Web Based Graphics & Virtual Reality Systems

Simple Animation, Spline and Interpolation and Particle System

Recap

- In the previous lectures, we have a detail study on the graphics rendering pipeline
- Our focuses were on rendering one single frame
- This lecture, we will study the methods of creating animations, so that smooth movements of objects can be created

Computer Animation

- Sequence of still images being displayed continuously
  - Moving pictures
- For a real-time rendering engine, if we can render fast enough, we can create real-time animation
  - Commonly require **> 30 fps**
  - Otherwise, human eyes will notice flicker

Computer Animation

- High quality (Movies)

- Lower quality (Games)

Techniques on Computer Animations

- No matter the rendering quality, the ways to **create** and form the animation are similar
- Traditional ways to create cartoon animation are on papers and transparent slides
- But for CG, all are inside the computer
  - 3D model
  - Lighting/ rendering
  - Motion of each frame
  - Etc.

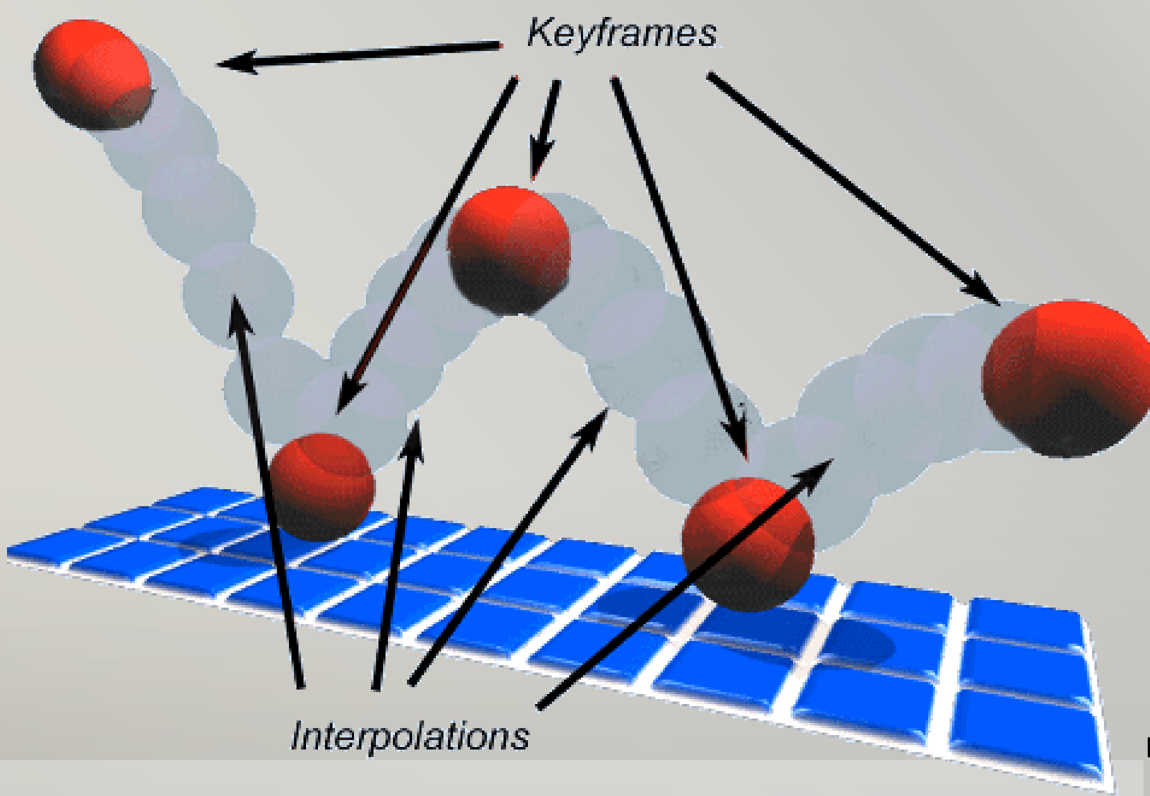
Key-Frame Animation

- A most basic way to create animation
- Animator to create "Key" frames

http://www.idleworm.com

Interpolation

- In CG, frames in-between keys are generated by computers
  - Interpolation in the time dimension



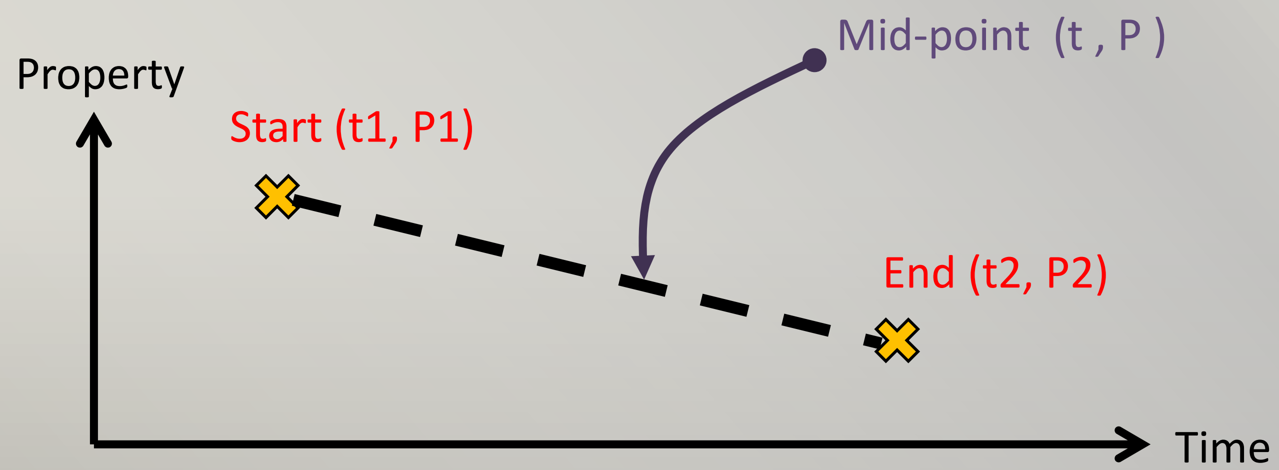
http://www.erimez.com

Linear Interpolation

- Simplest way to interpolate a certain property
  - Given 2 points,

    Start (t1, P1) and End (t2, P2)
  - Any mid-point at time t:

$$P = P1*(t2-t) /(t2-t1) + P2* (t-t1)/(t2-t1)$$



Example of Simple animation with Linear Interpolation

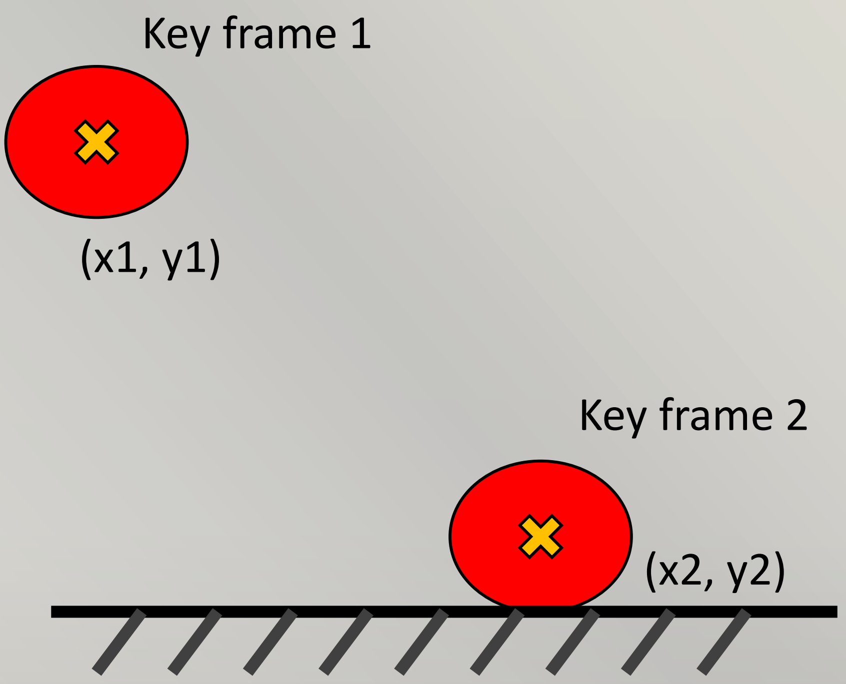
- E.g. the properties concerned are the x and y positions
- Consider a ball
  - At Key frame 1

    Position = (x1, y1) = (2,4)

