# Lecture 30 of 41

## Animation 3 of 3: Inverse Kinematics Control & Ragdoll Physics

William H. Hsu
Department of Computing and Information Sciences, KSU

KSOL course pages: http://bit.ly/hGvXlH / http://bit.ly/eVizrE
Public mirror web site: http://www.kddresearch.org/Courses/CIS636
Instructor home page: http://www.cis.ksu.edu/~bhsu

**Readings:**
Last class: Particle System Handout
Today: §5.3, Eberly 2e – see http://bit.ly/ieUq45; CGA Handout
Next class: Chapter 14, Eberly 2e
Reference: Wikipedia, *Inverse Kinematics*, http://bit.ly/hr8r2u
Reference: Wikipedia, *Ragdoll Physics*, http://bit.ly/3oggUZ





# Lecture Outline

- Reading for Last Class: Particle System Handout
- Reading for Today: §5.3, Eberly 2e; CGA Handout
- Reading for Next Class: Chapter 14, Eberly 2e
- Last Time: Lab on Particle Systems; Dissection of Working Program
- Today: Animation Part 3 of 3 – Inverse Kinematics
  - Autonomous agents (robots, swarms) *vs.* hand-animated movement
  - Forward kinematics and control
  - Inverse kinematics for autonomous movement in robotics
  - Jacobians and iterative minimization models
  - Rag doll physics
- End of Material on: Particle Systems, Collisions, CGA
- Also Conclusion of Physically-Based Modeling (PBM)
- Next Class: Ray Tracing, Part 1 of 2
  - Vectors: light/shadow (L), reflected (R), transmitted/refracted (T)
  - Basic recursive ray tracing: ray trees





# Where We Are

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

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.

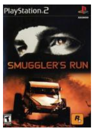




# Acknowledgements: Inverse Kinematics

**David C. Brogan**
Visiting Assistant Professor, Computer Science Department, University of Virginia
http://www.cs.virginia.edu/~dbrogan/
Susquehanna International Group (SIG)
http://www.sig.com

Computer Science at the University of Virginia




**Steve Rotenberg**
Visiting Lecturer
Graphics Lab
University of California – San Diego
CEO/Chief Scientist, PixelActive
http://graphics.ucsd.edu

UCSD

**Renata Melamud**
Ph.D. Candidate
Mechanical Engineering Department
Stanford University
http://micromachine.stanford.edu/~rmelamud/

Stanford Mechanical Engineering





# Review [1]: Uses of Particle Systems

- Explosions
  - Large
  - Fireworks
- Fire
- Vapor
  - Clouds
  - Dust
  - Fog
  - Smoke
  - Contrails
- Water
  - Waterfalls
  - Streams
- Plants


*Command & Conquer 4: Tiberian Twilight*
© 2010 Electronic Arts, Inc.
Wikipedia: http://bit.ly/gFGMjO

Adapted from slides ♥ 2008 R. Malhotra, CSU San Marcos
CS 536 Intro to 3-D Game Graphics, Spring 2008 – http://bit.ly/hNhUuE

California State University SAN MARCOS

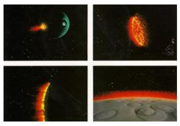




# Review [2]: History of Particle Systems

*Spacewar!* © 1962 S. Russell *et al.*
Wikipedia: http://bit.ly/eaiWUW


*Asteroids* © 1979 L. Rains & E. Logg
Wikipedia: http://bit.ly/hwfEQk



*Star Trek II* © 1983 Paramount
Wikipedia: http://bit.ly/eXwrhb

- *Spacewar!* (1962) Used Pixel Clouds as Explosions
- *Asteroids* (1979) First "Physically-Based" PS/Collision Model in Games
- *Star Trek II* (1983) Particle Fountain: http://youtu.be/Qe9qSLYK5q4
- *Hey, Hey, 16K* © 2000 M. J. Hibbett, Video © 2004 R. Manuel http://youtu.be/Ts96J7HhO28

Adapted from slides ♥ 2008 R. Malhotra, CSU San Marcos
CS 536 Intro to 3-D Game Graphics, Spring 2008 – http://bit.ly/hNhUuE

California State University SAN MARCOS

7

## Review [3]: Definition & Physically-Based Model

- A particle system is a collection of a number of individual elements or *particles.*
- *Particle systems control a set of particles that act autonomously but share some common attributes.*

- Particle is a point in 3D space.
- Forces (e.g. gravity or wind) accelerate a particle.
- Acceleration changes velocity.
- Velocity changes position

