

## Lecture 27 of 41

# Interaction Handling Lab 5: Using Particle Systems

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course pages: <http://bit.ly/hGvXIH> / <http://bit.ly/eVizrE>

Public mirror web site: <http://www.kddresearch.org/Courses/CIS636>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

### Readings:

Today: §8.3 – 8.4, 4.2, 5.0, 5.6, 9.1, Eberly 2<sup>e</sup> – see <http://bit.ly/ieUq45>

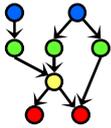
Next class: §9.1, **Particle System Handout**

Wikipedia, *Human-Computer Interaction*: <http://bit.ly/bqrQTg>

Wikipedia, *User Interface*: <http://bit.ly/eaTY7u>

Wikipedia, *Particle System*: <http://bit.ly/hzZofl>

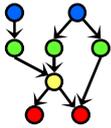




## Lecture Outline

- Reading for Last Class: Chapter 7, §8.4, Eberly 2<sup>e</sup>
- Reading for Today: §8.3 – 8.4, 4.2, 5.0, 5.6, 9.1, Eberly 2<sup>e</sup>
- Reading for Next Class: §9.1, **Particle System Handout**
- Last Time: Picking
  - \* OpenGL modes: rendering (default), feedback, selection
  - \* Name stack, hit records, rendering in selection mode
  - \* Selection buffer and other buffers with color coding
- Today: Interaction Handling
  - \* Human-Computer Interaction (HCI)
  - \* Perceptual Principles: Legibility, Consistency, Redundancy
  - \* Mental Models: Realism, User Expectations
  - \* Attention: Access Cost/Benefit, Multiple Sources, Sensory Modes
  - \* Memory: Self-Explanatory GUIs, Predictive Aids, Reusable Skills
- Next Class: Particle Systems





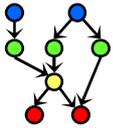
## Where We Are

21	<b>Lab 4a: Animation Basics</b>	<b>Flash animation handout</b>
22	Animation 2: Rotations; Dynamics, Kinematics	Chapter 17, esp. §17.1 – 17.2
23	Demos 4: Modeling & Simulation; Rotations	Chapter 10 <sup>1</sup> , 13 <sup>2</sup> , §17.3 – 17.5
24	<b>Collisions 1: axes, OBBs, Lab 4b</b>	<b>§2.4.3, 8.1, GL handout</b>
25	Spatial Sorting: Binary Space Partitioning	Chapter 6, esp. §6.1
26	<b>Demos 5: More CGA; Picking; HW/Exam</b>	<b>Chapter 7<sup>2</sup>; § 8.4</b>
27	<b>Lab 5a: Interaction Handling</b>	<b>§ 8.3 – 8.4; 4.2, 5.0, 5.6, 9.1</b>
28	Collisions 2: Dynamic, Particle Systems	§ 9.1, <b>particle system handout</b>
	<b>Exam 2 review; Hour Exam 2 (evening)</b>	<b>Chapters 5 – 6, 7<sup>2</sup> – 8, 12, 17</b>
29	<b>Lab 5b: Particle Systems</b>	<b>Particle system handout</b>
30	<b>Animation 3: Control &amp; IK</b>	<b>§ 5.3, CGA handout</b>
31	Ray Tracing 1: intersections, ray trees	Chapter 14
32	Lab 6a: Ray Tracing Basics with POV-Ray	<b>RT handout</b>
33	Ray Tracing 2: advanced topic survey	Chapter 15, <b>RT handout</b>
34	Visualization 1: Data (Quantities & Evidence)	<b>Tufe handout (1)</b>
35	<b>Lab 6b: More Ray Tracing</b>	<b>RT handout</b>
36	Visualization 2: Objects	<b>Tufe handout (2 &amp; 4)</b>
37	Color Basics; Term Project Prep	<b>Color handout</b>
38	Lab 7: Fractals & Terrain Generation	<b>Fractals/Terrain handout</b>
39	<b>Visualization 3: Processes; Final Review 1</b>	<b>Tufe handout (3)</b>
40	<b>Project presentations 1; Final Review 2</b>	–
41	<b>Project presentations 2</b>	–
	<b>Final Exam</b>	<b>Ch. 1 – 8, 10 – 15, 17, 20</b>

Lightly-shaded entries denote the due date of a written problem set; heavily-shaded entries, that of a machine problem (programming assignment); blue-shaded entries, that of a paper review, and the green-shaded entry, that of the term project.