    Time = 0
  - At Key frame 2

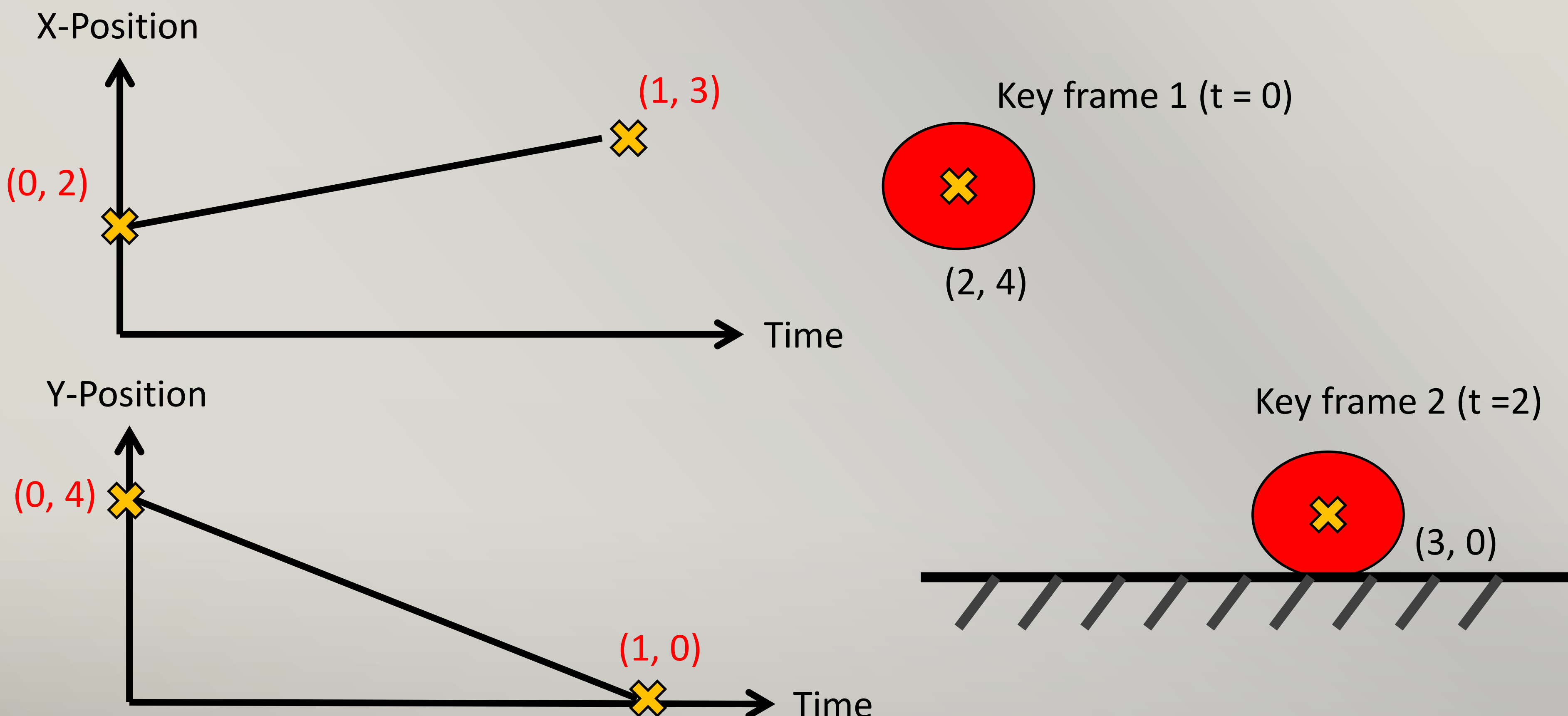
    Position = (x2, y2) = (3,0)

    Time = 1



Example of Simple animation with Linear Interpolation

- So the interpolation happens in 2 time lines



Example of Simple animation with Linear Interpolation

- To obtain position at time t = 0.6

X = x1*(t2-t) /(t2-t1) +
x2* (t-t1)/(t2-t1)

X = 2*(1-0.6) /(1.0) +
3* (0.6-0)/(1.0)

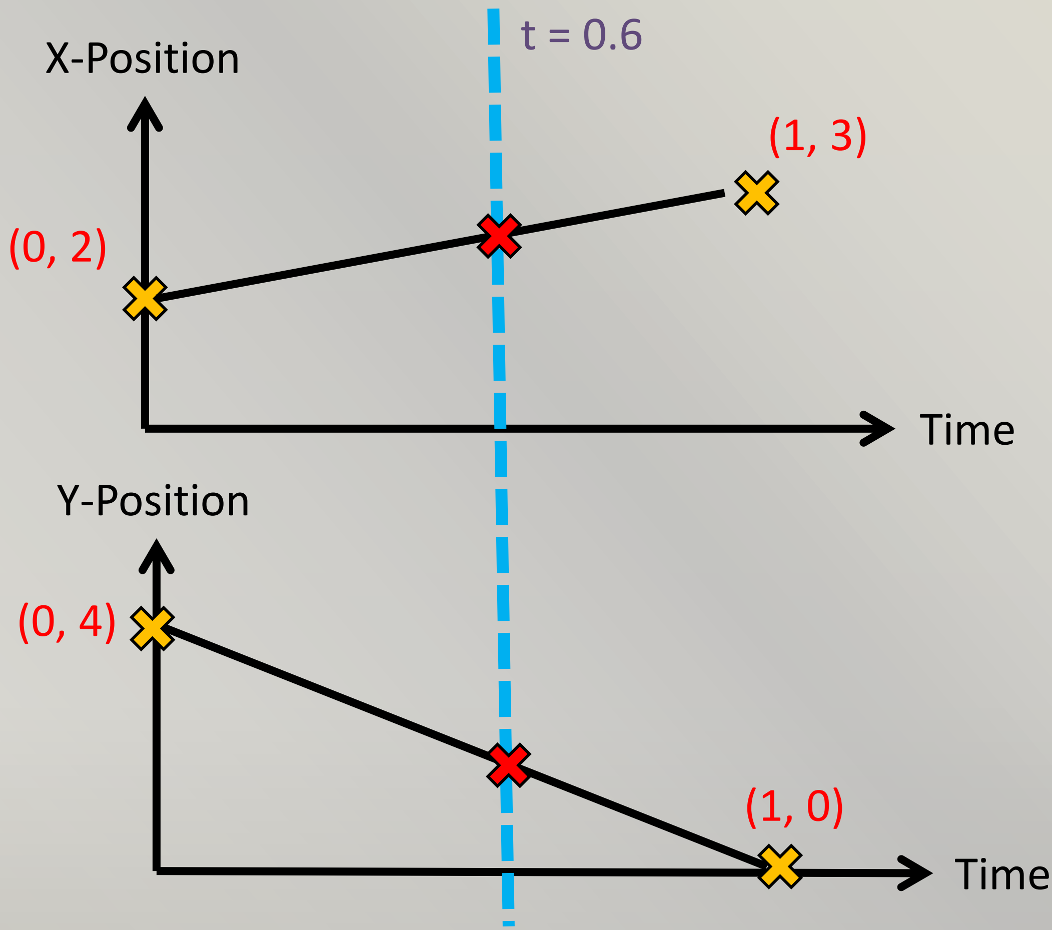
= 0.8+1.8 = 2.6

- Similar for y

Y = y1*(t2-t) /(t2-t1) +
y2* (t-t1)/(t2-t1)

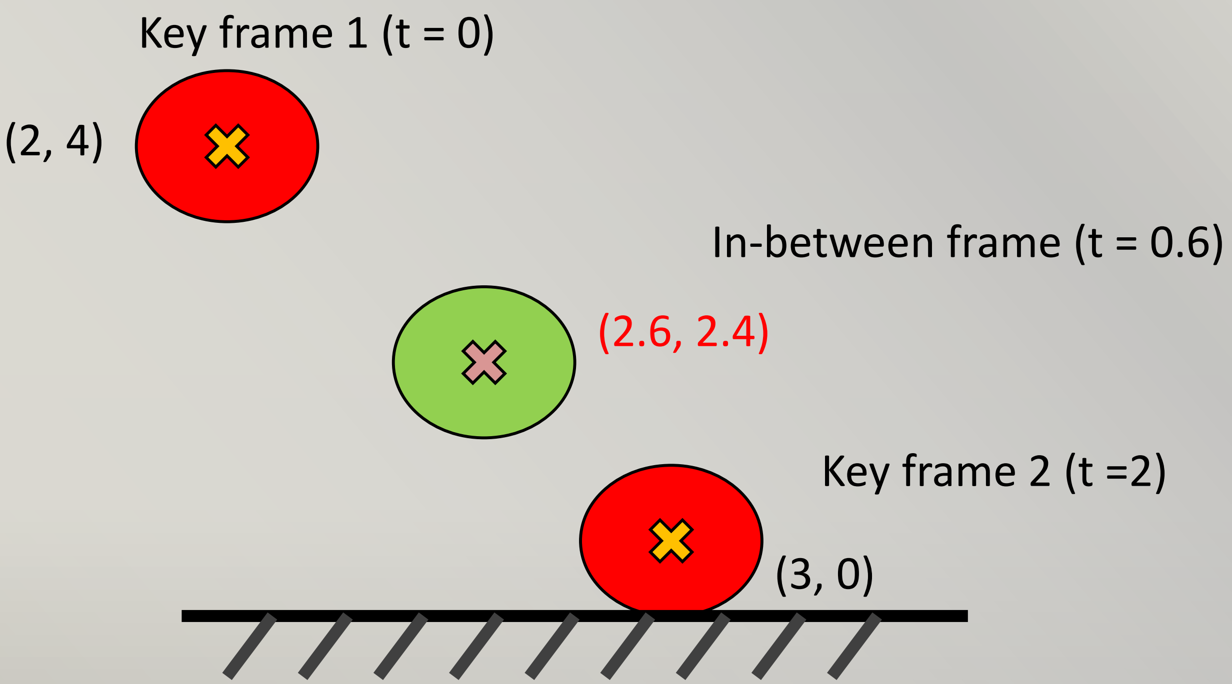
Y = 4*(1-0.6) /(1.0) +
0* (0.6-0)/(1.0)

= 2.4+0.0 = 2.4



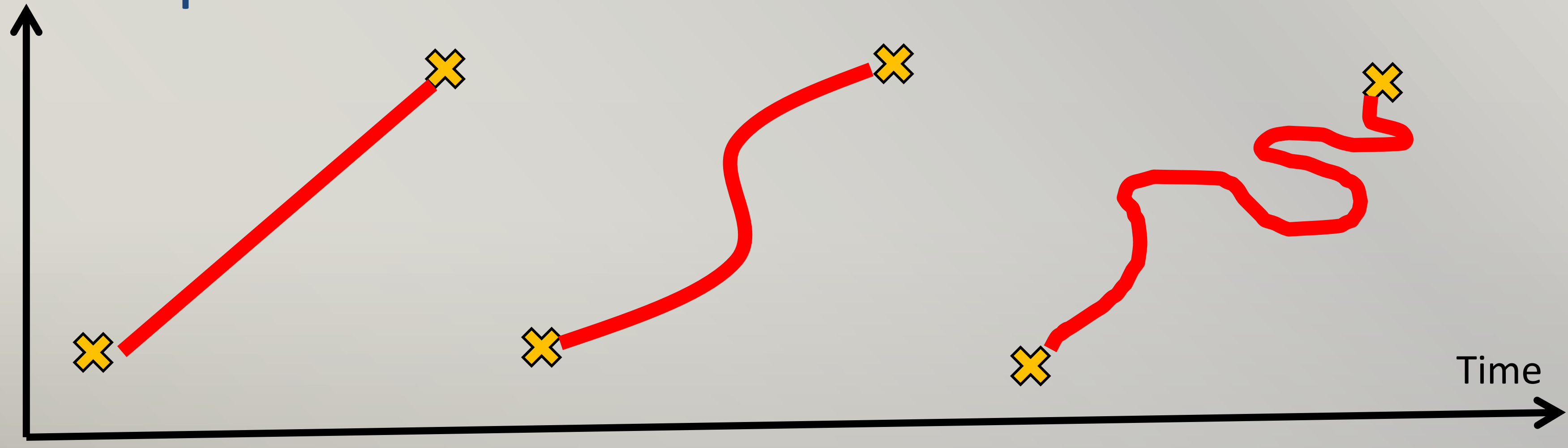
Example of Simple animation with Linear Interpolation

- So the in-between frame at t = 0.6, the ball will be at (2.6, 2.4)



Curved Interpolation

- The path from one point to another can vary
  - Straight path : Linear interpolation
  - Curved path : Spline interpolation
- The problem is how to define the curve ?



