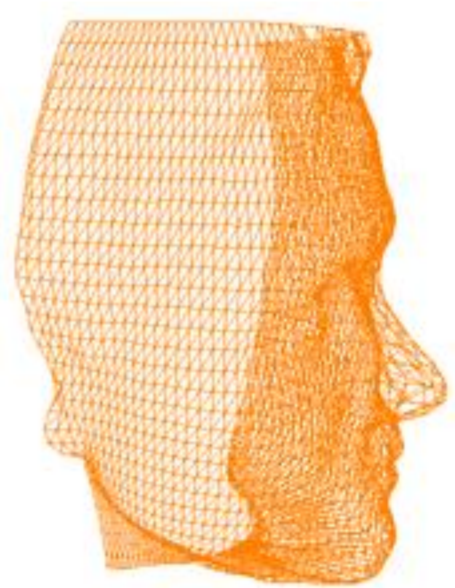
COSC422 Advanced Computer Graphics



8 Motion Data and Skeletal Animation

Semester 2
2019

R. Mukundan (mukundan@canterbury.ac.nz)
Department of Computer Science and Software Engineering
University of Canterbury, New Zealand.

Lecture Outline

- Motion capture data
  - Mocap data formats
  - Joint Hierarchy
  - BVH Representations and Keyframe Animation
  - Animating skeletons using BVH data
- Implementations using Assimp

Motion Capture (mocap)

- Motion capture data can be used to animate a character model based on realistic human movements
- An actor performs the movements and the joint positions and angles are computed from various sensors
  - Body sensors: Magnetic motion capture, accelerometers, gyroscopes
  - Optical sensors: Reflective markers, LEDs
  - Hybrid sensing, sensor fusion

Wikipedia.org

- The joint positions are then mapped to a character mesh via a skeletal structure, animated using joint angles.

Motion Capture Data Formats

- Acclaim : (ASF, AMC)
- **Biovision: BVH** (Biovision Hierarchy)
- 3D Biomechanics (c3d.org): C3D
- 3D Max Character Studio: BIP (Biped)
- Autodesk: **FBX** (Film box)

:

ASF, AMC, BVH are editable files in ASCII format.

ASF: Acclaim Skeleton File

AMC: Acclaim Motion Capture data

Joints

Motion data specifies

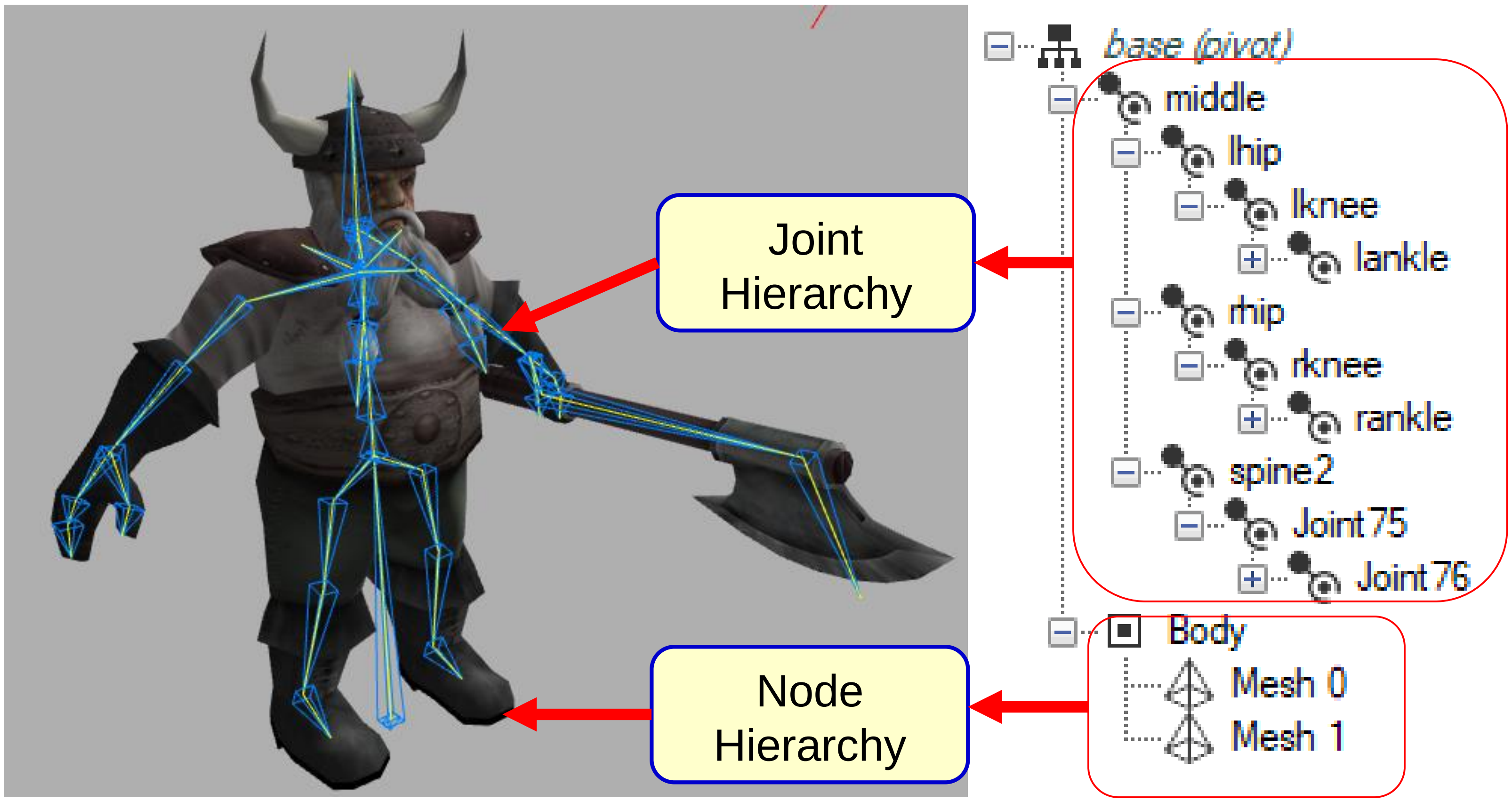
- the **rotational transformations** at the joints of a character model
- the movement (**translation**) of the whole character model

For character animation, we require a set of **joints** and a hierarchical structure similar to the node hierarchy.

Note:

A **node** hierarchy (scene graph) represents the mesh model, while a **joint** hierarchy (skeleton) specifies how motion transformation must be applied to the model.

