

Animation

Modifying scene parameters as a function of time

3) Scripting

5) Keyframing

7) Inverse Kinematics

9) Behavioral Animation

11) Dynamics

Scripting

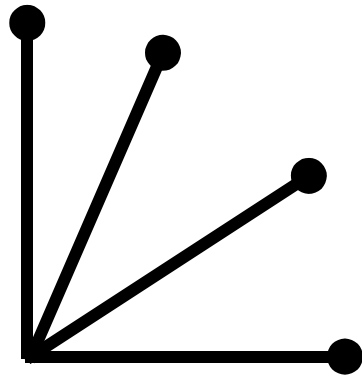
Specifying the parameters at every frame

```
define spinningCube()  
    rotAngle = pi*frameNumber / 50
```

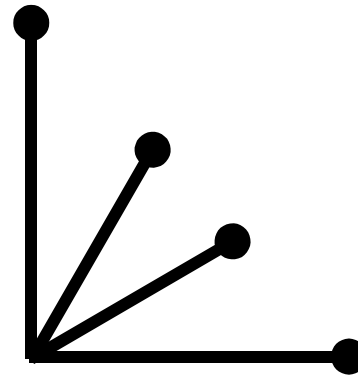
```
define carScript()  
    carTranslation = 10*(frameNumber / 100)  
    wheelRotation = pi*frameNumber / 5
```

Keyframing

Specify only the important frames,
interpolate the frames in-between



interpolating angle

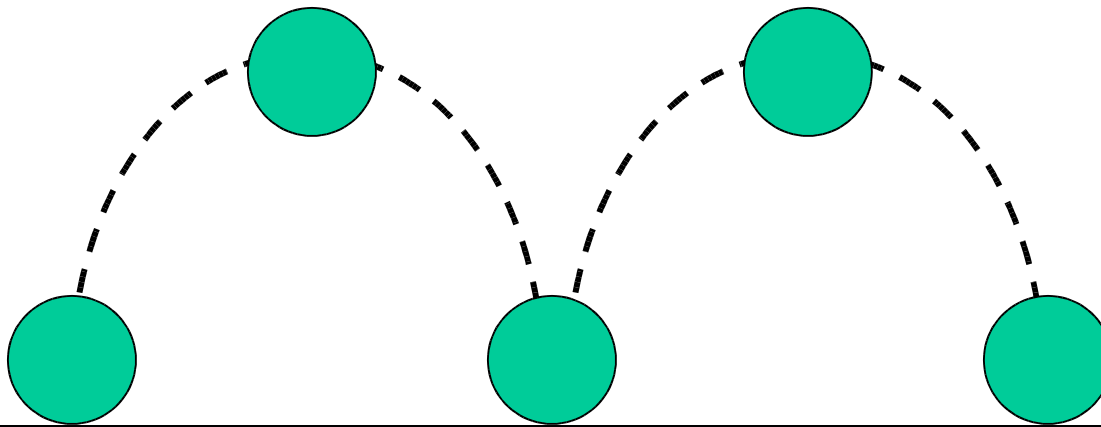


interpolating endpoints

What and how to interpolate is important

Interpolating Translations

Interpolate using a cubic B spline



Use an arc length parameterization

Interpolating Rotations

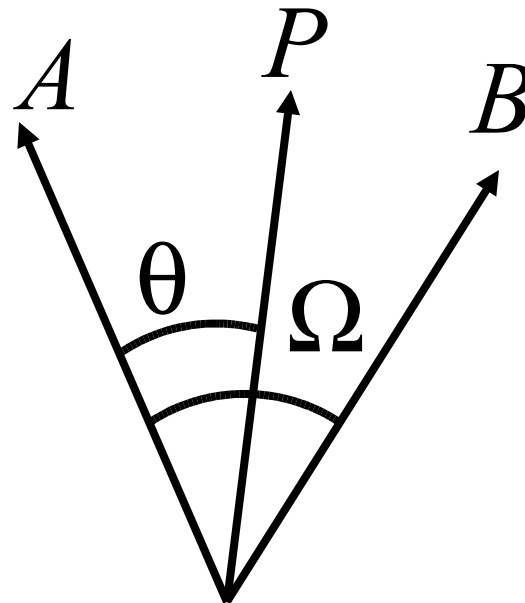
Interpolate using spherical interpolation

$$P = \alpha A + \beta B$$

$$A \cdot P = \cos \theta$$

$$A \cdot B = \cos \theta$$

$$|A| = |B| = |P| = 1$$



Interpolating Rotations

Solving for the coefficients

$$A \cdot P = \alpha A \cdot A + \beta A \cdot B \quad P \cdot P = \alpha P \cdot A + \beta P \cdot B$$

