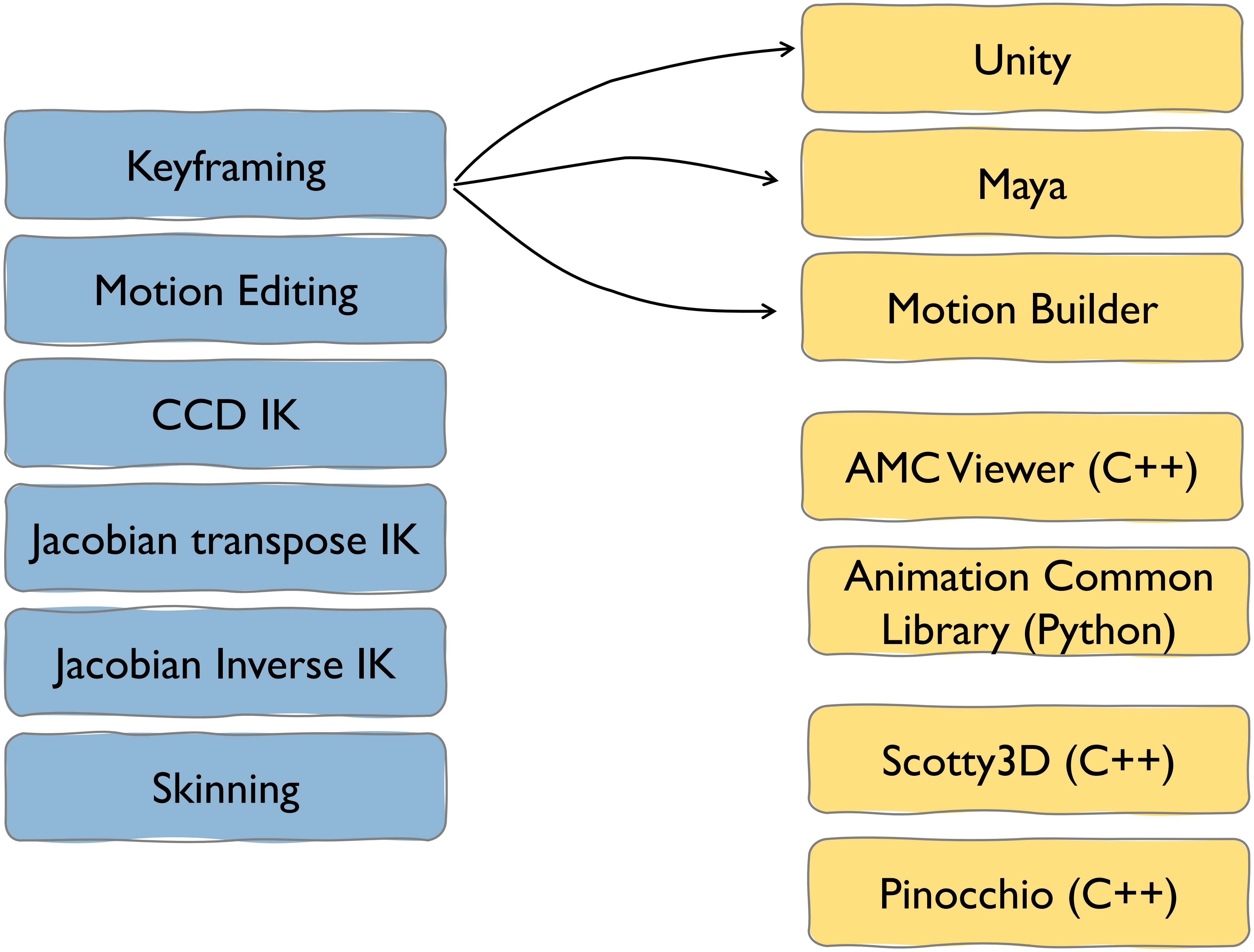
Mini Project I

Yanzhe Yang
Jan 29, 2020

Instructions and basic requirement

- http://graphics.cs.cmu.edu/nsp/course/15464-s20/www/assignments/miniProject1.htm