Adapted from slides ♥ 2008 R. Malhotra, CSU San Marcos
CS 536 Intro to 3-D Game Graphics, Spring 2008 – http://bit.ly/hNhUuE



8

## Review [4]: More Attributes of Particles

- Position
- Velocity
- Life Span
- Size
- Weight
- Representation
- Color
- Owner

Adapted from slides ♥ 2008 R. Malhotra, CSU San Marcos
CS 536 Intro to 3-D Game Graphics, Spring 2008 – http://bit.ly/hNhUuE



9

## Review [5]: Four Ways to Represent Particles

- Points
- Lines
- Texture-mapped quads
- Point Sprites

Adapted from slides ♥ 2008 R. Malhotra, CSU San Marcos
CS 536 Intro to 3-D Game Graphics, Spring 2008 – http://bit.ly/hNhUuE



10

## Kinematics

- The study of object movements irrespective of their speed or style of movement

Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd



11

## Degrees of Freedom (DOFs) [1]: Translational & Rotational

- The variables that affect an object's orientation
- How many degrees of freedom when flying?

- So the kinematics of this airplane permit movement anywhere in three dimensions
- Six
  - x, y, and z positions
  - roll, pitch, and yaw

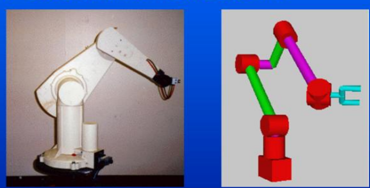
Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd



12

## Degrees of Freedom (DOFs) [2]: Robot Arm

- How about this robot arm?

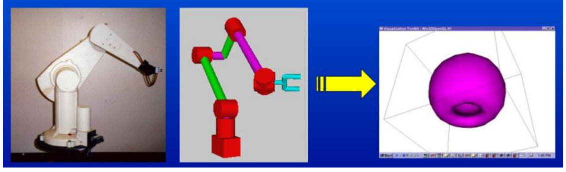


- Six again
  - 2-base, 1-shoulder, 1-elbow, 2-wrist

Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd

## Configuration Space

- The set of all possible positions (defined by kinematics) an object can attain







## Work Space vs. Configuration Space

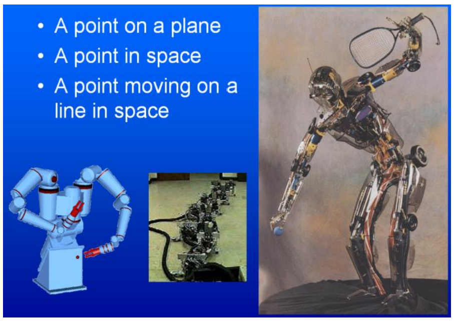
- Work space
  - The space in which the object exists
  - Dimensionality
    - $R^3$ for most things, $R^2$ for planar arms
- Configuration space
  - The space that defines the possible object configurations
  - Degrees of Freedom
    - The number of parameters that necessary and sufficient to define position in configuration





## More Examples

- A point on a plane
- A point in space
- A point moving on a line in space

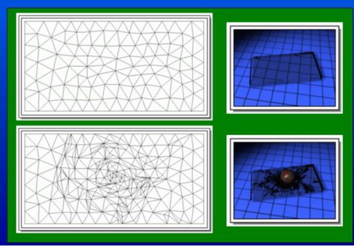






## Controlled DOFs

- DOFs that you can actually control (position explicitly)

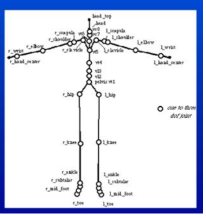






## Hierarchical Kinetic Modeling

- A family of parent-child spatial relationships are functionally defined
  - Moon/Earth/Sun movements
  - Articulations of a humanoid
- Limb connectivity is built into model (joints) and animation is easier

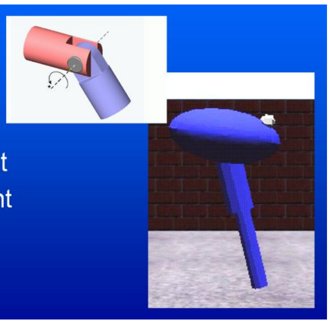






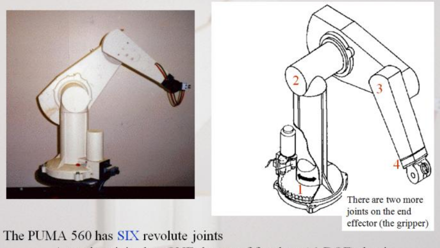
## Robot Parts & Terms

- Links
- End effector
- Frame
- Revolute Joint
- Prismatic Joint



19

## Example: Puma 560 Robot




The PUMA 560 has SIX revolute joints

A revolute joint has ONE degree of freedom ( 1 DOF) that is defined by its angle

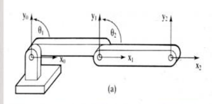
Wikipedia, *Programmable Universal Machine for Assembly (PUMA )*: http://bit.ly/fBMRaM