Green, blue and red letters denote exam review, exam, and exam solution review dates.





## Acknowledgements: Picking, Interaction, Particles



### Edward Angel

Professor Emeritus of Computer Science  
Founding Director, ARTS Lab  
University of New Mexico

<http://www.cs.unm.edu/~angel/>



### François Guimbretière

Associate Professor  
Department of Computer Science  
Cornell University

<http://www.cs.cornell.edu/~francois/>



Cornell University  
Department of Computer Science

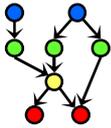


### Hubert Pak Ho Shum

Postdoctoral Researcher  
Advanced Science Institute  
RIKEN (理研)

<http://hubertshum.com/info/>



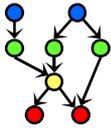


## Review [1]: Interactive Computer Graphics

- **More Sophisticated Interactive Programs**
  - \* Modes of interaction
  - \* Tools for building
- **Techniques**
  - \* Picking: select objects from display (three methods covered)
  - \* Rubberbanding: interactive drawing of lines, rectangles
  - \* Display lists: retained mode graphics

Adapted from slides ♥ 2005-2008 E. Angel, University of New Mexico  
*Interactive Computer Graphics*, 4<sup>th</sup> & 5<sup>th</sup> edition slides, <http://bit.ly/gvxfPV>



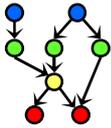


## Review [2]: Picking – Three Approaches

- **1. Hit List**
  - \* Most general approach
  - \* Difficult to implement
- **2. Buffered Object IDs**
  - \* Write to back buffer or some other buffer
  - \* Store object IDs as objects rendered
- **3. Rectangular Maps**
  - \* Easy to implement for many applications
  - \* e.g., simple paint programs

Adapted from slides ♥ 2005-2008 E. Angel, University of New Mexico  
*Interactive Computer Graphics*, 4<sup>th</sup> & 5<sup>th</sup> edition slides, <http://bit.ly/gvxfPV>



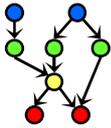


## Review [3]: OpenGL Rendering Modes

- **OpenGL: Can Render in One of Three Modes**
  - \* **GL\_RENDER**
    - Normal rendering to frame buffer
    - Default
  - \* **GL\_FEEDBACK**
    - Provides list of primitives rendered
    - No output to frame buffer
  - \* **GL\_SELECTION**
    - Each primitive in view volume generates *hit record*
    - Record placed in *name stack*
    - Stack can be examined later
- **Mode Selected by `glRenderMode(mode)`**

Adapted from slides ♥ 2005-2008 E. Angel, University of New Mexico  
*Interactive Computer Graphics*, 4<sup>th</sup> & 5<sup>th</sup> edition slides, <http://bit.ly/gvxfPV>





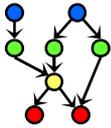
## Review [4]: OpenGL Name Stack Functions

- `void glInitNames(void);`
  - \* Creates empty name stack
  - \* Must call to initialize stack prior to pushing names
- `void glPushName(GLuint name);`
  - \* Adds name to top of stack
  - \* Maximum dimension: implementation-dependent
  - \* Must contain at least 64 names
  - \* Can query state variable `GL_NAME_STACK_DEPTH`
  - \* Pushing too many values causes `GL_STACK_OVERFLOW`
- `void glPopName();`
  - \* Removes name from top of stack
  - \* Popping value from empty stack causes `GL_STACK_UNDERFLOW`
- `void glLoadName(GLuint name);`
  - \* Replaces top of stack with name
  - \* Same as calling `glPopName(); glPushName(name);`

Adapted from tutorial ♥ 2001-2009 A. R. Fernandes  
Lighthouse 3D, <http://www.lighthouse3d.com>

Lighthouse3d.com 





## Review [5]: Example - OpenGL Selection Mode

- #define BODY 1
- #define HEAD 2
- ...
- void renderInSelectionMode ()
 

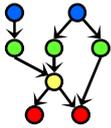
```

      {
          glInitNames ();           // 1. create empty name stack (NS)
          glPushName (BODY);       // 2. push first name
          // 3. hit record (HR) for each primitive intersecting view volume
          drawBody ();
          // 4. empty stack & save HRs to selection buffer (SB)
          glPopName ();
          Same as glLoadName (HEAD);
          glPushName (HEAD);      // 5. new name; no HR, same SB
          drawHead ();           // 6. new HR for each primitive in VV
          drawEyes ();          // 7. update HR with new max/min depths
          glPopName ();         // 8. empty NS; write HRs to SB
          drawGround ();        // 9. new HRs; empty NS, depth update only
      }
      
```