- Due Feb 17, 2020 at start of class



Keyframing

Motion Editing

CCD IK

Jacobian transpose IK

Jacobian Inverse IK

Skinning

Unity

Maya

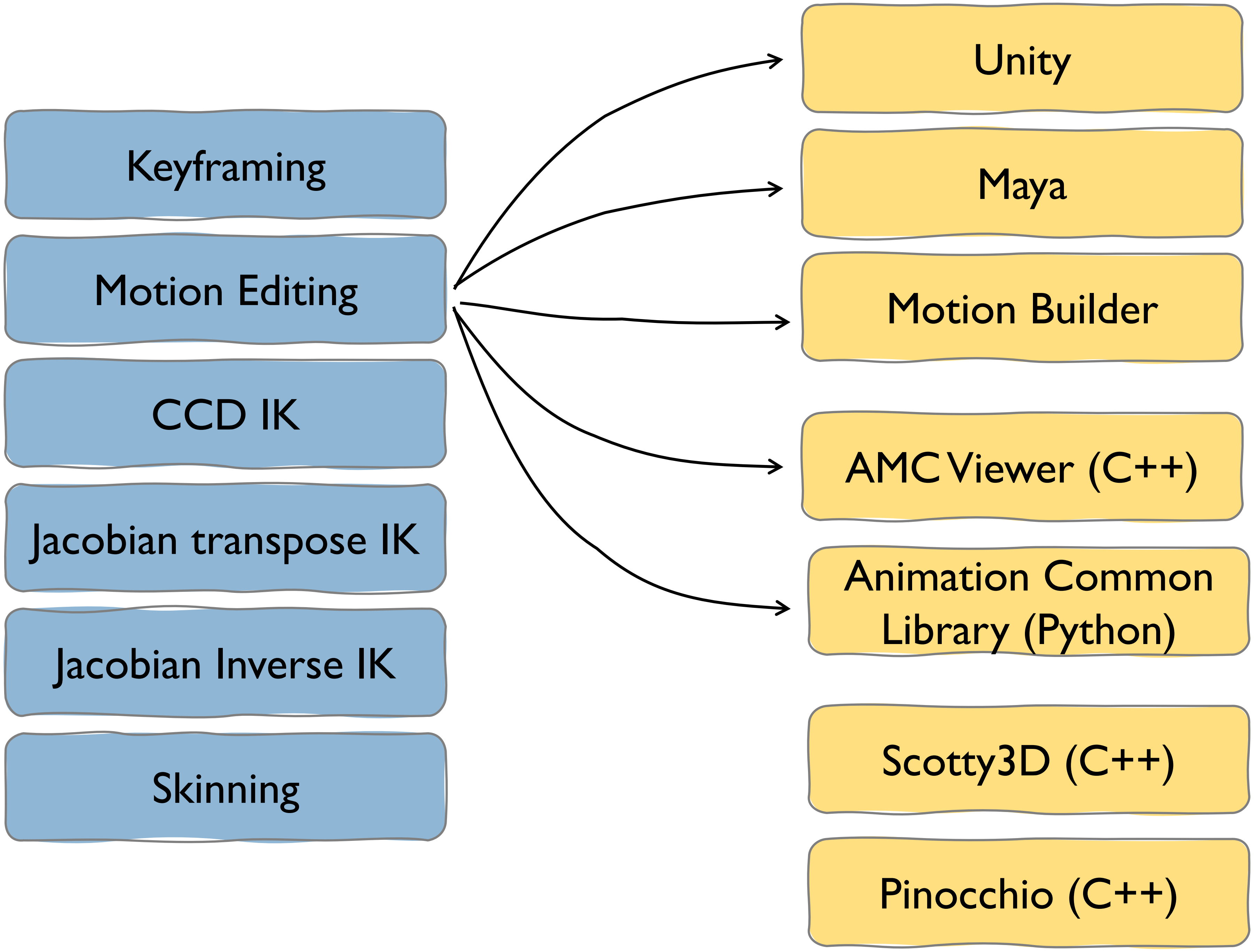
Motion Builder

AMC Viewer (C++)