Adapted from slides ♥ 2002 R. Melamud, Stanford University
Mirrored at CMU 16-311 Introduction to Robotics, http://generalrobotics.org

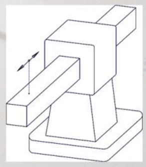


20

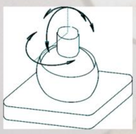
## Joint Types: Revolute, Prismatic, Spherical



Revolute Joint
1 DOF ( Variable - Y)



Prismatic Joint
1 DOF (linear) (Variables - d)



Spherical Joint
3 DOF ( Variables - $Y_1$, $Y_2$, $Y_3$)

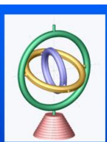
Adapted from slides ♥ 2002 R. Melamud, Stanford University
Mirrored at CMU 16-311 Introduction to Robotics, http://generalrobotics.org

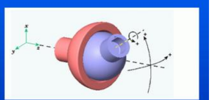


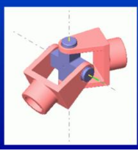
21

## More Complex Joints

- 3 DOF joints
  - Gimbal
  - Spherical (doesn't possess singularity)
- 2 DOF joints
  - Universal







Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd

Computer Science at the University of Virginia



22

## Hierarchical Representation

- Model bodies (links) as nodes of a tree
- All body frames are local (relative to parent)
  - Transformations affecting root affect all children
  - Transformations affecting any node affect all its children

Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd

Computer Science at the University of Virginia



23

## Forward vs. Inverse Kinematics

- Forward Kinematics
  - Compute configuration (pose) given individual DOF values
- Inverse Kinematics
  - Compute individual DOF values that result in specified end effector position

Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd

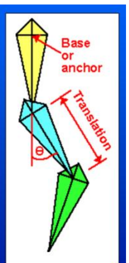
Computer Science at the University of Virginia



24

## Forward Kinematics [1]: Definition & General Approach

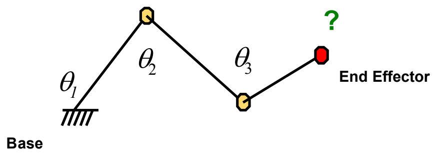
- Traverse kinematic tree and propagate transformations downward
  - Use stack
  - Compose parent transformation with child's
  - Pop stack when leaf is reached
- High DOF models are tedious to control this way




Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd

Computer Science at the University of Virginia

# Forward Kinematics [2]: Illustration




$$\vec{\mathbf{x}} = \mathrm{f}(\vec{\boldsymbol{\theta}})$$

Choi

$$\mathbf{e} = f(\mathbf{\Phi})$$

Rotenberg

Adapted from slides ♥ 2002 K. J. Choi, Seoul National University Graphics and Media Lab (http://graphics.snu.ac.kr) – mirrored at: http://bit.ly/hnzSAN





# Forward Kinematics [3]: Joint Angles to Bone Coordinates

- The local and world matrix construction within the skeleton is an implementation of *forward kinematics*
- Forward kinematics refers to the process of computing world space geometric descriptions (matrices…) based on joint DOF values (usually rotation angles and/or translations)

Adapted from slides ♥ 2004 – 2005 S. Rotenberg, UCSD
CSE169: Computer Animation, Winter 2005 – http://bit.ly/f0ViAN





# Inverse Kinematics [1]: Definition & General Approach

- Given end effector position, compute required joint angles
- In simple case, analytic solution exists
  - Use trig, geometry, and algebra to solve

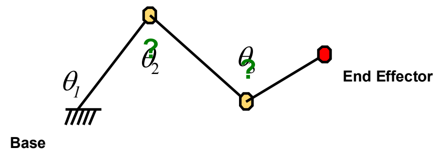
Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd





