CS248 Lecture 14

Character Animation and Physics

Zhenglin Geng

February 15th, 2018

Overview

- Basics on character animation
- Articulated rigid bodies
- Inverse kinematics

Basics on Character Animation

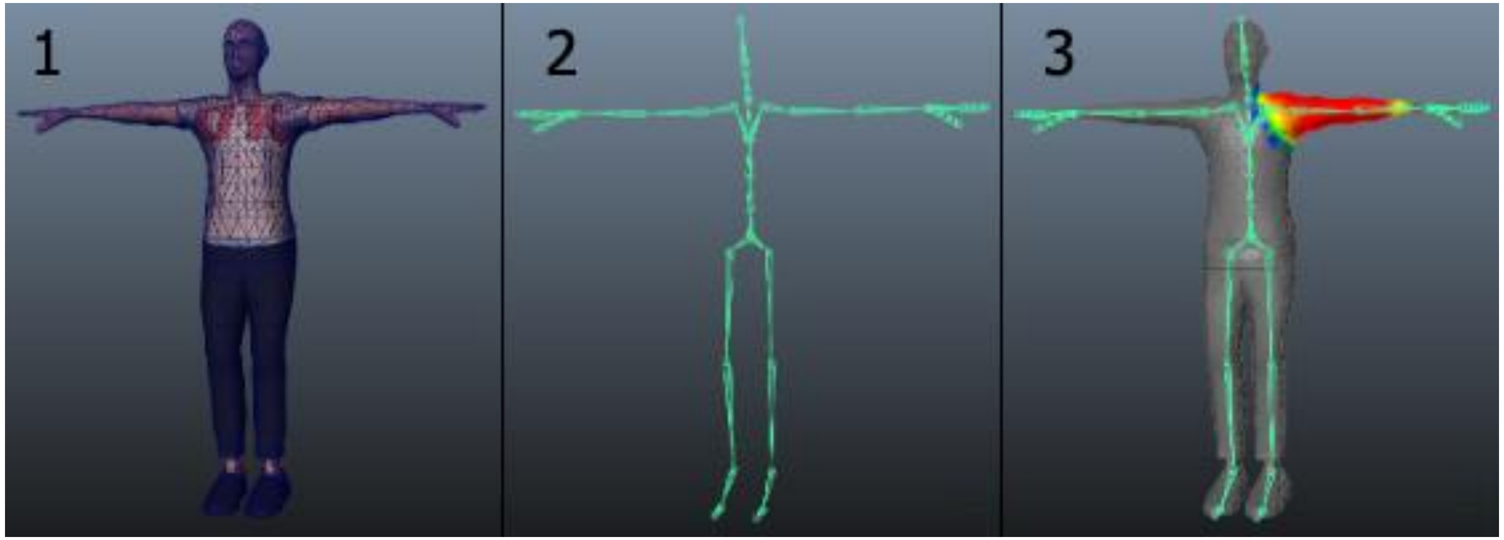
3D Animation

Battlefield 3 Animation (Upright, Crouch, Prone)

3D Animation

Battlefield 4 Animation (Running)

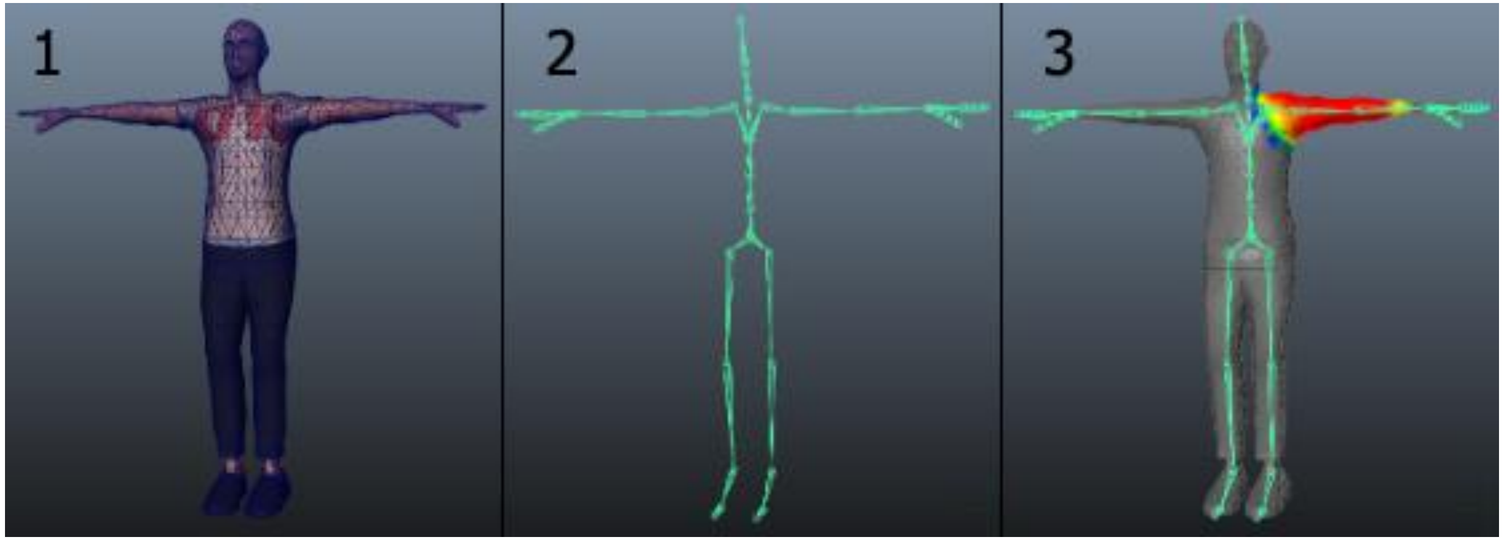
Prepare your own character



Modeling **Rigging** **Skinning**

Retargeting

Prepare your own character



Modeling

- Sensible topology
- T-Pose

Rigging

- HIPS - spine - chest - shoulders - arm - forearm - hand
- HIPS - spine - chest - neck - head
- HIPS - UpLeg - Leg - foot - toe - toe_end

Skinning

- Use an automated process initially
- Incrementally editing and refining

Animating Characters

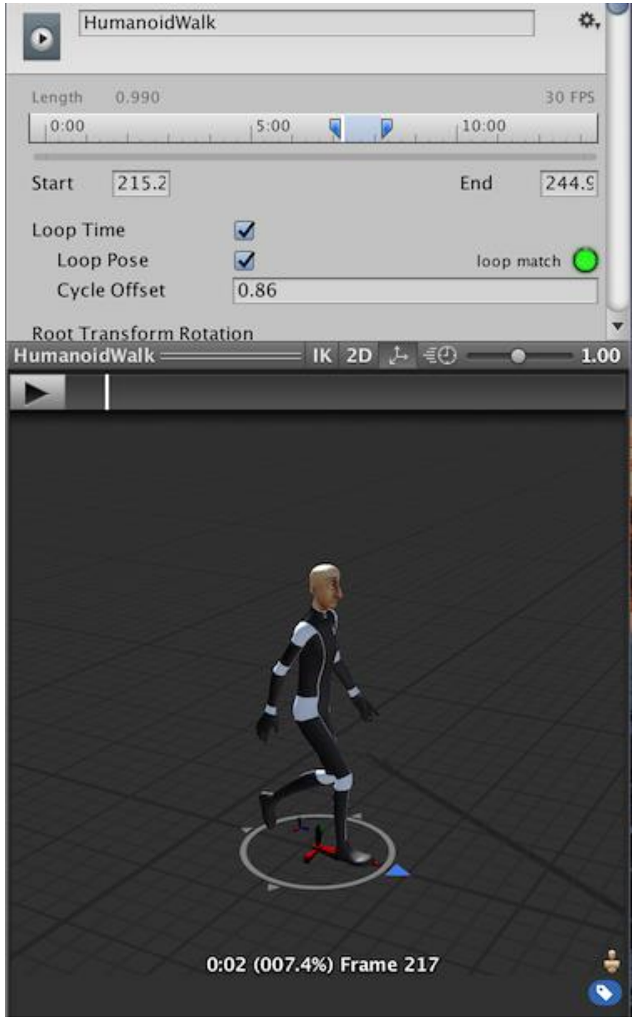
Animation from external sources

- Mocap
- 3DS Max, Maya or Blender
- Unity’s asset store
- Multiple clips cut and sliced from a single imported timeline.