Animation Common Library (Python)

Scotty3D (C++)

Pinocchio (C++)



Keyframing

Motion Editing

CCD IK

Jacobian transpose IK

Jacobian Inverse IK

Skinning

Unity

Maya

Motion Builder

AMC Viewer (C++)

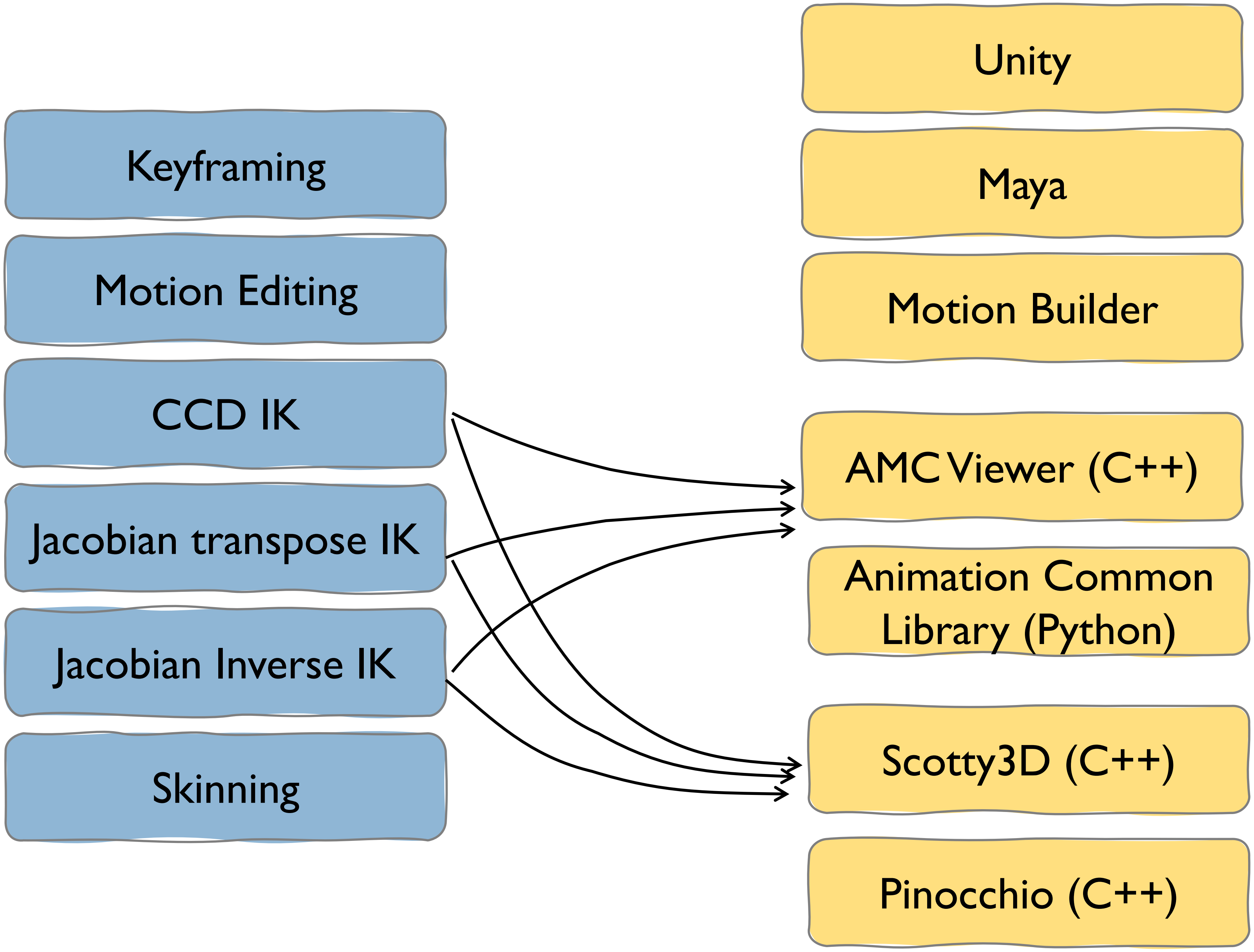
Read ASF/AMC file
View Mocap in interface

