Computer Animation and Visualisation

Coursework 1

Daniel Holden

s0822954@staffmail.ed.ac.uk
Skinning

- Given we have some movement or animation of a character skeleton.
- Now we have to deform some character's mesh based upon these movements.
- This process is called *skinning*.
- This is the task for Coursework 1.
To get the position of vertex $v_l$ in the global space $v_g$

- Compose the rotation and translations Matrices in the joint chain
- Multiply in front the position of $v$ in the local space $v_l$
Local Vertex Position

- Our Character's vertex positions are given in global space.
- To get the local vertex position we can make use of the skeleton rest pose.
- This is configuration in which the skeleton fits inside the mesh.

\[
\begin{align*}
\hat{T}_1 \hat{R}_1 \hat{T}_2 \hat{R}_2 \hat{T}_3 \hat{R}_3 v_l &= v_g \\
\hat{T}_1 \hat{R}_1 \hat{T}_2 \hat{R}_2 \hat{T}_3 \hat{R}_3 &= \hat{M} \\
v_l &= \hat{M}_r^{-1} v_g
\end{align*}
\]
Rigid Deformation

• Now we know how to get the vertex local coordinates.

• We can then change the rotation and translation matrices of the joints to move the vertices.

• But this only works if vertices are rigidly assigned to a single bone.

• We want skin to deform more smoothly.
Linear Blending

- Assume vertices are rigidly tied to many bones.
- Give weights for how much each vertex is associated with each of those bones.
- Do a weighted sum of all the rigid transformations for each bone $b$ associated.

\[ v_g = \sum_{b} w_b M_b \hat{M}_b^{-1} v_r \]
Suggested Workflow

- Calculate global transforms for each joint in rest pose.
- Render the skeleton.
- Apply an animation to the skeleton.
- Compute global transforms for each joint in animated pose.
- Compute inverse global transforms for joints in rest pose.
- Use equation in previous slide to compute linear blending and new vertex positions per frame.
C++

- Assignment Requires use of C++
- Difficult language to learn
- De-facto for graphics and animation
- Learning it is therefore considered part of the course materials.
- Providing code runs well enough to be assessed, quality shouldn't matter.
- If it runs at one frame per minute I can't mark it.
- Otherwise performance and memory leaks won't be penalized.
Compiling/Running Assignment

- Place all source files in “src”
- Place all header files in “include”
- All animations are in “resources”
- Run “make” to build executable
- Edit or Create any files as required
Demo
Code Explanation
Final Result Demo
Marking

• Fully skinned character doing animations without bugs or errors should get 70%

• Bonus Marks For:
  – Animation Keyframe Interpolation
  – Motion Blending
  – Dual Quaternion Skinning
  – Inverse Kinematics
  – User Interaction/Keyframing
Frequent Questions / Problems

- Make copy of mesh vertices. If you update their positions frame by frame errors will accumulate.
- Calculate joint matrices once per frame, not per vertex.
- Don't forget to transform normals too.
- Don't translate normals when you transform them.
- It should run on DICE or I can't mark it.
- Some DICE machines occasionally act weird with OpenGL. Try a different one.
Frequent Questions / Problems

• The skeleton structure is stored by making each joint specifying the id of its parent joint.

• Vertex weights are stored as a Vector3 of joint ids (you can safely cast from float to int) and a Vector3 of weights.

• Weights might not sum exactly to one – you can renormalize them if you like but it isn't strictly required.
Frequent Questions / Problems

• You can use your computer but don't e-mail me asking for help with setup or installing anything.

• You can use custom setups, libraries, OpenGL versions, or C++ versions as long as it is easy for me to get working.
Questions?

Daniel Holden
s0822954@staffmail.ed.ac.uk