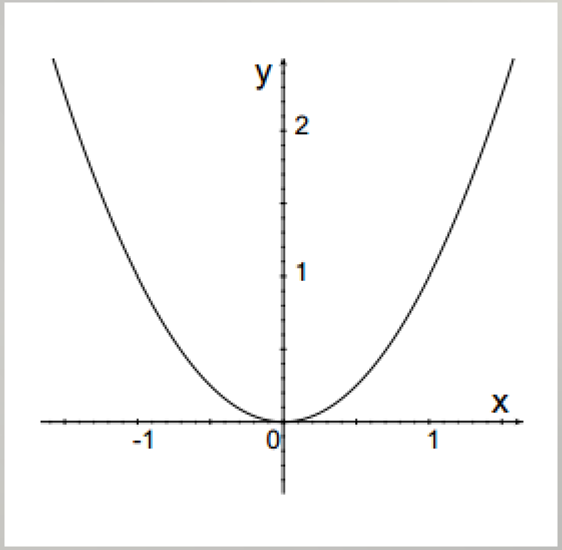
Spline Curves

Curves

- Curves can be defined by
  - Function, or
  - Control points in a parametric way
- In case of function, normally **polynomial** is used
  - Easiest example :

    A quadratic polynomial

    $y = x^2$



Curves by Polynomial

- The general form a polynomial:

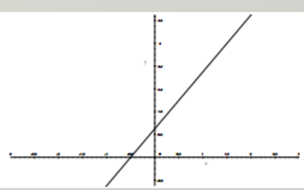
  $y = a + bx + cx^2 + dx^3 + \ldots$

  The degree of a polynomial corresponds with the highest coefficient that is non-zero

- The higher the degree, the more variation the curve can be

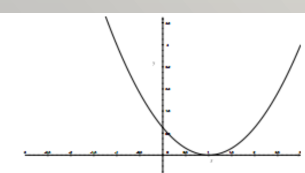
Linear

Example: $y = 1 + 2x$



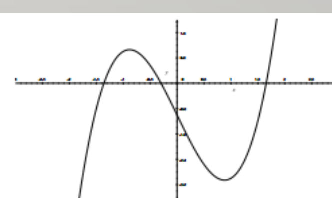
Quadratic

Example: $y = 1 - 2x + x^2$



Cubic

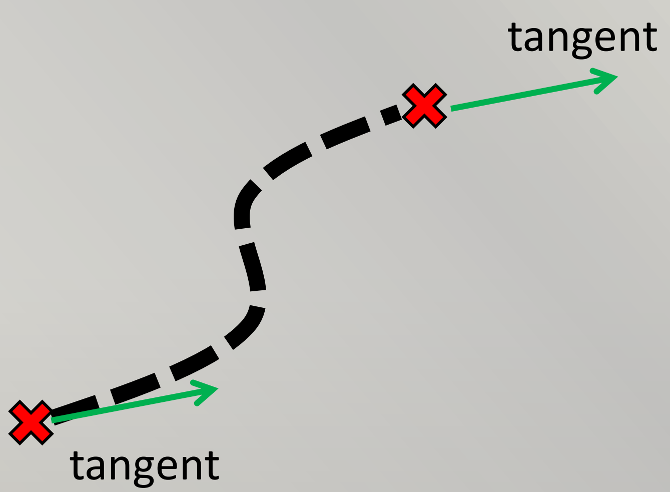
Example: $y = -1 - 7/2x + 3/2x^3$



Parametric Curves

- One of the problems of a polynomial curve is that the constants a,b,c,d …
  - They have no intuitive meaning
- In most of the case, when we define a curve, we use
  - Points passing through

    control points
  - Tangents at the points

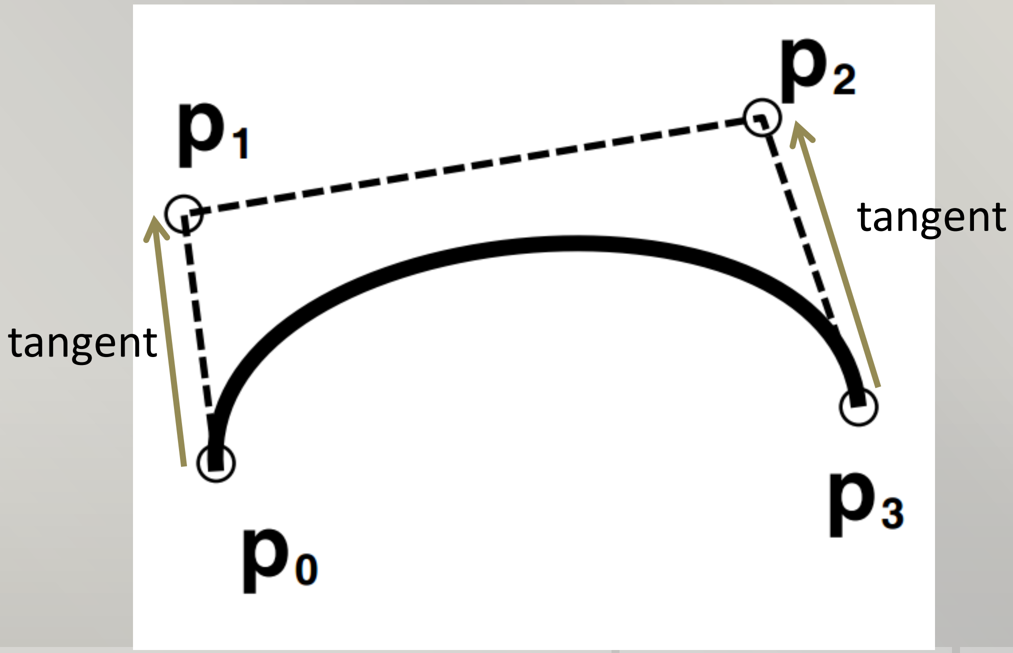


Parametric Curves

- Types of parametric curves
  - Hermite
  - Catmull-Rom
  - Bezier
  - B-Spline
  - Etc.
- They are similar, just have different properties

Bezier Curves

- One of the most popular parametric curve
  - E.g. true type fonts, animation software
  - Usually, the cubic Bezier curve is used
- p_0, p_1, p_2, and p_3 are 4 control points to define the curve
  - Note that p_0p_1 and p_2p_3 forms tangents at p_0 and p_3 respectively



Bezier Curves

- The curve is defined by

$$\mathbf{c}(u) = \Sigma_{i=0}^{3} \mathbf{p}_i B_i(u)$$

  where p_i is the 4 control points

- $B_i(u)$ are the basis functions in terms of the parameter u as follow:

$$B_0(u) = (1-u)^3$$
$$B_1(u) = 3u(1-u)^2$$
$$B_2(u) = 3u^2(1-u)$$
$$B_3(u) = u^3$$

Bezier Curves

- It is easy to notice that when u= 0,

  $B_0(0)$ =1, $B_1(0)$ =0 , $B_2(0)$ =0, $B_3(0)$ =0

  (You can also try when u=1 yourself)

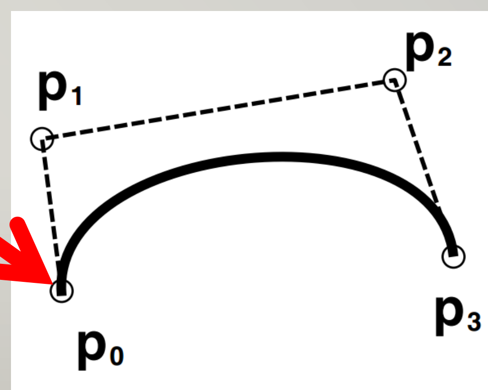
- So, the point return from c(0) will be

  p_0 *1 + p_1 *0 + p_2 *0 + p_3 *0

  $= p_0$

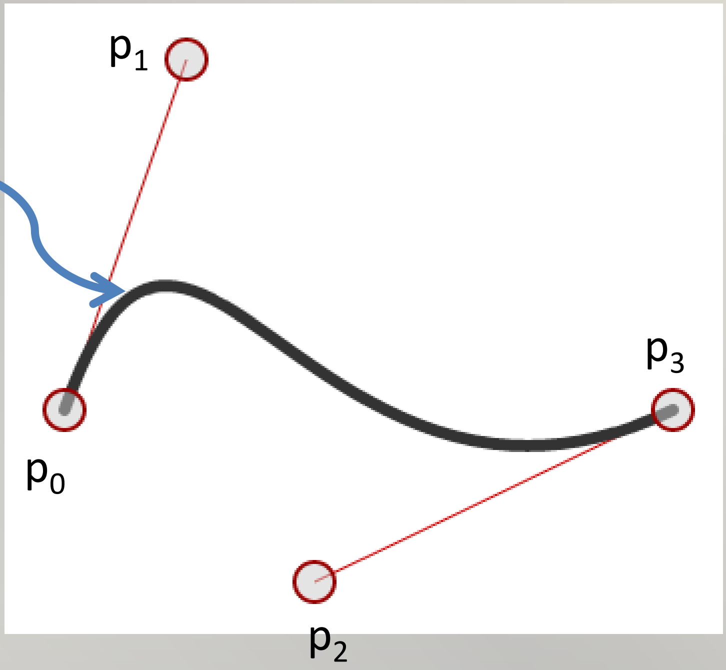
$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i^3(u)$$

That is the first control point !!!!