Node and Joint Hierarchies

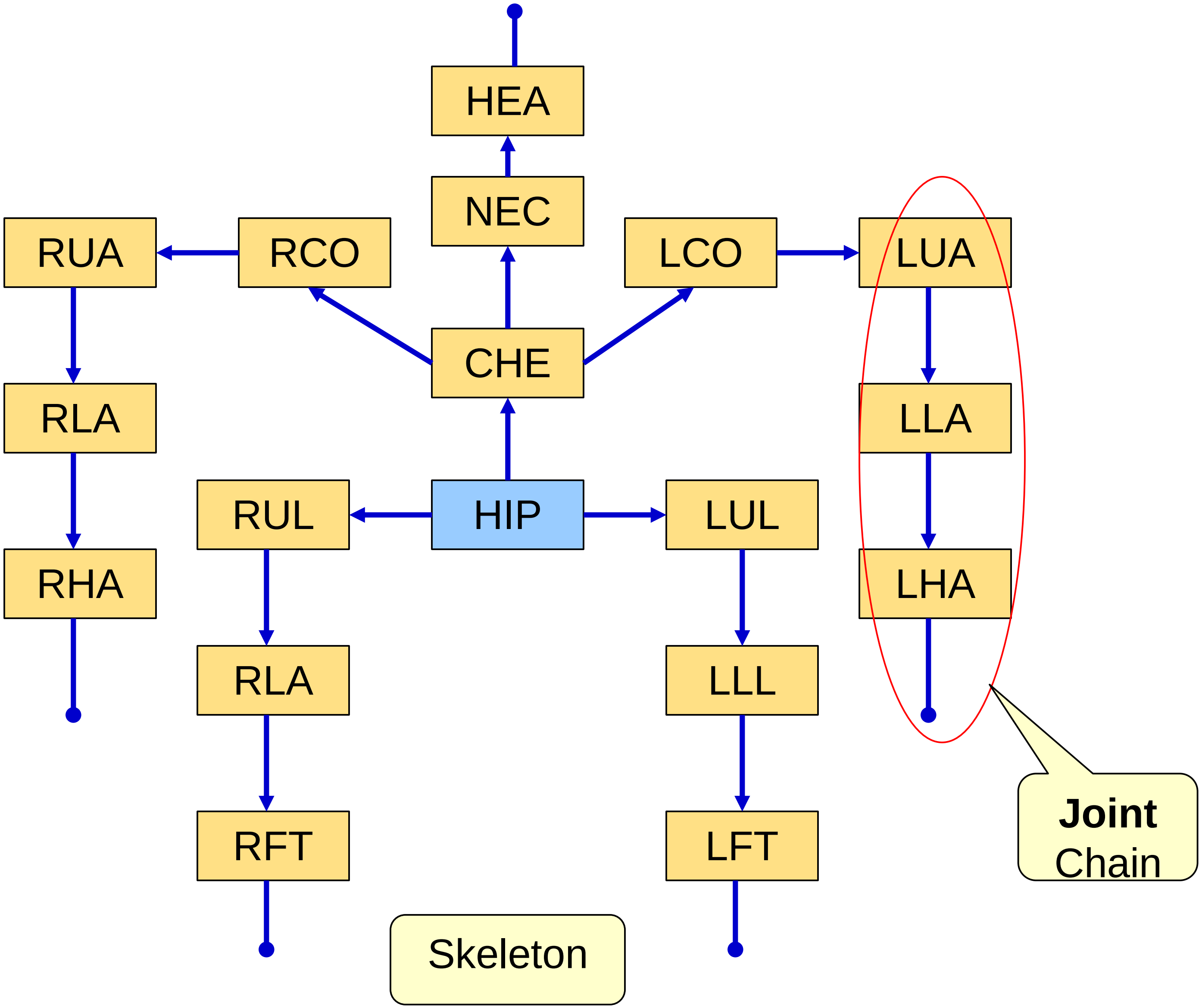


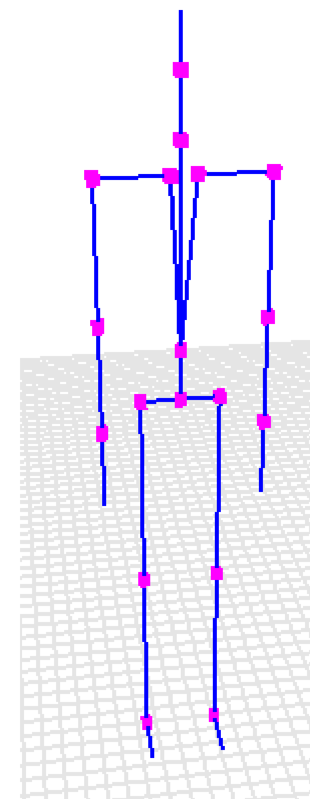
Notation: = Node = Joint

http://www.open3mod.com/

<u>Note</u>: A joint does not contain any mesh data

Joint Hierarchy of a Character Model





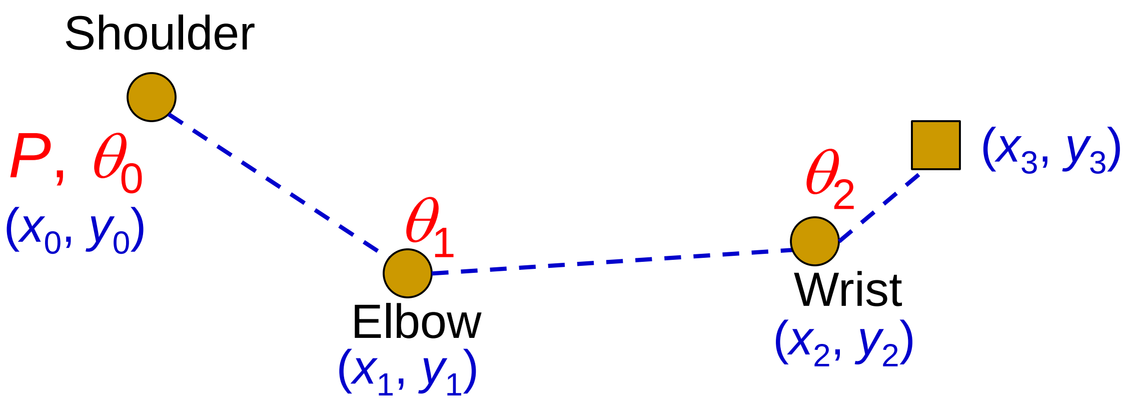
Animation Using Motion Data

- Motion capture data are always associated with a skeleton structure consisting of one or more joint chains.
- However, motion capture files do not contain any mesh definitions.
- In this section, we look at the structure of motion capture data and ways to create skeletal animations using them.
- We will consider animations of rigged character models (models with attached skeletons as shown in slide 6) in the next section.

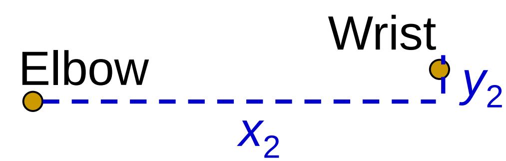
Motion Capture Data: Example

```
HIERARCHY
ROOT Hips
{
        OFFSET 0.00000 0.00000 0.00000
        CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
        JOINT LHipJoint
        {
                OFFSET 0 0 0
                CHANNELS 3 Zrotation Yrotation Xrotation
                JOINT LeftUpLeg
                {
                        OFFSET 1.36306 -1.79463 0.83929
                        CHANNELS 3 Zrotation Yrotation Xrotation
                        JOINT LeftLeg
                        {
                                :
                                :
                }
        }
        JOINT RHipJoint
                :
                :
}
MOTION
Frames: 2752
Frame Time: .0083333
9.3722 17.8693 -17.3198 0 0 0 0 0 0 -17 0 0 0 0 0 0 0 0 0 0 0 0 0 17 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9.3722 17.8693 -17.3198 -3.2316 -7.5970 -2.0168 0.0000 0.0000 0.0000 -13.8102 2.5002
3.3502 1.1281 6.7462 18.9656 0.8229 0.7402 -17.4237 0.0598 -1.5779 -4.3405 0.0000
0.0000 0.0000 22.6124 4.5864 1.0031 -1.9501 -8.7613 25.0515 -0.6734 -4.2652 -17.6334
-0.0434 -1.3453 3.6994 1.1730 -0.3951 2.3019 2.7039 -0.4627 0.0030 2.1315 -0.2318 -
1.2745 -6.7079 -3.7653 -9.3282 -3.0132 -3.2335 27.8377 -0.3184 -2.5573 10.5560 -
0.0000 0.0000 0.0000 -59.1897 -43.6055 29.6611 154.4073 -36.8037 -111.3575 -0.0000
0.0000 -10.1195 38.9427 -13.7335 -10.8605 -7.1250 0.0000 -0.0000 7.7209 12.6535 -
5.8971 -0.0000 0.0000 0.0000 87.3917 1.1336 15.0356 -10.9661 18.2362 -1.7653 0.0000 -
0.8001 -1.0527 -16.5784 0.0606 -1.5880 -4.3683 0.0000 0.0000 0.0000 23.1403 4.6447 -
7.3492 -2.1090 -9.0895 26.0757 -0.5356 -5.7244 -17.0969 -0.0685 -1.6881 4.6445 1.5604
-0.4589 10.1001 4.1127 -0.2882 2.1111 3.4417 -0.1474 -3.4522 -7.5852 -2.7489 -8.1076
-0.0810 -2.6948 12.0186 1.0235 -1.6659 5.0637 -0.0000 0.0000 0.0000 -74.2111 -30.3846
35.7068 153.3469 -37.8464 -110.7145 -0.0000 0.0000 -2.1706 47.8795 -17.2711 -18.1772
-7.1250 0.0000 -0.0000 12.5973 5.6069 -11.5965 -0.0000 0.0000 0.0000 85.0422 -1.6817
30.9269 -27.3229 38.4854 -9.6994 0.0000 -0.0000 -20.1460 -21.3943 -22.9966 8.7021
        :
        :
```