Adapted from tutorial ♥ 2001-2009 A. R. Fernandes  
Lighthouse 3D, <http://www.lighthouse3d.com>

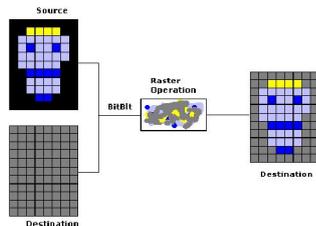
Lighthouse3d.com 





## Review [6]: BitBlitting, Sprites, & XOR Write

- Usual (Default) Mode
  - \* Source replaces destination:  $d' = s$
  - \* Cannot write temporary lines this way – why?
    - Cannot recover what was “under” line in fast, simple way
    - Consequence: cannot *deselect* (toggle select) easily
- Solution: Exclusive OR Mode (XOR)
  - \*  $d' = d \oplus s$
  - \* Suppose we use XOR mode to scan convert line  $P_0P_1$
  - \* Can draw it again to erase it!

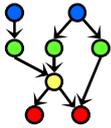


Visual Basic Explorer © 2002 S. Christensen & B. Abreu  
<http://bit.ly/gXstAM>

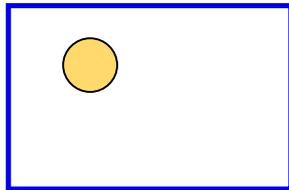
Adapted from slides ♥ 2005-2008 E. Angel, University of New Mexico  
 Interactive Computer Graphics, 4<sup>th</sup> & 5<sup>th</sup> edition slides, <http://bit.ly/qvxfPV>

THE UNIVERSITY of  
NEW MEXICO

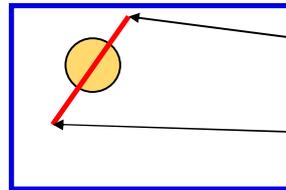




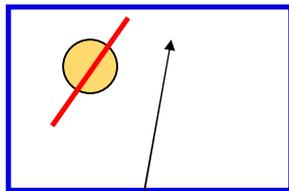
## Review [7]: Example – Rubberband Lines



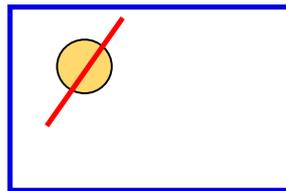
Initial display



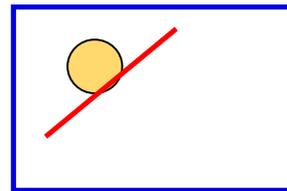
Draw line with mouse  
in XOR mode



Mouse moved to  
new position



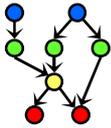
Original line redrawn  
with XOR



New line drawn  
with XOR

Adapted from slides ♥ 2005-2008 E. Angel, University of New Mexico  
*Interactive Computer Graphics*, 4<sup>th</sup> & 5<sup>th</sup> edition slides, <http://bit.ly/gvxfPV>





## Review [8]: OpenGL Display List Functions

- **Creating Display List**

```
* GLuint id;  
* void init()  
{  
    id = glGenLists( 1 );  
    glNewList( id, GL_COMPILE );  
    /* other OpenGL routines */  
    glEndList();  
}
```

- **Calling Created List**

```
* void display()  
{  
    glCallList(id);  
}
```

- \* **Documentation:** <http://bit.ly/gJYana>

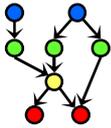
- \* **Tutorial © 2005 S. H. Ahn:** <http://bit.ly/eN3R8c>

Adapted from slides ♥ 2005-2008 E. Angel, University of New Mexico  
*Interactive Computer Graphics*, 4<sup>th</sup> & 5<sup>th</sup> edition slides, <http://bit.ly/qvxfPV>