Animation created and edited within Unity

- Position, rotation and scale of GameObjects

Use standard format FBX



HumanoidWalk

Length 0.990 30 FPS

Start 215.2 End 244.9

Loop Time ☑

Loop Pose ☑ loop match ●

Cycle Offset 0.86

Root Transform Rotation

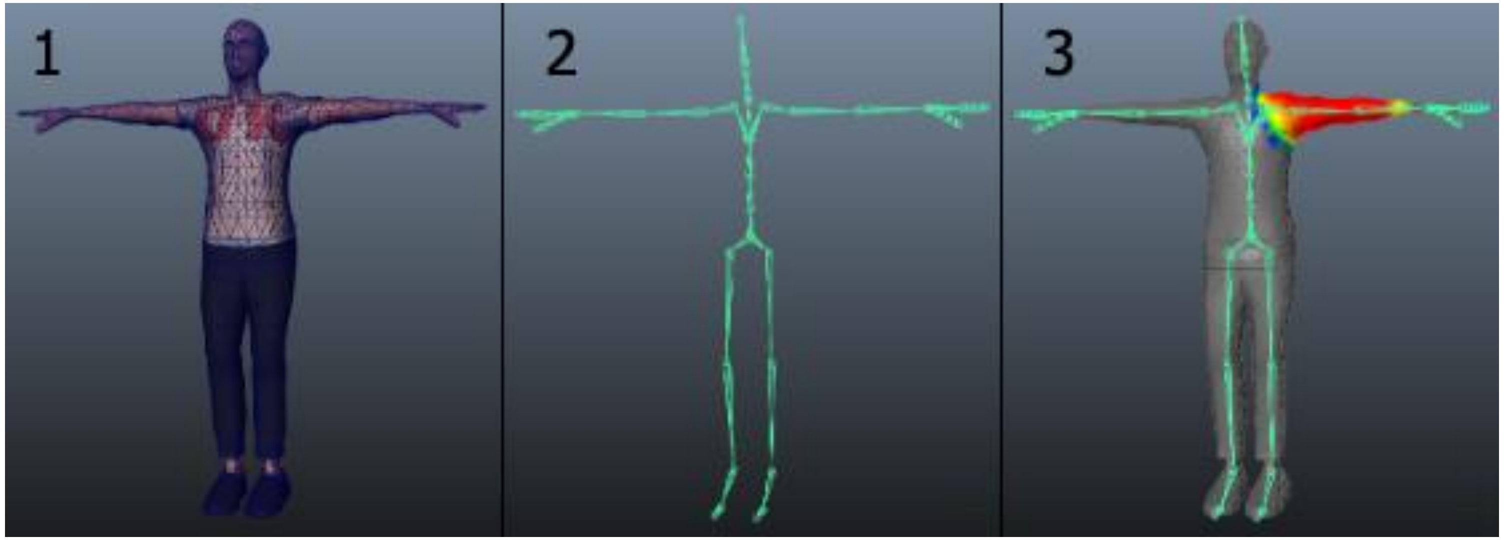
HumanoidWalk IK 2D 1.00

0:02 (007.4%) Frame 217

Using Humanoid Characters

How to obtain humanoid models

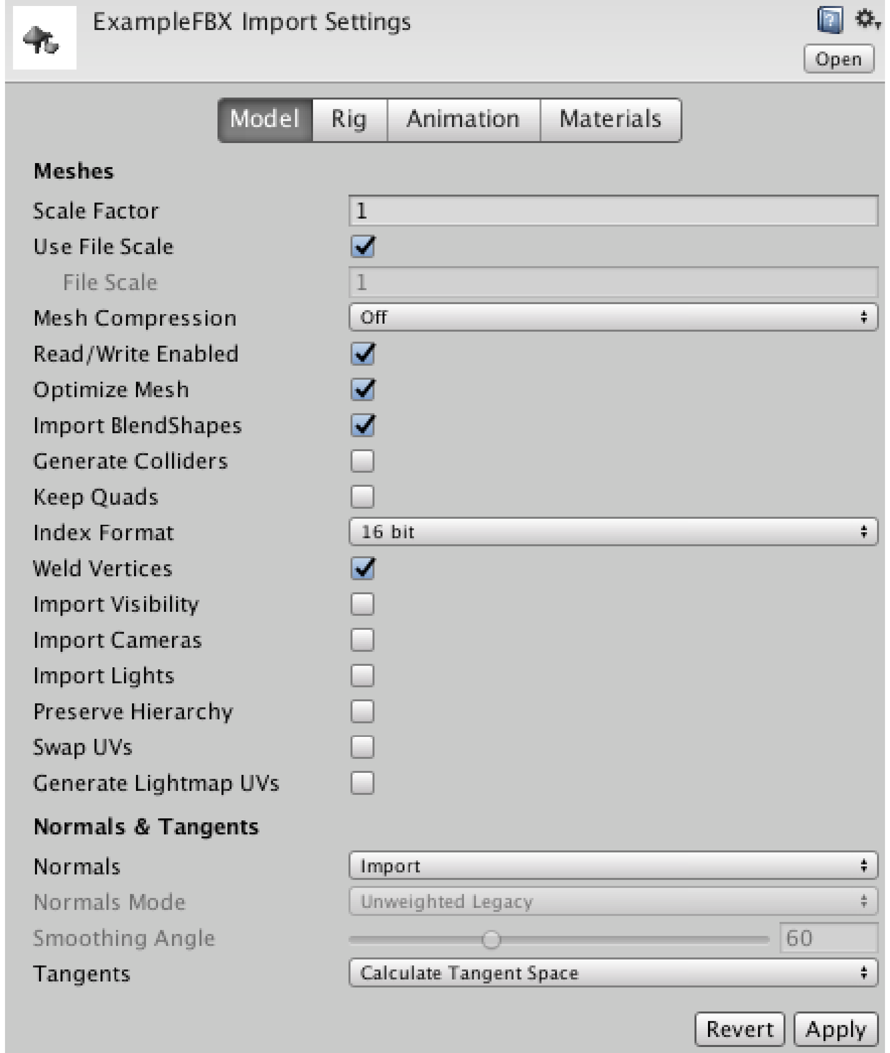
- Procedural character modeling or character generator such as *Poser*, *MakeHuman* or *Mixamo*
- Unity assets store
- Build from scratch



Importing Models

Model

- 3D Model, such as a character, a building or a piece of furniture



ExampleFBX Import Settings

Meshes

Scale Factor	1
Use File Scale	☑
File Scale	1
Mesh Compression	Off
Read/Write Enabled	☑
Optimize Mesh	☑
Import BlendShapes	☑
Generate Colliders	□
Keep Quads	□
Index Format	16 bit
Weld Vertices	☑
Import Visibility	□
Import Cameras	□
Import Lights	□
Preserve Hierarchy	□
Swap UVs	□
Generate Lightmap UVs	□

Normals & Tangents

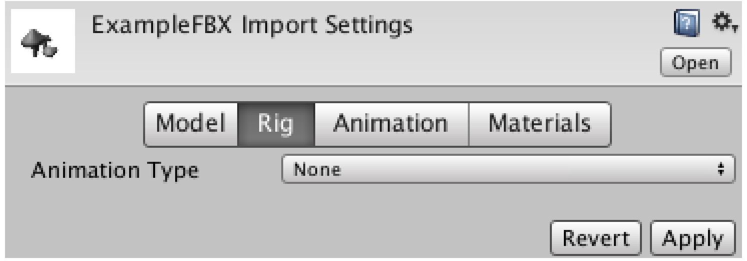
Normals	Import
Normals Mode	Unweighted Legacy
Smoothing Angle	60
Tangents	Calculate Tangent Space

Stanford University

Importing Models

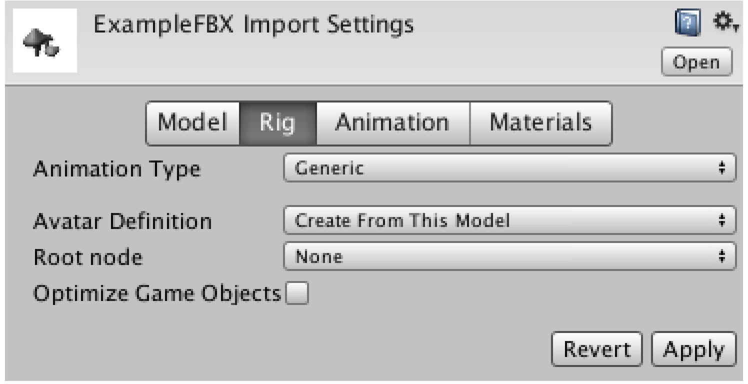
Rig

- Animation type:
  - None
  - Legacy
  - Generic
  - Humanoid



ExampleFBX Import Settings

Animation Type: None



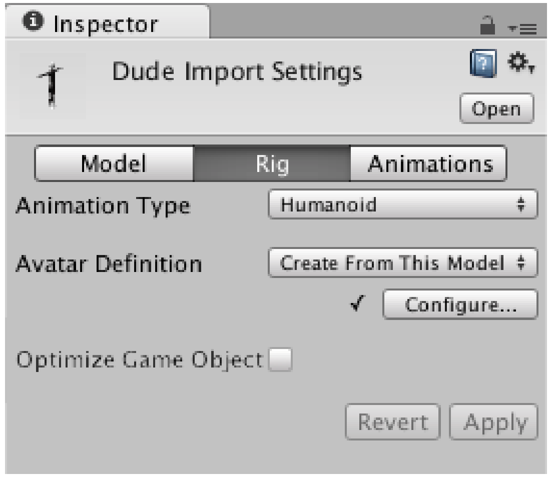
ExampleFBX Import Settings

Animation Type: Generic

Avatar Definition: Create From This Model

Root node: None

Optimize Game Objects □



Inspector

Dude Import Settings

Animation Type: Humanoid

Avatar Definition: Create From This Model

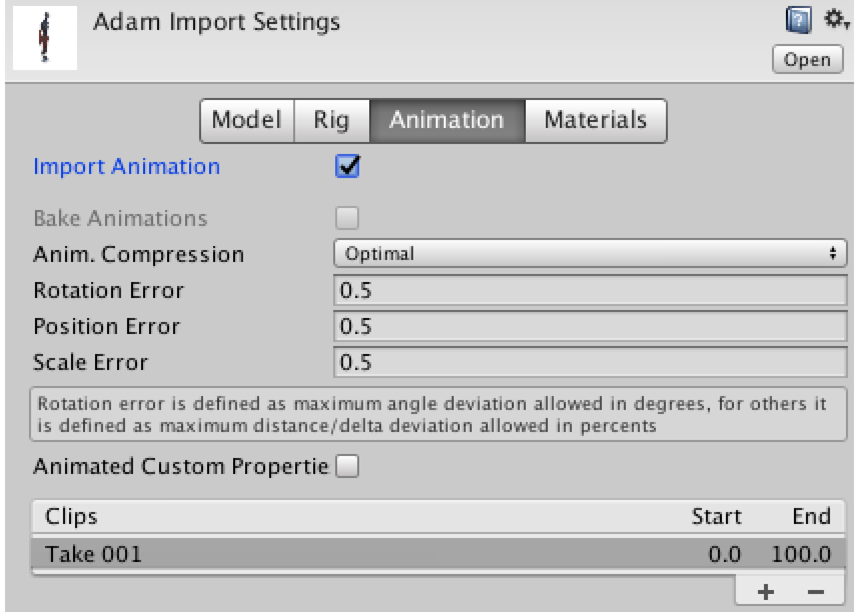
✓

Optimize Game Object □

Importing Models

Animation

- Animation clips



Adam Import Settings

Import Animation ☑

Bake Animations □

Anim. Compression: Optimal

Rotation Error: 0.5

Position Error: 0.5

Scale Error: 0.5

Rotation error is defined as maximum angle deviation allowed in degrees, for others it is defined as maximum distance/delta deviation allowed in percents

Animated Custom Propertie □

Clips	Start	End
Take 001	0.0	100.0

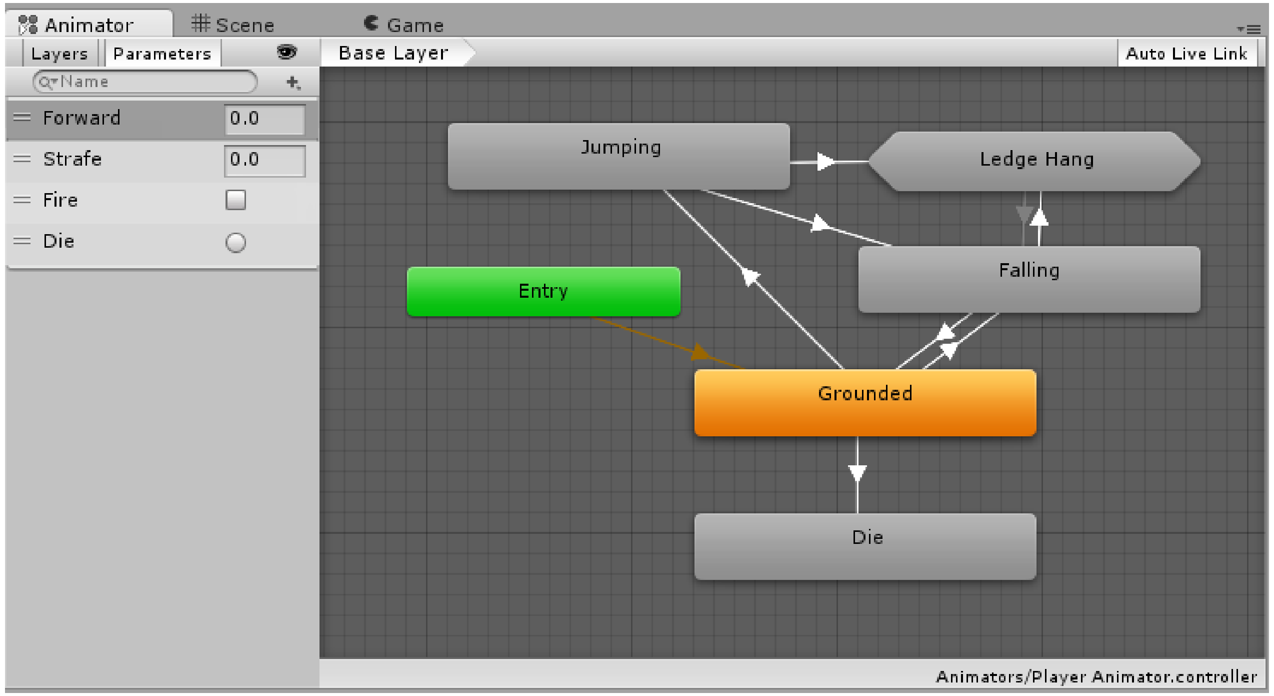
Demo

Review

- **Prepare your character animation**
  - Modeling, rigging, skinning
  - Retargeting
  - Obtain humanoid models: *Poser, MakeHuman, Mixamo*
  - Animating characters
  - Working with FBX

Animator Controllers

- **Animator Controller**
    - arrange and maintain a set of Animation Clips and associated Animation Transitions for a character or object
- **Animation State Machine**
    - a flow-chart of Animation Clips and Transitions
- **States** (animation clips)
- **State transition**



