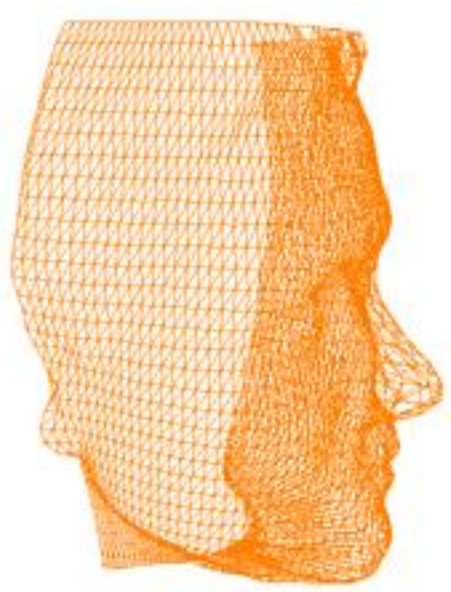
COSC422 Advanced Computer Graphics



10 Motion Capture Data and Skeletal Animation

Semester 2
2021

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
  - Joints and 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

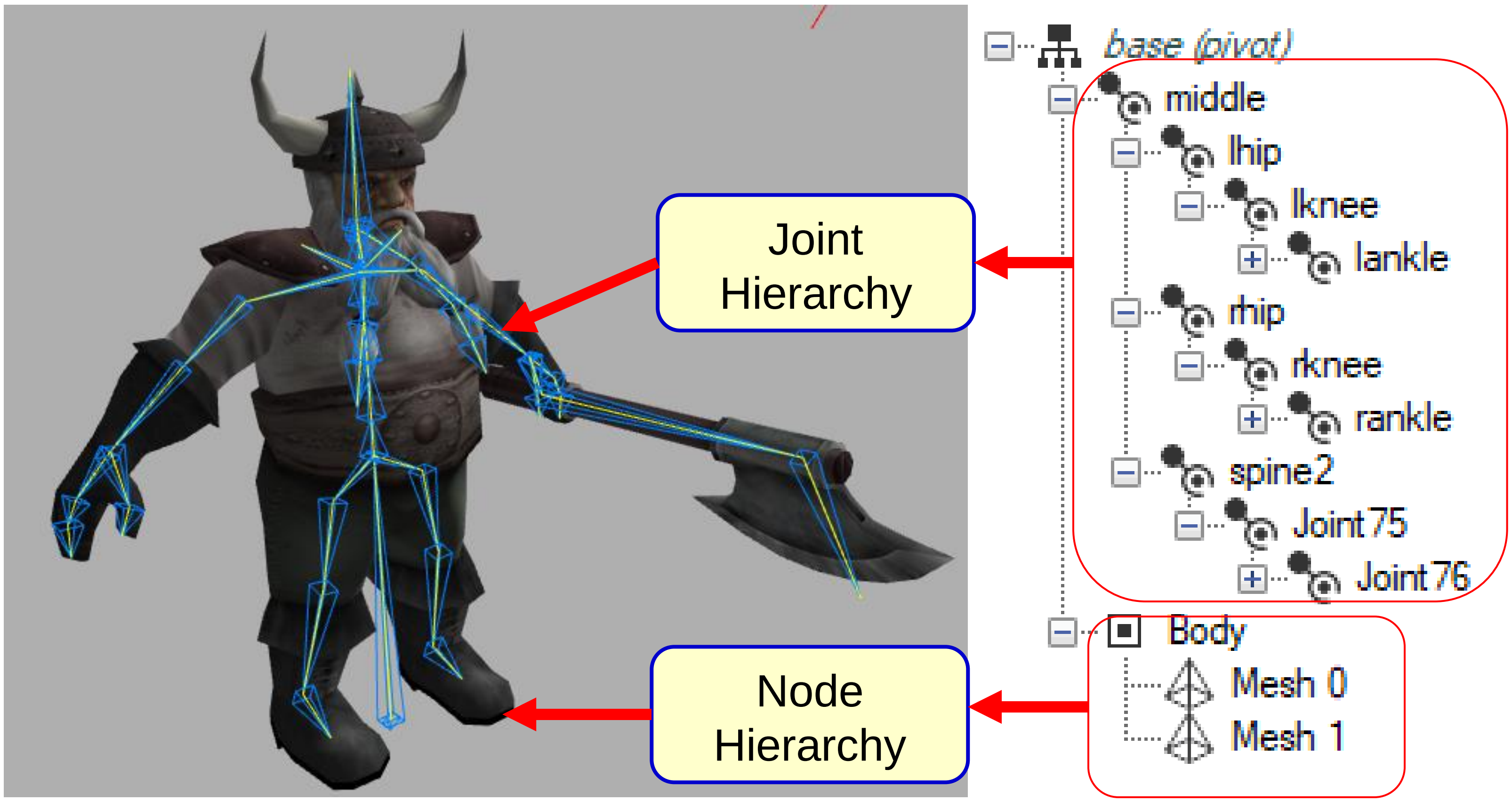
For character animation, we require a set of **joints** and a hierarchical structure similar to the node hierarchy.

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.

(Both hierarchies are part of Assimp's node hierarchy)