Hierarchical Data: 2D Example



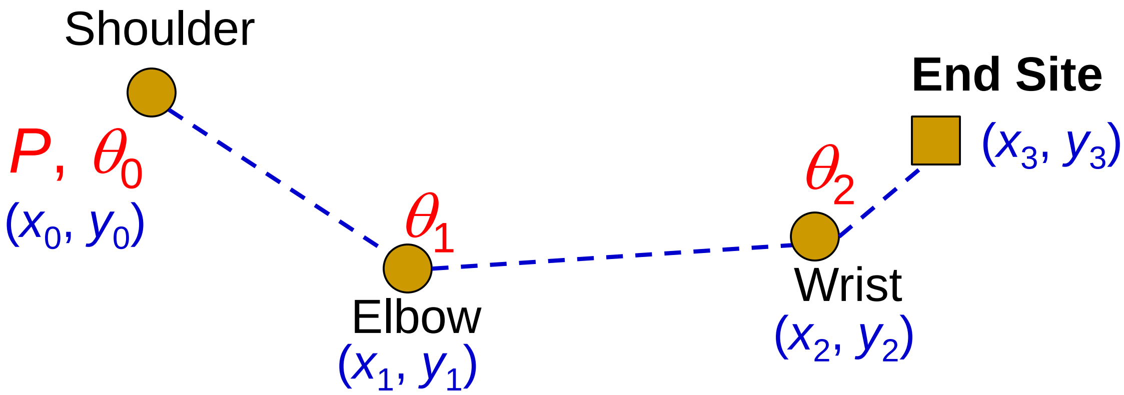
- **Skeleton parameters (Offsets):**



  (x_0, y_0), (x_1, y_1), (x_2, y_2) , (x_3, y_3)

  - These are fixed parameters. Each offset is defined **relative** to the joint's parent.
  - The skeleton parameters uniquely define the location of the joints in the initial configuration.
  - In the above example, (x_0, y_0) is the initial position of shoulder joint from the origin in the initial configuration.

Hierarchical Data: 2D Example



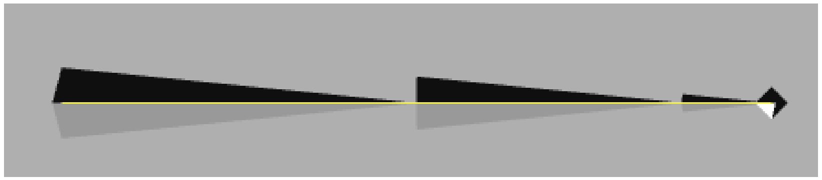
- **Animatable parameters (Channel):**

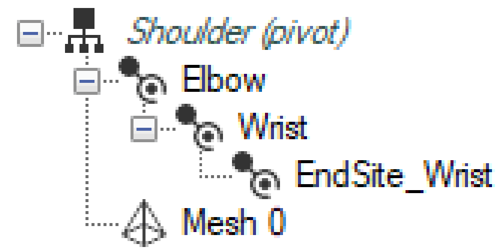
  P (global position), $\theta_0, \theta_1, \theta_2$ (**joint angles**)

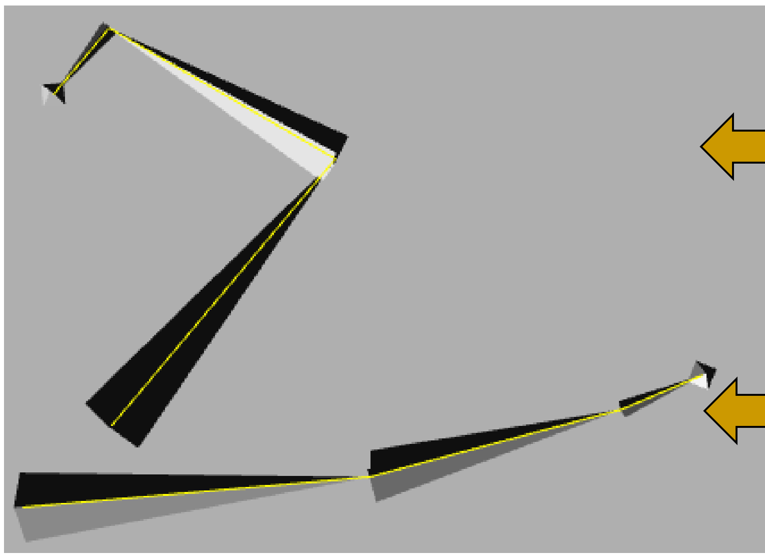
  - These parameters are defined for each **animation frame.**
  - Joint angles specify the rotation of a link from initial configuration.
  - The root joint will have a translation parameter P specifying the motion of the whole joint chain.
  - The leaf node of a joint chain is known as the "end site" and it does not contain any animatable parameters.

2D Example (Test.bvh)

Initial Configuration:







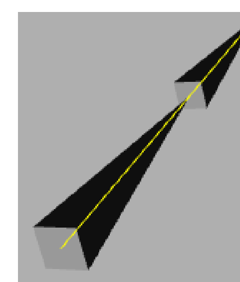
Keyframe 9
$P = (1,\ 0.18)$
$\theta_0 = 50,\ \ \theta_1 = 100,\ \ \theta_2 = 80$

Keyframe 1
$P = (0.1,\ 0)$
$\theta_0 = 5,\ \ \theta_1 = 10,\ \ \theta_2 = 8$

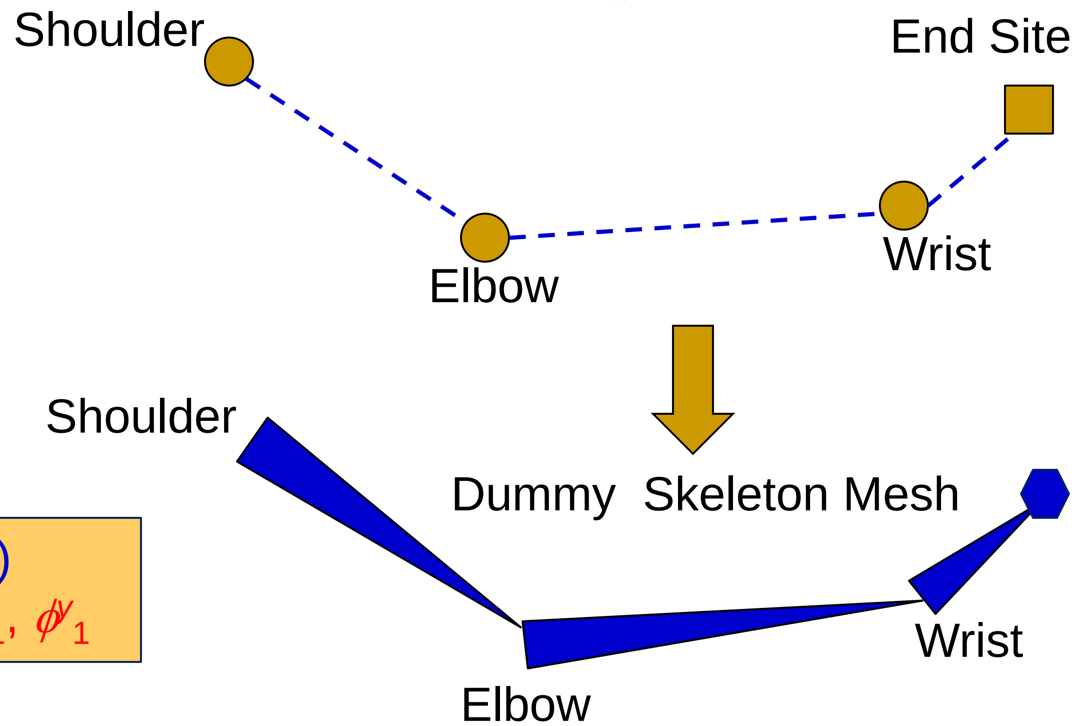
<u>Note:</u>

Motion files do not contain mesh data.