Animation Common Library (Python)

Load and save BVH/FBX file
Save into mp4 files

Scotty3D (C++)

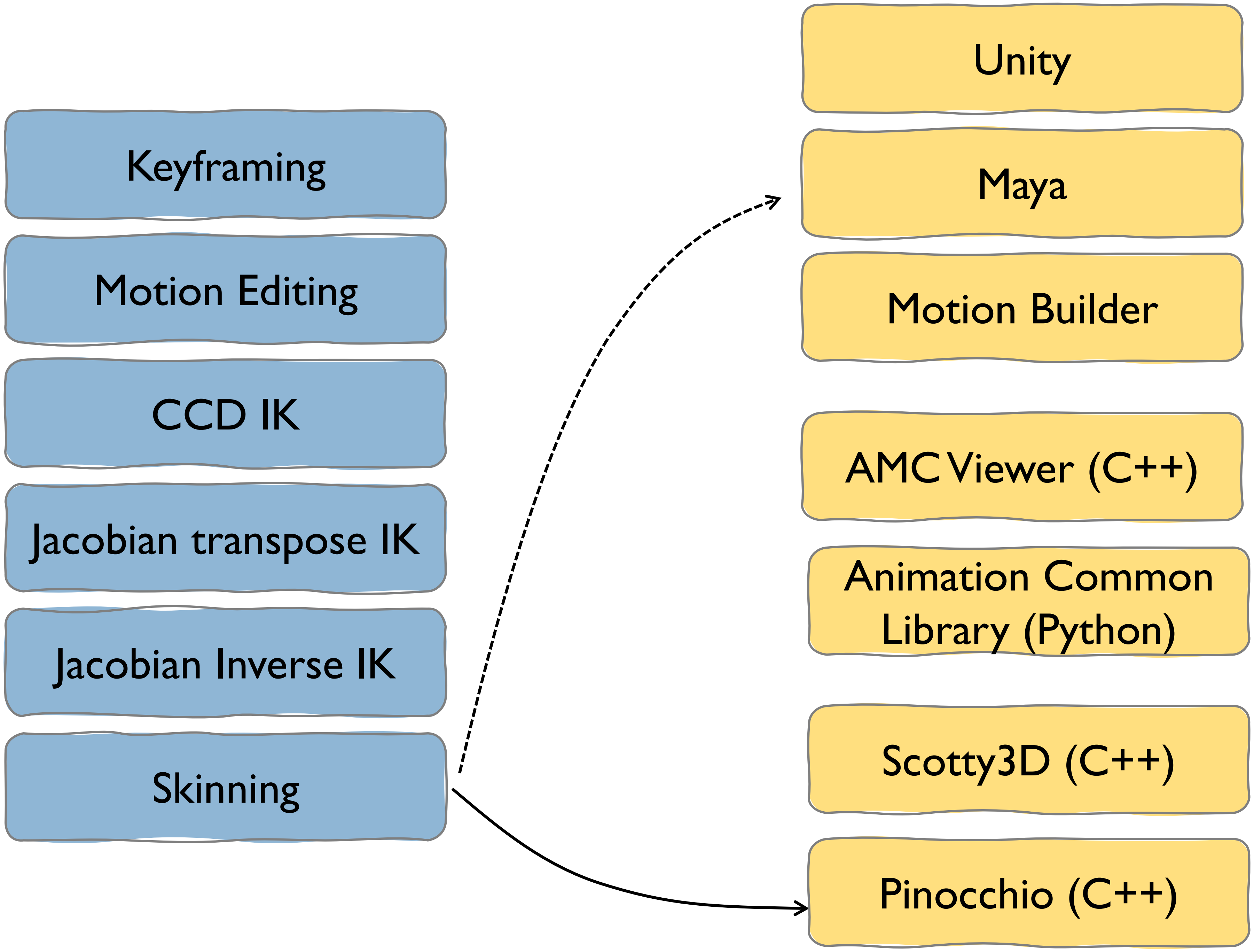
Pinocchio (C++)