# Inverse Kinematics [2]: Illustration

For more on characters & IK, see:
Advanced Topics in CG Lecture 05




$$\vec{\boldsymbol{\theta}} = \mathrm{f}^{-1}(\vec{\mathbf{x}})$$

Choi

$$\mathbf{\Phi} = f^{-1}(\mathbf{e})$$

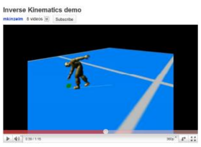
Rotenberg

Adapted from slides ♥ 2002 K. J. Choi, Seoul National University Graphics and Media Lab (http://graphics.snu.ac.kr) – mirrored at: http://bit.ly/hnzSAN





# Inverse Kinematics [3]: Demos




© 2008 M. Kinzelman
http://youtu.be/I52yZ491kPo


© 2007 A. Brown
http://youtu.be/6JdLOLazJJ0


© 2008 T. Komura, H. S. Lim, & R. W. H. Lau
http://youtu.be/FJTBMnP6oCM


© 2011 K. Iyer
http://youtu.be/YvRBWIRAPsE





# Inverse Kinematics [4]: Analytic Solution for 2-Link Case

$$x^2 + y^2 = a_1^2 + a_2^2 - 2a_1a_2\cos(\pi - \theta_2)$$

$$\cos\theta_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

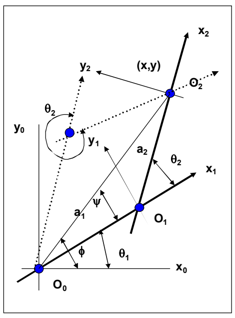
for greater accuracy

$$\tan^2\frac{\theta_2}{2} = \frac{1-\cos\theta}{1+\cos\theta} = \frac{2a_1a_2 - x^2 - y^2 + a_1^2 + a_2^2}{2a_1a_2 + x^2 + y^2 - a_1^2 - a_2^2}$$

$$= \frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}$$

$$\theta_2 = \pm 2\tan^{-1}\sqrt{\frac{\left(a_1^2 + a_2^2\right)^2 - \left(x^2 + y^2\right)}{\left(x^2 + y^2\right) - \left(a_1^2 - a_2^2\right)^2}}$$

**Two solutions: elbow up & elbow down**




Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd

31

## Inverse Kinematics [5]: Iterative IK Solutions

- Frequently analytic solution is infeasible
- Use Jacobian
- Derivative of function output relative to each of its inputs
- If y is function of three inputs and one output

$$y = f(x_1, x_2, x_3)$$

$$\delta y = \frac{\partial f}{\partial x_1} \cdot \delta x_1 + \frac{\partial f}{\partial x_2} \cdot \delta x_2 + \frac{\partial f}{\partial x_3} \cdot \delta x_3$$

- Represent Jacobian J(X) as a 1x3 matrix of partial derivatives

Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd



32

## Jacobian [1]: 6x6 DOF Case

- In another situation, end effector has 6 DOFs and robotic arm has 6 DOFs
- $f(x_1, \ldots, x_6) = (x, y, z, r, p, y)$
- Therefore J(X) = 6x6 matrix

$$\begin{bmatrix} \frac{\partial f_x}{\partial x_1} & \frac{\partial f_y}{\partial x_1} & \frac{\partial f_z}{\partial x_1} & \frac{\partial f_r}{\partial x_1} & \frac{\partial f_p}{\partial x_1} & \frac{\partial f_y}{\partial x_1} \\ \frac{\partial f_x}{\partial x_2} & & & & & \\ \frac{\partial f_x}{\partial x_3} & & & & & \\ \frac{\partial f_x}{\partial x_4} & & & & & \\ \frac{\partial f_x}{\partial x_5} & & & & & \\ \frac{\partial f_x}{\partial x_6} & & & & & \end{bmatrix}$$

Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd



33

## Jacobian [2]: Solution

- Relates velocities in parameter space to velocities of outputs

$$\dot{Y} = J(X) \cdot \dot{X}$$

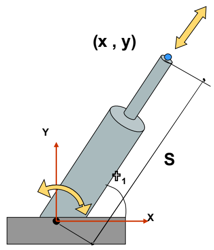
- If we know $Y_{current}$ and $Y_{desired}$, then we subtract to compute $Y_{dot}$
- Invert Jacobian and solve for $X_{dot}$

Adapted from slides ♥ 2000 – 2005 D. Brogan, University of Virginia
CS 551, Advanced CG & Animation – http://bit.ly/hUXrqd



