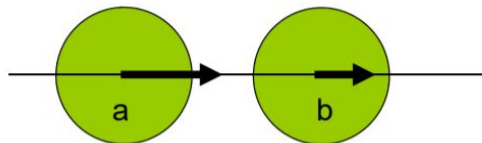
The **energy** of an object is defined as one half of its mass multiplied by its velocity squared:

$$Energy = \frac{1}{2}mv^2$$

coefficient of restitution

In a collision, energy is conserved in the entire system, but not necessarily in the form of velocities. (It can become permanent deformation, heat, light, etc..)

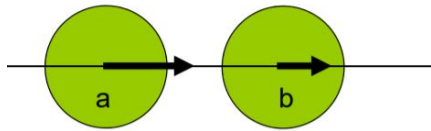
This loss of velocity is expressed as the **Coefficient of Restitution** (COR). The COR, e , is how much less the relative velocities of the objects are after impact than they were before impact:



$$v'_b - v'_a = -e(v_b - v_a)$$

(the negative sign is there to indicate the "bounce")

The Physics of Collisions – Combining Momentum and Restitution Laws



Starting with these two equations:

$$m_a v_a + m_b v_b = m_a v'_a + m_b v'_b$$

$$v'_b - v'_a = -e(v_b - v_a)$$

Treat the two initial velocities as inputs and solve for the two resulting velocities. This gives:

with Immoveable Objects

To treat the case of mass b being an immoveable object, such as the ground or a solid wall, solve for the resulting velocities taking the limit:

$$\begin{aligned} \lim_{m_b \rightarrow \infty} v'_a &= \frac{m_a v_a + m_b v_b + e m_b (v_b - v_a)}{m_a + m_b} \\ &= \lim_{m_b \rightarrow \infty} \left[\frac{m_a v_a}{m_a + m_b} + \frac{m_b v_b}{m_a + m_b} + \frac{e m_b (v_b - v_a)}{m_a + m_b} \right] \\ &= [0 + v_b + e(v_b - v_a)] \end{aligned}$$

Since mass b is immoveable, its velocity is zero, so that a's post-collision velocity is:

$$v'_a = -e v_a$$

Collisions – Experimentally Determining the Coefficient of Restitution

Velocities are hard to measure live, but distances are not.

So, drop the object from a height h , and measure its bounce to a height h' :

Before the bounce:

$$v_2^2 = 0^2 + 2gh$$

$$v = \sqrt{2gh}$$

After the bounce:

$$0^2 = v'^2 - 2gh'$$

$$v' = \sqrt{2gh'}$$

$$|v'| = e|v|$$

$$e = \frac{v'}{v} = \frac{\sqrt{2gh'}}{\sqrt{2gh}} = \sqrt{\frac{h'}{h}}$$

elastic collisions

The Physics of Collisions – Totally Elastic Collisions

What happens when $e=1$? The two fundamental equations are

$$m_b v'_b + m_a v'_a = m_b v_b + m_a v_a$$

$$v'_b - v'_a = -e(v_b - v_a) = (v_a - v_b)$$

Rearranging:

$$m_a (v'_a - v_a) = m_b (v_b - v'_b)$$

$$v'_a + v_a = v_b + v'_b$$

The Physics of Collisions – Elastic Collisions

$$m_a (v'_a - v_a) = m_b (v_b - v'_b)$$
$$v'_a + v_a = v_b + v'_b$$

Then, multiplying the two together gives:

$$m_a v'^2_a - m_a v_a^2 = m_b v_b^2 - m_b v'^2_b$$

Or:

$$\frac{1}{2} m_b v'^2_b + \frac{1}{2} m_a v'^2_a = \frac{1}{2} m_b v_b^2 + \frac{1}{2} m_a v_a^2$$

This shows that energy is conserved when the Coefficient of Restitution is 1.0

plastic collisions

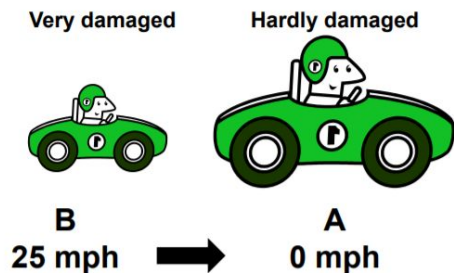
The Physics of Collisions – Totally Plastic Collisions

$$v'_a = \frac{m_a v_a + m_b v_b + e m_b (v_b - v_a)}{m_a + m_b}$$
$$v'_b = \frac{m_a v_a + m_b v_b - e m_a (v_b - v_a)}{m_a + m_b}$$

If $e=0$, then the two objects stick together and end up with the same resulting velocity:

$$v'_a = v'_b = \frac{m_a v_a + m_b v_b}{m_a + m_b}$$

One of my Jury Duties: Two vehicles collide.
One is very damaged, the other hardly at all.
What happened? Who's right?