$$\cos \theta = \alpha + \beta \cos \Omega$$

$$1 = \alpha \cos \theta + \beta \cos(\Omega - \theta)$$

$$\alpha = \frac{\sin(\Omega - \theta)}{\sin \Omega}$$

$$\beta = \frac{\sin \theta}{\sin \Omega}$$

Interpolating Rotations

Letting $\theta = \Omega u$ $u \in [0,1]$

$$P = \frac{\sin(\Omega - \theta)}{\sin \Omega} A + \frac{\sin \theta}{\sin \Omega} B$$

$$P = \frac{\sin((1-u)\Omega)}{\sin \Omega} A + \frac{\sin(\Omega u)}{\sin \Omega} B$$

Interpolate using quaternions

Interpolating Rotations

Spherical linear interpolation is discontinuous

Need a higher order interpolation

Interpolate using B splines

Must constrain the curve to be on the surface of the unit sphere in 4D

Interpolating Time

Interpolate using 2 splines

Kinetic spline: $keyframe \Rightarrow time$

$$(0, t_0), \dots, (n-1, t_{n-1})$$

Position spline: $keyframe \Rightarrow parameter$

$$(0, X_0), \dots, (n-1, X_{n-1})$$

Allows position and time to be edited independently

Kinematics

The study or specification of motion,
independent of the underlying physics
that created the motion

Articulated Figure

A figure made up of a series of links (bones)
connected at joints

Kinematics

State Vector θ

The vector of independent variables describing the state of the articulated figure

Degrees of Freedom (DOF)

Number of variables in the state vector

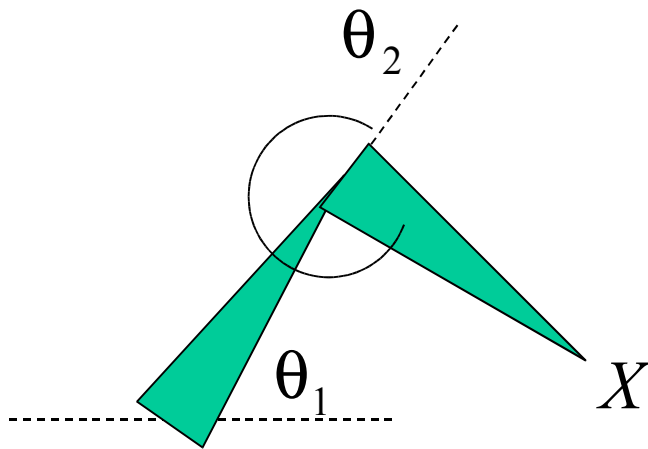
End Effector X

The pose of the end of a chain of links

Forward Kinematics

Given the state vector,
calculate the pose of the end effector

$$X = f(\theta)$$

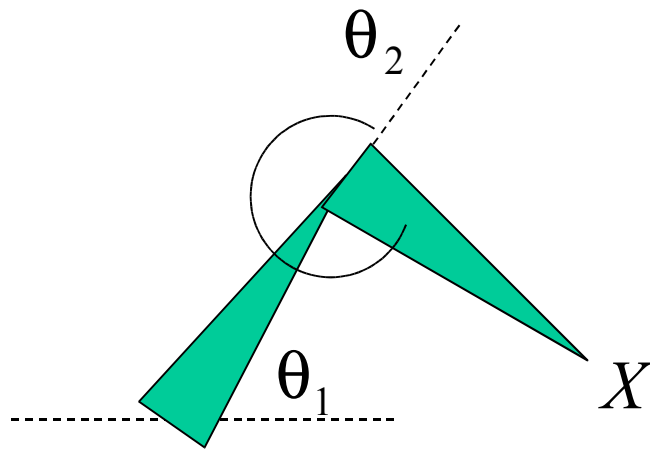


$$X = \begin{bmatrix} l_1 \cos \theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin \theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix}$$

Inverse Kinematics

Given the pose of the end-effector,
calculate the state vector

$$\theta = f^{-1}(X)$$



$$\theta = \begin{bmatrix} -(l_2 \sin \theta_2)x + (l_1 + l_2 \cos \theta_2)y \\ (l_2 \sin \theta_2)y + (l_1 + l_2 \cos \theta_2)x \\ \cos^{-1} \frac{(x^2 + y^2 - l_1^2 - l_2^2)}{2l_1l_2} \end{bmatrix}$$