THE UNIVERSITY of  
NEW MEXICO



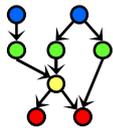


## Human-Computer Interaction (HCI)

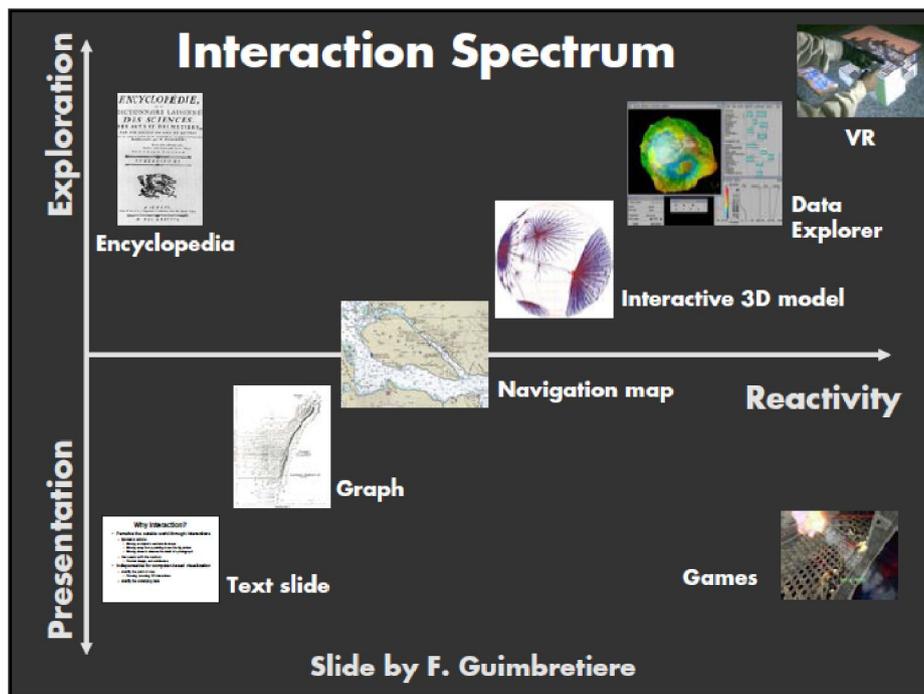
- **Study, Planning, & Design of Interaction between People, Computers**
  - \* **Study:** intersection of computer science, behavioral science, design
  - \* **Planning:** information management tasks, media
  - \* **Design:** graphical user interfaces, displays, algorithms, systems
- **Related Areas**
  - \* **Cognitive Science & Cognitive Modeling**
  - \* **Ergonomics & Human Factors**
    - HCI: more emphasis on computers (vs. other artifacts)
    - Some overlap within information technology
  - \* **Computational Information & Knowledge Management (CIKM)**
  - \* **Software Engineering: Operating Systems**
  - \* **Computer Graphics**
  - \* **Wearable Computers & Ubiquitous Communication/Computing**

Adapted from Wikipedia, *Human-Computer Interaction*  
<http://bit.ly/bqrQTg>





# Interaction Spectrum for HCI & Print Media

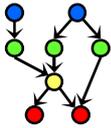


Adapted from slide ♥ 2004 F. Guimbretiere, Cornell University  
 Stanford CS448B: Visualization, Fall 2004, <http://bit.ly/h0hRzU>



Cornell University  
 Department of Computer Science



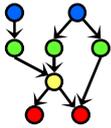


## HCI Design [1]: Perceptual Principles

- **1. Make Displays Perceivable (Legible, Audible, etc.)**
  - \* User must discern characters, objects, processes to interact
  - \* Usual sense of perceivability: legibility (readability)
- **2. Avoid Absolute Judgment Limits**
  - \* Don't ask user to sense level of variable based on one channel
  - \* Examples: color, size, (audio) volume
- **3. Use Top-Down Processing: Be Consistent with Past Experience**
- **4. Exploit Redundancy**
  - \* Present signal in alternative forms: color & shape, voice & print, etc.
  - \* Example: traffic light (color & position, sometimes sound)
- **5. Use Discriminable Elements to Minimize Confusion**
  - \* Allow visual elements to overlap only when necessary
  - \* Example: A423B9 more similar to A423B8 than 92 to 93

Adapted from material ♥ 2008 Wikimedia Foundation (from Wickens et al., 2004)  
*Human-Computer Interaction*, <http://bit.ly/bqrQTg>



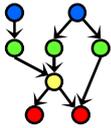


## HCI Design [2]: Mental Models Principles

- **6. Maintain Pictorial Realism**
  - \* UI elements should match real-world elements
  - \* Organization & layout should match those of real-world
- **7. Follow User's Expectations Regarding Moving Parts**
  - \* Motion paths should follow behavior of processes they model
  - \* Example: mark on altimeter should move up as altitude increases

Adapted from material ♥ 2008 Wikimedia Foundation (from Wickens et al., 2004)  
*Human-Computer Interaction*, <http://bit.ly/bqrQTg>