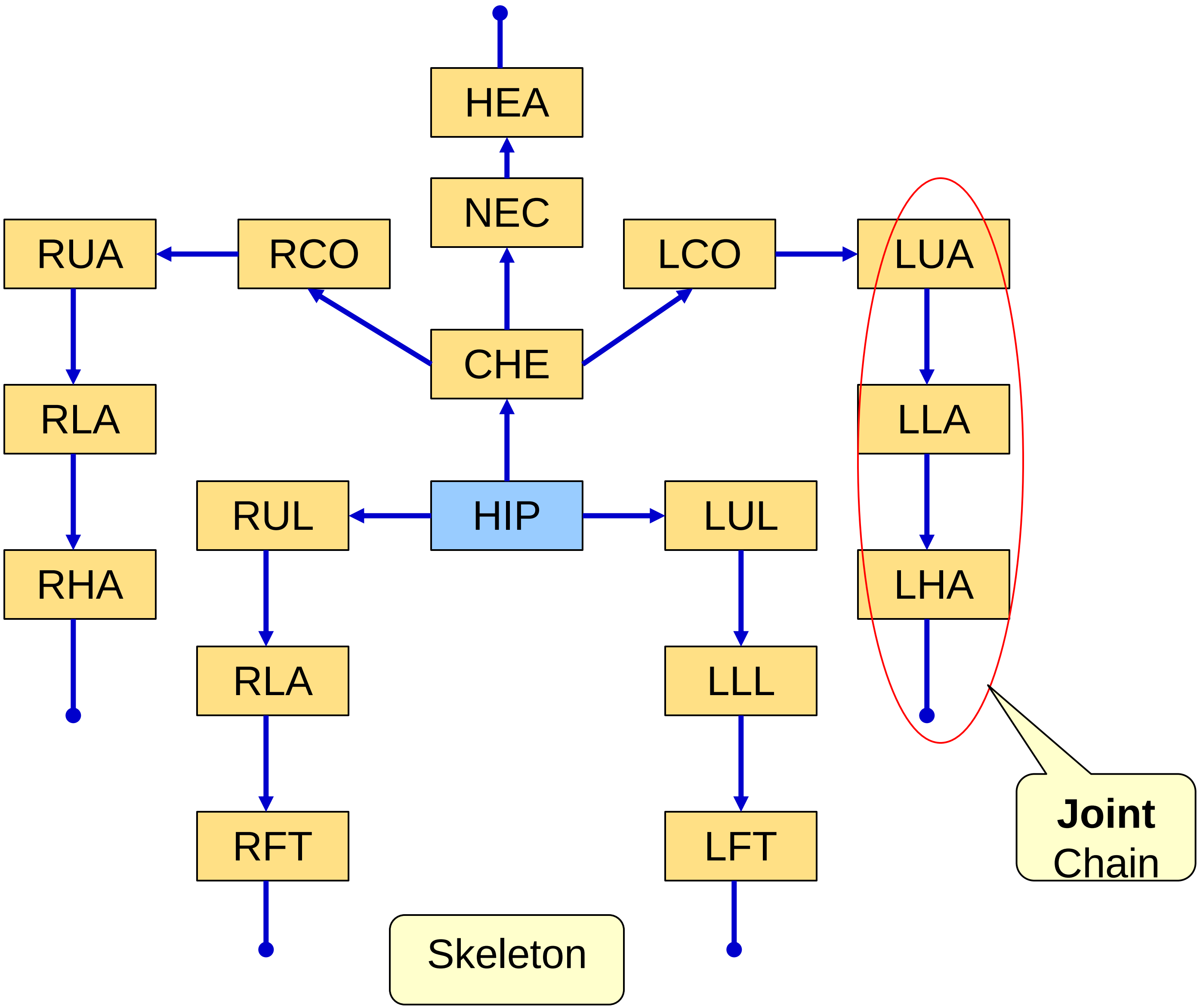
Node and Joint Hierarchies

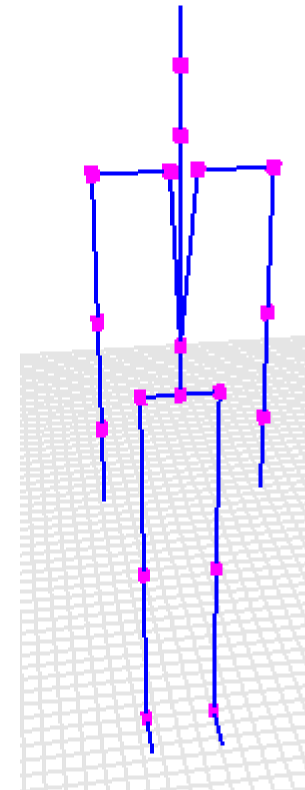


Notation: = Node = Joint

Note: 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

Format BVH

```
HIERARCHY
ROOT hip
{
  OFFSET 0 0 0
  CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
  JOINT abdomen
  {
    OFFSET 0 20.6881 -0.73152
    CHANNELS 3 Zrotation Xrotation Yrotation
    JOINT chest
    {
      OFFSET 0 11.7043 -0.48768
      CHANNELS 3 Zrotation Xrotation Yrotation
      JOINT neck
      {
        OFFSET 0 22.1894 -2.19456
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT head
        {
          OFFSET -0.24384 7.07133 1.2192
          CHANNELS 3 Zrotation Xrotation Yrotation
          End Site
          {
            OFFSET 1 0 0
```

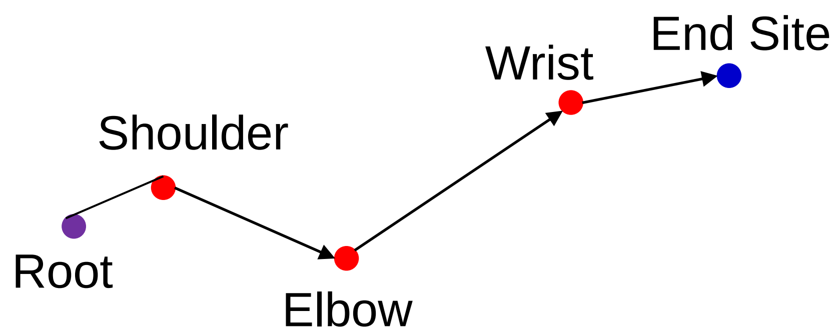
Joint Hierarchy **(HIERARCHY)**

```
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 17 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
9.3728 17.8666 -17.3192 -3.1009 -7.5883 -2.0638 0.0000 0.0000 0.0000 -13.9682 2.5229
9.3692 17.8658 -17.3215 -3.2095 -7.5354 -2.0862 0.0000 0.0000 0.0000 -13.8218 2.5041
9.3726 17.8597 -17.3260 -2.9561 -7.3832 -2.1994 0.0000 0.0000 0.0000 -14.1244 2.3274
9.3733 17.8536 -17.3317 -2.9110 -7.3527 -2.2688 0.0000 0.0000 0.0000 -14.1889 2.2258
9.3689 17.8530 -17.3350 -3.0434 -7.3807 -2.2750 0.0000 0.0000 0.0000 -14.0111 2.2992
9.3676 17.8526 -17.3375 -3.0382 -7.2530 -2.2686 0.0000 0.0000 0.0000 -14.0037 2.2152
9.3648 17.8499 -17.3463 -3.3020 -7.0875 -2.2158 0.0000 0.0000 0.0000 -13.6701 2.0739
9.3598 17.8506 -17.3603 -3.9143 -7.0800 -2.0917 0.0000 0.0000 0.0000 -12.9330 2.0239
9.3594 17.8548 -17.3683 -4.1461 -7.0678 -1.9856 0.0000 0.0000 0.0000 -12.6724 1.9580
9.3610 17.8550 -17.3711 -4.0020 -6.9770 -1.9689 0.0000 0.0000 0.0000 -12.8399 1.9037
9.3649 17.8511 -17.3802 -3.9286 -6.8873 -1.9864 0.0000 0.0000 0.0000 -12.9495 1.7424
9.3686 17.8470 -17.3957 -3.9244 -6.8022 -1.9956 0.0000 0.0000 0.0000 -12.9865 1.4917
9.3650 17.8439 -17.4052 -4.0625 -6.7236 -1.9711 0.0000 0.0000 0.0000 -12.7883 1.4626
9.3582 17.8410 -17.4103 -4.3298 -6.5852 -1.8964 0.0000 0.0000 0.0000 -12.3962 1.6119
```