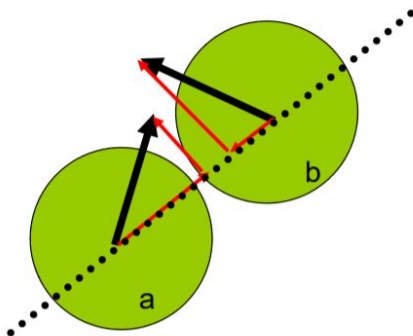


$$\begin{aligned} v_a &= 0. \text{ mph} \\ v_b &= 25 \text{ mph} = 11.2 \text{ m/sec} \\ m_a &= 2.0 \\ m_b &= 1.0 \\ e &= .30 \end{aligned}$$

$$v'_a = \frac{v_b + e v_b}{m_a + m_b} = \frac{11.2(1.3)}{3} = 4.9 \text{ m/sec}$$

$$v'_b = \frac{m_b v_b - e m_a v_b}{m_a + m_b} = \frac{11.2(1 - .3 * 2)}{3} = 1.5 \text{ m/sec}$$

oblique impacts. (You don't need to know the v_a' and v_b' equations.)
The Physics of Collisions – Oblique Impacts



Oblique Impacts are then handled by using vector math to determine the direct and tangential velocity components with respect to the **Line of Impact**.

The direct components are changed using the equations we just derived.

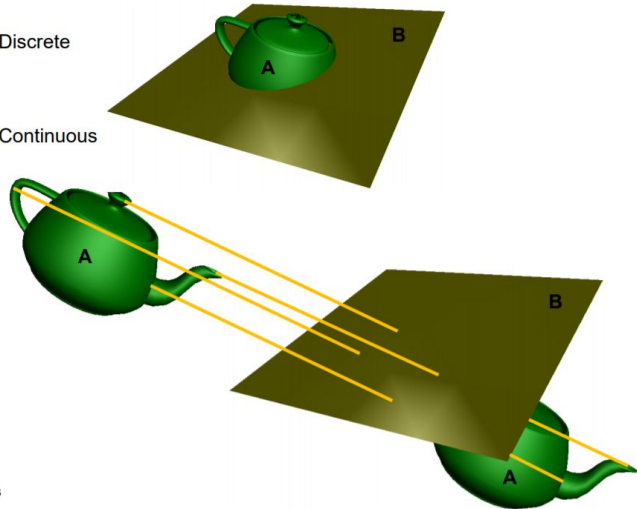
The tangential components are left unchanged. (This assumes no friction.)

The new components are then combined to produce the resulting velocity vectors.

- Collision Detection:
Discrete vs. continuous.

1. Discrete

2. Continuous



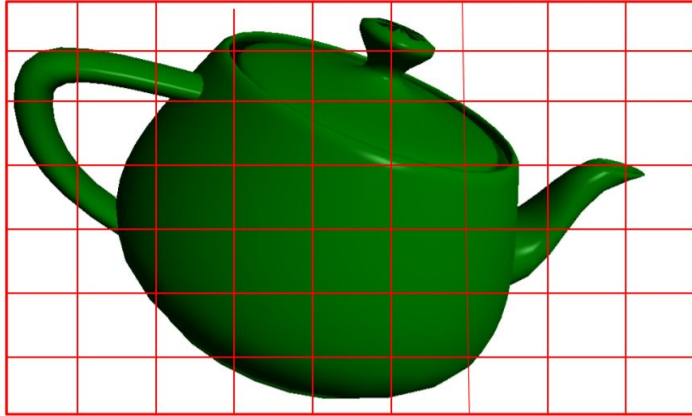
Detecting Collisions Between Two Objects

1. Do as many fast rejections as you can
 2. Do hierarchical fast rejections
 3. Discrete: compare all edges of Object A against all faces of Object B
 3. Continuous: create “pseudo-edges” by connecting respective points in Object A across the time step, then compare all these pseudo-edges of Object A against all faces of Object B
- Avoiding having to compute collisions at all (bounding boxes, bounding spheres, hierarchical bounding)

Bounding boxes

Try to Simplify the Intersection Test– Break the Scene into a Grid

4



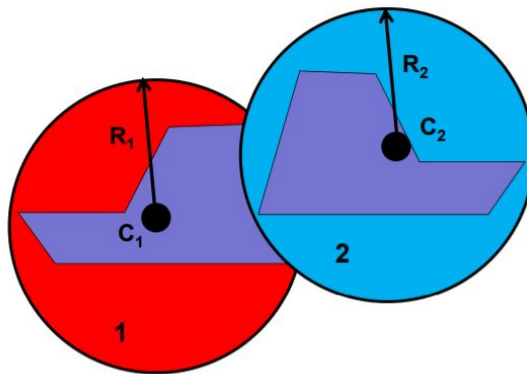
Discrete: You only have to do intersection tests against objects that live in the same grid square.