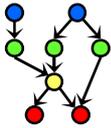


## HCI Design [3]: Principles Based on Attention

- **8. Minimize Information Access Cost**
  - \* Reduce cognitive load, especially time/effort overhead
  - \* Make convenience/accessibility proportional to frequency of access
- **9. Place Multiple Information Sources Close Together & Integrate Them**
  - \* Ensure design compatibility (layout, style, symbols, controls, etc.)
  - \* Tradeoff: usability aspects – understandability vs. efficiency
  - \* Use convenient shortcuts in proportion to frequency of need
  - \* User education needs
- **10. Multiple Information Channels**
  - \* Multisensory channels and integration (e.g., audio-visual)
  - \* Take advantage of brain's ability to merge senses

Adapted from material ♥ 2008 Wikimedia Foundation (from Wickens et al., 2004)  
*Human-Computer Interaction*, <http://bit.ly/bqrQTg>



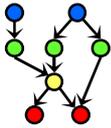


## HCI Design [4]: Memory Principles

- **11. Replace Memory with Visual Information: Knowledge in The World**
  - \* Don't rely on user to remember menus, shortcuts & macros
  - \* Present GUI elements in self-explanatory way to user
  - \* Balance usability for new users and experienced users
- **12. Develop Methods for Predictive Aiding**
  - \* Proactive (as opposed to reactive) user feedback
  - \* Example: road signs depicting km to destination, progress bars
- **13. Ensure Consistency**
  - \* Fulfill user's expectations of similarity towards lookalike GUIs
  - \* Maximize skill transfer

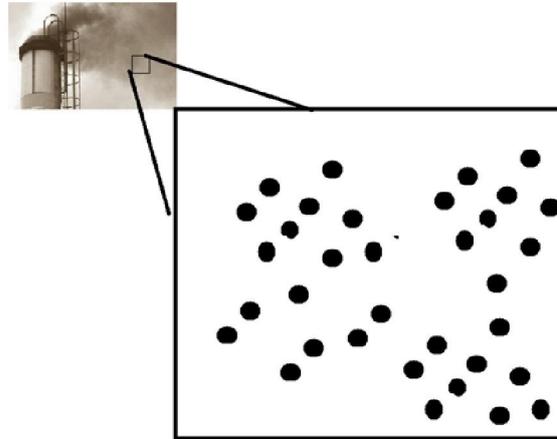
Adapted from material ♥ 2008 Wikimedia Foundation (from Wickens et al., 2004)  
*Human-Computer Interaction*, <http://bit.ly/bqrQTg>





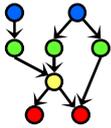
## Particle Systems [1]: Natural Effects

- The use of particle systems is a way of modelling fuzzy objects, such as
  - Fire
  - Clouds
  - Smoke
  - Water*etc.*
- These objects don't have smooth well-defined surfaces and are non-rigid
- Use small particles to represent them



Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/ig6KTK>



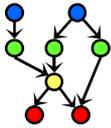


## Particle Systems [2]: Advantages

- Complex systems can be created with little human effort.
- The level of detail can be easily adjusted.
  - For example, if a particle system object is in the distance, then it can be modelled to low detail (few particles), but if it is close to the camera, then it can be modelled in high detail (many particles)

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





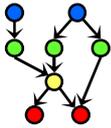
## Particle Systems [3]: Basic Model

A particle system is a collection of many minute particles that model some object. For each frame of an animation sequence the following steps are performed:

1. New particles are generated
2. Each new particle is assigned its own set of attributes
3. Any particles that have existed for a predetermined time are destroyed
4. The remaining particles are transformed and moved according to their dynamic attributes
5. An image of the remaining particles is rendered

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





## Particle Generation

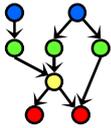
- Particles are generated using stochastic methods.
- the designer controls the mean number of particles generated per frame and the variance.
- So the number of particles generated at frame F is:

$$N_{partsF} = N_{mean} + rand() \cdot Variance$$

$-1.0 \leq rand() \leq 1.0$  a uniformly distributed random number

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





## Particles per Area

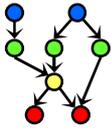
- A second method generates a certain number of particles per screen area. So MeanParts and VarianceParts refer to a number per unit screen area:

$$N_{partsF} = N_{meanSAF} + rand() \cdot VarianceSAF \cdot ScreenArea$$

- This method is good for controlling the level of detail required. Note: SAF means per screen area for frame F.