Bezier Curves : Example

- Given control points: p_0=(100,260), p_1 =(160,85), p_2 =(255,345), p_3 =(404,260),
- So to find a point with u = 0.2
- First compute all B_i
  - i.e. $B_0(0.2)$, $B_1(0.2)$, $B_2(0.2)$, $B_3(0.2)$



Bezier Curves : Example

- $B_0(0.2) = (1 - 0.2)^3 = 0.512$
- $B_1(0.2) = 3*0.2\ (1 - 0.2)^2 = 0.384$
- $B_2(0.2) = 3*0.2^2\ (1 - 0.2) = 0.096$
- $B_3(0.2) = (0.2)^3 = 0.008$

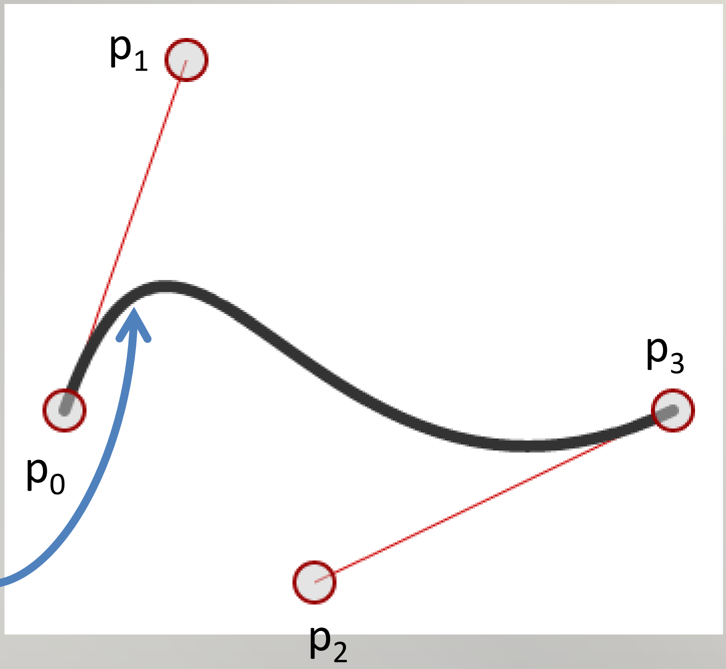
- Then, we evaluate C(0.2) by summing all the multiplication between B_i and p_i
  - Pretty like a weighted sum of all control points

$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i(u)$$

Bezier Curves : Example

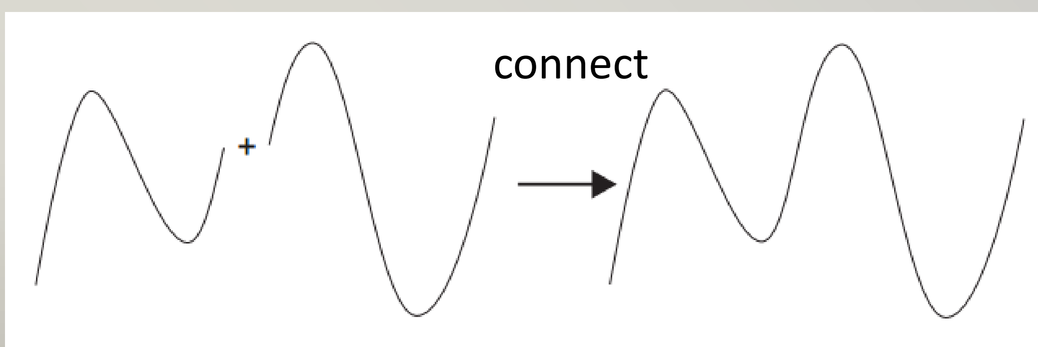
Because p_0=(100,260), p_1 =(160,85), p_2 =(255,345), p_3 =(404,260)

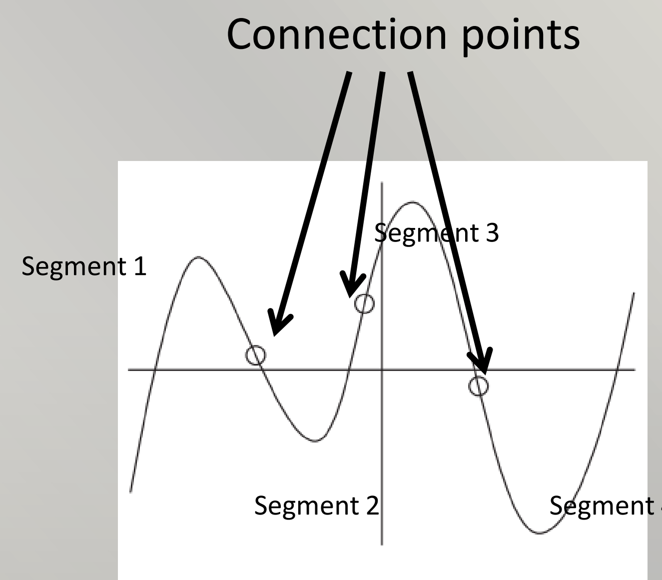
- c(0.2) = 0.512*(100,260) +
  0.384*(160,85) +
  0.096*(255,345) +
  0.008*(404,260)
  =(51.2, 133.12) +
  (61.44, 32.64) +
  (24.48, 33.12) +
  (3.232, 2.08)
  = (140.352, 200.96)



Spline Curves

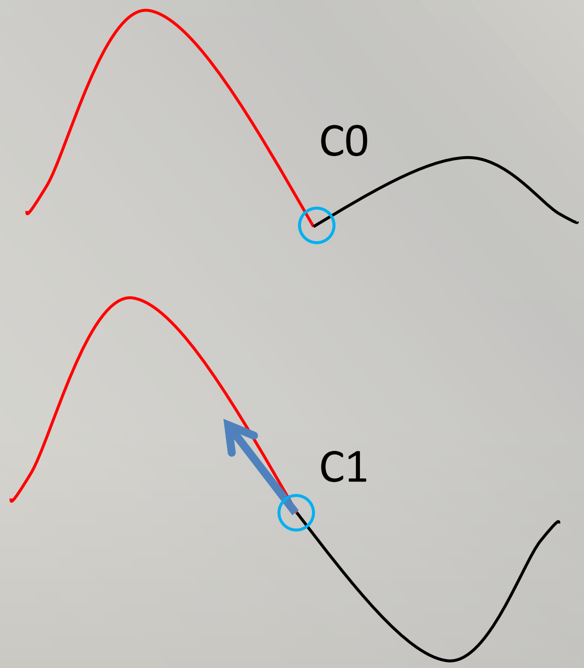
- The degree of freedom of a single cubic curve is limited
  - E.g. a cubic curve at most can have 2 turnings
- **<u>Piecewise curve</u>** is to construct a curve with higher degree of freedom
  - Cubic curves are connected together





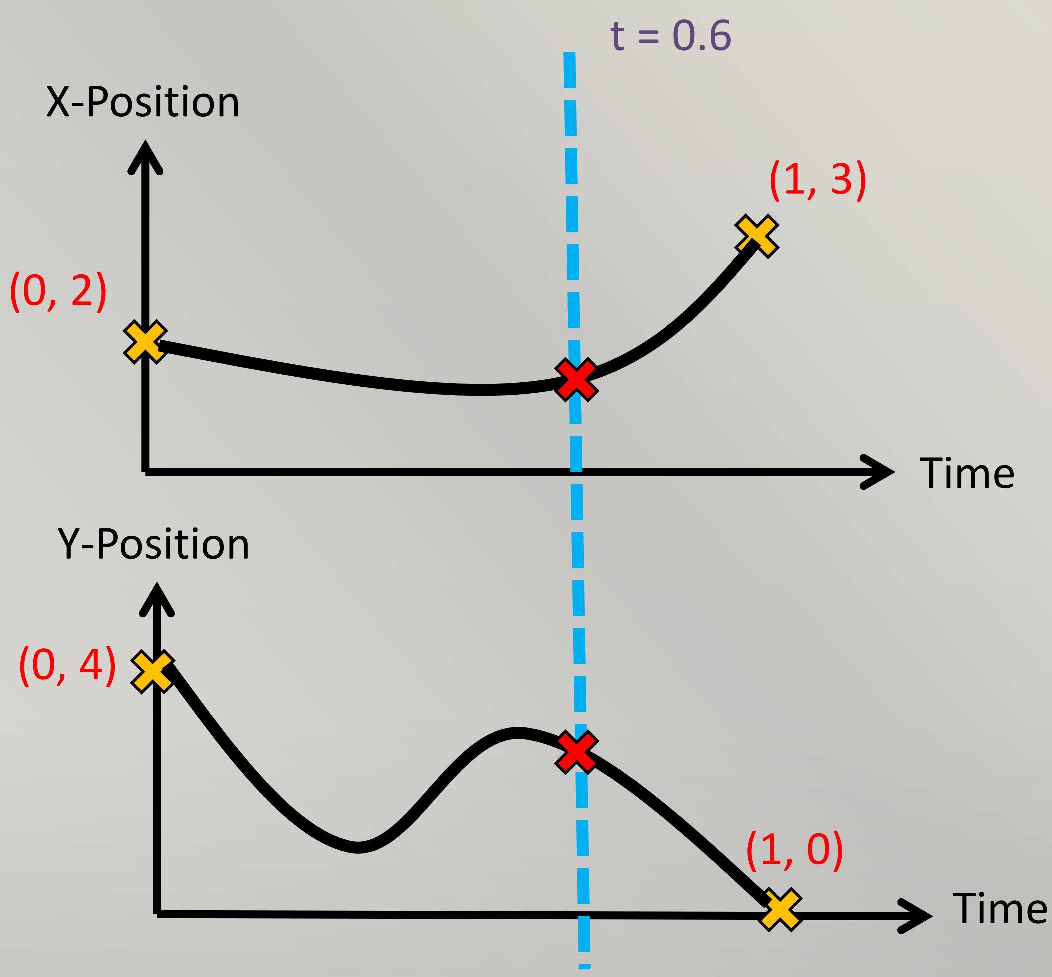
Spline Curves: Degree of continuity

- When 2 curves are connected to from spine, continuity becomes a problem
- We introduce here 2 types of continuity
- C^0 continuity
  - Not continuous
  - Simply connect
- C^1 continuity
  - Ensure matching with tangent



Curved Interpolation

- Return to our original problem of interpolation using curve
  - Put t = 0.6
  - Compute the interception point on curve
  - So as to obtain positions of in the animation



Interpolated Key frame Animation

- Apart from positions, other properties of the object can also be interpolated over time, like
  - Color
  - Alpha
  - Rotation/ Orientation
  - Scale and etc.
- Therefore, these properties can be interpolated between the key frames

Advanced Techniques in Animation

- Weakness of key-frame animation
  - Intensive human involvement
    - Need a key whenever a movement occurs
  - Realistic motion is difficult to model by hands
    - E.g. fluid motions, human movements, etc.