bounding spheres

Try to Simplify the Intersection Test-- A Bounding Sphere



Try to Simplify the Intersection Test- Bounding Spheres



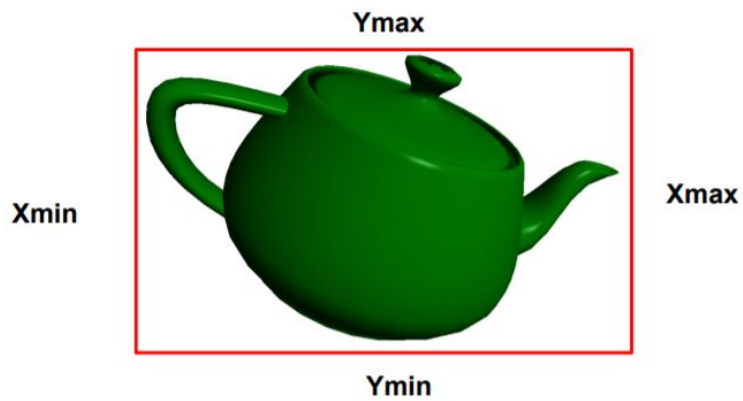
These spheres overlap if:

$$\text{Distance}(C_1, C_2) < R_1 + R_2$$

To avoid the square root:

$$\text{Distance}^2(C_1, C_2) < (R_1 + R_2)^2$$

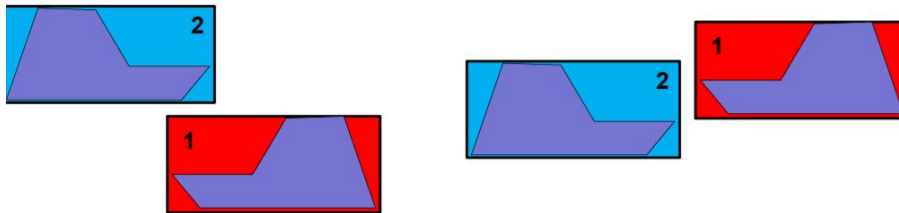
Try to Simplify the Intersection Test-- A Bounding Box



Try to Simplify the Intersection Test- Bounding Boxes

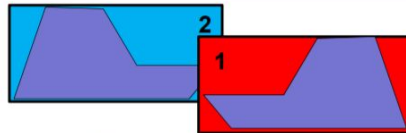
8

Quickly compare two objects by fitting each with a bounding box and then comparing the two bounding boxes.



These boxes **do not** overlap if:

$$X_{\max_1} < X_{\min_2} \parallel Y_{\max_1} < Y_{\min_2} \parallel X_{\max_2} < X_{\min_1} \parallel Y_{\max_2} < Y_{\min_1}$$



These boxes **do** overlap if:

$$X_{\max_1} > X_{\min_2} \ \&\& \ Y_{\max_1} > Y_{\min_2} \ \&\& \ X_{\max_2} > X_{\min_1} \ \&\& \ Y_{\max_2} > Y_{\min_1}$$



Oregon State University
Computer Graphics

Try to Simplify the Intersection Test- -- Two Types of Bounding Boxes

9

Axis-Aligned Bounding Box (AABB)

Check for overlap by looking for overlap in just X, then just Y, then just Z



Arbitrary-Oriented Bounding Box (AOBB)

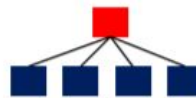
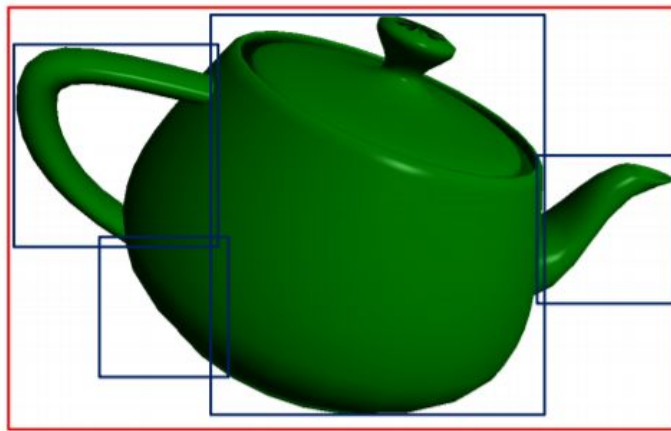
This is a tighter fit around the object, but the overlap comparison is more involved



Oregon State University
Computer Graphics

mjb - August 27, 2019

A Hierarchy of Bounding Boxes



<http://web.engr.oregonstate.edu/~mjb/cs491/Handouts/collision-detection.1pp.pdf>