Pro: Need compiling
Cons: Live interface

Pro: Easy to setup
Cons: No live interface

Check 15-462/662



Keyframing

Motion Editing

CCD IK

Jacobian transpose IK

Jacobian Inverse IK

Skinning

Unity

Maya

Motion Builder

AMC Viewer (C++)

Animation Common Library (Python)

Scotty3D (C++)

Pinocchio (C++)

Outline

- Animation representation

  Data structure for hierarchical modeling

- File formats

  How to read/load files?

  Some popular formats: BVH, FBX, and ASF/AMC

- Introduction to start code

- A quick introduction to Unity and Motion Builder

Outline

- Animation representation

  Data structure for hierarchical modeling

- File formats

  How to read/load files?

  Some popular formats: BVH, FBX, and ASF/AMC

- Introduction to start code

- A quick introduction to Unity and Motion Builder

Human figures are modeled as hierarchical linkages

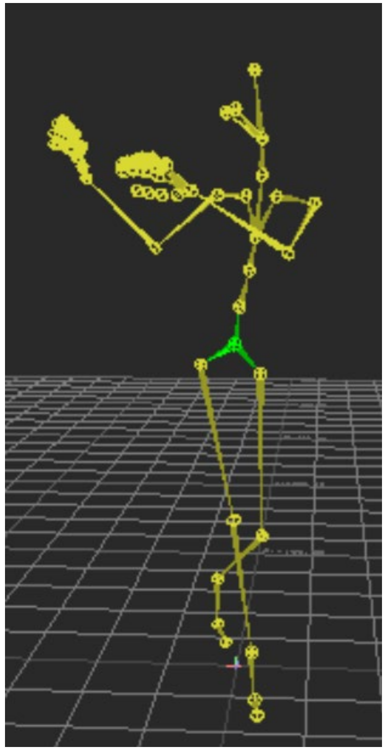
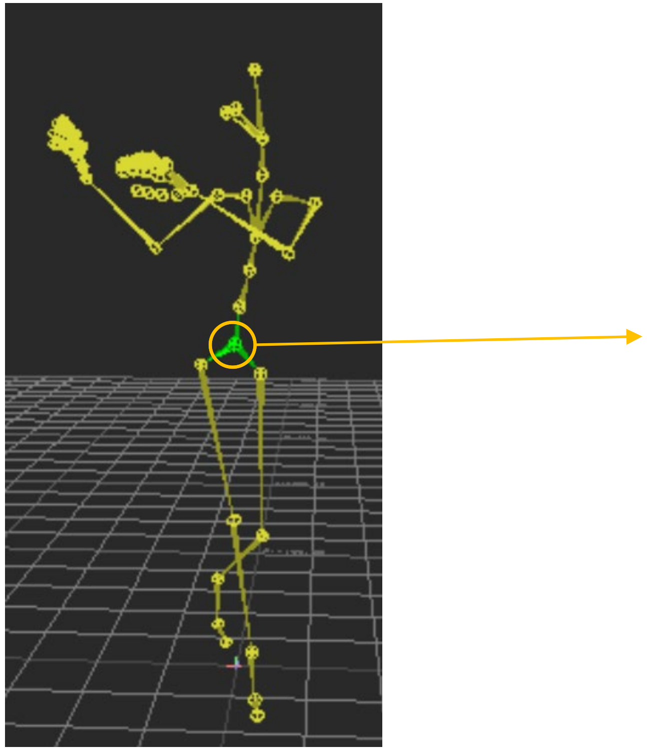