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>



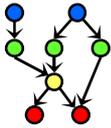


## Particle Emission Rate as a Function of Time

- The designer may want to change the number of particles generated as time changes and can do this by a simple linear function:
  - $\text{MeanPartsF} = \text{InitialMeanParts} + \text{DeltaMeanParts} \times (F - F')$
- The designer could do this by some function other than linear if needed or desired. So the designer must specify the initial parameters for the above equations and then everything is automatic.

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





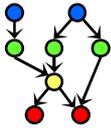
## Particle Attributes

Each new particle has the following attributes:

- initial position
- initial velocity (speed and direction)
- initial size
- initial color
- initial transparency
- shape
- lifetime

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





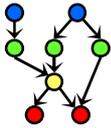
## Review: Dynamics & Kinematics

- **Dynamics: Study of Motion & Changes in Motion**
  - \* Forward: model forces over time to find state, e.g.,
    - Given: initial position  $p_0$ , velocity  $v_0$ , gravitational constants
    - Calculate: position  $p_t$  at time  $t$
  - \* Inverse: given state and constraints, calculate forces, e.g.,
    - Given: *desired* position  $p_t$  at time  $t$ , gravitational constants
    - Calculate: position  $p_0$ , velocity  $v_0$  needed
  - \* Wikipedia: <http://bit.ly/hH43dX> (see also: “Analytical dynamics”)
  - \* For non-particle objects: rigid-body dynamics (<http://bit.ly/dLvejg>)
- **Kinematics: Study of Motion without Regard to Causative Forces**
  - \* Modeling systems – e.g., articulated figure
  - \* Forward: from angles to position (<http://bit.ly/eh2d1c>)
  - \* Inverse: finding angles given desired position (<http://bit.ly/hsyTb0>)
  - \* Wikipedia: <http://bit.ly/hr8r2u>



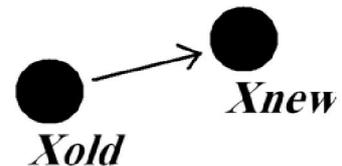
Forward Kinematics  
© 2009 Wikipedia





## Particle Dynamics

- A particle's position in each succeeding frame can be computed by knowing its velocity (speed and direction of movement).
- This can be modified by an acceleration force for more complex movement, e.g., gravity simulation.



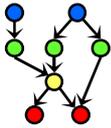
$$x_{new} = x_{old} + v_{old} \Delta t$$

$$v_{new} = v_{old} + a \Delta t$$

$$ma = F$$

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>



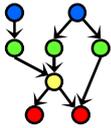


## Particle Extinction

- Particle is destroyed when it
  - Has no more lifetime
    - When a particle is created it can be given a lifetime in frames. After each frame, this is decremented and when the lifetime is zero, the particle is destroyed.
  - Fades out
    - When the color/opacity are below a certain threshold the particle is invisible and is destroyed.
  - Moves away
    - When a particle has left the region of interest, e.g., is a certain distance from its origin, it could be destroyed.

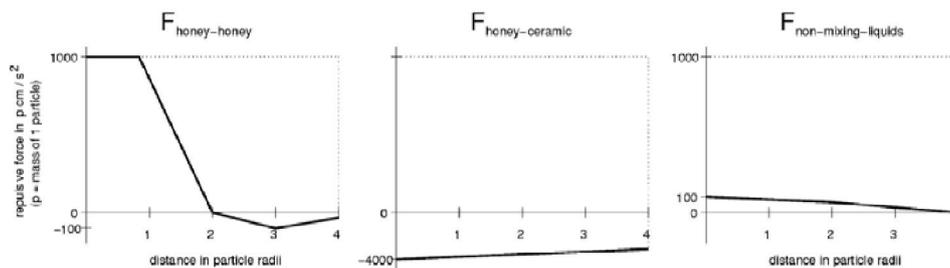
Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





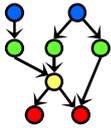
## Modeling Water

- To model liquid, viscosity and friction has to be taken into account. The force applied is the sum of all forces from neighboring particles plus the gravitational force.
- The external forces can be divided into components:
  - adhesion forces that attract particles to each other and to other objects,
  - viscosity forces that limit shear movement of particles in the liquid, and
  - friction forces that dampen movement of particles in contact with objects in the environment.