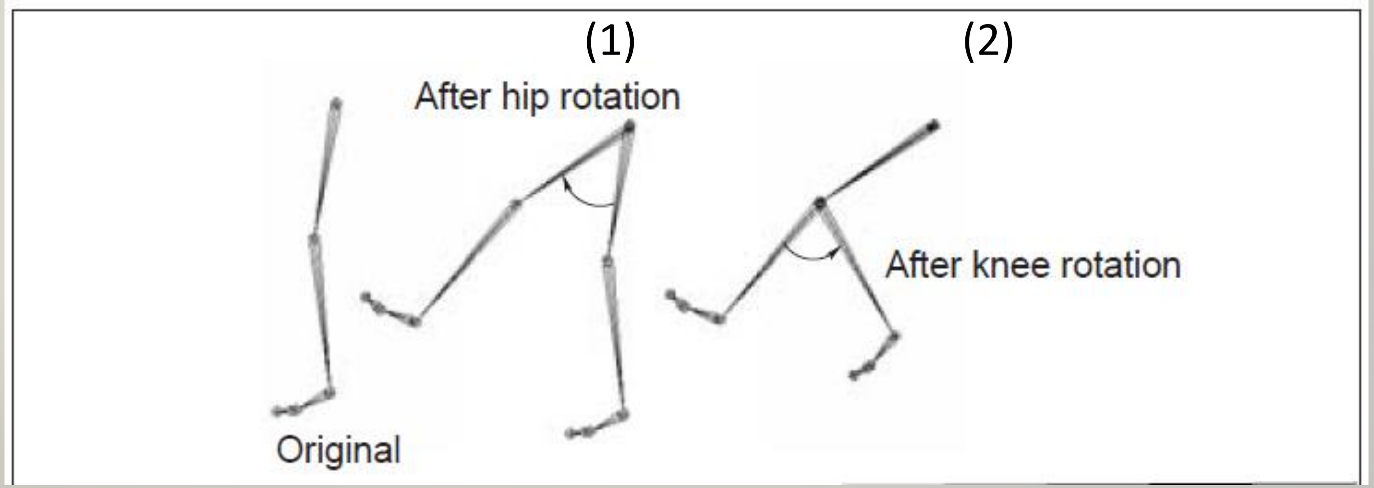
- A wide range of animation techniques are available besides key-frame animation

Inverse Kinematics (I.K)

- Character Animation
  - Forward Kinematics

    Simplest and straight forward

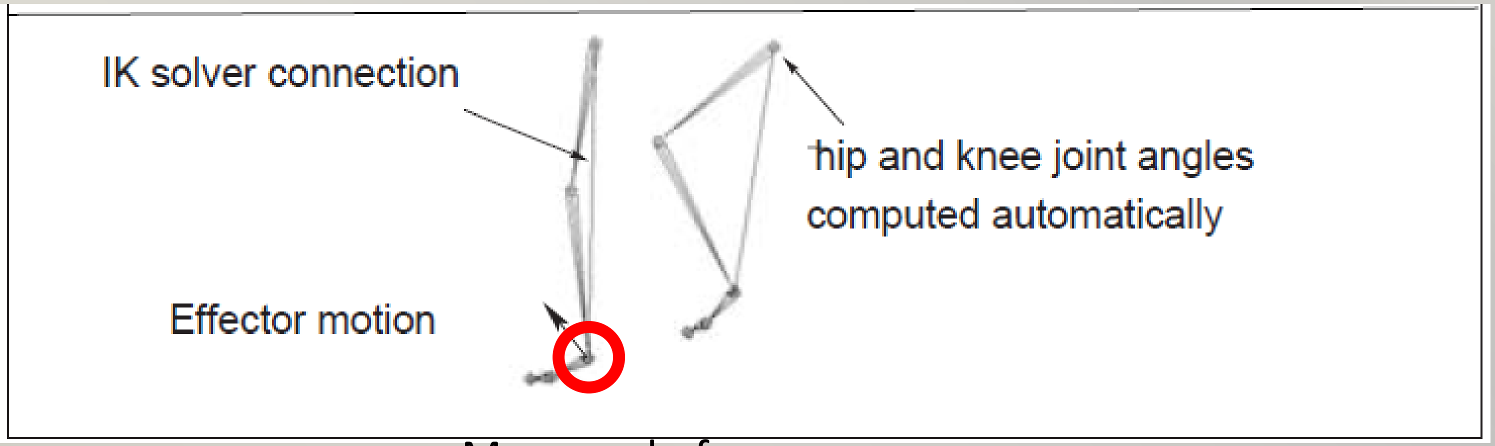
    Move each joint one after another



Inverse Kinematics (I.K)

Opposite to forward kinematics

- Move one end-effector (end point)
- The whole other joints will move accordingly



Move end of
the joint

Motion Capture

- Use motion capturing device to record motions of human
- Reapply/Retarget onto virtual character
  - ✖ More realistic and natural human like motion

Organic motion

Facial Animation

- A special kind of motion capture
- Facial expressions of human are captured and retarget on character

cubicmotion

Crowd Animation

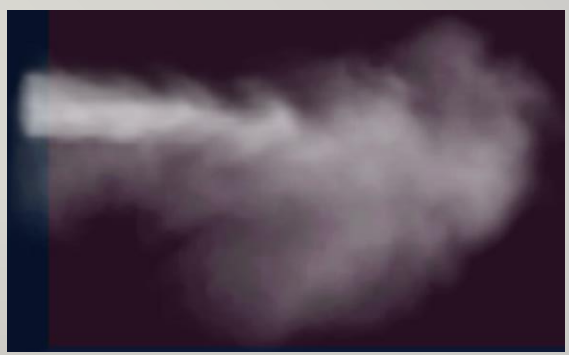
- Avoid editing a huge number of similar characters which are moving in groups
  - Roughly move similarly
  - But, each of them has minor difference

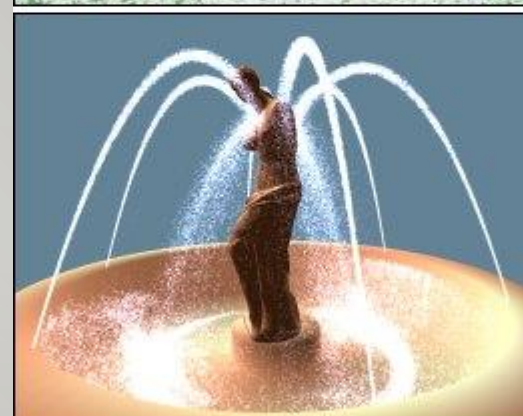
    Some randomness

Procedural/ Physical based Animation

- Usually to simulate natural phenomenon
- Particle system

- Fluid dynamics

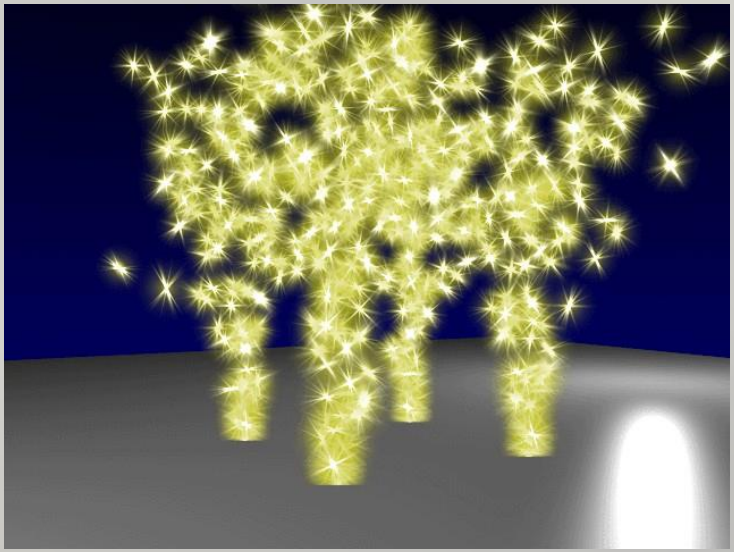




Particle System

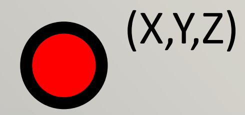
What is a particle system?