Figure 1. Example of a human figure

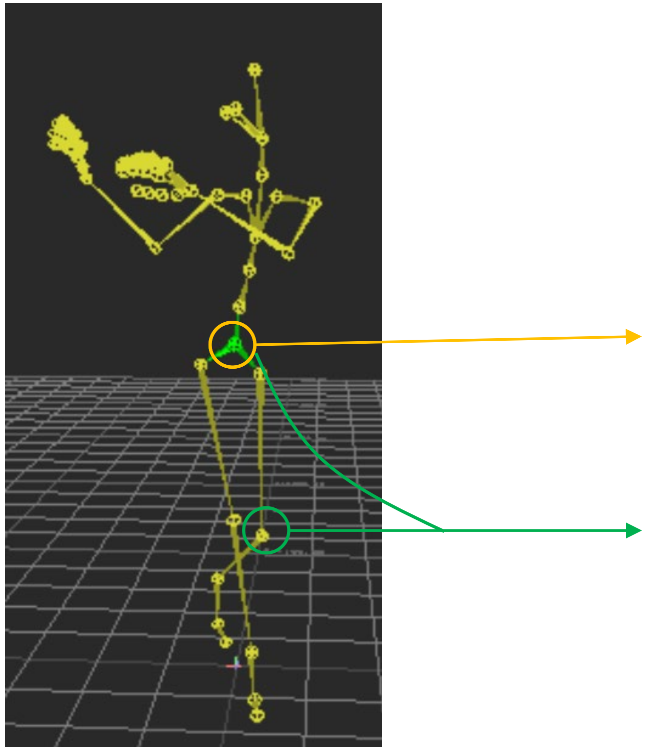
Human figures are modeled as hierarchical linkages



Root node (6 DOF):
Translation of the root repositions the whole structure

Figure 1. Example of a human figure

Human figures are modeled as hierarchical linkages

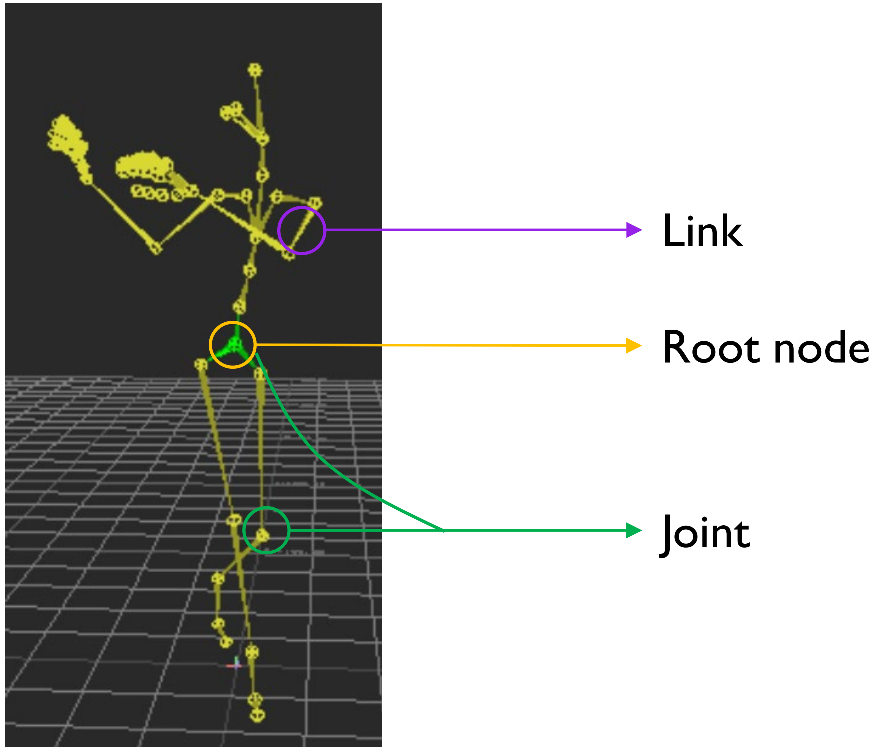


Root node (6 DOF):
Translation of the root repositions the whole structure

Joint (3 DOF or 6 DOF):
Transformations are relative to its parent joint.
For rigid body, joint translation is constant when relative to its parent joint.