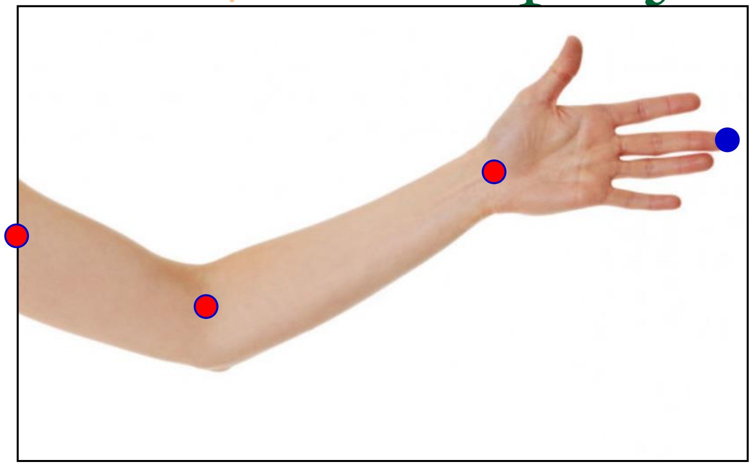
Animation Frames **(MOTION)**

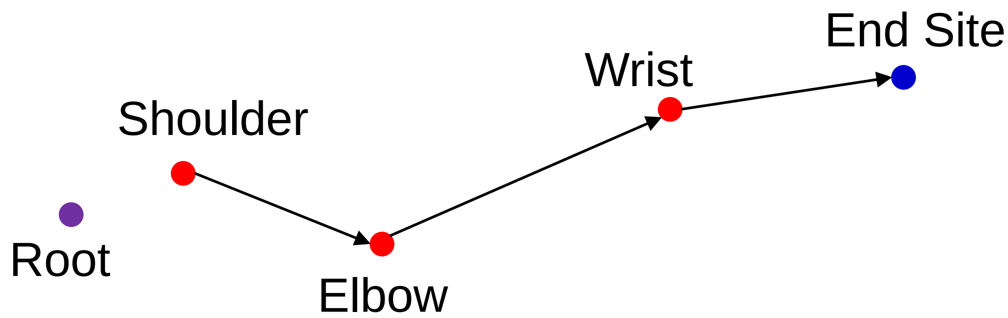
Motion Data

- A motion data file contains
  - A structure specifying a joint hierarchy
  - Skeleton parameters (offsets)
    - The position of each joint ***relative to*** its parent
  - Animation data (key frames or channels)
  - No mesh data


End Site
Wrist
Shoulder
Root
Elbow

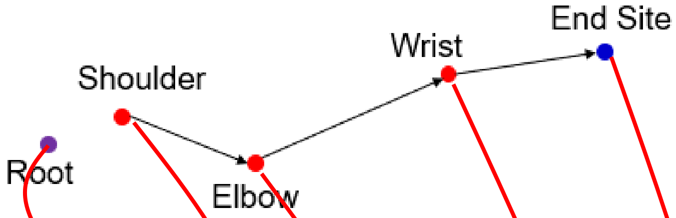
A Simple Joint Chain





- A joint is a point about which a rotation can be performed. A sequence of rotations about a joint is stored as a channel.
- Each joint’s position is specified relative to the parent’s joint.
- Special types of joints:
  - The root (has 6 degrees of freedom)
  - The end point of a joint chain is called the “End Site”

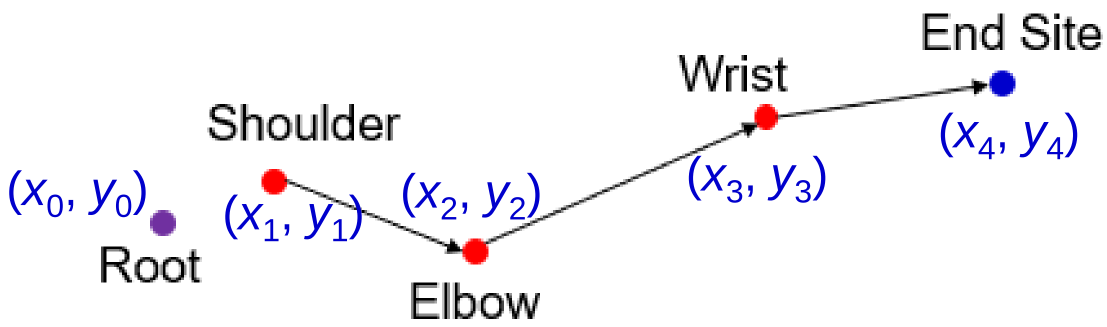
Test Model

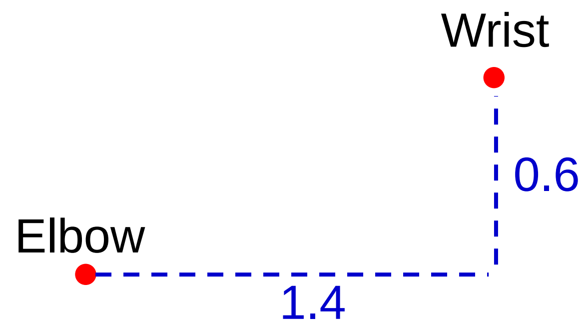



```
HIERARCHY
ROOT root
{
    OFFSET 0.00000 0.00000 0.00000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT Shoulder
    {
        OFFSET 0.5  0.3  0.0
        CHANNELS 3 Zrotation Yrotation Xrotation
        JOINT Elbow
        {
            OFFSET 1.0  -0.5 0.0
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT Wrist
            {
                OFFSET 1.4  0.6 0.0
                CHANNELS 3 Zrotation Yrotation Xrotation
                End Site
                {
                    OFFSET 1.0 0.1 0.0
                }
            }
        }
    }
}
```

Test.bvh

Hierarchical Data: 2D Example





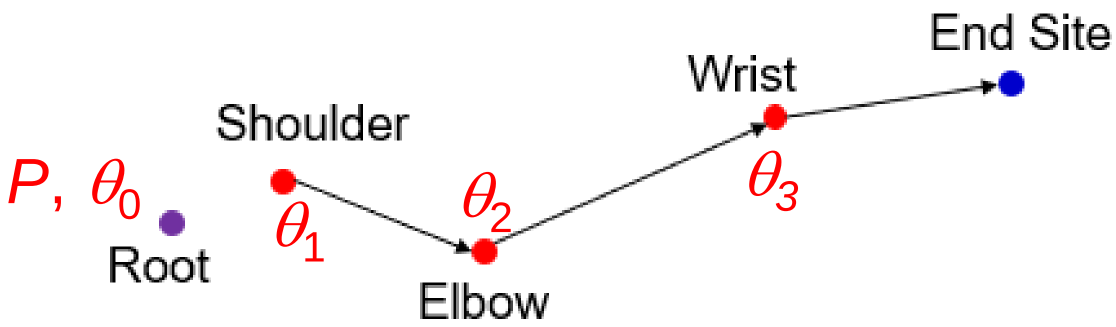
Relative Coordinates

- **Skeleton parameters (Offsets):**

  (x_0, y_0), (x_1, y_1), (x_2, y_2) , (x_3, y_3) , (x_4, y_4)

  - These are fixed parameters. Each offset is defined **relative** to the joint's parent. The skeleton parameters uniquely specify the locations of the joints in the initial configuration.
  - In the above example, (1.4, 0.6, 0) is the initial position of the wrist joint relative to the elbow joint. The absolute position of the wrist joint in the initial configuration is (2.9, 0.4, 0).

Hierarchical Data: 2D Example



- **Animatable parameters (Channel)**:

  P (global position), θ_0, θ_1, θ_2, θ_3 (joint angles)
  - These parameters are defined for each **animation frame.**
  - The root joint will have a translation parameter P specifying the motion of the whole joint chain in world space.
  - A joint angle specifies the rotation applied to a joint in its initial configuration.
  - The leaf node of a joint chain is known as the "end site" and it does not contain any animatable parameters.