- A collection of point masses that obeys some physical laws
  - e.g. Gravity, air flow, water stream, spring behaviors
- Commonly used to simulate motions in physical phenomena
  - Like fire, water spring, etc



Physical Laws

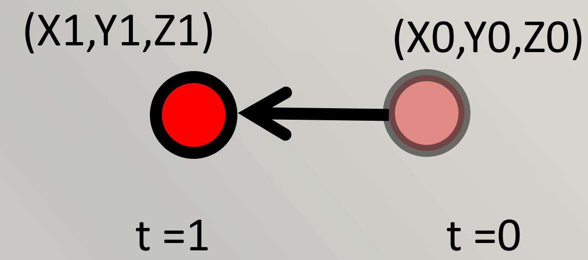
- Physical properties related to motion of a particle include
  - Position (X,Y,Z)



  - Velocity (Speed + Direction)

    Rate of **change of position**

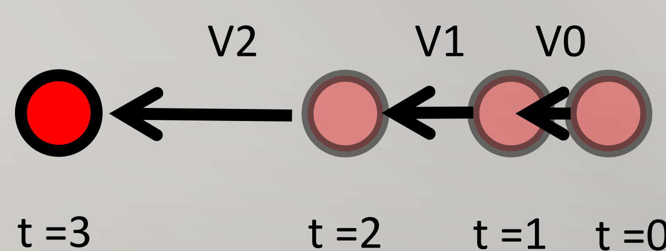
    $$\langle Vx, Vy, Vz \rangle = \left\langle \frac{X1-X0}{dt}, \frac{Y1-Y0}{dt}, \frac{Z1-Z0}{dt} \right\rangle$$



  - Acceleration

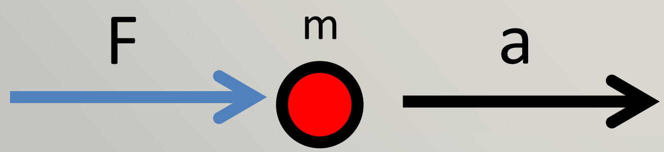
    Rate of **change of velocity**

    $$\left\langle \frac{V1-V0}{dt}, \frac{V1-V0}{dt}, \frac{V1-V0}{dt} \right\rangle$$

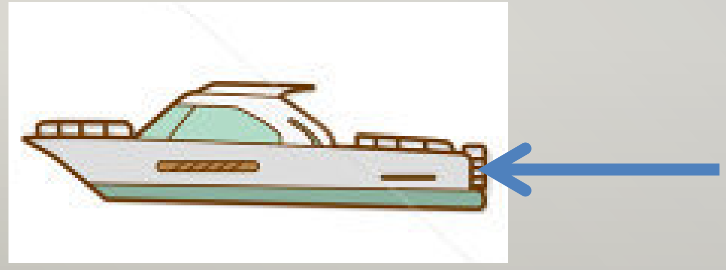


Particle in an acceleration field

- Remember, F = ma
  - F is force, m is mass
  - a is acceleration

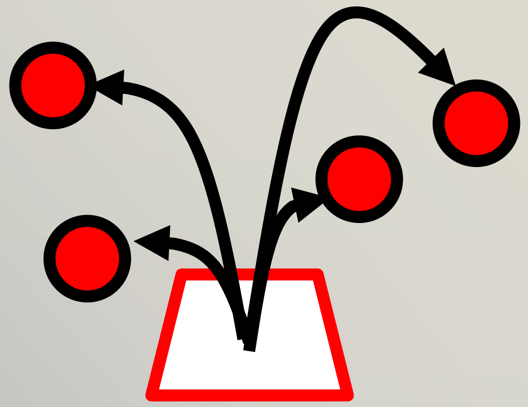


- Everything moves by "force"
  - A jet engine provides force to move a boat forward
  - You fall because there is gravitational force



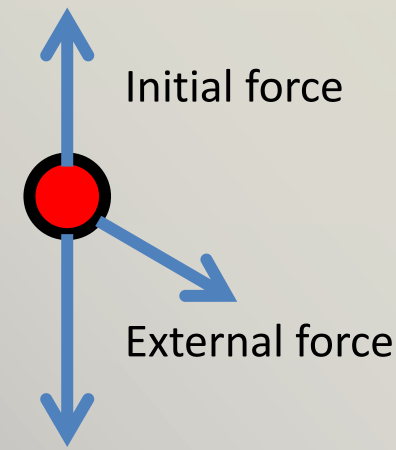
A Particle System

- Particle source
  - Where particles are generated
- Initial force and velocity
  - The initial force to move the particle when it is generated from the source
  - May initialize with different forces (include different directions)
- External force
  - Force being applied to the particle constantly
  - E.g. gravitational force



Solving Motions of a particle System

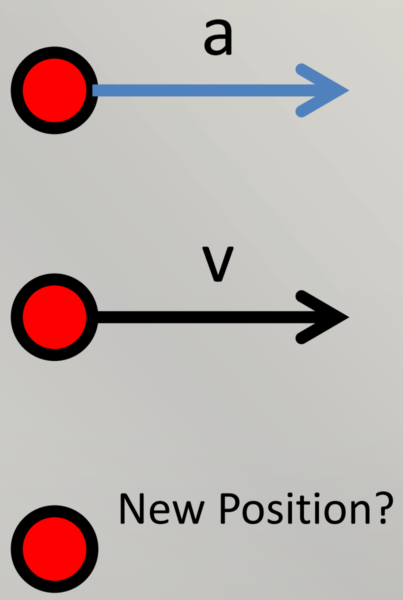
- A particle is affected by many different forces
- We need to combine all of them
- At every time frame, the velocity and position of a particle are updated
- In general terms, we are solving a differential equation



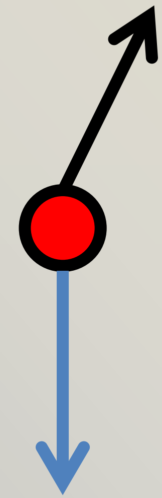
Solving Motions of a particle System

- The simplest approach is called “Euler method”
- Solve the integration using information from only the last time frame

1. a(t) = F(t)/m
2. v(t+1) = v(t) + a(t)*dt
3. x(t+1) = x(t) + v(t)*dt



A 2D Example



- A 2D particle has mass 0.1kg at
  - **x(0)** = (0.0, 0.0)
- Initial velocity of a particle
  - **v(0)** = <1.0, 2.0> m/s
- Subject to an external gravitational force
  - **f** = <0.0, -10.0> N
- Let's compute the position changes from t =0 to t =3 with **dt = 0.1s**
  - i.e. x(t=3)

A 2D Example

- **x(0)** = (0.0, 0.0)
- Start from t=0, we try to solve x(t=1)

a(0) = f/m

= <0,-10>/0.1

= <0,-100>

v(1) = v(0)+<0,-100>*0.1

= <1, 2> + <0,-10>

= <1, -8>

x(1) = x(0) + v(0)*0.1

=(0, 0)+ <1, 2>*0.1 = **(0.1,0.2)**

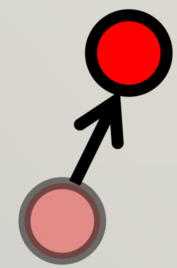
1 a(t) = F(t)/m

2 v(t+1) = v(t) + a(t)*dt

3 x(t+1) = x(t) + v(t)*dt

A 2D Example

- We finished the first time frame, x(1) = (0.1,0.2)
- Then, we go next time frame t = 2
- Repeat same steps



a(1) = f/m = <0,-100>

v(2) = v(1)+<0,-100>*0.1

= <1, -8> + <0,-10>

= <1, -18>

x(2) = x(1) + v(1)*0.1

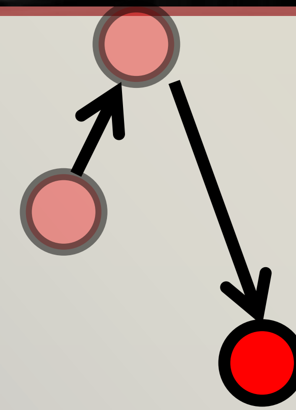
=(0.1, 0.2)+ <1, -8>*0.1 = **(0.2, -0.6)**

1 a(t) = F(t)/m

2 v(t+1) = v(t) + a(t)*dt

3 x(t+1) = x(t) + v(t)*dt

A 2D Example



- x(t=2) = (0.2,-0.6)
- Repeat again for time frame t=3

a(1) = f/m = <0,-100>

v(3) = v(2)+<0,-100>*0.1

= <1, -18> + <0,-10>

= <1, -28>

x(3) = x(2) + v(2)*0.1

=(0.2, -0.6)+ <1, -18>*0.1 = **(0.3, -2.4)**

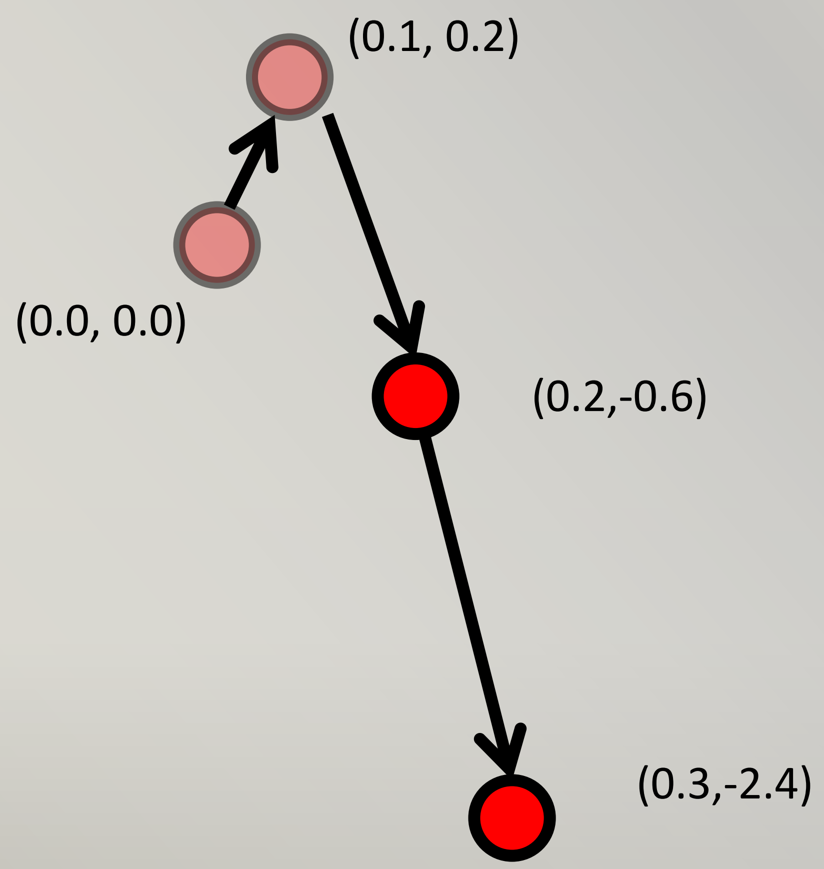
1 a(t) = F(t)/m

2 v(t+1) = v(t) + a(t)*dt

3 x(t+1) = x(t) + v(t)*dt

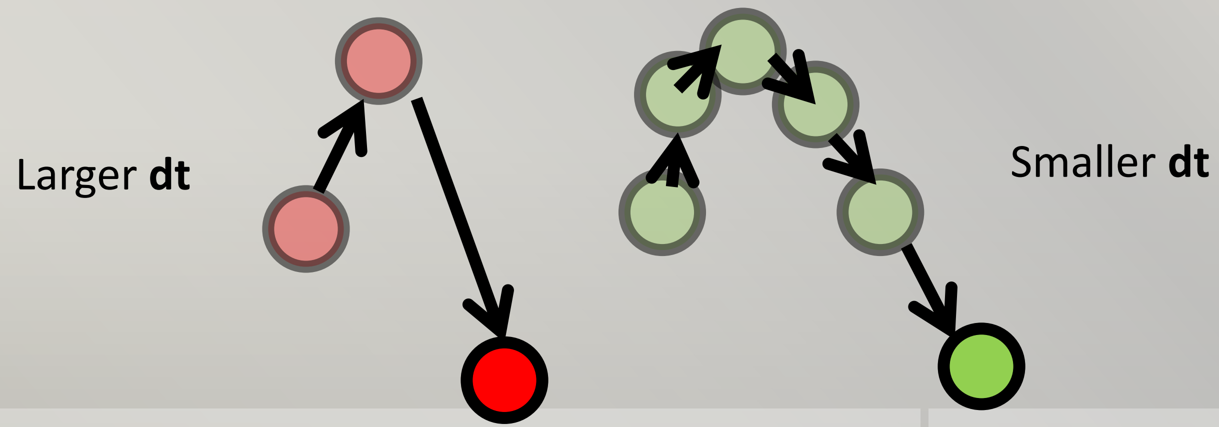
A 2D Example

- Finally, we have x(t=3) = (0.3,-2.4)