Inverse Kinematics

The kinematic function is usually highly nonlinear and hard to invert

Consider it as locally linear

$$X = f(\theta)$$

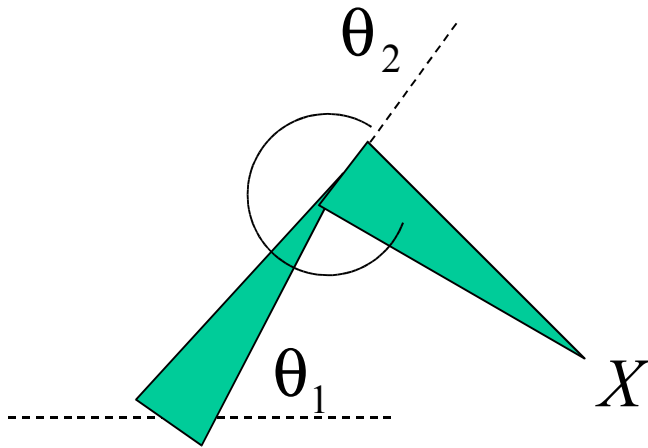
$$X + dX = f(\theta) + J(\theta)d\theta + \dots$$

$$dX \approx J(\theta)d\theta$$

$$d\theta \approx J^{-1}(\theta)dX$$

Inverse Kinematics

Calculating the Jacobian



$$X = \begin{bmatrix} l_1 c_1 + l_2 c_{12} \\ l_1 s_1 + l_2 s_{12} \end{bmatrix}$$

$$X' = \begin{bmatrix} -l_1 s_1 \theta_1' - l_2 s_{12} (\theta_1' + \theta_2') \\ l_1 c_1 \theta_1' + l_2 c_{12} (\theta_1' + \theta_2') \end{bmatrix}$$

$$X' = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix} \begin{bmatrix} \theta_1' \\ \theta_2' \end{bmatrix}$$

$$X' = J(\theta) \begin{bmatrix} \theta_1' \\ \theta_2' \end{bmatrix}$$

Behavioral Animation

Animating by describing an actor's behavior

An actor's behavior defines how the actor interacts with other actors and the environment

```
TRex()  
  if(player is close)  
    eatPlayer()  
  else if(can see player)  
    chasePlayer()  
  else  
    wander()
```

Behavioral Animation

Boids (birds) [Reynolds87]

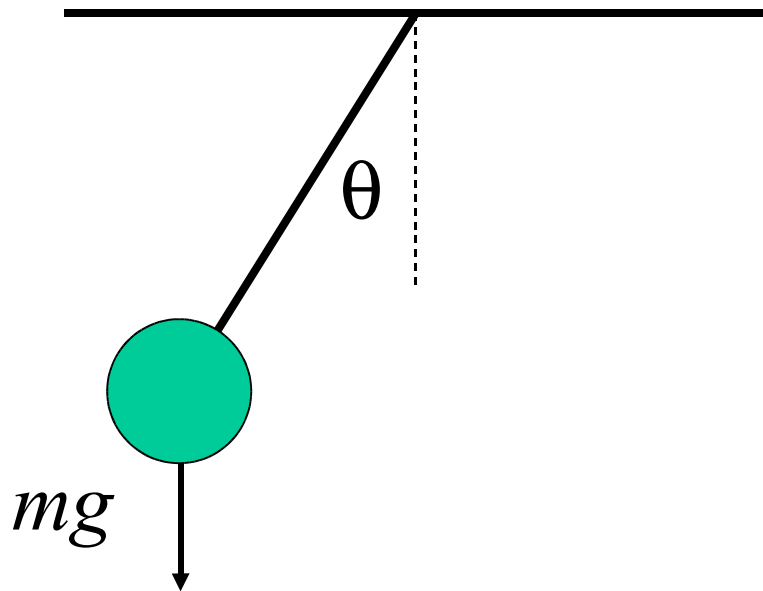
- 1) Collision Avoidance
avoid collisions with nearby flockmates
- 3) Velocity Matching
match velocity with nearby flockmates
- 5) Flock Centering
stay near centroid of nearby flockmates

Behavioral Animation



Dynamics

Using “physics” to define the animation



Model choice is important

(1)

$$\dot{X} = V$$

$$\dot{V} = \frac{F}{m}$$

(2)

$$\dot{X} = V$$

$$\dot{P} = F$$

$$\dot{\theta} = \omega$$

$$\dot{L} = T$$

Can use “augmented” laws of physics

Dynamics

Particle Systems [Reeves83]

Represent “fuzzy” objects such as fire and smoke as a collection of particles

Each particle contains local state

- Position
- Velocity
- Age
- Lifespan
- Rendering properties
- ...

Dynamics

Particle Systems [Reeves83]

- 1) Create new particles
- 3) Assign initial attributes
- 5) Remove dead particles
- 7) Update particles
- 9) Render particles