BVH Data

Test.bvh

```
HIERARCHY
ROOT root
{
    OFFSET 0.00000 0.00000 0.00000
    CHANNELS 6 Xposition Yposition Zposition  Zrotation Yrotation Xrotation
    JOINT Shoulder
    {
        OFFSET 0.5  0.3  0.0
        CHANNELS 3  Zrotation Yrotation Xrotation
        JOINT Elbow
        {
            OFFSET 1.0  -0.5 0.0
            CHANNELS 3  Zrotation Yrotation Xrotation
            Joint Wrist
            {
                OFFSET 1.4 0.6 0.0
                …
```

Specifies what each channel (animatable parameter) represents.

...Continued on next slide.

Animation Channels

Test.bvh

MOTION

Channel name

Frames: 10

Frame Time: 0.1

Root	Shoulder	Elbow	Wrist
0 0 0 0 0 0	3 0 0	8 0 0	-3 0 0
0 0 0 0 0 0	6 0 0	16 0 0	-6 0 0
0 0 0 0 0 0	9 0 0	24 0 0	-9 0 0
0 0 0 0 0 0	12 0 0	32 0 0	-12 0 0
0 0 0 0 0 0	15 0 0	40 0 0	-15 0 0
0 0 0 0 0 0	18 0 0	48 0 0	-18 0 0
0 0 0 0 0 0	21 0 0	56 0 0	-21 0 0
0 0 0 0 0 0	24 0 0	64 0 0	-24 0 0
0 0 0 0 0 0	27 0 0	72 0 0	-27 0 0
0 0 0 0 0 0	30 0 0	80 0 0	-30 0 0

tick = 0

Duration

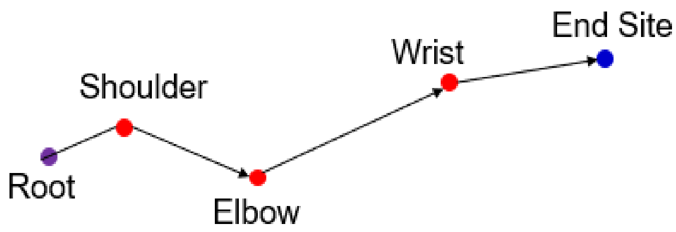
tick = 9

The above list contains 10 key frames.

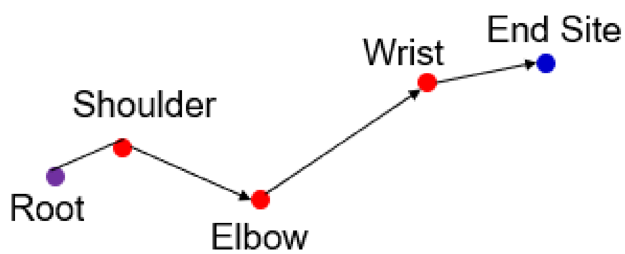
The channel name establishes the correspondence between a channel and a joint.

Animation Channels

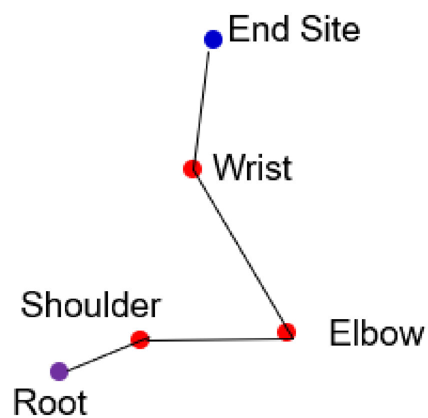
Test.bvh



Initial Configuration
(Base Pose or Zero Pose)
All joint angles zero

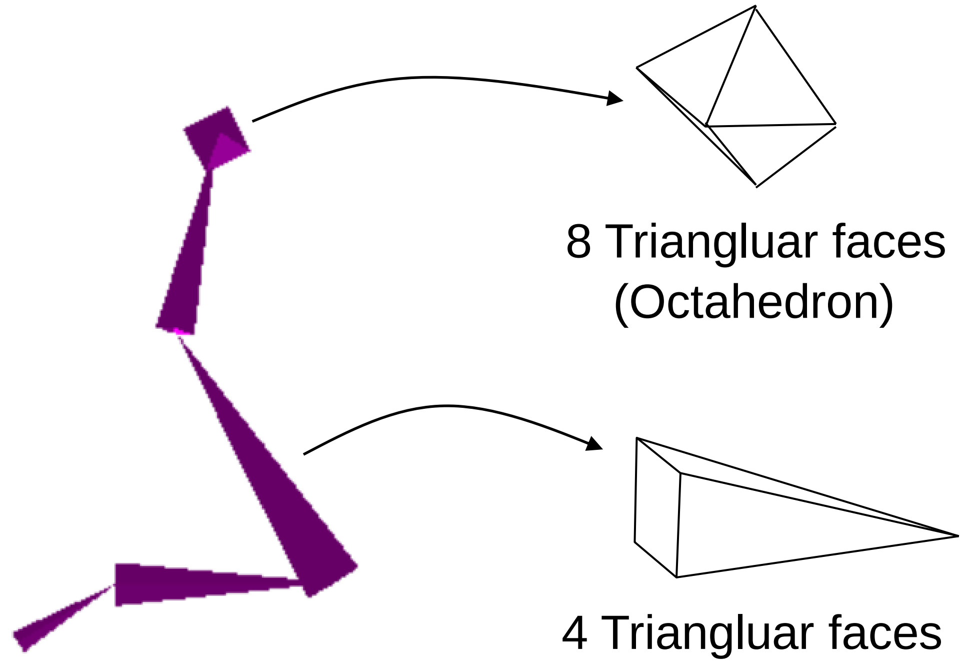


Tick = 0



Tick = Duration
Final Configuration

Test Model

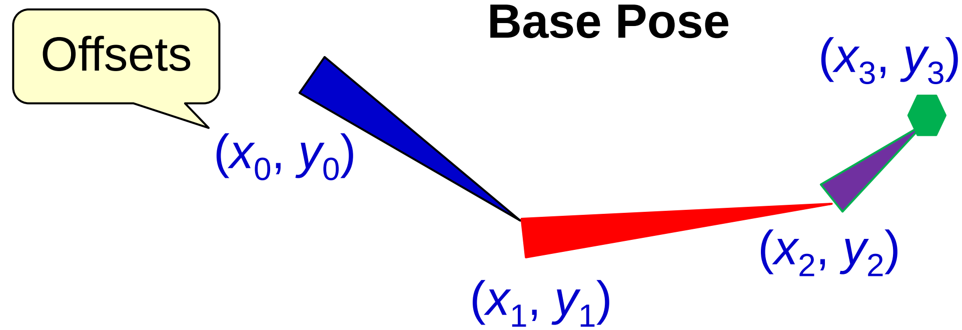


- Assimp creates a **dummy mesh** for each link of joint chains when a motion file is loaded.

```
scene = aiImportFile  ( “Test.bvh”,
aiProcessPreset_TargetRealtime_MaxQuality |
aiProcess_Debone );
```