34

## Another IK Problem: Revolute & Prismatic Joints Combined




**Finding ϑ:**

$$\theta = \arctan\left(\frac{y}{x}\right)$$

**More Specifically:**

$$\theta = \arctan 2\left(\frac{y}{x}\right)$$

**arctan2() specifies that it's in the first quadrant**

**Finding S:**

$$S = \sqrt{(x^2 + y^2)}$$

Adapted from slides ♥ 2002 R. Melamud, Stanford University
Mirrored at CMU 16-311 Introduction to Robotics, http://generalrobotics.org



35

## Ragdoll Physics [1]: Definition

- Type of Procedural Animation
  - Automatically generates CGA directives (rotations)
  - Based on simulation
  - Rigid-body dynamics
- Articulated Figure
  - Gravity
  - No autonomous movement
  - Used for inert body
    - Usually: character death (car impact, falling body, *etc.*)
    - Less often: unconscious, paralyzed character
- Collisions with Multiple Bodies
  - Inter-character
  - Character-object

*Falling Bodies* © 1997 – 2001 Animats
http://www.animats.com



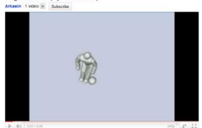
36

## Ragdoll Physics [2]: Demos


© 2007 N. Picouet
http://youtu.be/ohNqCb--aSs


© 2006 P. Pelt
http://youtu.be/6JdLOLazJJ0
See also: http://youtu.be/5_QIsl0fyaU


© 2009 M. E. Cerquoni
http://youtu.be/uW_DK2qvKv8




© 2010 M. Heinzen (Arkaein)
http://bit.ly/gUj9Su / http://youtu.be/FJTBMnP6oCM

# Physically-Based Modeling (PBM) [1]: Looking Back

- Particle Dynamics
  - Emitters
    - 0-D (points), 1-D (lines), 2-D (planes, discs, cross-sections)
    - *e.g.*, fireworks (0-D); fountains (0/1/2-D); smokestacks, jets (2-D)
  - Simulation: birth-death process, functions of particle age/trajectory
- Rigid-Body Dynamics
  - Constrained systems of connected parts
  - Examples: falling rocks, colliding vehicles, rag dolls
- Articulated Figures
- More References
  - ACM, *Intro to Physically-Based Modeling*: http://bit.ly/hhQvXd
  - Wikipedia, *Physics Engine*: http://bit.ly/h4PIRt
  - Wikipedia, *N-Body Problem*: http://bit.ly/1ayWwe

Rocks fall
Everyone dies





# Physically-Based Modeling (PBM) [2]: Applications in Movies & Games


*Star Wars Episode I: The Phantom Menace* © 1999 Lucasfilm, Inc. http://youtu.be/d4PSMXUCi-0





# Summary

- Reading for Today: §5.3, Eberly 2ᵉ; CGA Handout
- Reading for Next Class: Chapter 14, Eberly 2ᵉ
- Last Class: Lab on Particle Systems; Dissection of Working Program
- Today: Computer-Generated Animation Concluded
  - CGA of autonomous agents (robots, swarms) *vs.* animation by hand
  - Degrees of freedom (DOFs) and kinds of joints
  - Forward kinematics (FK)
    - Forward problem illustrated
    - Control problem
  - Inverse kinematics (IK)
    - IK (finding angles) *vs.* mechanical problem of finding forces
    - Analytical models
    - Iterative models (Jacobian-based)
  - Ragdoll physics
- Next Class: Ray Tracing, Part 1 of 2





# Terminology

- Emitter – Point, Line, Plane or Region from which Particles Originate
- Particle Fountain – Particle System with Directional Emitter
- Sprite (Wikipedia: http://bit.ly/gyInPg)
  - Definition: 2-D image or animation made part of larger scene
  - Point sprite (Saar & Rotzler, 2008): http://bit.ly/fkjBPY
- Joints: Parts of Robot / Articulated Figure That Turn, Slide
  - Revolute: able to turn (rotate), forming angle between bones
  - Prismatic (*aka* slider): "bone" slides through – http://bit.ly/hScjoe
  - Spherical (*aka* ball joint): "bone" rotates around socket
  - Cylindrical (*aka* hinge): flaps wrap around joint, joined to surfaces
- Effectors: Parts of Robot / Articulated Figure That Act (*e.g.*, Hand, Foot)
- Bones: Effectors, Other Parts That Rotate about, Slide through Joints
- Procedural Animation: Automatic Generation of Motion *via* Simulation
  - Ragdoll physics: procedural animation for inert characters
  - Other types: particle systems, *N*-body dynamics