Open3D Model Viewer and Assimp create a **dummy mesh** when a motion file is loaded.



Hierarchical Data: 3D Example



Root (Shoulder)

Offset: (x_0, y_0, z_0)
Position: (X_P, Y_P, Z_P)
Rotation: $\theta^z_0, \psi^x_0, \phi^y_0$

Elbow

Offset: (x_1, y_1, z_1)
Rotation: $\theta^z_1, \psi^x_1, \phi^y_1$

Wrist

Offset: (x_2, y_2, z_2)
Rotation: $\theta^z_2, \psi^x_2, \phi^y_2$

Channel

Offset: (x_3, y_3, z_3)

A channel is the sequence of values of animatable parameters of a particular joint.

BVH Data

Test.bvh

A BVH file contains two sections:
HIERARCHY and **MOTION**

```
HIERARCHY
ROOT Shoulder
{
    OFFSET 0.00000 0.00000 0.00000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT Elbow
    {
        OFFSET 2.0  0.0  0.0
        CHANNELS 3 Zrotation Yrotation Xrotation
        JOINT Wrist
        {
            OFFSET 1.5  0.0 0.0
            CHANNELS 3 Zrotation Yrotation Xrotation
            End Site
            {
                OFFSET 0.5 0.0 0.0
            }
```

Specifies what each channel (animatable parameter) value represents.

...Continued on next slide.

BVH Representation

Test.bvh

MOTION

Frames: 10

Frame Time: 0.1

Shoulder	Elbow	Wrist
0.1 0. 0. 5. 0. 0.	10. 0. 0.	8. 0. 0.
0.2 0.2 0. 10. 0. 0.	20. 0. 0.	16. 0. 0.
0.3 0.4 0. 15. 0. 0.	30. 0. 0.	24. 0. 0.
0.4 0.6 0. 20. 0. 0.	40. 0. 0.	32. 0. 0.
0.5 0.8 0. 25. 0. 0.	50. 0. 0.	40. 0. 0.
0.6 0.10 0. 30. 0. 0.	60. 0. 0.	48. 0. 0.
0.7 0.12 0. 35. 0. 0.	70. 0. 0.	56. 0. 0.
0.8 0.14 0. 40. 0. 0.	80. 0. 0.	64. 0. 0.
0.9 0.16 0. 45. 0. 0.	90. 0. 0.	72. 0. 0.
1. 0.18 0. 50. 0. 0.	100. 0. 0.	80. 0. 0.

3 **channels**, one for each joint.
The end site does not have a channel.

The above list contains 10 key frames.

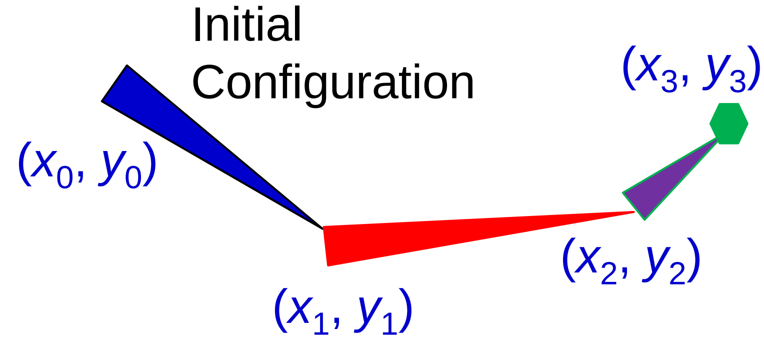
A channel is a set of values of animation parameters for a single joint.

BVH – Another Example

```
HIERARCHY
ROOT Hips
{
        OFFSET  0.00    0.00    0.00
        CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
        JOINT LeftUpLeg
        {
                OFFSET   3.430000        0.000000        0.000000
                CHANNELS 3 Zrotation Xrotation Yrotation
                JOINT LeftLowLeg
                {
                        OFFSET   0.000000        -18.469999       0.000000
                        CHANNELS 3 Zrotation Xrotation Yrotation
                        JOINT LeftFoot
                        {
                                OFFSET   0.000000        -17.950001       0.000000
                                CHANNELS 3 Zrotation Xrotation Yrotation
                                End Site
                                {
                                        OFFSET   0.000000        -3.119996        0.000000
                                }
                        }
                }
        }
        JOINT RightUpLeg
        {
                OFFSET  -3.430000        0.000000        0.000000
                CHANNELS 3 Zrotation Xrotation Yrotation
                JOINT RightLowLeg
                {
                        OFFSET   0.000000        -18.809999       0.000000
                        CHANNELS 3 Zrotation Xrotation Yrotation
                        JOINT RightFoot
                        {
                                OFFSET   0.000000        -17.570000       0.000000
                                CHANNELS 3 Zrotation Xrotation Yrotation
                                End Site
                                {
                                        OFFSET   0.000000        -3.250000        0.000000
                                }
```

Modelling a 2D Skeleton

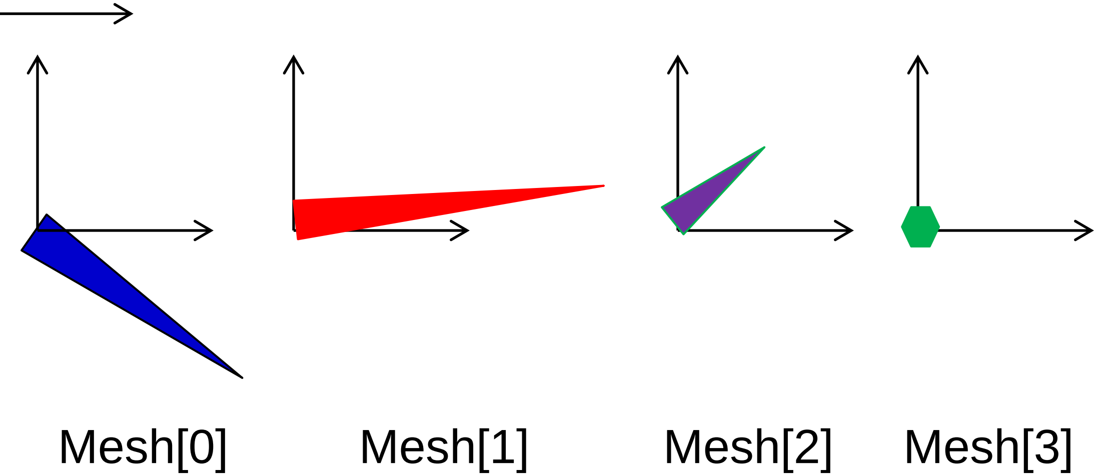
A node hierarchy for the dummy mesh created using skeleton parameters:



Scene

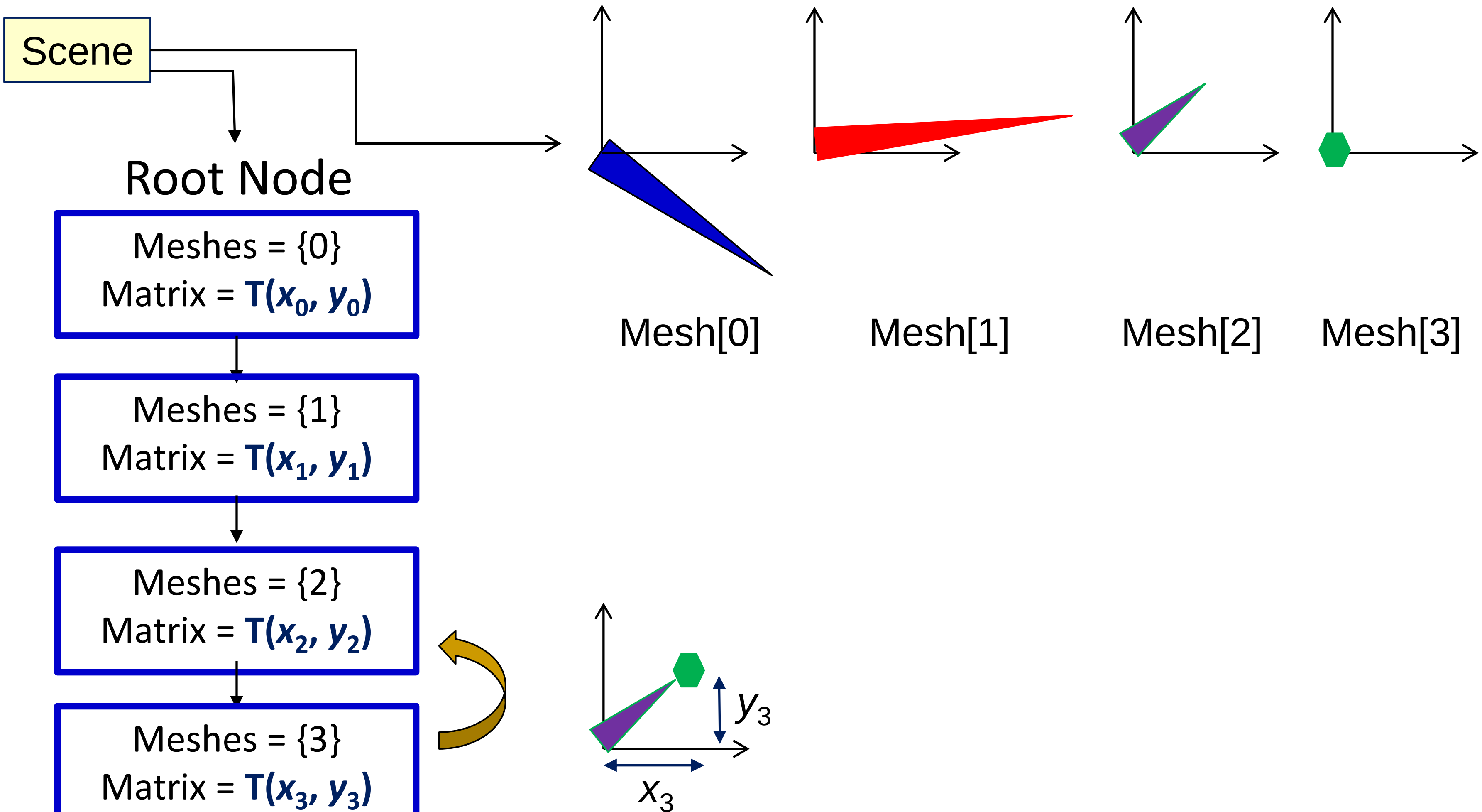
Root Node

Meshes = {0} Matrix = **T(x_0, y_0)**
Meshes = {1} Matrix = **T(x_1, y_1)**
Meshes = {2} Matrix = **T(x_2, y_2)**
Meshes = {3} Matrix = **T(x_3, y_3)**

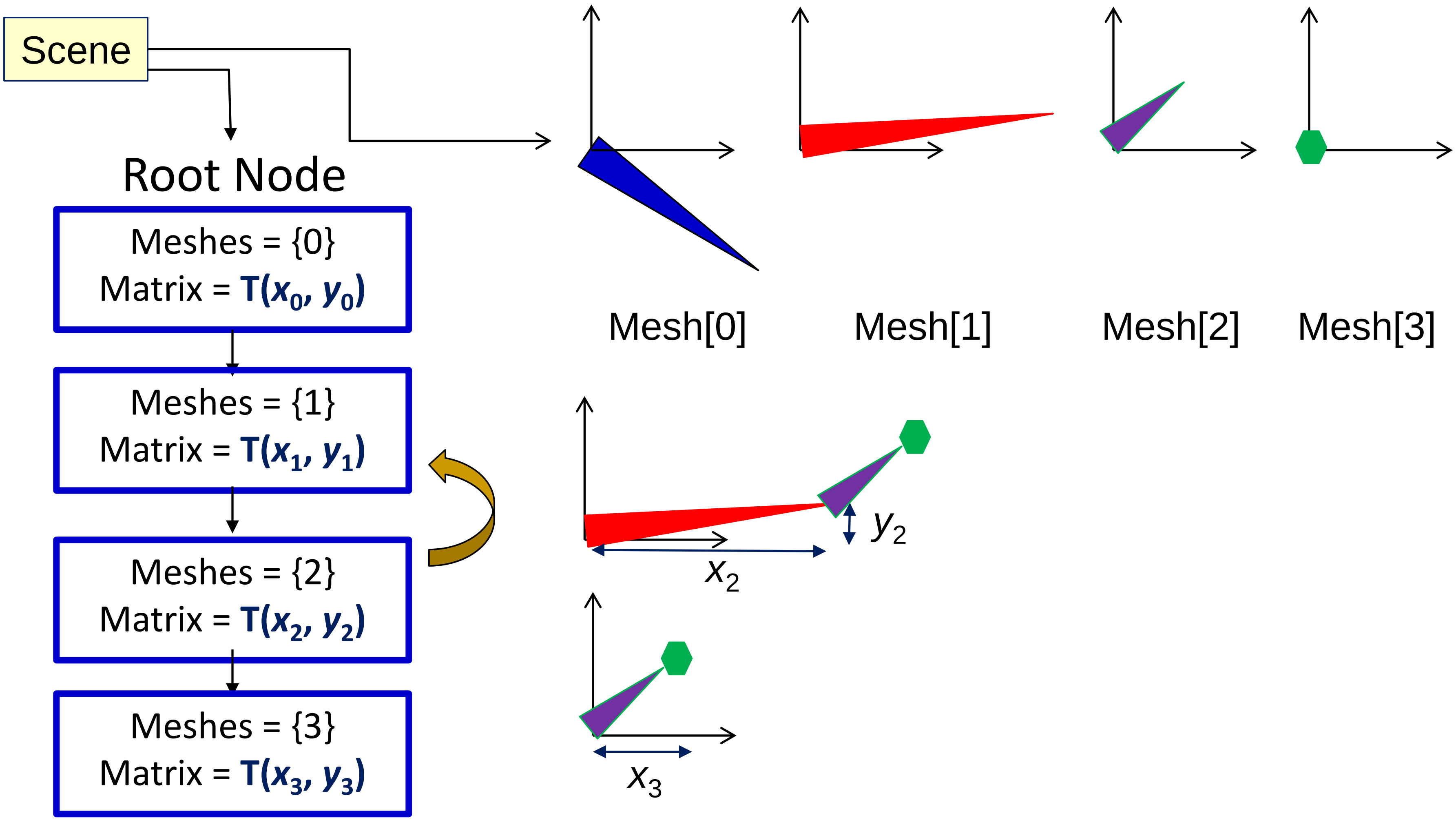


Note: The matrices give the transformation from a node's local coordinate system to its parent's coordinate system.