Forward 0.0

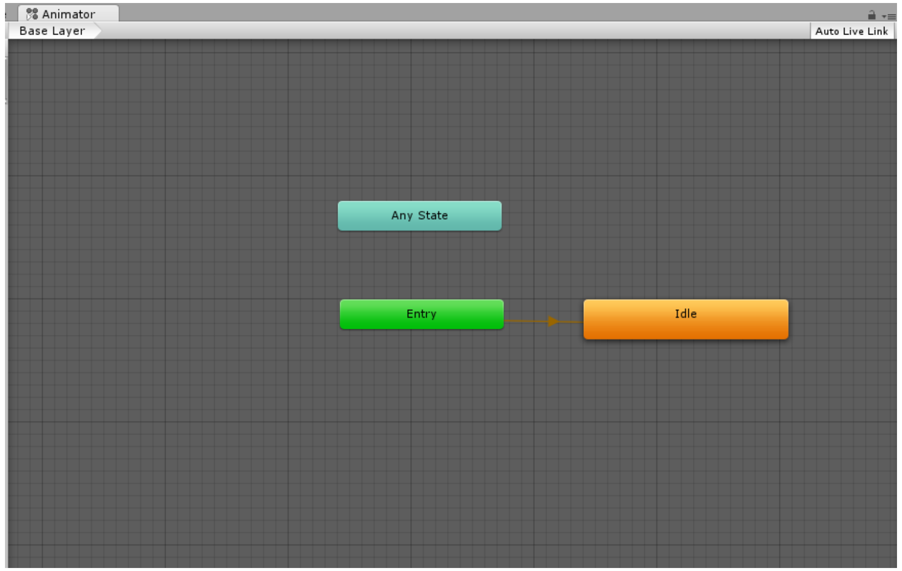
Strafe 0.0

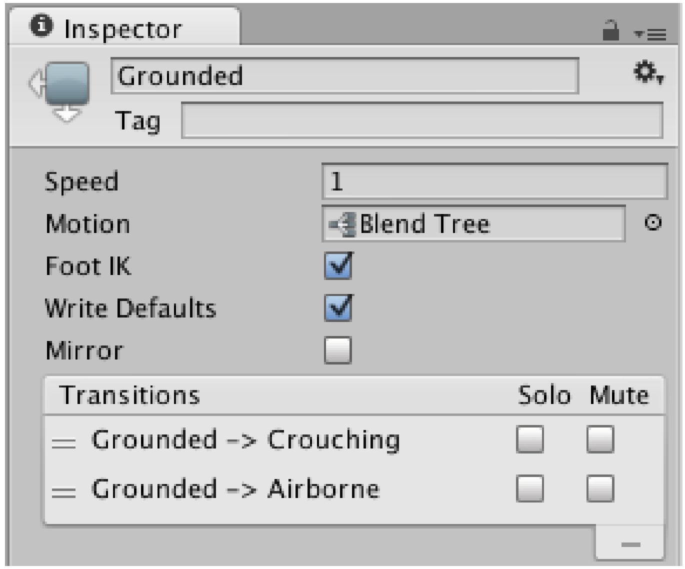
Fire □

Die ○

Animation States

- **Animation state**
  - An individual animation sequence (or blend tree) which will play while the character is in that state
- **Default state**
  - The state that the machine will be in when it is first activated
- **Any state**
  - Can be used to go to a specific state regardless of which state you are currently in
  - Cannot be the end point of a transition




Inspector

Grounded

Tag

Speed 1

Motion Blend Tree

Foot IK ☑

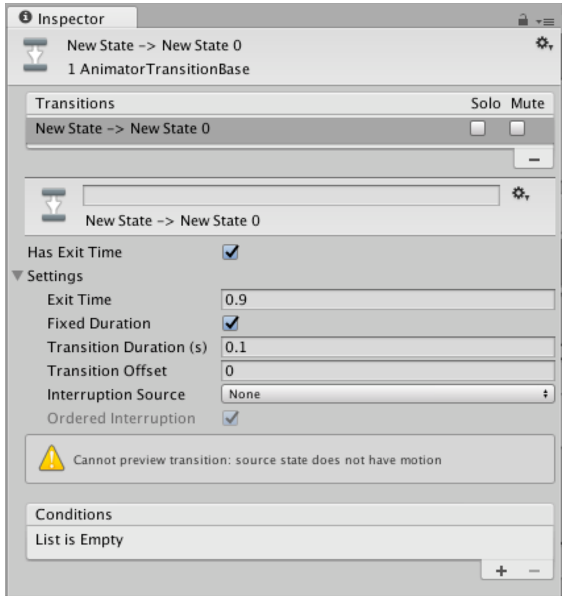
Write Defaults ☑

Mirror □

Transitions	Solo	Mute
Grounded -> Crouching	□	□
Grounded -> Airborne	□	□

Animation Transitions

- **Animation transition**
  - Switch or blend from one animation state to another
  - How to blend between states
  - Under what conditions they should activate (parameters)
- **Transition properties**
  - Exit time: the exact time at which the transition can take effect
  - Interruption source: control the circumstances under which this transition may be interrupted. Read this document for more details.
- **Transition graph**
  - Duration in/out
  - Transition offset



Inspector

New State -> New State 0
1 AnimatorTransitionBase

Transitions	Solo	Mute
New State -> New State 0	□	□

New State -> New State 0

Has Exit Time ☑

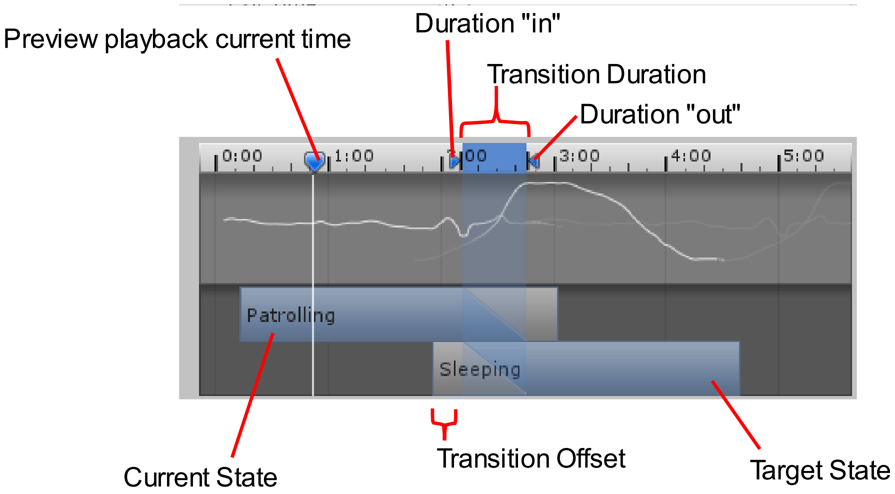
Settings

Exit Time	0.9
Fixed Duration	☑
Transition Duration (s)	0.1
Transition Offset	0
Interruption Source	None
Ordered Interruption	☑

Cannot preview transition: source state does not have motion

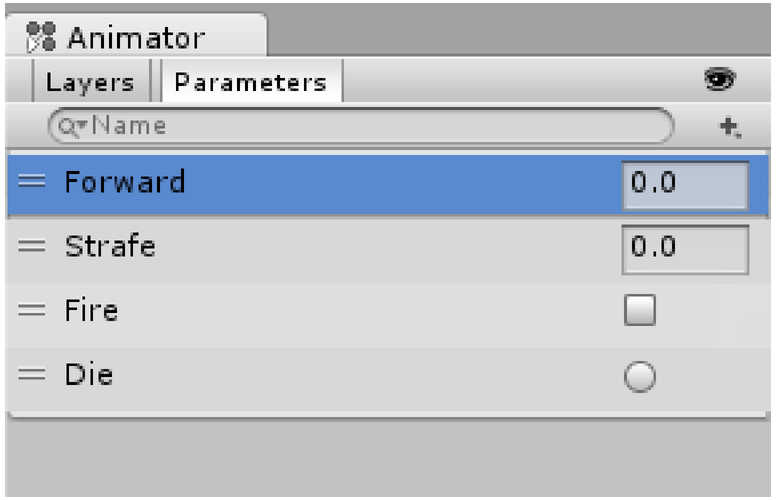
Conditions

List is Empty

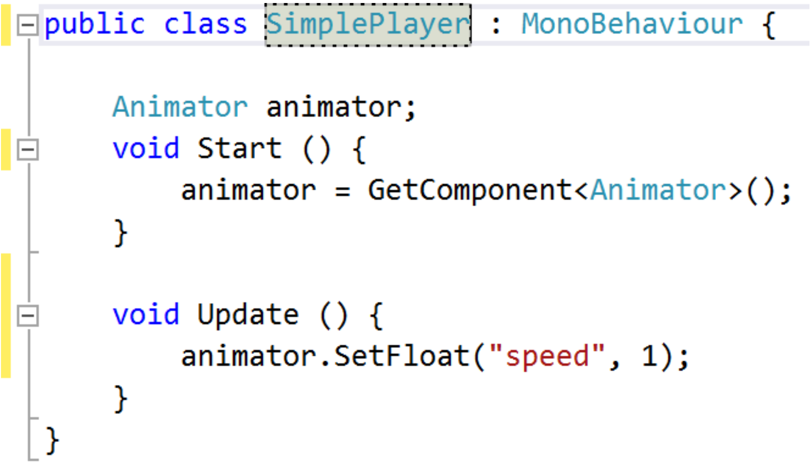


Animation Parameters

- **Animation Parameters**
  - Variables that can be accessed and assigned from scripts
  - Used to control or affect the flow of the state machine
- **Types:**
  - Int
  - Float
  - Bool
  - Trigger



Forward	0.0
Strafe	0.0
Fire	□
Die	○



```
public class SimplePlayer : MonoBehaviour {

    Animator animator;
    void Start () {
        animator = GetComponent<Animator>();
    }

    void Update () {
        animator.SetFloat("speed", 1);
    }
}
```

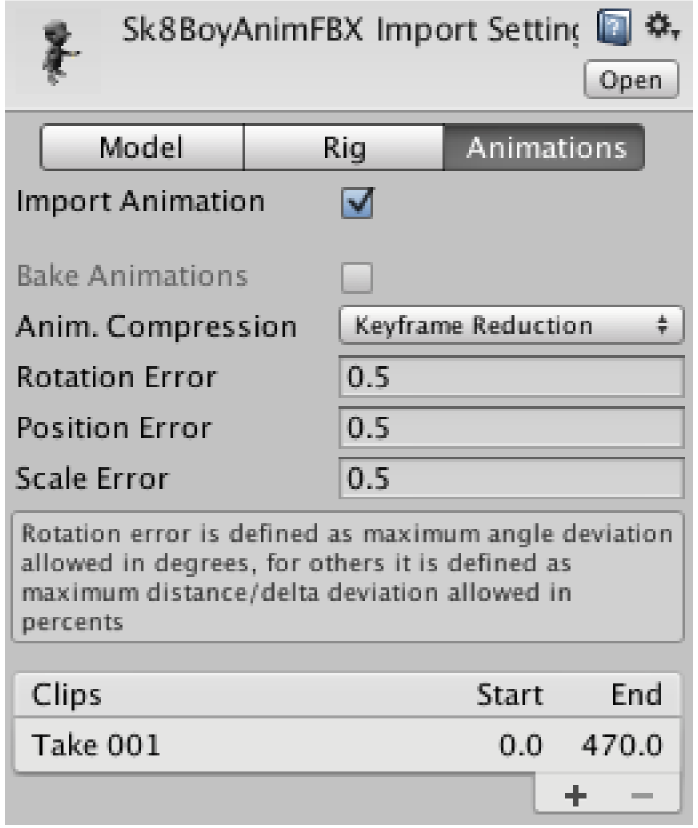
Demo

Review

- **Animation basics**
  - Animator
  - Animator controller
  - Animation state
  - Animation transition
  - Animation parameter

Splitting Animation Clips

- **Models with unsplit animations**
  - Walk animation 1-33
  - Run animation 41-57
  - Kick animation 81-97