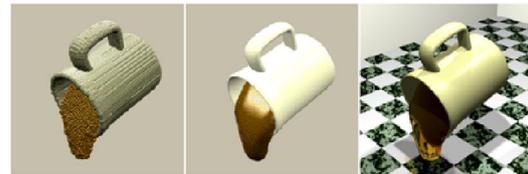
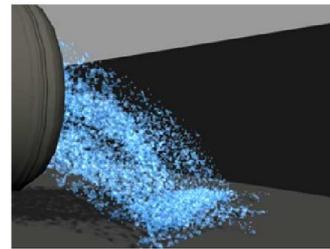
Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





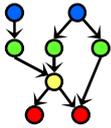
## Particle Systems [2]:

- There are two categories of methods
  - rendering individual particles
    - Works well for modelling waterfalls or spray
    - Faster
    - Give only a general idea about liquid or other continuous materials
  - rendering the surface of the object the particles
    - More accurate description
    - Much slower, especially with a system that is changing over time



Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>



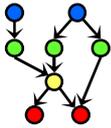


## Particle Systems for Multi-Body Systems

- Simulating multi-body systems such as
  - Human bodies
  - Robotic arms
  - Cars
  - Any systems that are composed of multiple joints
- The computational time is much shorter than other methods

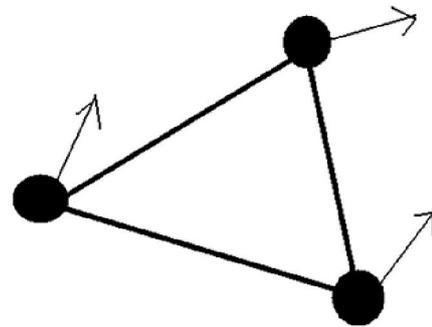
Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





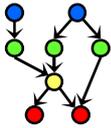
## Rigid Body Dynamics: How to Do It?

- Model rigid bodies by a collection of particles
- Assume the distance between the particles will remain the same if they belong to the rigid body
- Each particle will move based on the free particle dynamics
- The 3D position will be modified according to the distance constraints

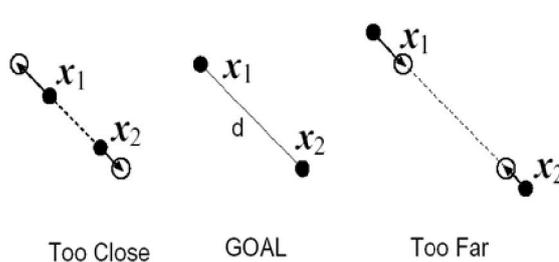


Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





## Distance between Particles: Keep Constant



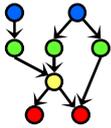
$$\Delta x_1 = \frac{m_1^{\text{inv}}}{m_1^{\text{inv}} + m_2^{\text{inv}}} \Delta d$$

$$\Delta x_2 = -\frac{m_2^{\text{inv}}}{m_1^{\text{inv}} + m_2^{\text{inv}}} \Delta d$$

- First integrate the position of the particles using the velocity and acceleration information
- To keep constant lengths of the links between points, the two particles are simply translated along the line connecting them offline

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





## Verlet Integration

- The new position  $\mathbf{x}'$  and velocity  $\mathbf{v}'$  are computed by

$$\begin{aligned}\mathbf{x}' &= \mathbf{x} + \mathbf{v} \cdot \Delta t \\ \mathbf{v}' &= \mathbf{v} + \mathbf{a} \cdot \Delta t,\end{aligned}$$

where  $\Delta t$  is the time step, and  $\mathbf{a}$  is the acceleration computed using Newton's law  $\mathbf{f} = m\mathbf{a}$  (where  $\mathbf{f}$  is the accumulated force acting on the particle).

- Instead of this rule the following rule is used here:

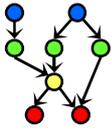
$$\begin{aligned}\mathbf{x}' &= 2\mathbf{x} - \mathbf{x}^* + \mathbf{a} \cdot \Delta t^2 \\ \mathbf{x}^* &= \mathbf{x}\end{aligned}$$

This is called Verlet integration and is used intensely when simulating molecular dynamics

- Advantage: more stable since the velocity is implicitly given

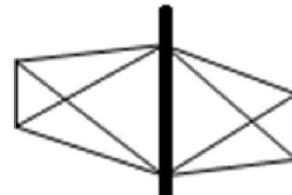
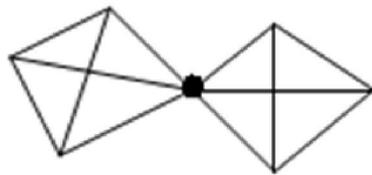
Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





## Modeling Joints

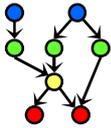
- It is possible to connect multiple rigid bodies by hinges, pin joints, and so on
  - Pin joint: two rigid bodies share a particle,
  - Hinge joint: two rigid bodies share two particles