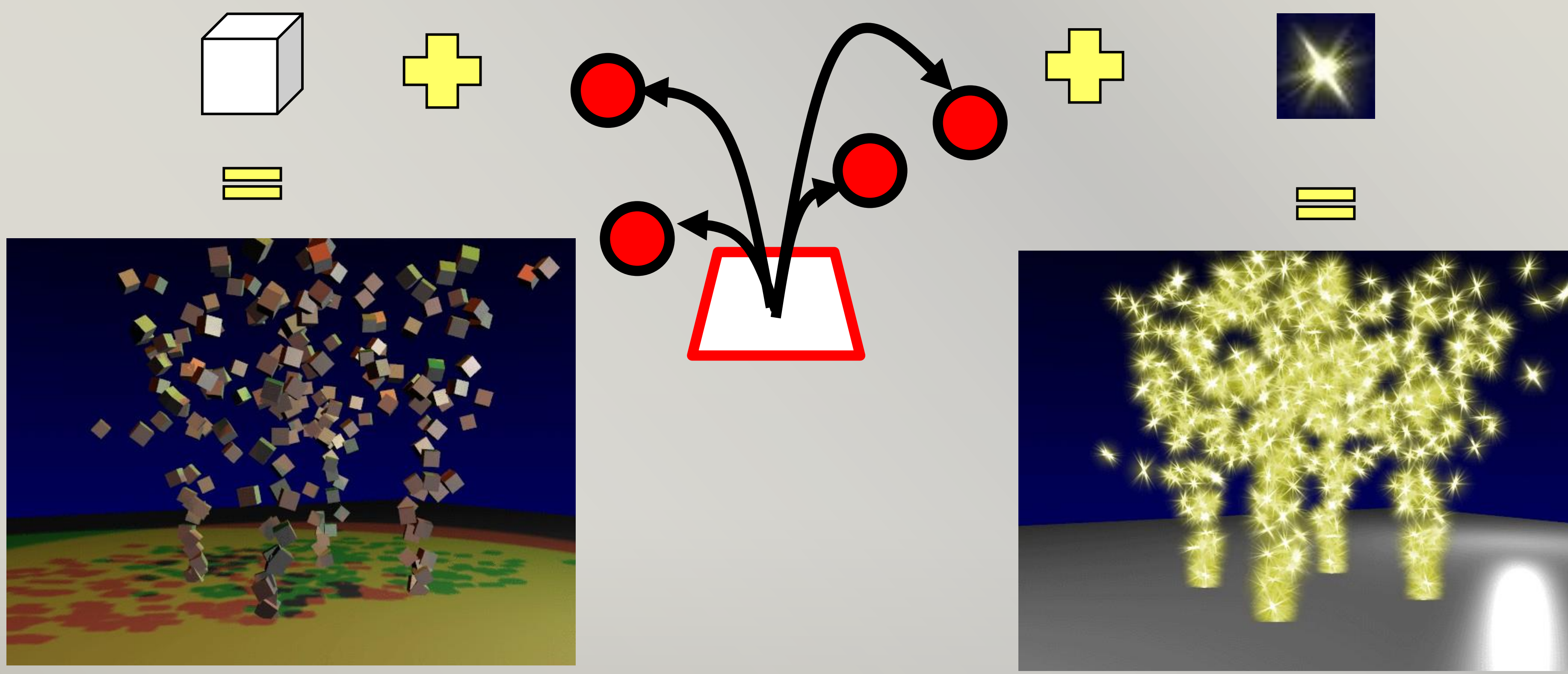
Particle System

- By repeatedly computing the equations to update velocity and positions of **all the particles** in each time frame, the particle system will work nicely
- When using Euler method, we find that smaller time steps (i.e. **dt**) will have a precise and smoother result



Particle System

- Usually particles will be attached with spites or shapes to create different effects

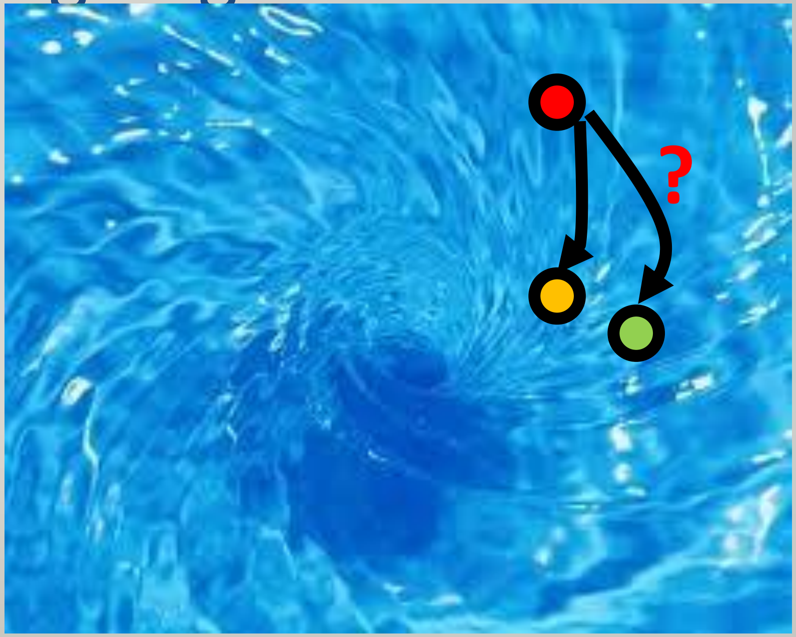


http://download.blender.org/documentation/html/x7193.html

More on particle based fluid animation

Fluid Dynamics

- Motion of fluid including
  - Liquid motion
  - Gas motion
- How fluid is going to move in next time frames



Fluid Dynamics

- Well studied in physics
- Governed by a rule : Navier Stoke Equation

$$\underbrace{\rho\left(\frac{\partial v}{\partial t}+v\cdot\nabla v\right)}_{\text{Velocity terms}}=\underbrace{\rho g}_{\text{Gravity}}-\underbrace{\nabla p}_{\text{Pressure: fluid moves from high pressure to low}}+\underbrace{\mu\nabla^2 v}_{\text{Viscosity}}$$

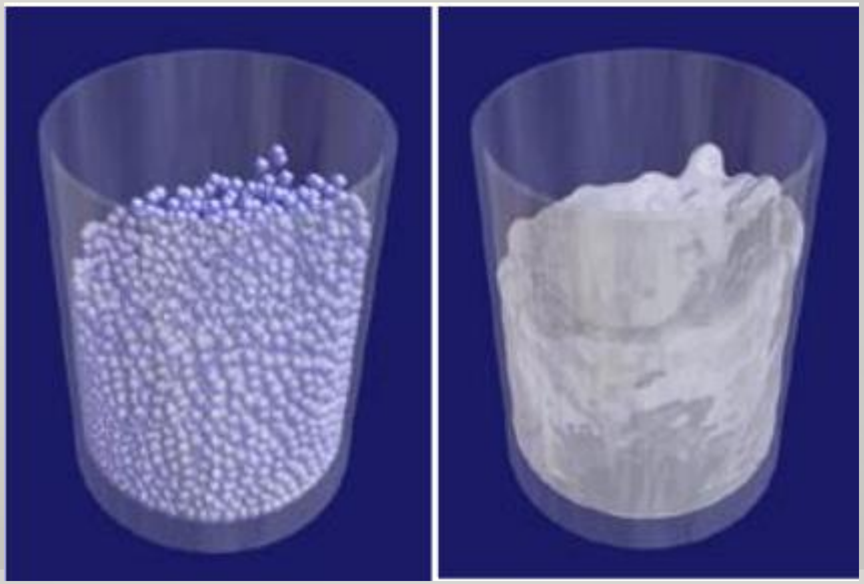
- An involving differental equation, rather out of scope in this basic graphics course

Smoothed Particle Hydrodynamics

- Smoothed Particle Hydrodynamics (SPH)
- A particle based approach
- Easier to understand and implement
- Less accurate when the number of particles are small
- Rendering is less trivial
- Much more time consuming than a simple particle system

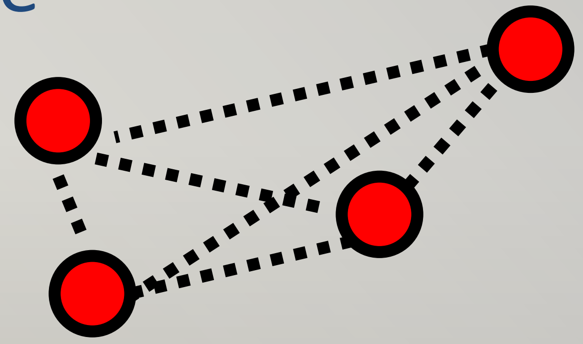
Fluid Dynamics with SPH

- Simulation of the dynamics
  - Move the particles
- Rendering of the fluid surface in case of liquid
  - Draw the surface of the liquid field