Import Animation ☑

Bake Animations □

Anim. Compression: Keyframe Reduction

Rotation Error: 0.5

Position Error: 0.5

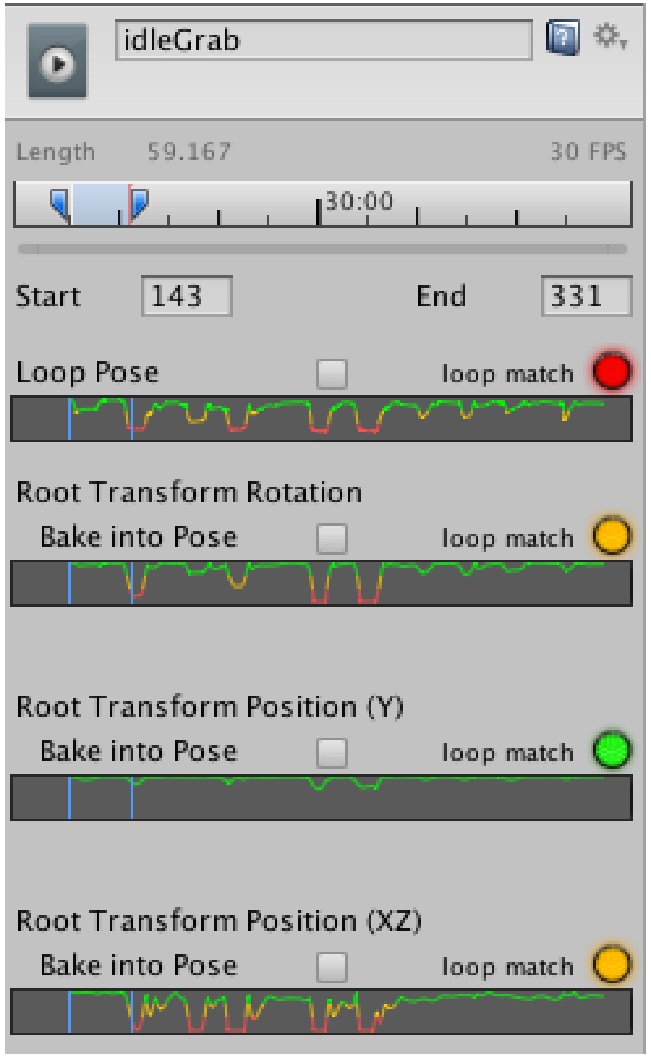
Scale Error: 0.5

Rotation error is defined as maximum angle deviation allowed in degrees, for others it is defined as maximum distance/delta deviation allowed in percents

Clips	Start	End
Take 001	0.0	470.0

Looping Animation Clips

- Loops can base on:
  - Pose
  - Rotation
  - Position



idleGrab

Length 59.167 30 FPS

Start 143 End 331

Loop Pose □ loop match ●

Root Transform Rotation

Bake into Pose □ loop match ●

Root Transform Position (Y)

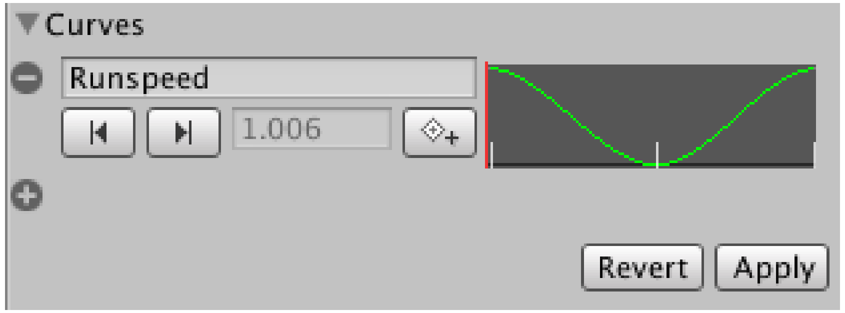
Bake into Pose □ loop match ●

Root Transform Position (XZ)

Bake into Pose □ loop match ●

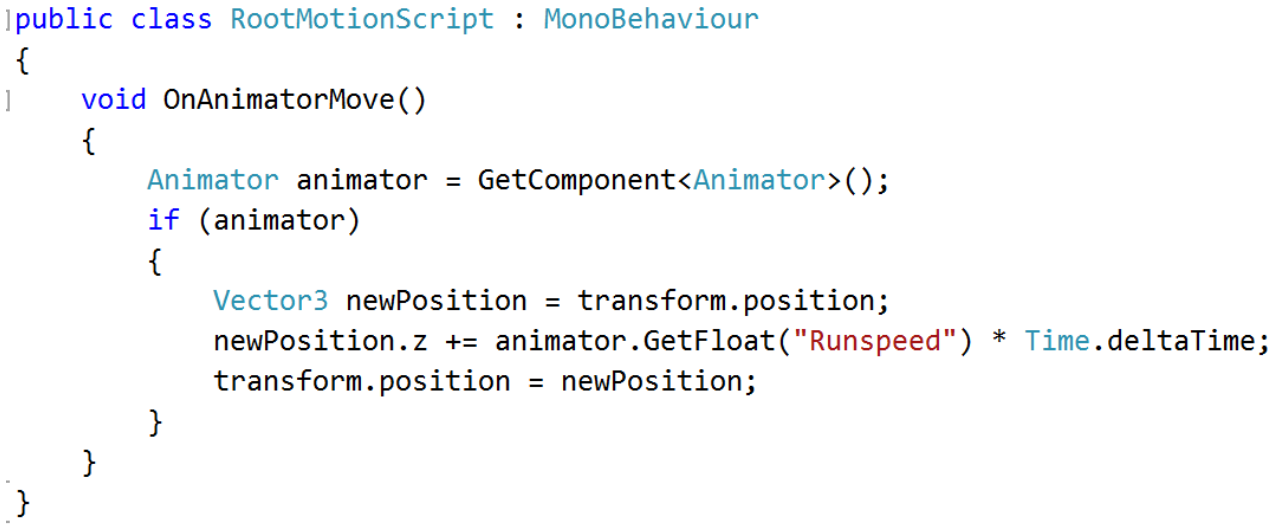
Root motion

- **Root motion**
  - Body transform
  - Root transform (XZ plane)
- **For animations comes as “in-place”**
  - Create a curve
  - Create a parameter
  - Control by script
  - “Handle by script”



Runspeed

1.006



```
public class RootMotionScript : MonoBehaviour
{
    void OnAnimatorMove()
    {
        Animator animator = GetComponent<Animator>();
        if (animator)
        {
            Vector3 newPosition = transform.position;
            newPosition.z += animator.GetFloat("Runspeed") * Time.deltaTime;
            transform.position = newPosition;
        }
    }
}
```

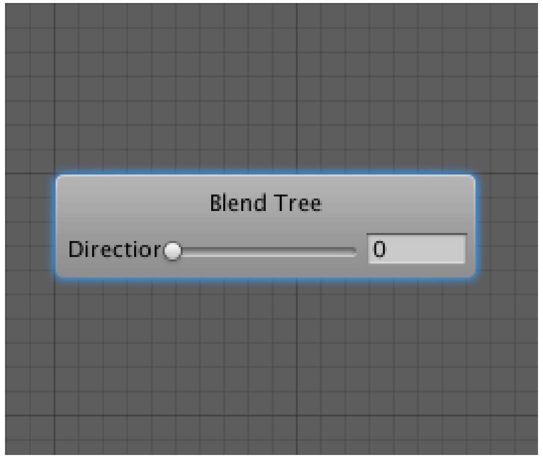
Demo

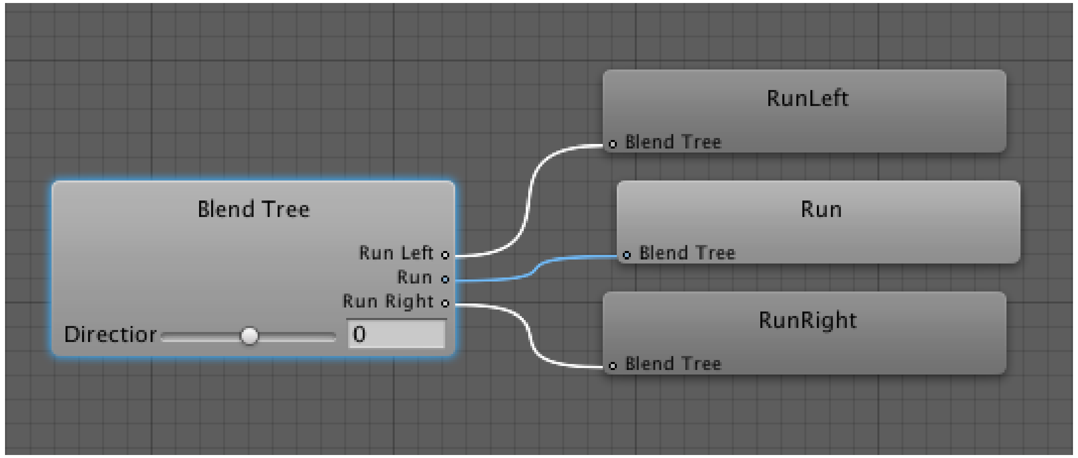
Review

- **Splitting animations**
- **Looping animations**
  - Pose
  - Rotation
  - Position
- **Root motion**
  - Create a curve
  - Control by script

Blend Trees

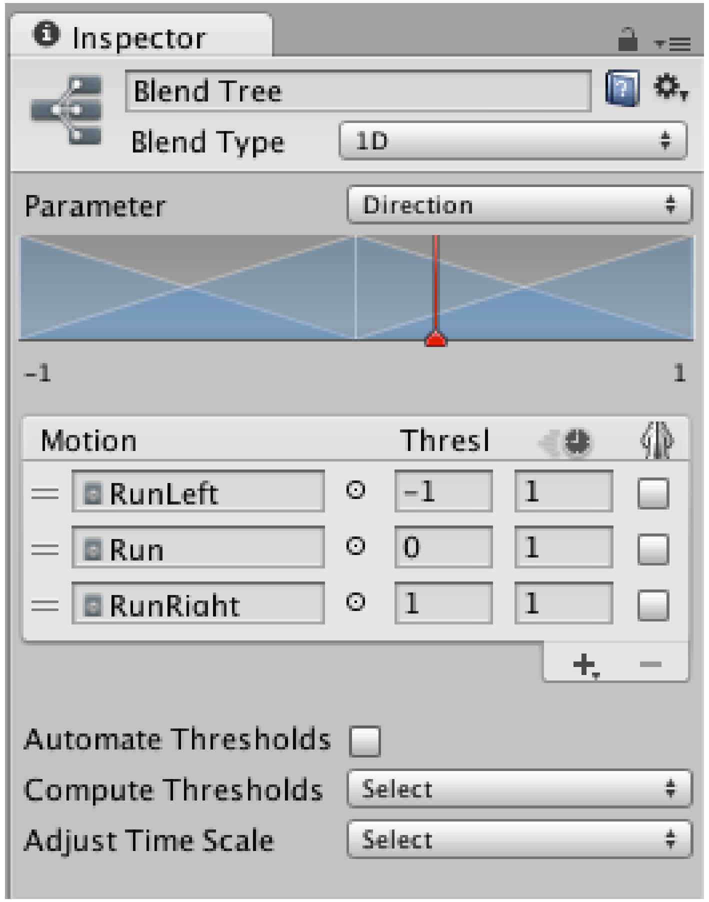
- **Blend trees**
  - Allow multiple animations to be blended smoothly
  - A special type of state of Animation State Machine
- **Transitions**
  - Transition from one animation state to another
  - Usually very quick
- **Using blend trees**
  - Create state > From New Blend Tree
  - Add animation clips using ‘+’ under motion





1D Blending

- **Blend types**
  - 1D
  - 2D
  - Direct blending
- **Blending parameter**
  - Animation parameter