Modelling a 2D Skeleton



Modelling a 2D Skeleton



Modelling a 2D Skeleton

Scene

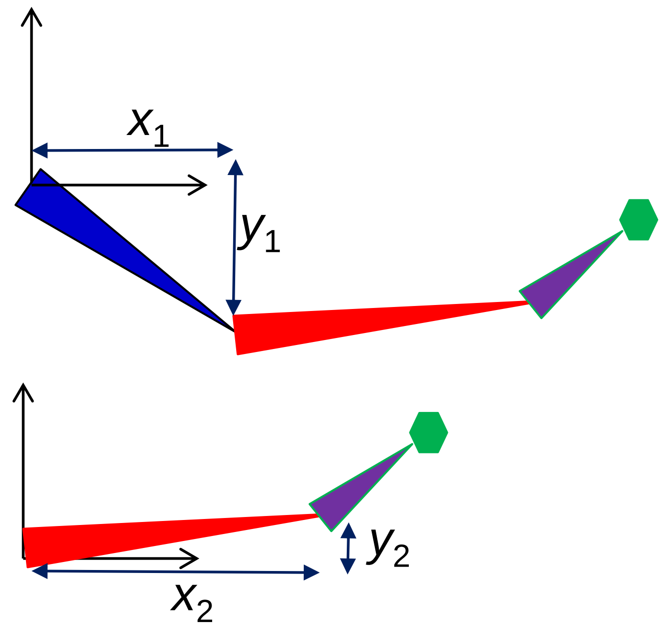
Root Node

Meshes = {0}
Matrix = **T(x_0, y_0)**

Meshes = {1}
Matrix = **T(x_1, y_1)**

Meshes = {2}
Matrix = **T(x_2, y_2)**

Meshes = {3}
Matrix = **T(x_3, y_3)**



Modelling a 2D Skeleton

Scene

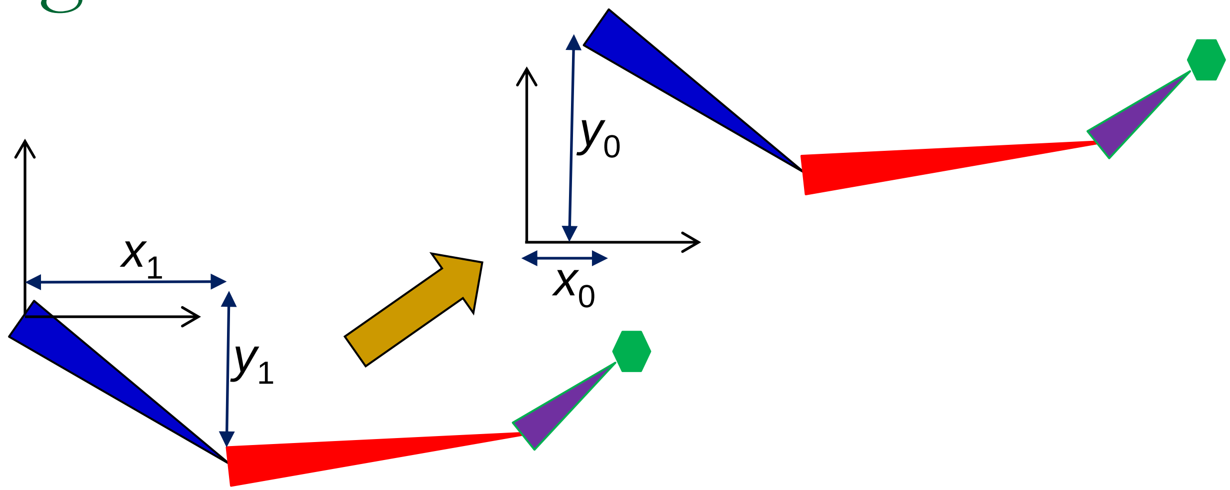
Root Node

Meshes = {0}
Matrix = **T(x_0, y_0)**

Meshes = {1}
Matrix = **T(x_1, y_1)**

Meshes = {2}
Matrix = **T(x_2, y_2)**

Meshes = {3}
Matrix = **T(x_3, y_3)**



Animating a 2D Skeleton

Each mesh is moved by transformations defined using animatable parameters:

Root Node

Meshes = {0}
Matrix = $\mathbf{T}(x_0, y_0)\ \mathbf{T}(P)\mathbf{R}(\theta_0)$

↓

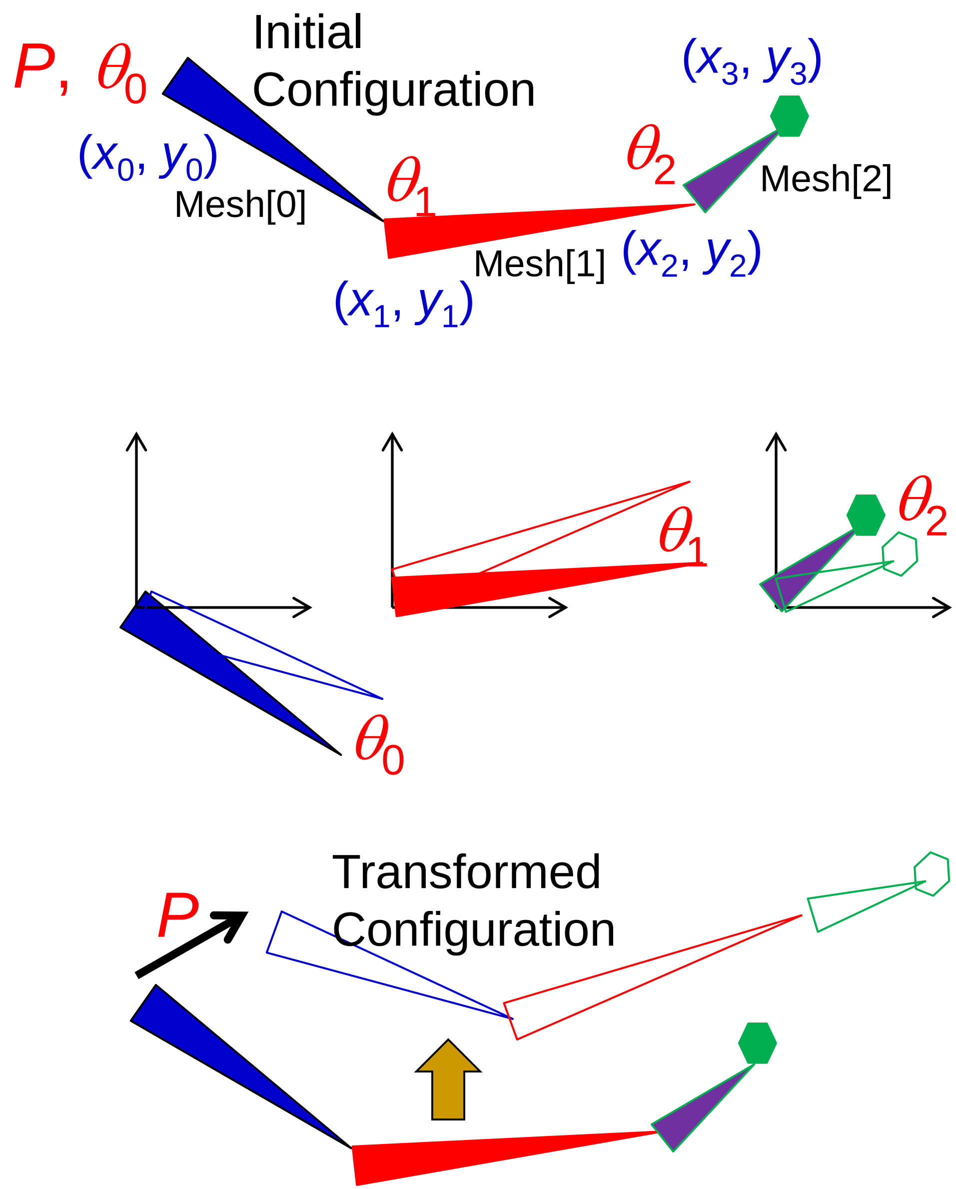
Meshes = {1}
Matrix = $\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)$

