Advanced Animation Programming

GPR-450

Daniel S. Buckstein

Hierarchies & Skeletal Animation: Data Formats

Weeks 5 – 7

License

Skeletal Animation

- *Morph target animation* is nice, but very computationally expensive: lots of data!
- Easier if we define a *few spatial nodes* and animate those instead of *every single vertex*
- These nodes are the “skeleton” and we animate them using kinematics!
- Can then give the illusion of life to our character by *skinning* the mesh (…later)

Skeletal Animation

- We are concerned with FK in the context of *character animation systems*
- Not only does each joint/node have its own local transformation…
- …but the joint also has ***key poses*** that describe the local transformation over time!

Skeletal Animation

- ***Joint/node pose data structure***:
- Note: all described ***locally*** for each joint!
- Key Pose:
  - Raw Euler angles (X, Y, Z)

    AND/OR
  - Quaternion

    Why do we have this choice???
  - Translation

    (will never change for revolute joints)
  - Scale

Skeletal Animation

- Joint/node pose data structure:

```
struct HierarchyNodePose
{
    vec4 quat_or_euler;
    vec3 translate;
    vec3 scale;  // ...maybe...
};
```

Skeletal Animation

- Add a set of *key poses* to the hierarchy:
  - Still want to decouple as many things as possible

```
struct HierarchyPoseSet
{
    const Hierarchy *hierarchy;
    // multi-dimensional array of poses:
    //    a set of keys per node
    HierarchyNodePose **keyPoseList;
    unsigned int keyPoseCount;
};
```

Skeletal Animation

- Add a single set of poses to hierarchy state:

```
struct HierarchyState
{
    const Hierarchy *hierarchy;
    mat4 *localTransformList;
    mat4 *worldTransformList;
    // current pose per node
    HierarchyNodePose *localPoseList;
};
```

Skeletal Animation

- Okay… but where do we get the data?
- Just as the Wavefront OBJ file format provides us with a nice “polygon soup” of mesh data…
- …there are also some useful file formats for skeleton/hierarchy information
- “***Motion Capture File Formats Explained***” (paper posted on Canvas)

http://www.dcs.shef.ac.uk/intranet/research/public/resmes/CS0111.pdf

Skeletal Animation

- BVH: "***B***io***v***ision ***H***ierarchical data"
- Created by Biovision, a motion capture services company
- ASCII format that specifies node relationships, initial transforms, and key-pose transforms
- Key pose values sorted by "channel" (e.g. rotation X, Y, Z, translation X, Y, Z…)

Skeletal Animation

- BVH: "***B***io***v***ision ***H***ierarchical data"
- Hierarchy:

Local offset from parent's location

List of joint attributes with animation data

Start of child joint

"Leaf" node

```
HIERARCHY
ROOT Hips
{
	OFFSET	20	0.00	0.00
	CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
	JOINT LeftHip
	{
		OFFSET		3.430000	0.000000	0.000000
		CHANNELS 3 Zrotation Xrotation Yrotation
		JOINT LeftKnee
		{
			OFFSET		0.000000	-18.469999	0.000000
			CHANNELS 3 Zrotation Xrotation Yrotation
			JOINT LeftAnkle
			{
				OFFSET	0.000000	-17.950001	0.000000
				CHANNELS 3 Zrotation Xrotation Yrotation
				End Site
				{
					OFFSET	0.000000	-3.119996
				}
			}
		}
	}
```

Skeletal Animation

- BVH: "***B***io***v***ision ***H***ierarchical data"
- Animations:

SAMPLE #
(keyframe)

```
MOTION
Frames:      222
Frame Time:  0.033333
```

0.00	39.88	-0.01	-1.79	-18.43	-1.74	5.02	-0.34	0.03
0.11	39.87	-0.01	-2.31	-17.29	-3.05	3.19	-4.22	0.24
0.22	39.88	0.02	-2.83	-16.47	-4.56	1.98	-6.94	0.24
0.32	39.92	0.19	-3.51	-17.17	-7.03	1.79	-6.29	0.20

......

CHANNEL #
(attribute)

Skeletal Animation

- BVH: "***B***io***v***ision ***H***ierarchical data"
- Problems with BVH format:
- Just like OBJ is considered "*polygon soup*", BVH could be considered "motion soup"
- Motion section is not organized by object, nor is it consistent; hard to track
- E.g. one joint may only have 1 channel but others may have 6

Skeletal Animation

- BVH: "***B***io***v***ision ***H***ierarchical data"
- Problems with BVH format:
- Furthermore, order of operations is not specified anywhere…
- Rotations: XYZ or ZYX?
  - They are not the same!!!
- A new format was developed to make up for BVH's shortcomings…

Skeletal Animation

- HTR: "***H***ierarchical ***T***ranslation & ***R***otation"
- Created for Motion Analysis software suite
- ASCII format that specifies hierarchy, motion, *and* mathematical operation information
- Organized per-joint (no more soup!)
- The most detail I've ever found on this file format has been from the paper on Canvas

Skeletal Animation

- HTR: "***H***ierarchical ***T***ranslation & ***R***otation"
- Header: describes interpretation info

Attribute names

Attribute values

```
[Header]
# KeyWord<space>Value<CR>
FileType htr
DataType HTRS
FileVersion 1
NumSegments 10
NumFrames 91
DataFrameRate 30
EulerRotationOrder ZYX
CalibrationUnits mm
RotationUnits Degrees
GlobalAxisofGravity Y
BoneLengthAxis Y
ScaleFactor 1.00
```