Inspector

Blend Tree

Blend Type: 1D

Parameter: Direction

-1 1

Motion	Thresl		
RunLeft	-1	1	□
Run	0	1	□
RunRight	1	1	□

Automate Thresholds □

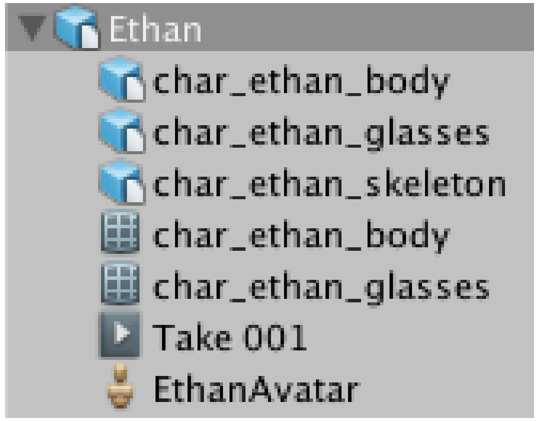
Compute Thresholds: Select

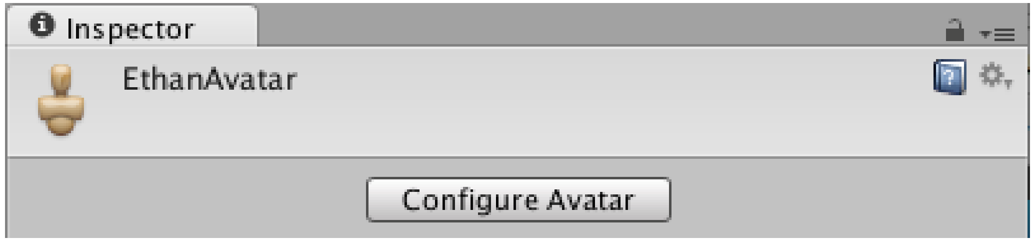
Adjust Time Scale: Select

Demo

Avatar

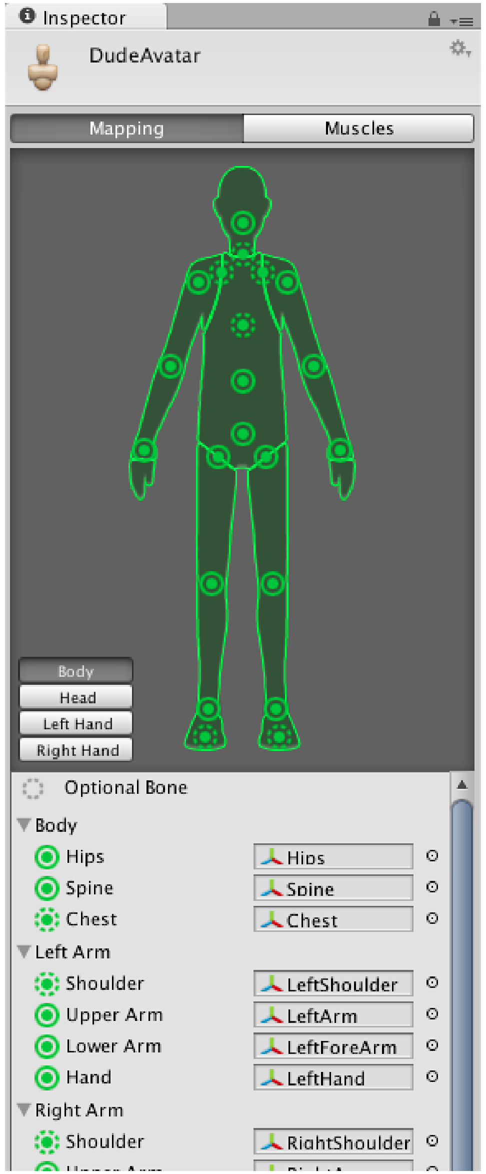
- **Avatar**
  - Mapping between simplified bone structure understood by Unity and the actual bones present in the skeleton
  - Allow for retargeting and inverse kinematics





Configuring the Avatar

- **Automatic avatar configuration**
  - Manual inspection is always recommended
  - Needs to have similar bone structure (rigging)
  - Needs to be T-pose (modeling)



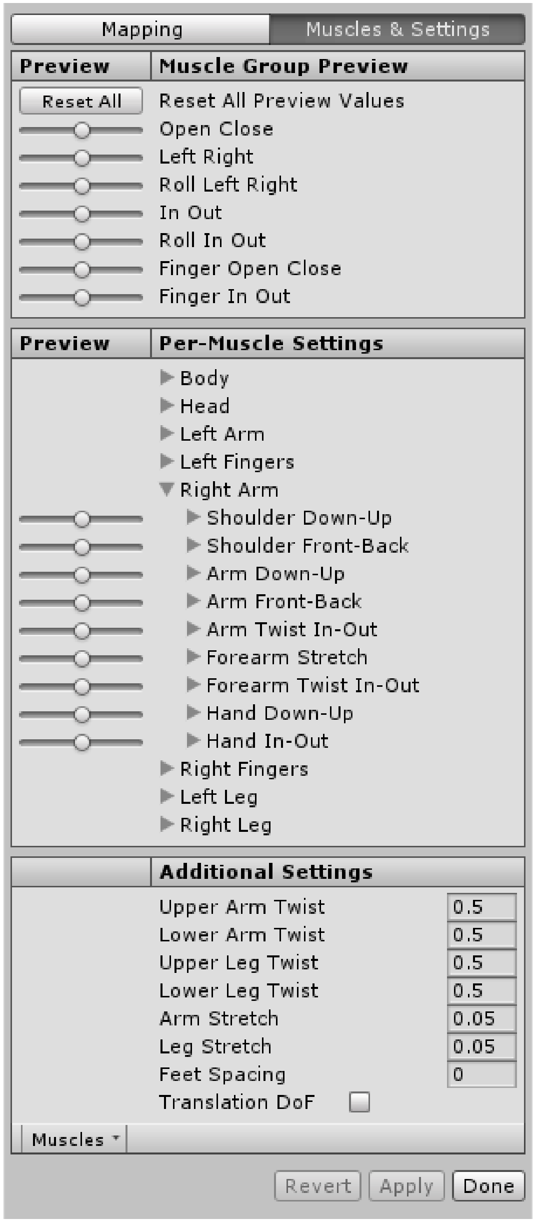
Body
- Hips: Hips
- Spine: Spine
- Chest: Chest

Left Arm
- Shoulder: LeftShoulder
- Upper Arm: LeftArm
- Lower Arm: LeftForeArm
- Hand: LeftHand

Right Arm
- Shoulder: RightShoulder

Muscle Setup

- **Muscle**
  - Control range of motion of different bones
  - Prevent visual artifacts and self-overlaps
- **Muscle group preview**
- **Per-Muscle Settings**



Preview | **Muscle Group Preview**

Reset All Preview Values
Open Close
Left Right
Roll Left Right
In Out
Roll In Out
Finger Open Close
Finger In Out

Preview | **Per-Muscle Settings**

- Body
- Head
- Left Arm
- Left Fingers
- Right Arm
  - Shoulder Down-Up
  - Shoulder Front-Back
  - Arm Down-Up
  - Arm Front-Back
  - Arm Twist In-Out
  - Forearm Stretch
  - Forearm Twist In-Out
  - Hand Down-Up
  - Hand In-Out
- Right Fingers
- Left Leg
- Right Leg

Additional Settings

Upper Arm Twist	0.5
Lower Arm Twist	0.5
Upper Leg Twist	0.5
Lower Leg Twist	0.5
Arm Stretch	0.05
Leg Stretch	0.05
Feet Spacing	0
Translation DoF	□

Demo

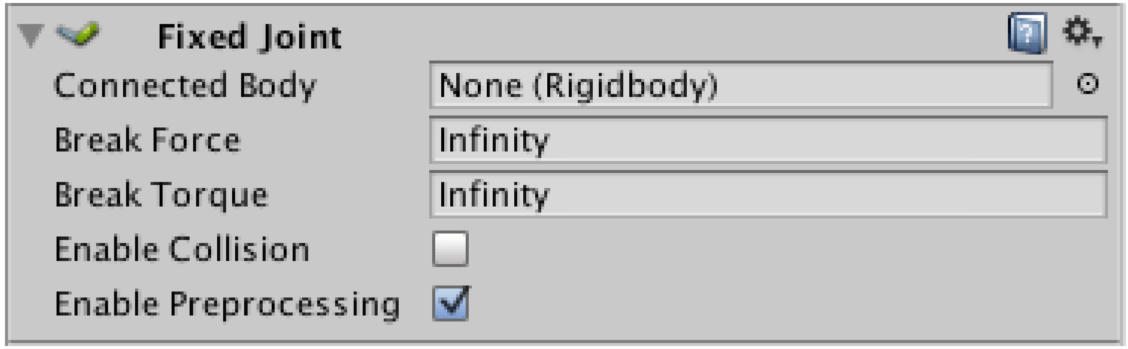
Review

- **Blend trees**
  - Blend trees vs transitions
  - Creating blend trees
  - Blending parameters
  - Blend types: 1D, 2D, Direct
- **Avatar**
  - Mapping, allow for retargeting and inverse kinematics
  - Configuring the avatar
  - Muscle: control range of motion

Articulated Rigid Bodies

Fixed Joint

- Restrict an object's movement to be dependent on another object
- Fixed joint vs parenting
  - Implemented through physics rather than transform hierarchy
  - Can break apart

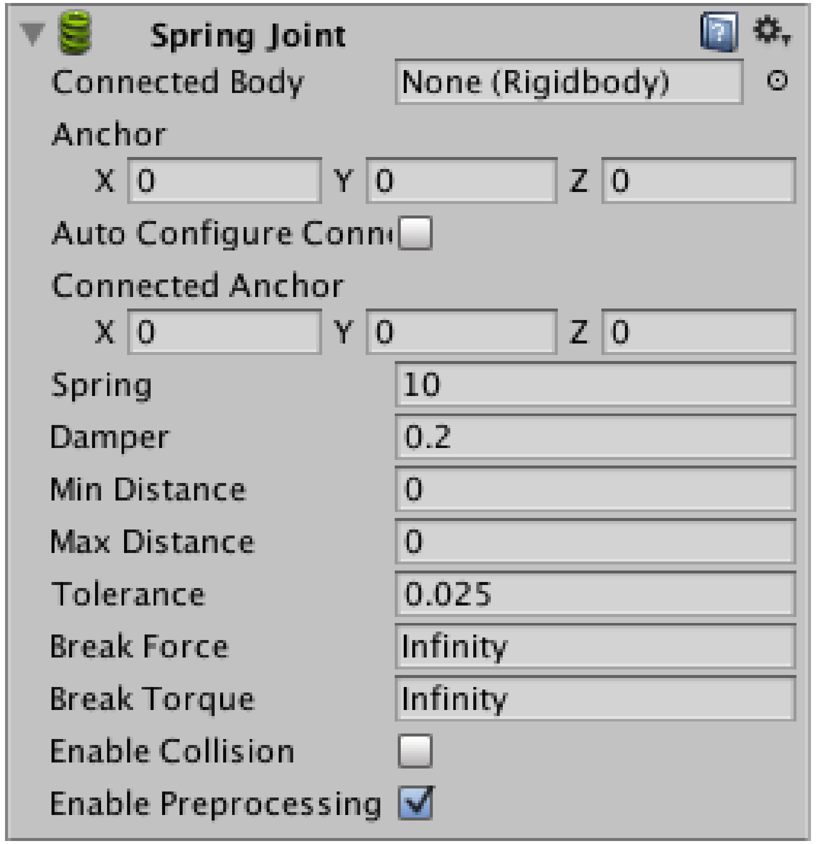


Fixed Joint

Connected Body	None (Rigidbody)
Break Force	Infinity
Break Torque	Infinity
Enable Collision	□
Enable Preprocessing	☑

Spring Joint

- Connect two rigid bodies through a spring
- Anchor
  - Point in object’s local space at which the joint is attached
- Connected anchor
  - Point in the connected object’s local space at which the joint is attached
- Auto configure connected
- Spring
- Damper



Spring Joint

Connected Body None (Rigidbody)