↓

Meshes = {2}
Matrix = $\mathbf{T}(x_2, y_2)\mathbf{R}(\theta_2)$

↓

Meshes = {3}
Matrix = $\mathbf{T}(x_3, y_3)$



The combined transformation matrix for mesh[1] is $\mathbf{T}(x_0, y_0)\ \mathbf{T}(P)\mathbf{R}(\theta_0)\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)$

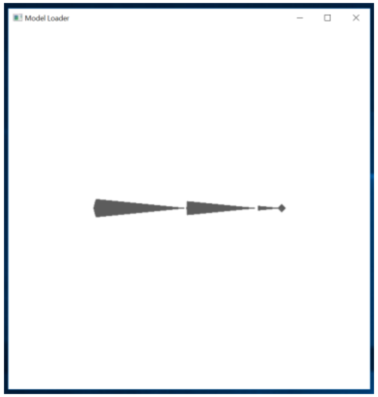
Animating a Skeleton

- Assimp creates a dummy mesh as shown on previous slides when a BVH file is loaded.
- The mesh can be animated only if it is “segmented” (i.e., the mesh for each link is assigned to a separate node as shown on previous slide, so that each link can be independently transformed using its animatable parameters).
- Please make sure that you use Assimp post process flag *aiProcess_Debone* when loading a BVH file:

```
scene = aiImportFile  ( fileName,
aiProcessPreset_TargetRealtime_MaxQuality |
aiProcess_Debone );
```

Test.bvh

See slides 14, 15



```
====================== Scene Data =======================
Number of animations = 1
Number of cameras = 0
Number of lights = 0
Number of materials = 1
Number of meshes = 4
Number of textures = 0
=================== Node Data ==========================
Node Name: Shoulder  Parent: NO PARENT  nchild = 1  nmesh = 1
Mesh indices: 0
Transformation:  1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
=================== Node Data ==========================
Node Name: Elbow  Parent: Shoulder  nchild = 1  nmesh = 1
Mesh indices: 1
Transformation:  1 0 0 2 0 1 0 0 0 0 1 0 0 0 0 1
=================== Node Data ==========================
Node Name: Wrist  Parent: Elbow  nchild = 1  nmesh = 1
Mesh indices: 2
Transformation:  1 0 0 1.5 0 1 0 0 0 0 1 0 0 0 0 1
=================== Node Data ==========================
Node Name: EndSite_Wrist  Parent: Wrist  nchild = 0  nmesh = 1
Mesh indices: 3
Transformation:  1 0 0 0.5 0 1 0 0 0 0 1 0 0 0 0 1
```

```
HIERARCHY
ROOT Shoulder
{
   OFFSET 0.00000 0.00000 0.00000
   CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
   JOINT Elbow
   {
      OFFSET 2.0  0.0  0.0
      CHANNELS 3 Zrotation Yrotation Xrotation
      JOINT Wrist
      {
            OFFSET 1.5  0.0 0.0
            CHANNELS 3 Zrotation Yrotation Xrotation
            End Site
            {
               OFFSET 0.5 0.0 0.0
            }
      }
   }
}
MOTION
Frames: 10
Frame Time: .1
0.1 0.0  0.0 5.0 0.0 0.0  10.0  0.0 0.0 8.0 0.0 0.0
0.2 0.02 0.0 10.0 0.0 0.0 20.0 0.0 0.0 16.0 0.0 0.0
0.3 0.04 0.0 15.0 0.0 0.0 30.0 0.0 0.0 24.0 0.0 0.0
0.4 0.06 0.0 20.0 0.0 0.0 40.0 0.0 0.0 32.0 0.0 0.0
0.5 0.08 0.0 25.0 0.0 0.0 50.0 0.0 0.0 40.0 0.0 0.0
0.6 0.10 0.0 30.0 0.0 0.0 60.0 0.0 0.0 48.0 0.0 0.0
0.7 0.12 0.0 35.0 0.0 0.0 70.0 0.0 0.0 56.0 0.0 0.0
0.8 0.14 0.0 40.0 0.0 0.0 80.0 0.0 0.0 64.0 0.0 0.0
0.9 0.16 0.0 45.0 0.0 0.0 90.0 0.0 0.0 72.0 0.0 0.0
1.0 0.18 0.0 50.0 0.0 0.0 100.0 0.0 0.0 80.0 0.0 0.0
```

Ticks

Test.bvh

Ticks are equally spaced markers along the time axis.
In a BVH file, each keyframe corresponds to a tick and can be used as an index to keyframes.

MOTION

Frames: 10

Frame Time: .1

```
0.1   0.  0.     5. 0. 0.      10.   0. 0.      8. 0. 0.     Tick 0
0.2  0.2 0.     10. 0. 0.      20.   0. 0.     16. 0. 0.
0.3  0.4 0.     15. 0. 0.      30.   0. 0.     24. 0. 0.
0.4  0.6 0.     20. 0. 0.      40.   0. 0.     32. 0. 0.
0.5  0.8 0.     25. 0. 0.      50.   0. 0.     40. 0. 0.
0.6  0.10 0.  30. 0. 0.      60.   0. 0.     48. 0. 0.
0.7  0.12 0.  35. 0. 0.      70.   0. 0.     56. 0. 0.
0.8  0.14 0.  40. 0. 0.      80.   0. 0.     64. 0. 0.
0.9  0.16 0.  45. 0. 0.      90.   0. 0.     72. 0. 0.
1.   0.18 0.  50. 0. 0.     100.  0. 0.     80. 0. 0.     Tick 9
```

Ticks

Test.bvh

In the previous example,

Duration (ticks) = Frames–1 = 9

Ticks per sec = 1/(Frame Time) = 10

The animation speed or frame rate is determined by “Ticks per sec”.

For example, if you require an animation speed of 30 frames per sec, you should set the “Frame Time” in the BVH file as 0.033

We can access animation parameters from motion data as follows:

channel

0.1	0.	0.	5.	0.	0.	10.	0.	0.	8.	0.	0.
0.2	0.2	0.	10.	0.	0.	20.	0.	0.	16.	0.	0.
0.3	0.4	0.	15.	0.	0.	30.	0.	0.	24.	0.	0.
0.4	0.6	0.	20.	0.	0.	40.	0.	0.	32.	0.	0.
0.5	0.8	0.	25.	0.	0.	50.	0.	0.	40.	0.	0.

tick

```
channel->mRotationKeys[tick].mValue
```

Assimp Classes

- Animation Object (aiAnimation)
  - **Array of Channels**: anim->mChannels
  - **Ticks per Second:** anim->mTicksPerSecond
  - **Duration in Ticks:** anim->mDuration
- Channels (aiNodeAnim): A channel is also known as an animation node.
  - The first channel corresponding to “Shoulder” is obtained as anim->mChannels[0], “Elbow” anim->mChannels[1], etc.
  - **Channel’s position Keys**: channel->mPositionKeys
  - **Channel’s rotation Keys:** channel->mRotationKeys.
  - Each channel has a name (channel->mNodeName) that corresponds to a joint’s name.

