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
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
Importing Models

Rig

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

Animation

- Animation clips
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**
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
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
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
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
Looping Animation Clips

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

```csharp
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
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)
Muscle Setup

- **Muscle**
  - Control range of motion of different bones
  - Prevent visual artifacts and self-overlaps

- **Muscle group preview**

- **Per-Muscle Settings**
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
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
Configurable joint

- Customizable joint, 4 sections
  - Position and rotation configuration
  - Limit and limit springs
  - Target and drive forces
  - Projection
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 (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
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
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
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: \( \theta_i \)
- 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 $l_i, \theta_i$, find $p_i, s$
- **Inverse kinematics**
  - Given the values for the end effectors in world space, compute the joint angles
  - Jacobian iterative method
    - $s = F(\theta)$
    - $J = \frac{\partial s}{\partial \theta}$
    - $s - s_{target} \approx J(\theta - \theta_{target})$ (Taylor expansion)
    - Given $s, s_{target}, \theta$, find $\theta_{target}$, iteratively
    - $s \in R^n, \theta \in R^m$, what is the dimension of $J$? $n \times m$
Inverse Kinematics (Review)

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

Diagrams showing the kinematic chain with base and end effector coordinates.
Coordinate Frames

- Coordinate transfer (from frame 2 to frame 1)
  - \(1p = \frac{1}{2}R \cdot 2p + \frac{1}{2}t\)
  - \(1p\) is \(p\) in frame 1, \(\frac{1}{2}R\) is the matrix rotating coordinates from frame 2 to frame 1, \(\frac{1}{2}t\) is the translation vector from frame 1 to frame 2

- Homogenous coordinate and transformation matrix
  - \(1P = \begin{bmatrix} 1p \\ 1 \end{bmatrix}, \frac{1}{2}T = \begin{bmatrix} \frac{1}{2}R & \frac{1}{2}t \\ 0 & 1 \end{bmatrix}\)
  - \(1P = \frac{1}{2}T \cdot 2P\)
  - \(1P\) is homogenous representation of \(1p\), \(\frac{1}{2}T\) is matrix transforming coordinates from frame 2 to frame 1

- Multiple coordinate frames:
  - \(WP = W_0T \cdot W_1T \cdot \frac{1}{2}T \cdot 2P = W_2T \cdot 2P\) (commutativity)

- Origin of the \(i\)th coordinate frame in world space
  - \(WP_i = W_0T \cdot W_1T \cdot \frac{1}{2}T \cdot 2P = W_iT \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}\), (last column of transformation matrix)
  - \(p_i = [w_{iT14}, w_{iT24}, w_{iT24}]^T\)
Forward Kinematics

- Calculate $\mathbf{w}_0^T$
  \[ \mathbf{w}_0^T = \begin{bmatrix} \cos \theta_0 & -\sin \theta_0 & 0 & 0 \\ \sin \theta_0 & \cos \theta_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

- Calculate $\mathbf{w}_i^T$, for $i = 1, \ldots, m - 1$
  \[ \mathbf{w}_i^T = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & l_{i-1} \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]
  \[ \mathbf{w}_i^T = \mathbf{w}_i^{-1} \mathbf{w}_i^T \]

- Calculate end effector in world frame
  \[ \mathbf{s} = \mathbf{w}_{m-1}^T \begin{bmatrix} l_{m-1} \\ 0 \\ 0 \\ 1 \end{bmatrix} \]
Jacobian Calculation

• $\mathbf{J} = \frac{\partial \mathbf{s}}{\partial \mathbf{\theta}}$ difficult to evaluate

• $\dot{s} = \mathbf{J} \dot{\mathbf{\theta}} = [\mathbf{J}_0 \hspace{1em} \mathbf{J}_1 \hspace{1em} \mathbf{J}_2] \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix}$ (with respect to time)

• $\mathbf{J}_i$ is the ith row of Jacobian $\mathbf{J}$

• $\dot{s} = \sum_{i=0}^{m-1} \dot{s}_i = \sum_{i=0}^{m-1} \omega_i \times (s - p_i)$

• $= \sum_{i=0}^{m-1} (v_i \times (s - p_i)) \omega_i$

• $\omega_i$ is angular velocity, $v_i$ is the rotation axis for joint $i$, $\omega_i$ is the magnitude: $\omega_i = \omega_i v_i$

• 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 $J_i$, for $i = 1, ..., m - 1$

$$W_i^T = W_{i-1}^T W_i^T$$

$$p_i$$ is the first three entries in the last column of $W_i^T$

$$J_i = v_i \times (s - p_i)$$
Eigen

- Matrix and Vector types
  ```cpp
  Eigen::Matrix4d T;
  Eigen::Vector3d v;
  ```

- Matrix access and assignment
  ```cpp
  J(i,j)=0.;
  ```

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

- Matrix column and cross product
  ```cpp
  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