Anchor

X 0 Y 0 Z 0

Auto Configure Conn□

Connected Anchor

X 0 Y 0 Z 0

Spring 10

Damper 0.2

Min Distance 0

Max Distance 0

Tolerance 0.025

Break Force Infinity

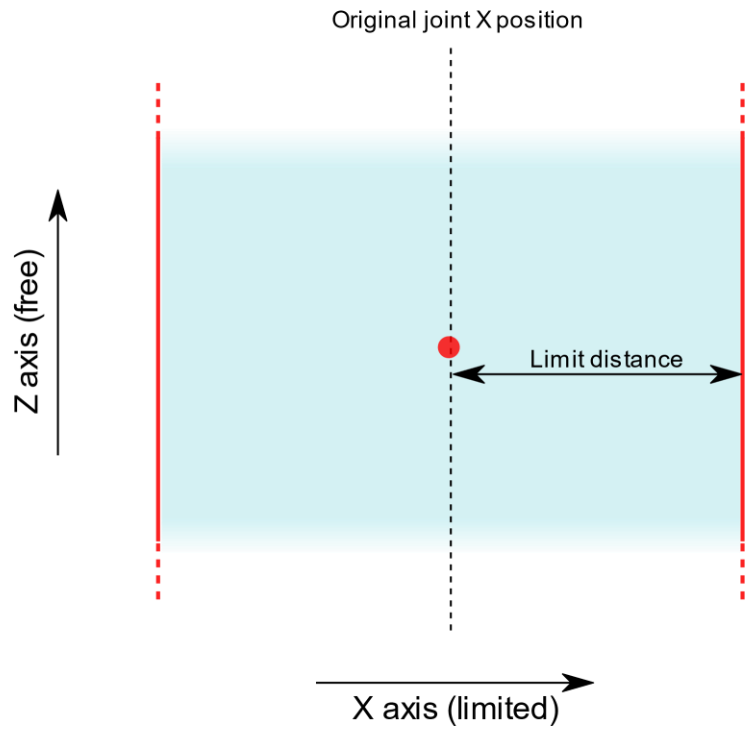
Break Torque Infinity

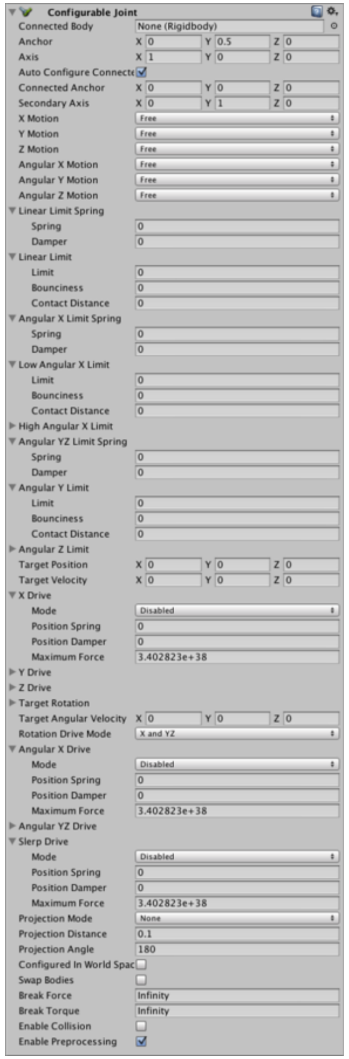
Enable Collision □

Enable Preprocessing ☑

Configurable joint

- Customizable joint, 4 sections
  - Position and rotation configuration
  - Limit and limit springs
  - Target and drive forces
  - Projection



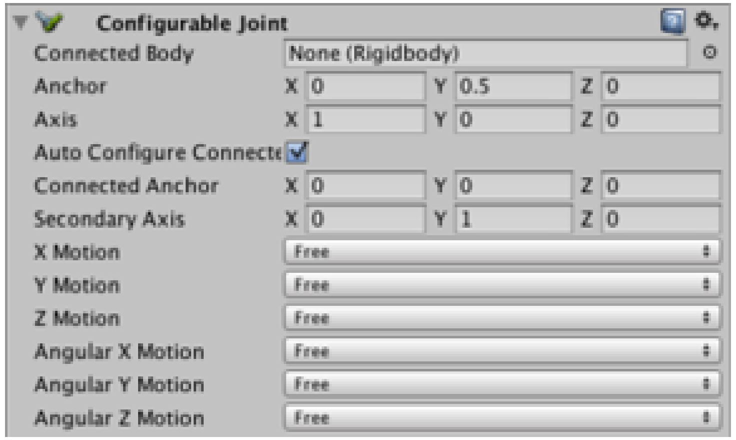


```
▼ Configurable Joint
  Connected Body                None (Rigidbody)
  Anchor                        X 0    Y 0.5  Z 0
  Axis                          X 1    Y 0    Z 0
  Auto Configure Connecte       ☑
  Connected Anchor              X 0    Y 0    Z 0
  Secondary Axis                X 0    Y 1    Z 0
  X Motion                      Free
  Y Motion                      Free
  Z Motion                      Free
  Angular X Motion              Free
  Angular Y Motion              Free
  Angular Z Motion              Free
▼ Linear Limit Spring
    Spring                      0
    Damper                      0
▼ Linear Limit
    Limit                       0
    Bounciness                  0
    Contact Distance            0
▼ Angular X Limit Spring
    Spring                      0
    Damper                      0
▼ Low Angular X Limit
    Limit                       0
    Bounciness                  0
    Contact Distance            0
▶ High Angular X Limit
▼ Angular YZ Limit Spring
    Spring                      0
    Damper                      0
▼ Angular Y Limit
    Limit                       0
    Bounciness                  0
    Contact Distance            0
▶ Angular Z Limit
  Target Position               X 0    Y 0    Z 0
  Target Velocity               X 0    Y 0    Z 0
▼ X Drive
    Mode                        Disabled
    Position Spring             0
    Position Damper             0
    Maximum Force               3.402823e+38
▶ Y Drive
▶ Z Drive
▶ Target Rotation
  Target Angular Velocity       X 0    Y 0    Z 0
  Rotation Drive Mode           X and YZ
▼ Angular X Drive
    Mode                        Disabled
    Position Spring             0
    Position Damper             0
    Maximum Force               3.402823e+38
▶ Angular YZ Drive
▼ Slerp Drive
    Mode                        Disabled
    Position Spring             0
    Position Damper             0
    Maximum Force               3.402823e+38
  Projection Mode               None
  Projection Distance           0.1
  Projection Angle              180
  Configured In World Spac      □
  Swap Bodies                   □
  Break Force                   Infinity
  Break Torque                  Infinity
  Enable Collision              □
  Enable Preprocessing          ☑
```

Stanford University

Configurable joint (1)

- Anchor
- Connected anchor
- To define local coordinate frame of the joint
  - Axis
  - Secondary axis
- X,Y,Z Motion
  - Free, locked, limited
- Angular X,Y,Z Motion
  - Free, locked, limited

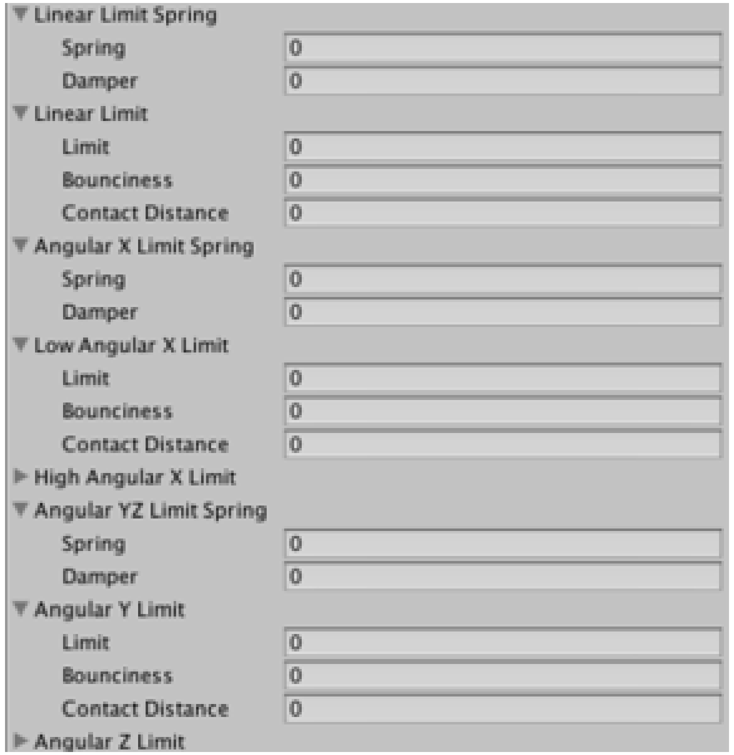


Configurable Joint

Connected Body	None (Rigidbody)					
Anchor	X	0	Y	0.5	Z	0
Axis	X	1	Y	0	Z	0
Auto Configure Connecte	☑					
Connected Anchor	X	0	Y	0	Z	0
Secondary Axis	X	0	Y	1	Z	0
X Motion	Free					
Y Motion	Free					
Z Motion	Free					
Angular X Motion	Free					
Angular Y Motion	Free					
Angular Z Motion	Free					

Configurable joint (2)

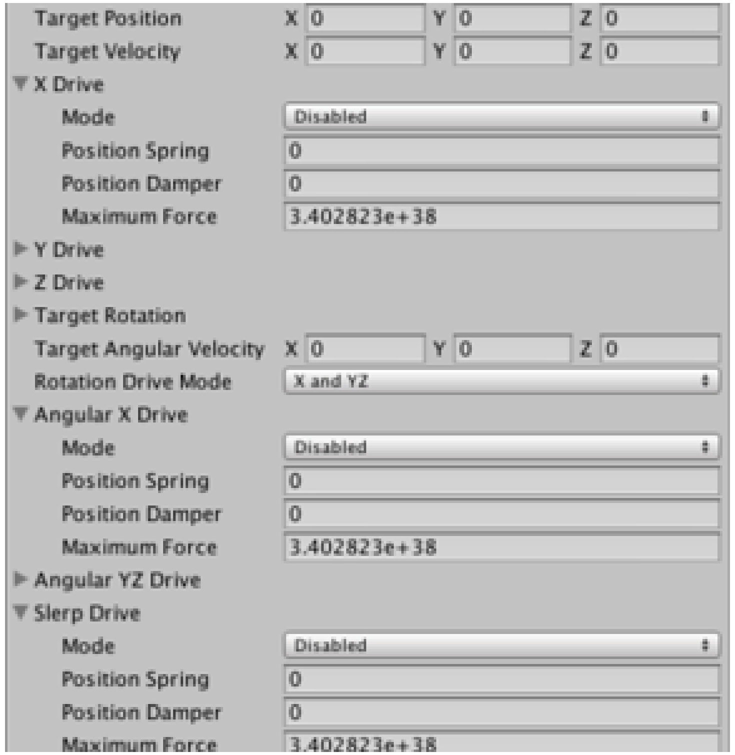
- Linear limit spring
  - Spring force applied to pull object back when it goes past the limit position
- Linear limit
  - Limit
    - Distance in world units
  - Bounciness
    - Bounce force applied to push is back when it reaches the limit distance
  - Contact distance
    - Tolerance
- Angular X
  - Limit spring, low limit, high limit
- Angular YZ
  - Limit spring, low limit, high limit



▼ Linear Limit Spring	
Spring	0
Damper	0
▼ Linear Limit	
Limit	0
Bounciness	0
Contact Distance	0
▼ Angular X Limit Spring	
Spring	0
Damper	0
▼ Low Angular X Limit	
Limit	0
Bounciness	0
Contact Distance	0
▶ High Angular X Limit	
▼ Angular YZ Limit Spring	
Spring	0
Damper	0
▼ Angular Y Limit	
Limit	0
Bounciness	0
Contact Distance	0
▶ Angular Z Limit	

Configurable joint (3)

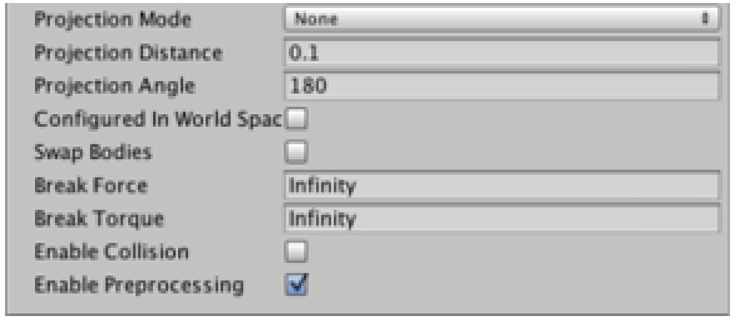
- Target position / velocity
  - Desired position / velocity
- X Drive
  - Drive force that moved toward target position/velocity along local X axis
  - Mode: disabled, position, velocity or both
  - Position spring, damper
  - Maximum force
- Y Drive, Z Drive
- Target rotation / angular velocity
- Angular X Drive
- Angular YZ Drive
- Slerp drive