A 2D Skeleton

The node hierarchy represents the mesh structure of the skeleton



Scene

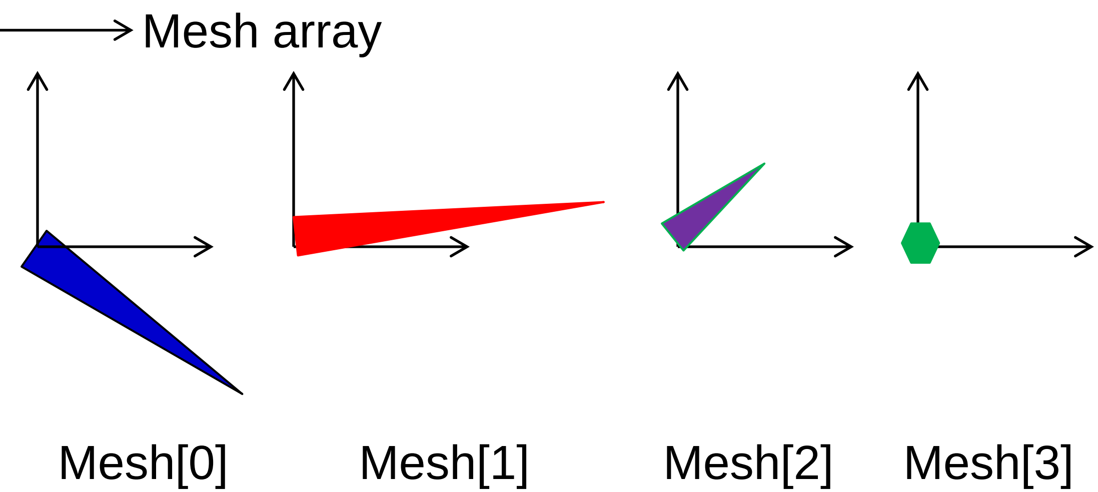
Root Node

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

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

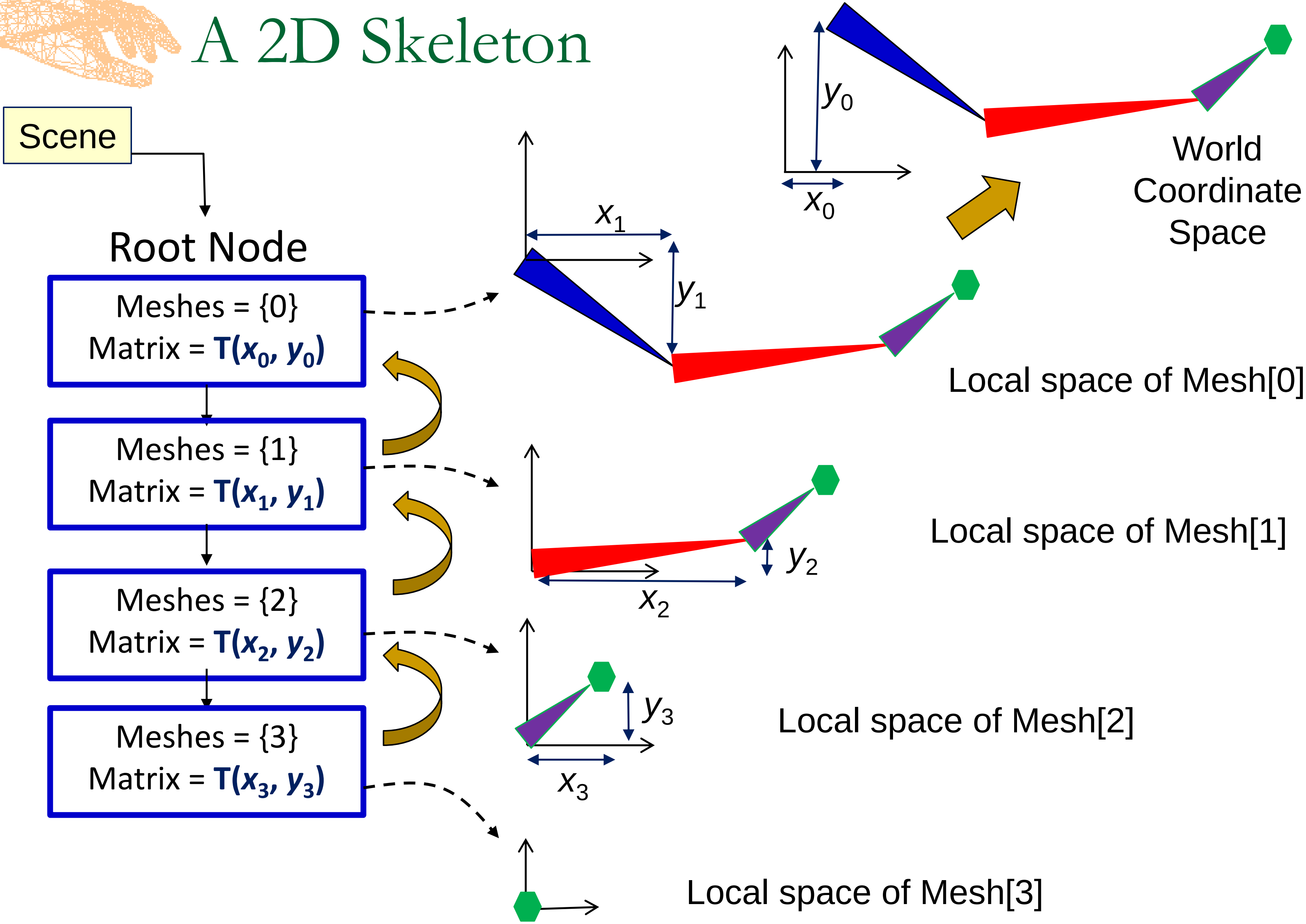
Meshes = {2}
Matrix = $\mathbf{T}(x_2, y_2)$

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



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

A 2D Skeleton



A 2D Skeleton

Scene

Root Node

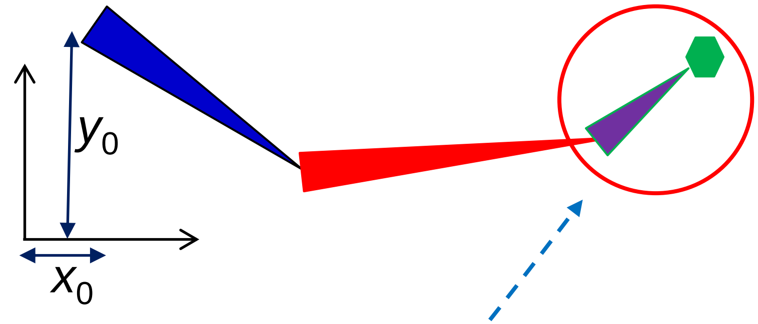
Meshes = {0}
Matrix = **T(x_0, y_0)**

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

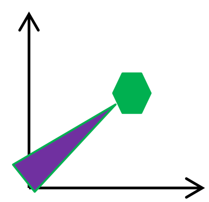
Meshes = {2}
Matrix = **T(x_2, y_2)**

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

World Coordinate Space



Transformation:
T(x_0, y_0) T(x_1, y_1)) T(x_2, y_2)



Local space of Mesh[2]

Animating a 2D Skeleton

Each mesh is moved by transformations defined using animatable parameters:

Root Node

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

↓

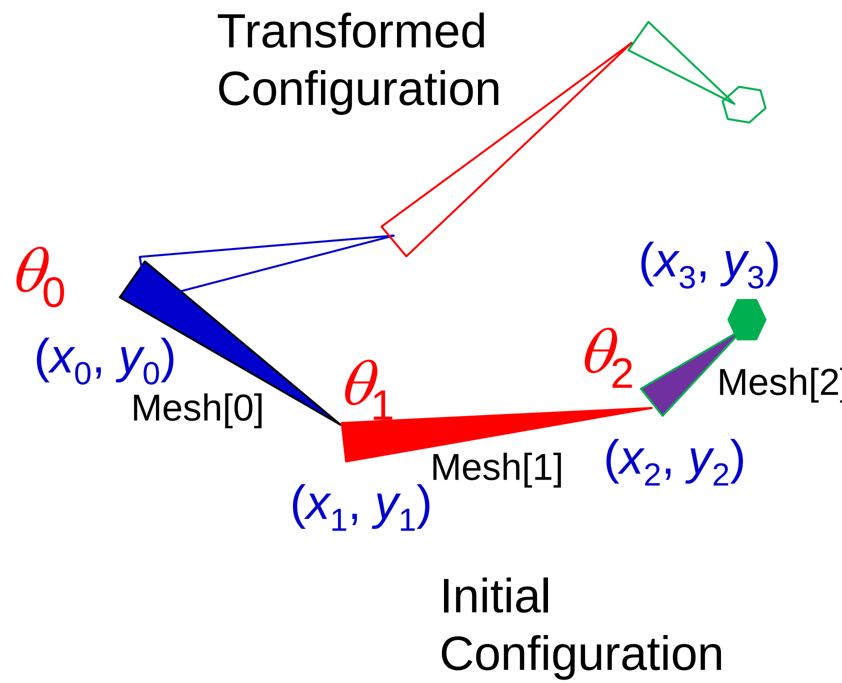
Meshes = {1}
Matrix = $\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)$

↓

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

↓

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



Animating a 2D Skeleton

Scene

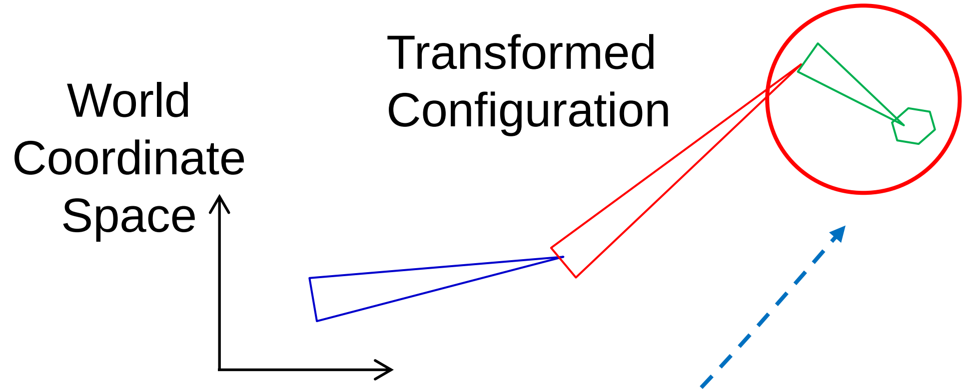
Root Node

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

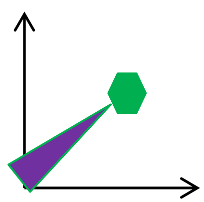
Meshes = {1}
Matrix = $\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)$

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

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



Transformation:
$\mathbf{T}(x_0, y_0)\mathbf{R}(\theta_0)\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)\ \mathbf{T}(x_2, y_2)\ \mathbf{R}(\theta_2)$



Local space of Mesh[2]

Global Position of the End Point

Scene

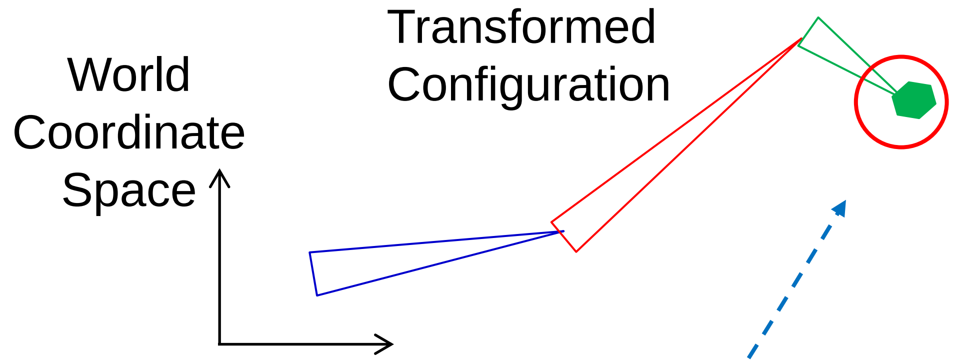
Root Node

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

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

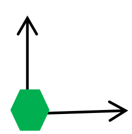
Meshes = {2}
Matrix = $\mathbf{T}(x_2, y_2)\mathbf{R}(\theta_2)$

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



Transformation:

$$\mathbf{T}(x_0, y_0)\mathbf{R}(\theta_0)\mathbf{T}(x_1, y_1)\ \mathbf{R}(\theta_1)\ \mathbf{T}(x_2, y_2)\ \mathbf{R}(\theta_2)\ \mathbf{T}(x_3, y_3)\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