Assimp Output for **Test.bvh**

- Number of animations (scene->mNumAnimations): 1
- Animation object (anim): scene->mAnimations[0]
- Number of channels (anim->mNumChannels): 3
- Animation duration in ticks (anim->mDuration): 9
- Number of ticks per sec (anim->mTicksPerSecond): 10

Please refer to the `printAnimInfo()` function of "assimp_extras.h" for examples of Assimp expressions used for accessing animation/channel parameters.

Position and Rotation Keys

Test.bvh

MOTION

Frames: 10

Frame Time: .1

Shoulder	Elbow	Wrist
0.1 0. 0. 5. 0. 0.	10. 0. 0.	8. 0. 0.
0.2 0.2 0. 10. 0. 0.	20. 0. 0.	16. 0. 0.
0.3 0.4 0. 15. 0. 0.	30. 0. 0.	24. 0. 0.
0.4 0.6 0. 20. 0. 0.	40. 0. 0.	32. 0. 0.
0.5 0.8 0. 25. 0. 0.	50. 0. 0.	40. 0. 0.
0.6 0.10 0. 30. 0. 0.	60. 0. 0.	48. 0. 0.
0.7 0.12 0. 35. 0. 0.	70. 0. 0.	56. 0. 0.
0.8 0.14 0. 40. 0. 0.	80. 0. 0.	64. 0. 0.
0.9 0.16 0. 45. 0. 0.	90. 0. 0.	72. 0. 0.
1. 0.18 0. 50. 0. 0.	100. 0. 0.	80. 0. 0.

The values of rotation keys are stored as **quaternions**.

Assimp Output

Channel: 0 Name: Shoulder
posKey 0: Value = 0.1 0 0
...
posKey 9: Value = 1 0.18 0
rotnKey 0: Value = 0.99 0 0 0.04
...
rotnKey 9: Value = 0.91 0 0 0.42
Channel: 1 Name: Elbow
posKey 0: Value = 2 0 0
rotnKey 0: Value = 0.99 0 0 0.08
...
rotnKey 9: Value = 0.64 0 0 0.76
Channel 2: Name = Wrist
posKey 0: Value = 1.5 0 0
rotnKey 0: Value = 0.99 0 0 0.07
...
rotnKey 9: Value = 0.76 0 0 0.64

Position and Rotation Keys

- Number of position keys of the first channel: 10
- Number of rotation keys of the first channel: 10
- Number of position keys of the second channel: 1
- Number of rotation keys of the second channel: 10

Important Notes:

If a position key has the same set of values for all ticks, then there will be only one key. Similarly, if the angles for a channel have the same values for all ticks, there will be only one rotation key.

For the second and subsequent channels, there will be only one position key which is the joint's offset value.

Animating a Skeleton

Initial Configuration

Root Node

Meshes = {0}
Matrix = $\mathbf{T}(x_0, y_0)$

Meshes = {1}
Matrix = $\mathbf{T}(x_1, y_1)$

Meshes = {2}
Matrix = $\mathbf{T}(x_2, y_2)$

Meshes = {3}
Matrix = $\mathbf{T}(x_3, y_3)$

Transformed Configuration

Root Node

Meshes = {0}
Matrix = $\mathbf{T}(x_0, y_0)\ \mathbf{T}(P)\mathbf{R}(\theta_0)$

Meshes = {1}
Matrix = $\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)$

Meshes = {2}
Matrix = $\mathbf{T}(x_2, y_2)\mathbf{R}(\theta_2)$

Meshes = {3}
Matrix = $\mathbf{T}(x_3, y_3)$

Position Key

Rotation Key

Animation Using BVH Data

- Use a timer function for animation:

```
glutTimerFunc(timeStep, update, 0);
```

- The timer callback function updates “tick”:

```
void update(int value) {
    if (tick > tDuration) return;
    updateNodeMatrices(tick);
    glutTimerFunc(timeStep, update, 0);
    tick++;
    glutPostRedisplay();
}
```

- For each channel *i*, get the values of position and rotation keys:

Check if valid

```
anim = scene->mAnimations[0];
chnl = anim->mChannels[i];
posn = chnl->mPositionKeys[tick].mValue;
rotn = chnl->mRotationKeys[tick].mValue;
```

aiVector3D

aiQuaternion

Animation Using BVH Data

- Convert the values of keys to matrices and construct the product matrix (position * rotation):

aiMatrix4x4

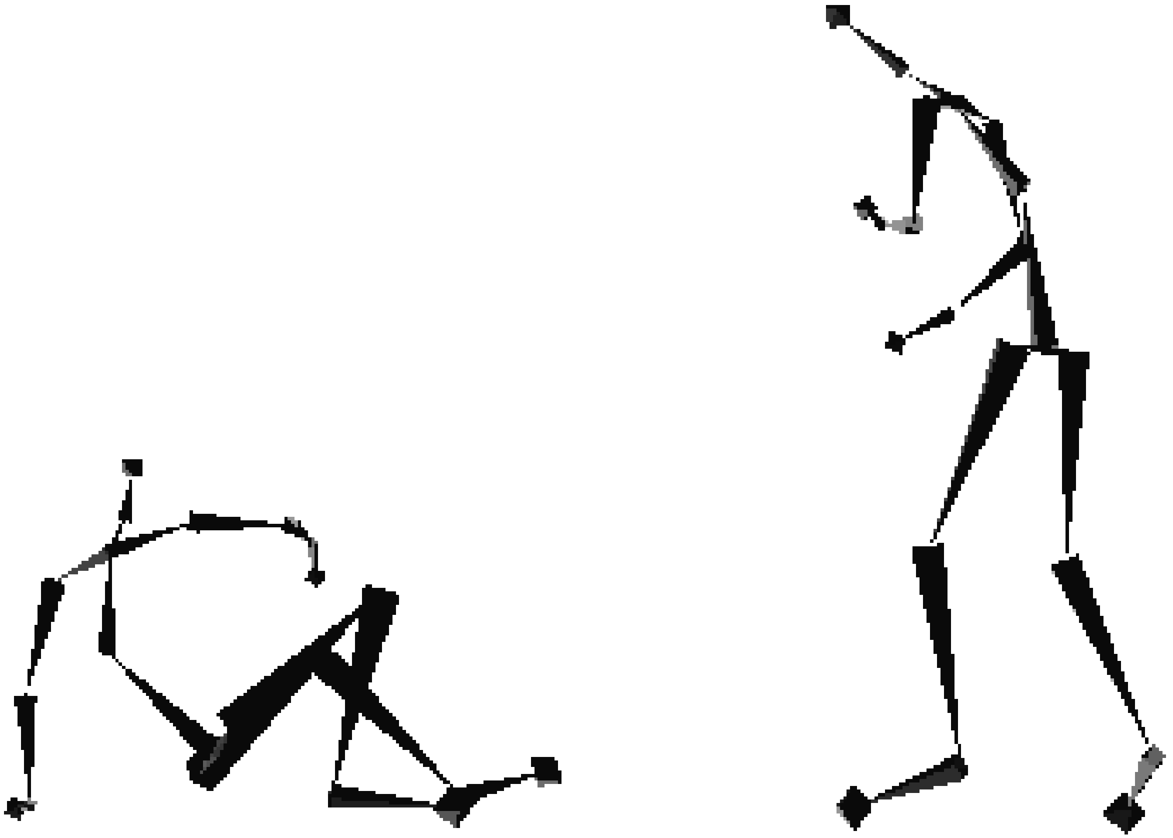
aiMatrix3x3

```
matPos.Translation(posn, matPos);
matRotn3 = rotn.GetMatrix();
matRot = aiMatrix4x4(matRotn3);
matprod = matPos * matRot;
```

- Find the node with the same name as the channel:

```
node = scene->mRootNode->FindNode(chnl->mNodeName);
```

- Replace the node's transformation matrix with the product matrix obtained above. Repeat this for every channel.



Boxing.bvh