Target Position	X 0 Y 0 Z 0
Target Velocity	X 0 Y 0 Z 0
▼ X Drive	
Mode	Disabled
Position Spring	0
Position Damper	0
Maximum Force	3.402823e+38
▶ Y Drive	
▶ Z Drive	
▶ Target Rotation	
Target Angular Velocity	X 0 Y 0 Z 0
Rotation Drive Mode	X and YZ
▼ Angular X Drive	
Mode	Disabled
Position Spring	0
Position Damper	0
Maximum Force	3.402823e+38
▶ Angular YZ Drive	
▼ Slerp Drive	
Mode	Disabled
Position Spring	0
Position Damper	0
Maximum Force	3.402823e+38

Configurable joint (4)

- Projection mode
  - (snap back when constraints unexpectedly violate)
  - None
  - Position and rotation
- Projection distance / angle
  - The distance/angle the joint must move beyond its constraints before the physics engine will attempt to snap it back to an acceptable position/rotation
- Configured in world space
- Swap bodies



Projection Mode	None
Projection Distance	0.1
Projection Angle	180
Configured In World Spac	□
Swap Bodies	□
Break Force	Infinity
Break Torque	Infinity
Enable Collision	□
Enable Preprocessing	☑

Apply forces and torques

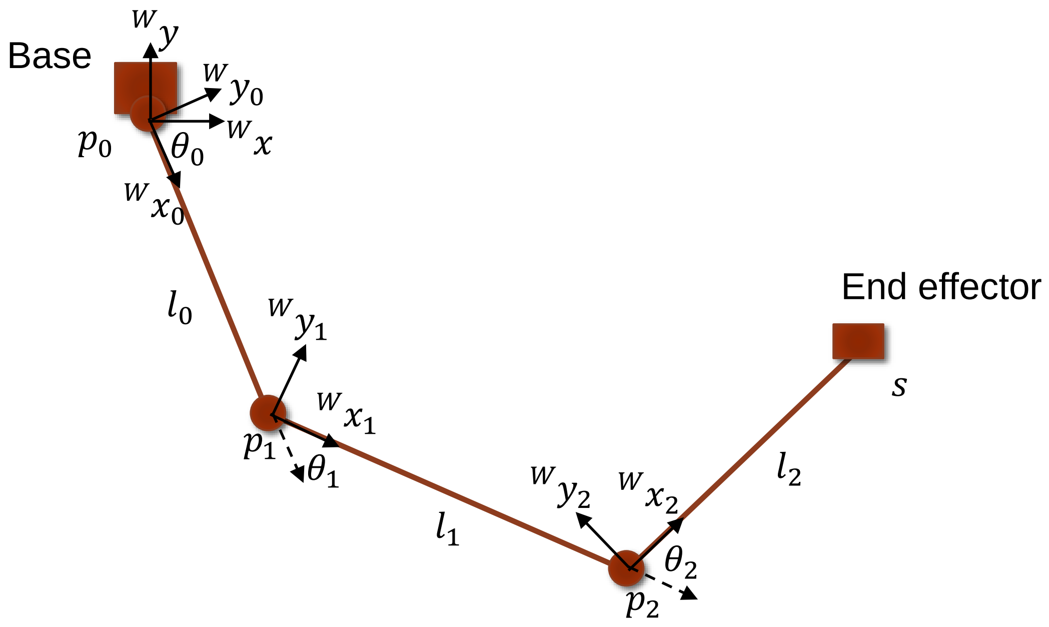
- Checkout Rigidbody class
  - public void **AddForce**(Vector3 force, ForceMode mode = ForceMode.Force)
  - public void **AddRelativeForce**(Vector3 force, ForceMode mode = ForceMode.Force)
  - public void **AddForceAtPosition**(Vector3 force, Vector3 position, ForceMode mode = ForceMode.Force)
  - public void **AddTorque**(Vector3 torque, ForceMode mode = ForceMode.Force)

Demo

Inverse Kinematics

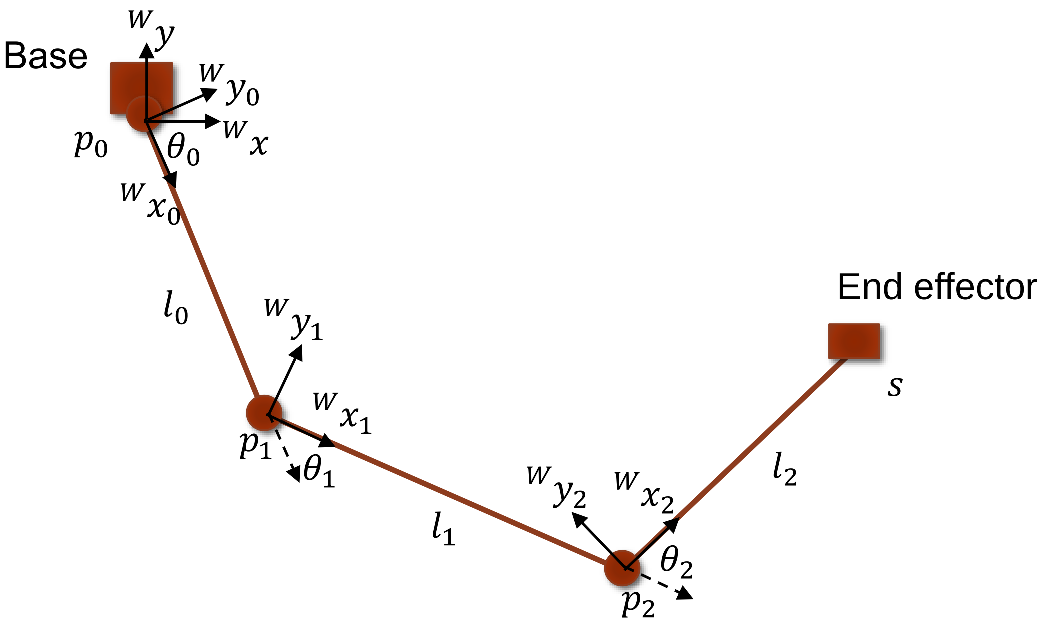
Inverse Kinematics (Review)

- Joints
  - Position: p_i
  - Angle: θ_i
- Lengths
  - l_i
- End effector
  - s
- Coordinate frames
  - $({}^{W}x, {}^{W}y, {}^{W}z), ({}^{i}x, {}^{i}y, {}^{i}z)$
  - Where are the z-axis? $(0,0,1)$
  - What is the coordinate of the end effector in frame 2? $(l_2, 0,0)$
  - What is the coordinate of p_i in frame i-1? $(l_{i-1}, 0,0)$



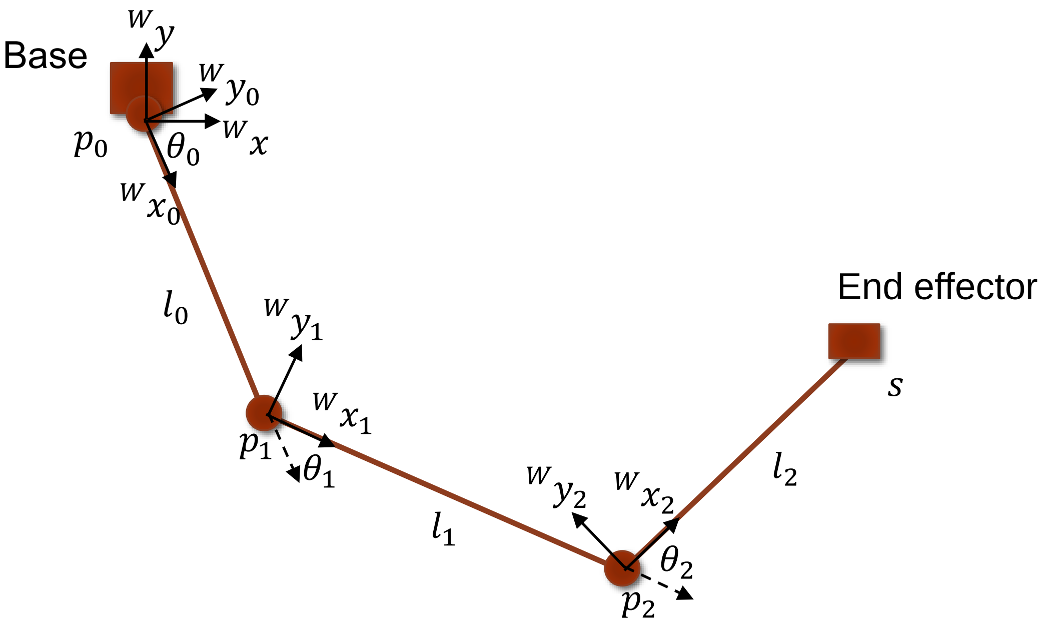
Inverse Kinematics (Review)

- Forward kinematics
  - Specify the base position/joint along with the other joint angles to prescribe motion
  - Given l_i, θ_i, find p_i, s
- Inverse kinematics
  - Given the values for the end effectors in world space, compute the joint angles
- Jacobian iterative method
  - $s = F(\boldsymbol{\theta})$
  - $\mathbf{J} = \frac{\partial s}{\partial \boldsymbol{\theta}}$
  - $s - s_{target} \approx \mathbf{J}\left(\boldsymbol{\theta} - \boldsymbol{\theta}_{target}\right)$ (Taylor expansion)
  - Given $s, s_{target}, \boldsymbol{\theta}$, find $\boldsymbol{\theta}_{target}$, iteratively
  - $s \in R^n, \theta \in R^m$, what is the dimension of $\mathbf{J}$? $n \times m$



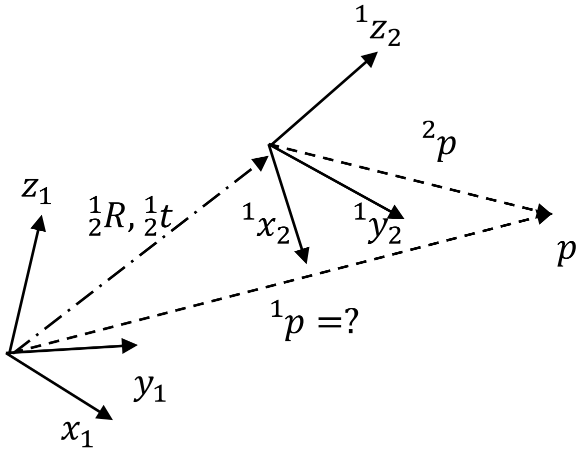
Inverse Kinematics (Review)

- $While\ |s - s_t| < thresh$
  - Compute $\mathbf{J}$
  - $\delta s = s_t - s$
  - Solve $\mathbf{J}\delta\boldsymbol{\theta} = \delta s$ to find $\delta\boldsymbol{\theta}$
  - Update with a small step α: $\boldsymbol{\theta} \mathrel{+}= \alpha\delta\boldsymbol{\theta}$
  - Update end effectors $s = F(\boldsymbol{\theta})$