A 3DOF pin joint (left) and 1DOF hinge joint (right)

Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>

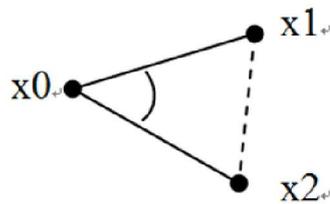




## Setting Joint Limits

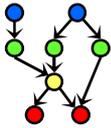
- A method for restraining angles is to satisfy a dot product constraint:

$$(\mathbf{x}_2 - \mathbf{x}_0) \cdot (\mathbf{x}_1 - \mathbf{x}_0) < \alpha.$$



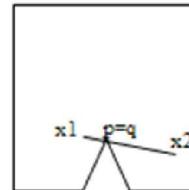
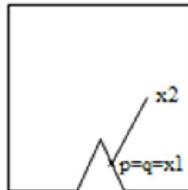
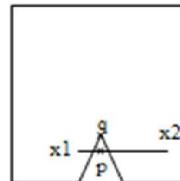
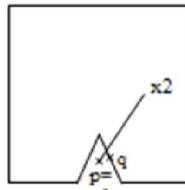
Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





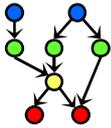
## Collision Detection Redux

- The collision detection will be done based on the points and the surfaces
- The colliding particle will be pushed out from the penetrating area
- The velocity of the colliding particle will be reduced to zero



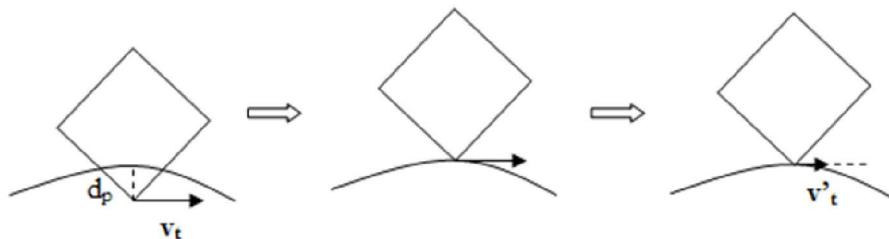
Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>





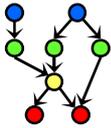
## Friction Force

- According to the Coulomb friction model, friction force depends on the size of the normal force between the objects in contact.
- The penetration depth  $d_p$  can be first measured when a penetration has occurs (before projecting the penetration point out of the obstacle),
- and then, the tangential velocity  $v_t$  is then reduced by an amount proportional to  $d_p$  (the proportion factor being the friction constant).



Adapted from slide ♥ 2008 H. P. H. Shum, RIKEN (理研)  
Computer Animation, <http://bit.ly/lq6KTK>

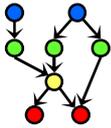




## Summary

- Reading for Last Class: Chapter 7, §8.4, Eberly 2<sup>e</sup>
- Reading for Today: §8.3 – 8.4, 4.2, 5.0, 5.6, 9.1, Eberly 2<sup>e</sup>
- Reading for Next Class: §9.1, **Particle System Handout**
- Last Time: Picking
  - \* OpenGL modes: rendering (default), feedback, selection
  - \* Name stack
  - \* Hit records
  - \* Rendering in selection mode using selection buffer
  - \* Color coding of pickable objects
- Today: Interaction Handling & Human Computer Interaction (HCI)
  - \* Spectrum of interaction
  - \* Kinds of interaction
    - User input: selection, control
    - Stimuli: much more when we cover visualization, color
- Next: Particle Systems, Collision Response





## Terminology

- **Picking: Allowing User to Select Objects in Scene**
  - \* **Selection mode:** mode when cursor (“mouse”) is active
  - \* **Name stack:** last in, first out data structure holding object names
  - \* **Hit records:** ID, depth info for intersections with view volume
  - \* **Selection buffer:** holds hits, depth (compare: frame/z-buffer)
  - \* **Color coding:** using color to represent pickable object ID
- **Human-Computer Interaction (HCI)**
  - \* aka **Man-Machine Interaction** (archaic), **Computer-Human Interaction**
  - \* Study, planning, design of interaction between humans & computers
  - \* **User interface**
    - Operation and control of machine
    - Feedback to human
- **Particle Systems – Simulation of Processes, Simple Physical Bodies**
  - \* **Events:** birth (emission), collision, death
  - \* **Properties:** mass, initial velocity, lifetime