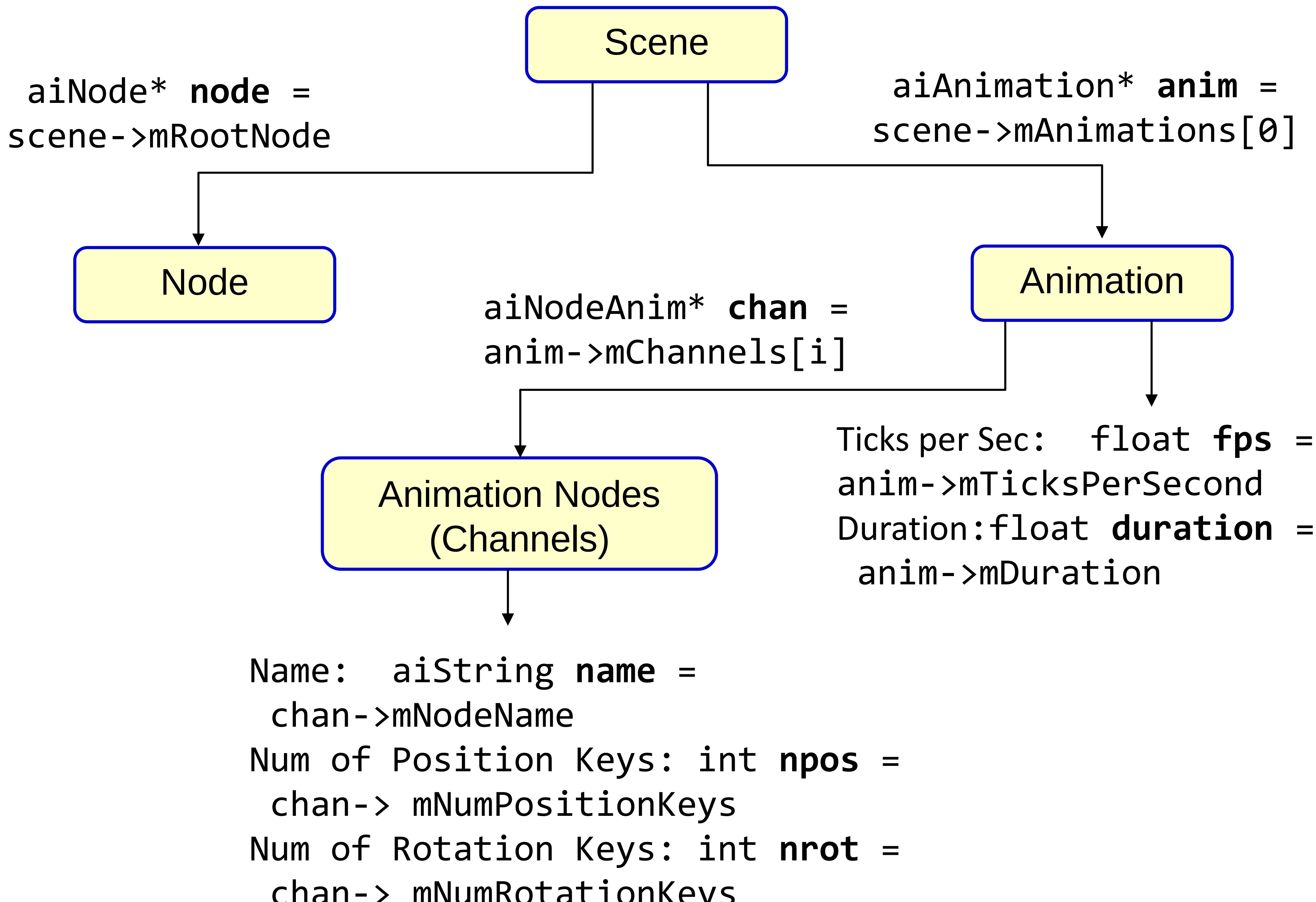


Local space of Mesh[3]

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.

Animation Classes



Assimp Output for **Test.bvh**

- Number of animations (scene->mNumAnimations): 1
- Animation object (anim): scene->mAnimations[0]
- Number of channels (anim->mNumChannels): 4
- Animation duration in ticks (anim->mDuration): 9
- Number of ticks per sec (anim->mTicksPerSecond): 10

  anim->mChannels[0]->mNodeName: root

  anim->mChannels[3]->mNodeName: wrist

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

MOTION

Frames: 10

Frame Time: .1

	Shoulder	Elbow	Wrist
0 0 0 0 0 0	3 0 0	8 0 0	-3 0 0
0 0 0 0 0 0	6 0 0	16 0 0	-6 0 0
0 0 0 0 0 0	9 0 0	24 0 0	-9 0 0
0 0 0 0 0 0	12 0 0	32 0 0	-12 0 0
0 0 0 0 0 0	15 0 0	40 0 0	-15 0 0
0 0 0 0 0 0	18 0 0	48 0 0	-18 0 0
0 0 0 0 0 0	21 0 0	56 0 0	-21 0 0
0 0 0 0 0 0	24 0 0	64 0 0	-24 0 0
0 0 0 0 0 0	27 0 0	72 0 0	-27 0 0
0 0 0 0 0 0	30 0 0	80 0 0	-30 0 0

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

<u>Assimp Output</u>

Channel: 0 Name: root
posKey 0: Value = 0 0 0
rotnKey 0: Value = 1 0 0 0
Channel: 1 Name: Shoulder
posKey 0: Value = 0.5 0.3 0
rotnKey 0: Value = 0.99 0 0 0.026
...
rotnKey 9: Value = 0.96 0 0 0.258
Channel 2: Name = Elbow
posKey 0: Value = 1.0 -0.5 0
rotnKey 0: Value = 0.99 0 0 0.069
...
rotnKey 9: Value = 0.76 0 0 0.642
Channel 3: Name = Wrist
posKey 0: Value = 1.0 -0.5 0
rotnKey 0: Value = 0.99 0 0 -0.026
...
rotnKey 9: Value = 0.96 0 0 -0.258

Position and Rotation Keys

- Number of position keys of the first channel: 1
- Number of rotation keys of the first channel: 1
- 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 corresponding to the joint's offset value.

Base Pose

When a BVH motion file is loaded, Assimp generates a node tree with transformation matrices constructed using only the offset values of each joint. This initial configuration corresponds to the base pose where all joint angles are zero.

Root

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

↓

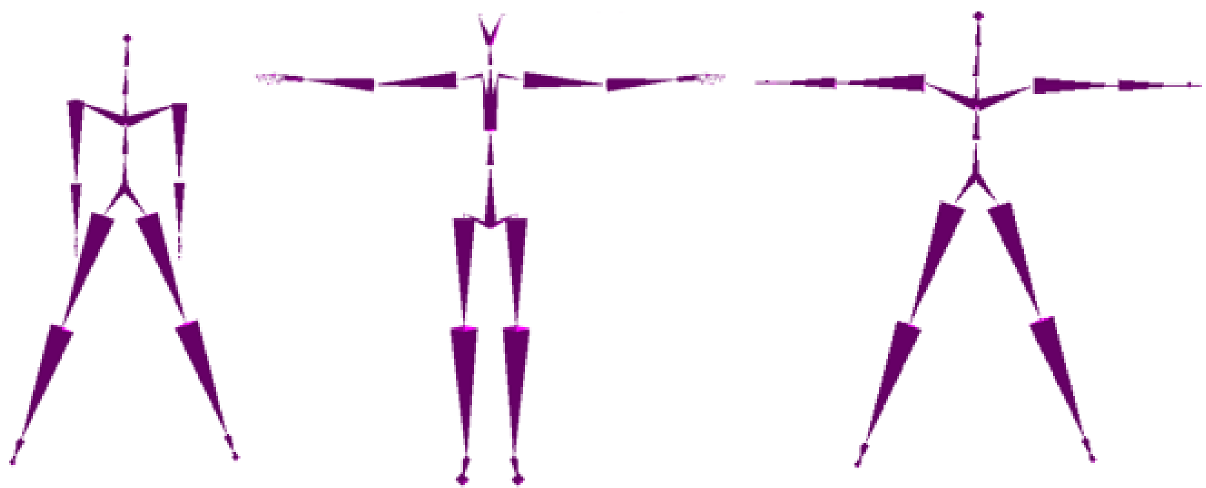
Meshes = {1}
Matrix = $\mathbf{T}(x_1, y_1)$