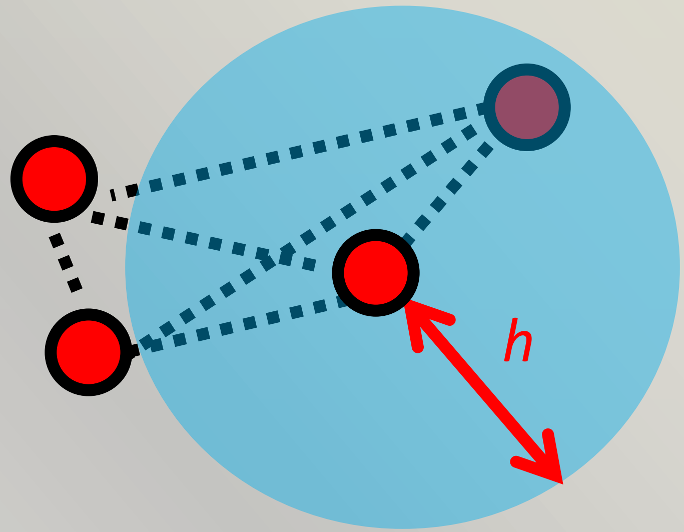
The Simulation

- Apart from individual particle's physics
- One major difference for SPH to simple particle system is
  - The modeling of inter-particle forces
- Each particle is attracted between each other with a certain distance



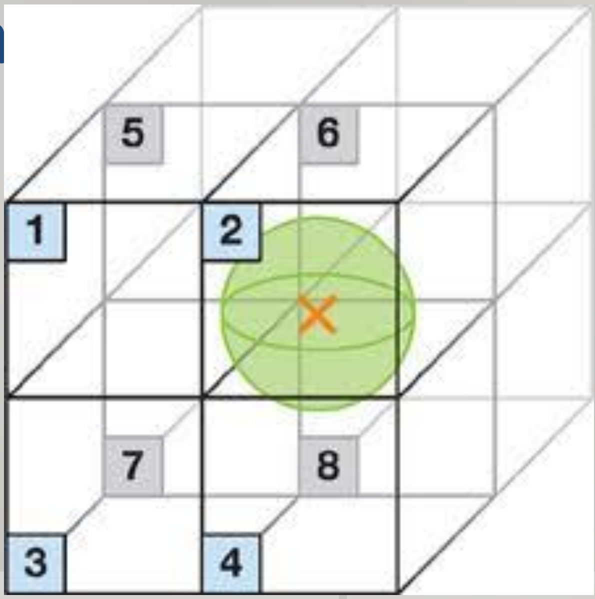
Inter-Particle Interaction

- If we consider every pairs of particles in the interaction
  - $O(n^2)$ potential computations for n particles!
- Usually, farther particles are ignored.
  - Defined with a kernel distance *h*



Spatial Search and Data Structure

- However, to find neighbors within kernel distance is not cheap
- Must quickly find all neighbors of each particle within a given radius
- Some spatial subdivision data lp
  - KD-tree
  - Octree
  - Hashed uniform grid



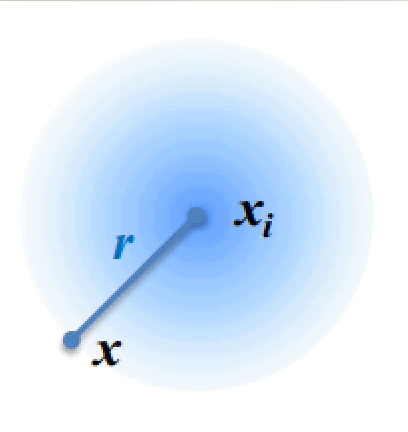
Forces from neighbor Particles

- Various forces are considered

$$f_i^{\text{pressure}} = -\nabla p(x_i) = -\sum_j \frac{m_j}{\rho_j} \nabla W(x_i - x_j)$$

$$f_i^{\text{external}} = \rho_i g$$

$$f_i^{\text{viscosity}} = \mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W(x_i - x_j)$$



- Example [Müller 03]: $W(r,h) = \frac{315}{64\pi h^9}(h^2 - r^2)^3, 0 \le r \le h$

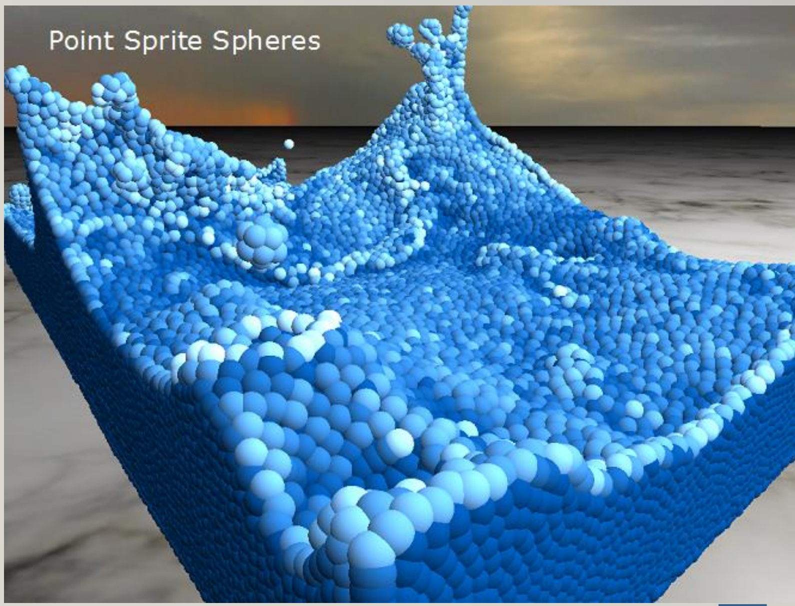
Summary of Major steps in SPH

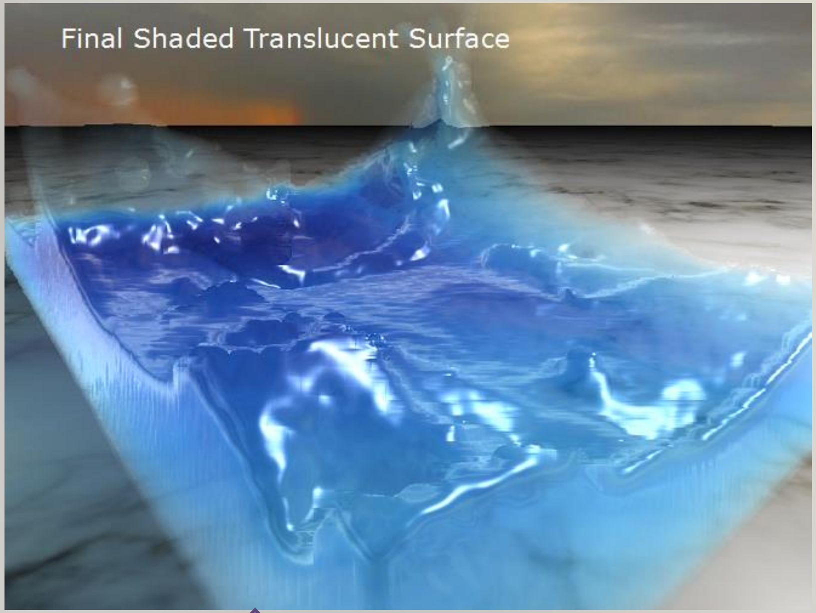
- Build spatial data structure on particles
  - To enable fast neighbor finding
- For each particle
  - find neighbors and compute density
- For each particle
  - find neighbors, compute force and update velocity
- For each particle
  - Find neighboring triangles, check collision, update position

Fluid Rendering

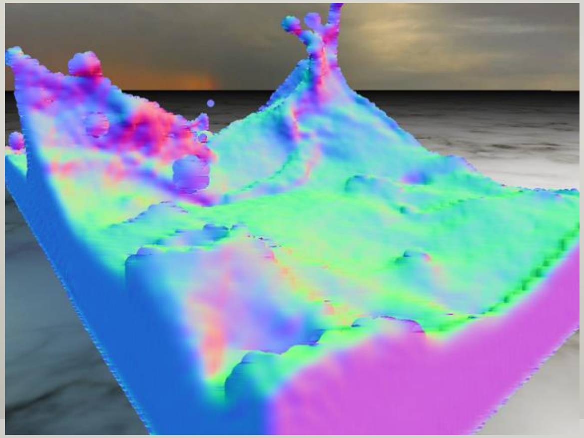
- Rendering of particles to fluid can be done in
  - 3D World Space, or
  - Screen Space
- 3D World Space Rendering or Isosurface extraction
  - Volume of liquid field is formed
  - Extract surface from the volume
- Screen Space Rendering
  - Render particles to depth buffer, thickness and normal buffer
  - Smooth depth buffer to avoid “particle appearance”
  - Render based on these buffers

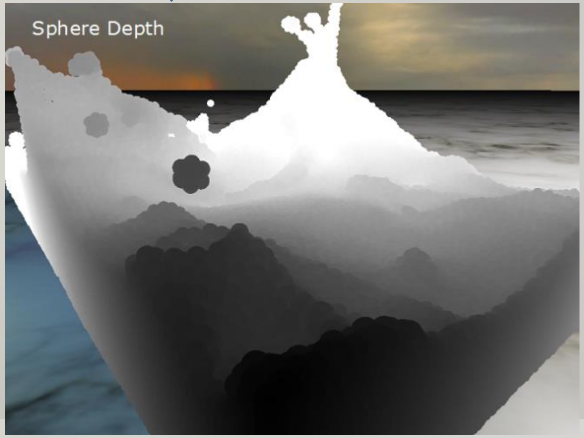
Screen space Rendering





Smooth out and rendering





Summary

- The basic animation technique: Key-frame animation is introduced
- In-between frames are done by interpolations of the character/object motions
- Beizer curve is a commonly used parametric approach to represent curves
- Other important techniques used in animation production are briefly introduced
  - I.K, motion capture, crowd animation, procedural animation

Summary

- Introduced a particular kind of method to create animation : Particle System
- Large group of particles are generated
- Each particle is moved based on physical laws
  - Acceleration
  - Velocity
  - Position
- Briefly introduce fluid dynamics with SPH which is a particle based approach