Skeletal Animation

- HTR: "***H***ierarchical ***T***ranslation & ***R***otation"
- Segment names & hierarchy:

```
[SegmentNames&Hierarchy]
# ObjectName<tab>ParentObjectName<CR>
root            GLOBAL
hips            root
r_hip           hips
r_knee1         r_hip
r_ankle1        r_knee1
r_toe1          r_ankle1
l_hip           hips
l_knee1         l_hip
l_ankle1        l_knee1
l_toe1          l_ankle1
```

Joint name

Parent joint name

Skeletal Animation

- HTR: "***H***ierarchical ***T***ranslation & ***R***otation"
- Base pose:

```
[BasePosition]
# ObjectName<tab>Tx<tab>Ty<tab>Tz<tab>Rx<tab>Ry<tab>Rz<tab>BoneLeng
root            0.000000        80.000000       0.000003        -90.
hips            0.000000        0.000000        0.000000        0.00
r_hip           0.000000        20.000000       0.000000        -75.
r_knee1         0.000000        41.231053       -0.000001       -151
r_ankle1        0.000000        41.231050       0.000000        -75.
r_toe1          0.000000        9.999999        0.000000        0.00
l_hip           0.000000        -20.000000      0.000000        104.
l_knee1         -0.000002       -41.231062      0.000001        -28.
l_ankle1        0.000000        -41.231046      0.000000        104.
l_toe1          0.000000        -9.999998       0.000000        0.00
```

Skeletal Animation

- HTR: "***H***ierarchical ***T***ranslation & ***R***otation"
- Motion data (per-joint): describes ***change from base pose*** for each joint independently!

```
[root]
# Tx<tab>Ty<tab>Tz<tab>Rx<tab>Ry<tab>Rz<tab>BoneScaleFactor<CR>
0       0.000000        0.000000        0.000000        0.000000
1       0.000000        0.000000        0.000000        0.000000
2       0.207726        -0.072885       -0.285990       0.000000
3       0.743440        -0.276966       -1.046562       0.000000
4       1.475947        -0.590377       -2.135620       0.000000
5       2.274052        -0.991254       -3.407070       0.000000
6       3.006560        -1.457725       -4.714815       0.000000
7       3.542274        -1.967931       -5.912759       0.000000
8       3.750000        -2.500000       -6.854808       0.000000
9       3.654445        -3.032069       -8.268361       0.000000
10      3.385555        -3.542275       -10.433482      0.000000
11      2.970000        -4.008746       -12.533063      0.000000
```

Skeletal Animation

- For any format, once you have parsed through all the data, need to compute *base local transformation*
- Given a base pose and a set of key poses, we need to compute the local transform for every key pose…
- …method differs from format to format…

Skeletal Animation

- How do we update a joint's *global* transform through forward kinematics at any time?
- How do we update a joint's *local* transform?
- ***2 steps***:

1) Calculate pose
2) Convert to 4x4 matrix

Skeletal Animation

- Updating local transformation state:

1) Calculate pose state:
   - Copy key pose values directly to state

     OR

   - Animate variables with respect to time
     - Update Euler angles using desired interpolation method
     - …OR use quaternion SLERP
     - Interpolate translation *if you know the translation is changing* (i.e. if not a purely revolute joint)

Skeletal Animation

- Updating local transformation state:

2) Convert to 4x4 homogeneous transformation
   - Convert multiple Euler angle rotations to single rotation matrix by concatenation
   - …OR convert Euler angles to rotation quaternion
   - …OR convert pre-computed quaternion to matrix
   - Store in 4x4 format (big T)

Skeletal Animation

- Have we seen this process before? ;)
- Yes, but let's simplify it:

$$^{\text{parent}}T_{n_t} = \begin{bmatrix} R_t & \vec{p}_t \\ 0 & 1 \end{bmatrix}$$

Local transform of node *n* at time *t*

$$R_t = \text{convert}(\hat{q}_k)$$

$$\vec{p}_t = \vec{p}_k$$

Transformation channels for key pose *k*

Skeletal Animation

- Example update algorithm (given the structs):

```
struct HierarchyPoseSet
{
    const Hierarchy *hierarchy;
    HierarchyNodePose **keyPoseList;
    unsigned int keyPoseCount;
};



struct HierarchyState
{
    const Hierarchy *hierarchy;
    mat4 *localTransformList;
    mat4 *worldTransformList;
    HierarchyNodePose *localPoseList;
};
```

```
////////////////////////////////////////////////
// copy set of key poses to state
// n is the node index and k is the pose index
for (n = 0; n < state->hierarchy->nodeCount; ++n)
  state->localPoseList[n]
    = poseSet->keyPoseList[n][k];   // *see below

////////////////////////////////////////////////
// update local matrices
for (n = 0; n < state->hierarchy->nodeCount; ++n)
  state->localTransformList[n] =
    convertPoseToMatrix(state->localPoseList[n]);

////////////////////////////////////////////////
// proceed to do kinematics...
```

Skeletal Animation

- We should now be comfortable with the concept of a *skeleton* (hierarchy of *joints*)
- Key poses for skeletal animation describe the set of *key angles*: the *known joint angles* at some point in time!
- For now we've just discussed how to apply FK at any *key pose*
- We'll worry about the in-betweens soon

(animation blending)

The end.

- Questions? Comments? Concerns?