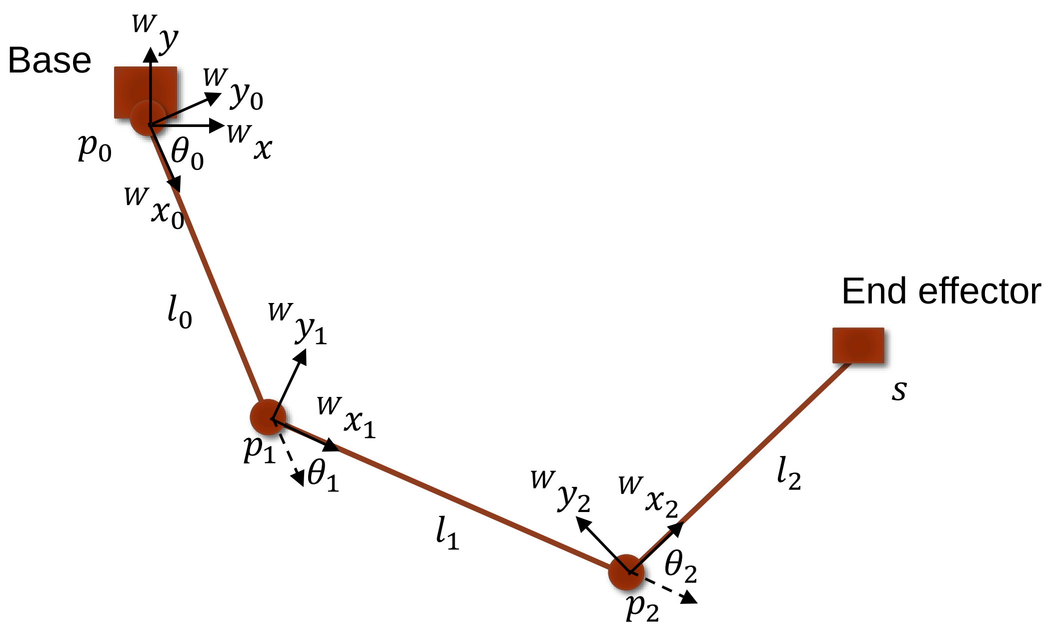
Coordinate Frames

- Coordinate transfer (from frame 2 to frame 1)
  - $^{1}p = {}^{1}_{2}R\ {}^{2}p + {}^{1}_{2}t$
    - ^{1}p is p in frame 1, ${}^{1}_{2}R$ is the matrix rotating coordinates from frame 2 to frame 1, ${}^{1}_{2}t$ is the translation vector from frame 1 to frame 2
- Homogenous coordinate and transformation matrix
  - $^{1}P = \begin{bmatrix} {}^{1}p \\ 1 \end{bmatrix}, {}^{1}_{2}T = \begin{bmatrix} {}^{1}_{2}R & {}^{1}_{2}t \\ 0 & 1 \end{bmatrix}$
  - $^{1}P = {}^{1}_{2}T\ {}^{2}P$
    - ^{1}P is homogenous representation of ^{1}p, ${}^{1}_{2}T$ is matrix transforming coordinates from frame 2 to frame 1
- Multiple coordinate frames:
  - $^{W}P = {}^{W}_{0}T\left({}^{0}_{1}T\left({}^{1}_{2}T\ {}^{2}P\right)\right) = {}^{W}_{\cancel{0}}T\,{}^{\cancel{0}}_{\cancel{1}}T\,{}^{\cancel{1}}_{\cancel{2}}T\ {}^{\cancel{2}}P = {}^{W}_{2}T\ {}^{2}P$ (commutativity)
- Origin of the ith coordinate frame in world space
  - $^{W}P_{i} = {}^{W}_{0}T\left({}^{0}_{1}T\left({}^{1}_{2}T\ {}^{2}P\right)\right) = {}^{W}_{i}T\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$ (last column of transformation matrix)
  - $p_{i} = \left[{}^{W}_{i}T_{14}, {}^{W}_{i}T_{24}, {}^{W}_{i}T_{24}\right]^{T}$



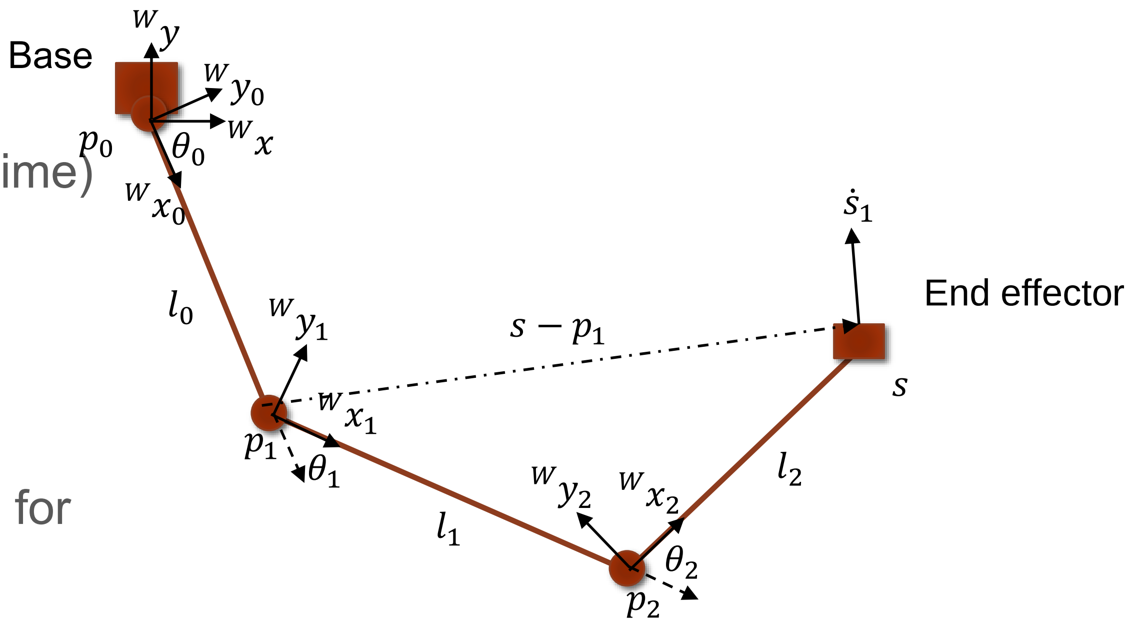
Forward Kinematics

- Calculate ${}^{W}_{0}T$
    - $${}^{W}_{0}T = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & 0 & 0 \\ \sin\theta_0 & \cos\theta_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
- Calculate ${}^{W}_{i}T$, for $i = 1, \ldots, m-1$
    - $${}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & l_{i-1} \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
    - $${}^{W}_{i}T = {}^{W}_{i-1}T\,{}^{i-1}_{i}T$$
- Calculate end effector in world frame
    - $$s = {}^{W}_{m-1}T \begin{bmatrix} l_{m-1} \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



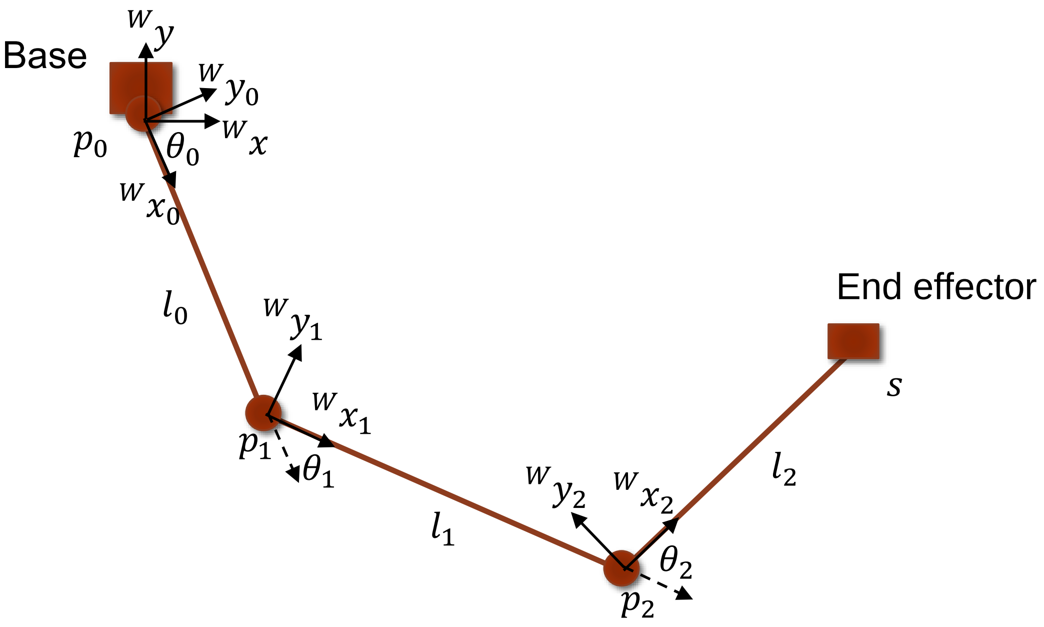
Jacobian Calculation

- $\mathbf{J} = \frac{\partial s}{\partial \boldsymbol{\theta}}$ difficult to evaluate
- $\dot{s} = \mathbf{J}\dot{\boldsymbol{\theta}} = [\mathbf{J_0} \quad \mathbf{J_1} \quad \mathbf{J_2}]\begin{bmatrix}\omega_0 \\ \omega_1 \\ \omega_2\end{bmatrix}$ (with respect to time)
- $\mathbf{J}_i$ is the ith row of Jacobian $\mathbf{J}$
- $\dot{s} = \sum_{i=0}^{m-1} \dot{s}_i = \sum_{i=0}^{m-1} \boldsymbol{\omega}_i \times (s - p_i)$
- $= \sum_{i=0}^{m-1} \big(v_i \times (s - p_i)\big)\omega_i$
- $\boldsymbol{\omega}_i$ is angular velocity, v_i is the rotation axis for joint i, ω_i is the magnitude: $\boldsymbol{\omega}_i = \omega_i v_i$
- The ith column of Jacobian
  - $\mathbf{J}_i = v_i \times (s - p_i)$



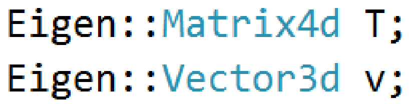
Jacobian Calculation

- Calculate ${}^{W}_{0}T$
  - ${}^{W}_{0}T = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & 0 & 0 \\ \sin\theta_0 & \cos\theta_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Calculate ${}^{W}_{i}T, p_i$ and $\mathbf{J}_i$, for $i = 1, \dots, m-1$
  - ${}^{i-1}_{\ \ i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & l_{i-1} \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  - ${}^{W}_{i}T = {}^{\ \ W}_{i-1}T\,{}^{i-1}_{\ \ i}T$
  - p_i is the first three entries in the last column of ${}^{W}_{i}T$
  - $\mathbf{J}_i = v_i \times (s - p_i)$



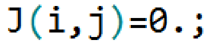
Eigen

- Matrix and Vector types



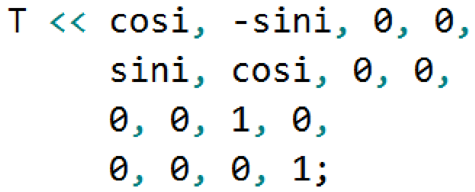
```
Eigen::Matrix4d T;
Eigen::Vector3d v;
```

- Matrix access and assignment



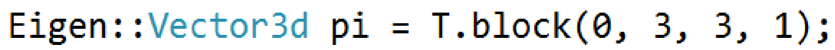
```
J(i,j)=0.;
```

- Initializing matrix



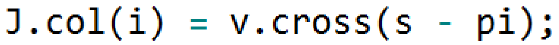
```
T << cosi, -sini, 0, 0,
     sini, cosi, 0, 0,
     0, 0, 1, 0,
     0, 0, 0, 1;
```

- Get block matrix: block(i,j,h,w)



```
Eigen::Vector3d pi = T.block(0, 3, 3, 1);
```

- Matrix column and cross product



```
J.col(i) = v.cross(s - pi);
```

Visual Studio Problems

- SAFESEH problem
  - Project Properties -> Linker -> Advanced -> Image Has Safe Exception Handlers, turn off
- Glut32.dll not found
  - Copy glut32.dll from lib to the directory that has .sln file

Review

- Basics on character animation
  - Prepare your model: modeling, rigging, skinning, (retargeting)
  - Obtain your model: Mixamo, unity assets store
  - Import models: use FBX
  - Animator, animator controllers, animation state machine, animation states, animation transitions, animation parameters
- Advanced materials on character animation
  - Splitting animation clips, looping animation clips, root motion
  - Blend trees, 1D blending, blending parameters
  - Avatar, avatar configuration, muscles
- Articulated rigid bodies
  - Fixed joint, spring joint
  - Configurable joint: limits and limit springs, targets and drive forces, projection
- Inverse kinematics
  - Forward kinematics, Jacobian calculation, Eigen