↓

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

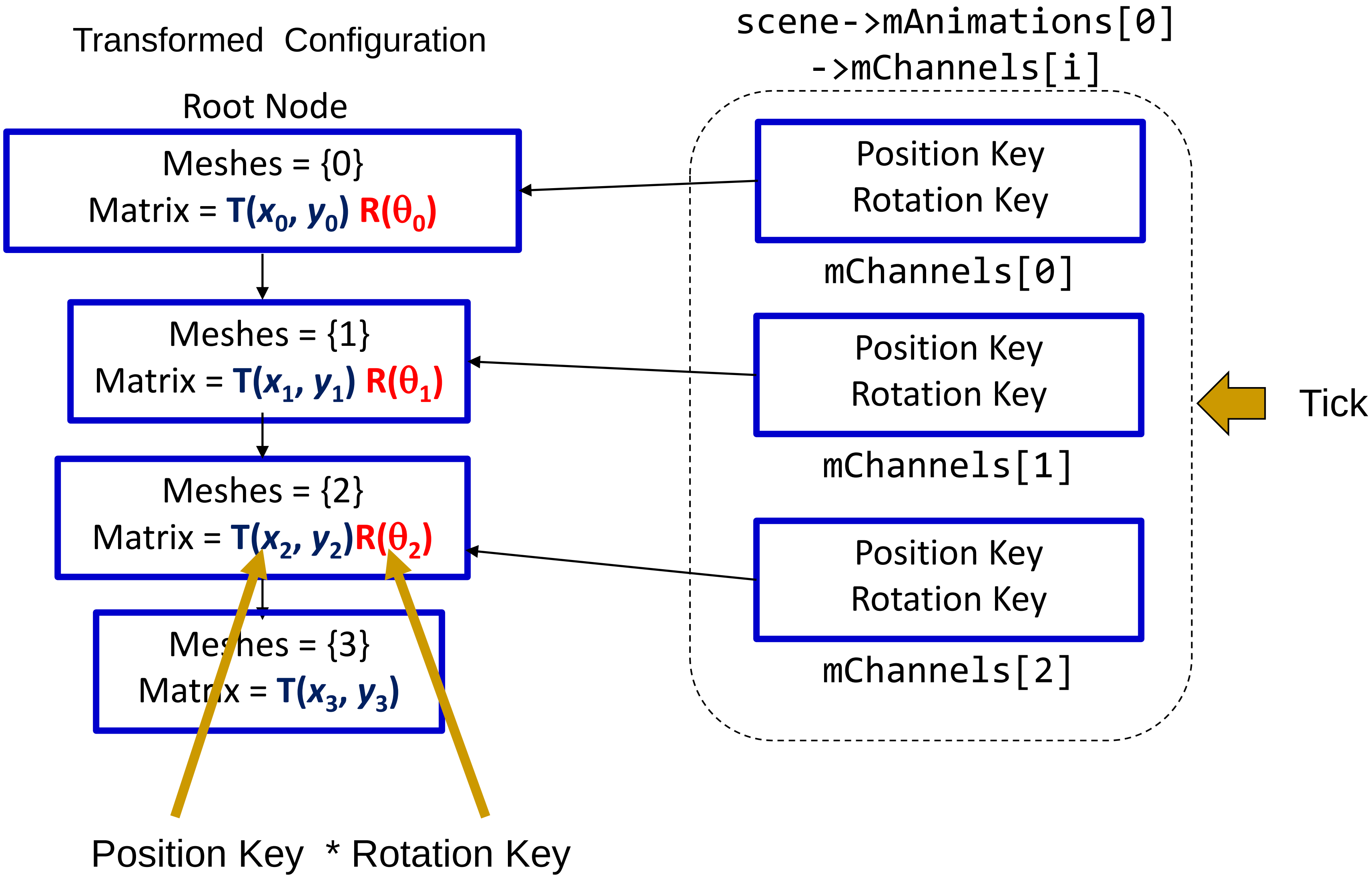
↓

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



3DS Max   DAZ-Studio   Motion Builder

Animating a Skeleton



Animation Using BVH Data

- Use a timer call-back function for animation:

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

- The value of `timeStep` may be computed based on the desired framerate (mTicksPerSec). See Slide 27.

- The timer callback function increments “tick” from 0 to “Duration”

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

Updating Node Matrices

- For each channel *i*, get the values of position and rotation keys, corresponding to the current value of "tick:

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

aiVector3D

aiQuaternion

Check if the array has a size > 1

- Note: For some channels, the number of position keys (`chnl->mNumPositionKeys`) or the number of rotation keys (`chnl->mNumRotationKeys`) may have a value 1.

Updating Node Matrices

- 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. See Slide 32

```
node->mTransformation = matprod;
```

Human Model Skeletons

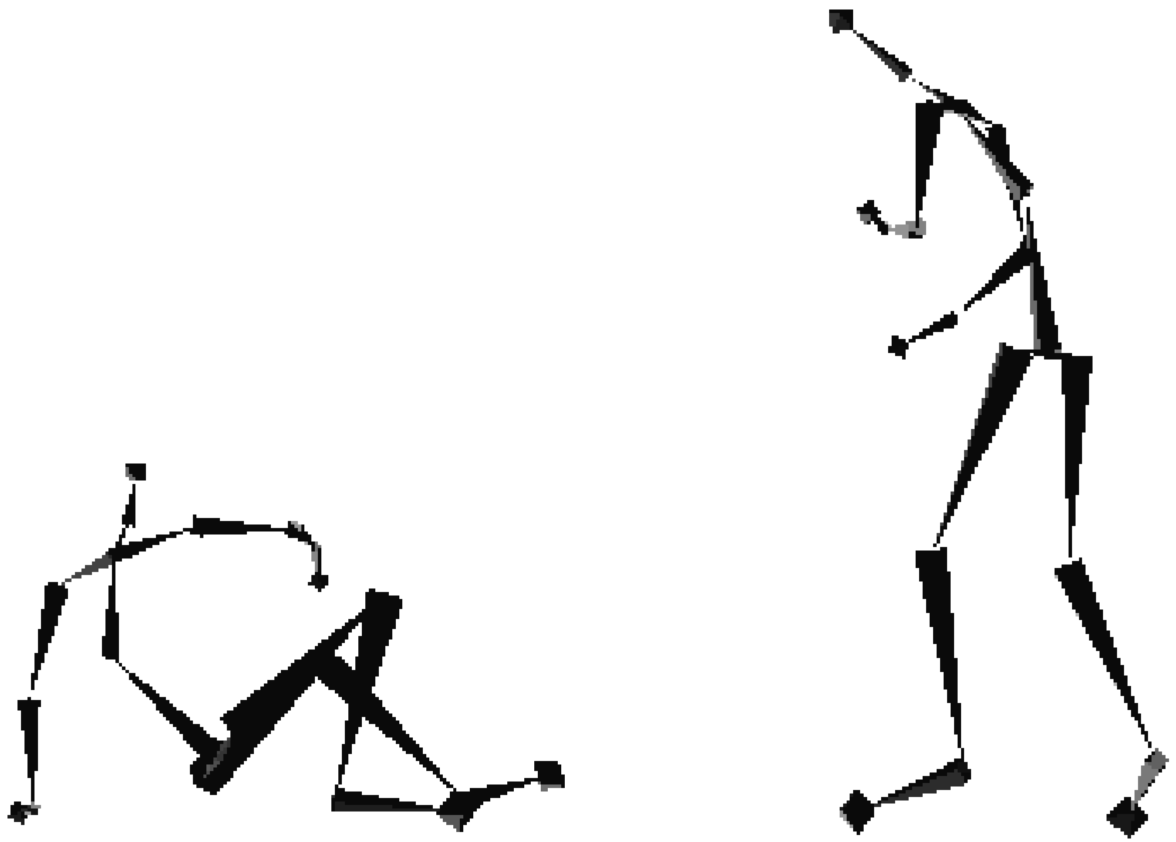
- BVH Repository:

https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture

This page contains links and information about Motion Capture software and datasets.

BVH conversions of the 2500-motion Carnegie-Mellon motion capture dataset:

1. Daz-friendly version (released July 2010, by B. Hahne)
2. 3dsMax-friendly version (released May 2009, by B. Hahne)
3. MotionBuilder-friendly version (released July 2008, by B. Hahne)



Boxing.bvh