Animation

- Networking
  - Input Devices
    - keyboard
    - mouse
    - controllers
  - Event Dispatcher
  - Platform Events

- Game Engine
  - Streaming
    - files, data
    - score, state
  - core functions: startup, loop, shutdown
  - Physics
    - gravity
    - friction

- Graphics Engine
  - Render Sequence
  - 3D World
    - lights
    - visibility
    - textures
  - 2D Menu

- CPU Management
  - Memory Management
  - Artificial Intelligence
    - opponent Strategy
    - path finding
    - learning
  - Sound Audio Engine

- Animations
- Particle Effects
- Collision Detection
Animations

1. Ogre Animation
2. Frames and Keyframes
3. Bind Pose and skinning
4. Clip Continuity and Additive Blending
Skeleton
void skeleton(void){
    glPushMatrix(); <-----push current matrix state
    glTranslatef(jtpos[spine][0],jtpos[spine][1],jtpos[spine][2]); <----move base of animation
    glRotatef(jtangle[spine][0],jtangle[spine][1],
            jtangle[spine][2],jtangle[spine][3]);  <------rotate animation base
    if (draw) glCallList(spineL); <----draw base as in pose position, previous rotations and translation are applied.
    head_neck(); rarm();larm();legs(); <---draw limbs, all affected by base rotation translation
    glPopMatrix(); <-----reset previous matrix state
    if (!draw) { <-----simplistic  way of making figure stay on ground
        jtpos[spine][1]-=foot_height;  draw=1; if (footpos){
            printf("jtpos[spine][1]=%e,foot_height=%e\n", 
            jtpos[spine][1],foot_height);
            footpos=0;
        }
        skeleton();
    }
    else
        draw=0;
}

Note how this works with nodes.
Frame values:

30 1.0 0.0 0.0 /*spine0*/
0.0 0.0 0.0 1.0 /*head1*/
45 1 0 0 /*neck2*/
0 0 0 1 /*pelvis3*/
0 0 0 1 /*rcollar4*/
0 -1 0 0 /*rshoulder5*/
0 -1.0 0 0 /*relbow6*/
90 -1 0 0 /*rhand7*/
45 -1 0 0 /*rhip8*/
0 -1 0 0 /*rknee9*/
0 1 0 0 /*rfoot10*/
0 1 0 0 /*lcollar11*/
30 -1 0 0 /*lshoulder12*/
90 -1.0 0 0 /*lelbow13*/
0 1 0 0 /*lhand14*/
0 1 0 0 /*lhip15*/
0 1 0 0 /*lknee16*/
0 1 0 0 /*lfoot17*/
0 8 0 /*spinePos*/
Key Frames

- To avoid having to store position data set for each frame a number of special (key) frames are stored showing the positions at a limited number of time intervals. Special frames are called key frames.
- The actual frames are derived from the key frames via (spline) interpolation.
- The frame data is applied to the object in its local coordinate bind pose position.
Skinning

Hypothetically a skeleton connects the joints but in practice there is no reason to draw it as a mesh representing skin covers the skeleton. Each mesh element is attached to a position between two joints, but that position may be affected by the rotation of nearby joints. The artists determine how the position varies as the joints move.
Skinning

- Mesh position can be interpolated from key frame data. (using quaternions).
- Determined as a weighted average of the affect of each joint it is attached to.
- Since key frame time positions may not correspond to actual frame position, one has to calculate the interpolation based on an time measure rather than on just frame count. That requires computing the time since the last frame.
InterClip Continuity

- Game animations are produced by piecing together a large number of fine grained animation clips.

- Transition between clips must be done carefully to avoid popping. Suppose clip A is running and we want to start clip B.
  - Frozen transition: The last frame of A is interpolated with successive frames of B.
  - Smooth transition. B starts before A ends and frames are interpolated with more and more weight given to B's values.
InterClip Continuity

Figure II.27. A smooth transition, in which the local clocks of both clips keep running during the transition.

Figure II.28. A frozen transition, in which clip A’s local clock is stopped during the transition.
InterClip Continuity

- A linear weighting factor like

\[ \beta(t) = \frac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}} \]

could be used,

- But a cubic weighting factor like \( \beta(t)^3 \) is smoother.

Figure II.29. A smooth transition, with a cubic ease-in/ease-out curve applied to the blend factor.
InterClip Continuity

- More than two clips can be combined

- But this requires that different weighting factors be applied to different joints, as well as a careful selection of the averaging technique.

Figure 11.31. Targeted movement can be implemented by blending together looping locomotion clips that move in each of the four principal directions.
Additive Blending

Another approach to transition is to calculate a deformation matrix for each joint. That is given a source pose $S_j$ and reference pose $R_j$ define

$$D_j = S_j R_j^{-1}$$

For instance, if $R$ represents the normal posture of an individual and $S$ is a “tired posture”. Then $D_j$ can be used to convert a pose $T_j$ from normal to tired by forming $D_j T_j$.

Of course that can be interpolate too.
Additive Blending Examples

Figure 11.38. Two single-frame difference animations A and B can cause a target animation clip to assume two totally different stances. (Character from Naughty Dog’s *Uncharted: Drake’s Fortune*.)
Additive Blending Examples

Figure 11.39. Additive blends can be used to add variation to a repetitive idle animation. Images courtesy of Naughty Dog Inc.
Additive Blending Examples

Figure 11.39. Additive blends can be used to add variation to a repetitive idle animation. Images courtesy of Naughty Dog Inc.
Additive Blending Examples

Figure 11.40. Additive blending can be used to aim a weapon. Screenshots courtesy of Naughty Dog Inc.
Ogre Examples

- Example: Implementing Animations
- Example: Animation info
- OGRE Manual: Skeletal Animation
- OGRE Documentation: Animation
Exercise (Optional)

- Implement Easy, Intermediate, Difficult questions in Intermediate tutorial 1
Possible Projects

- Implement a Key frame animation system with transitions and additive blending.
Animations

1. Ogre Animations
2. Intermediate tutorial 1
3. SkeletalAnimation.h
Ogre Animations

**Keyframed**

- A collection of related tracks: time and track values (joint positions):
  - NumericAnimationTrack [NumericKeyFrame]
  - NodeAnimationTrack [TransformKeyFrame]
  - VertexAnimationTrack [VertexMophKeyFame, VertexPoseKeyFrame]
    - MorphKeyFame has a copy of the mesh
    - Cannot be blended
  - Keyframe.h
Animation Topics

- Skeleton Bounding Box
- MeshMagick
- Manual Resource Loading