Figure 1. Example of a human figure

Joint hierarchy is a tree structure

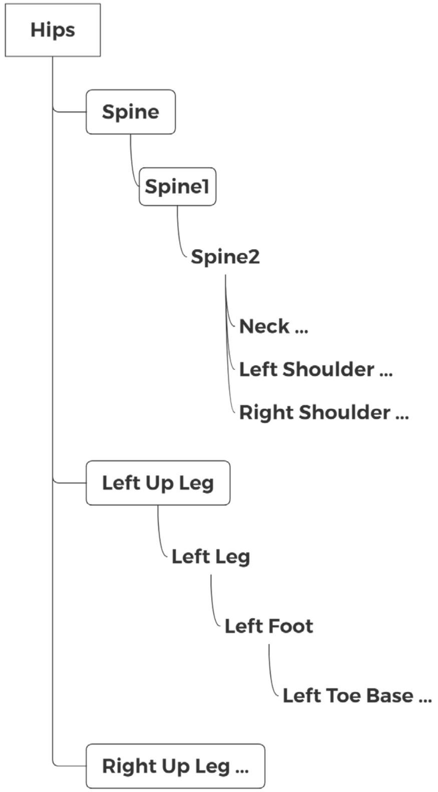


Link

Root node

Joint

Figure 1. Example of a human figure

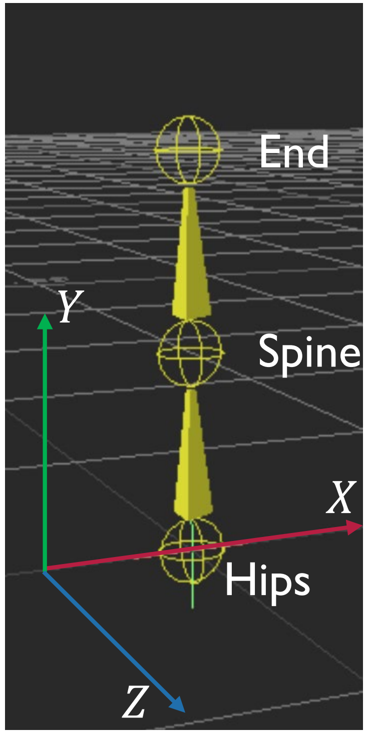


- Hips
  - Spine
    - Spine1
      - Spine2
        - Neck ...
        - Left Shoulder ...
        - Right Shoulder ...
  - Left Up Leg
    - Left Leg
      - Left Foot
        - Left Toe Base ...
  - Right Up Leg ...

Figure 2. Example of a tree structure.

How to represent a character animation?

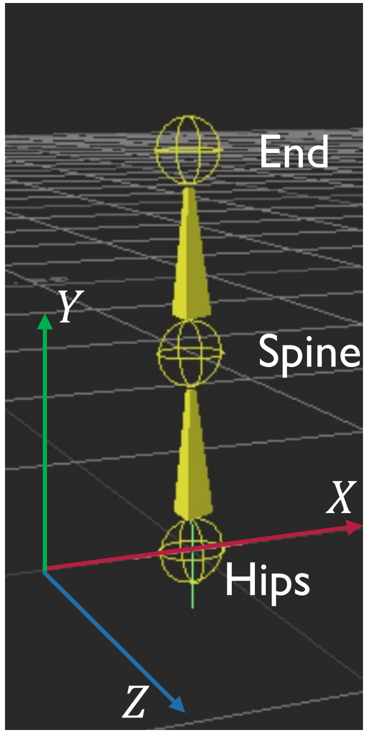
- Consider a simple example



Coordinate system:Y-up and right-hand

How to represent a character animation?

- Consider a simple example



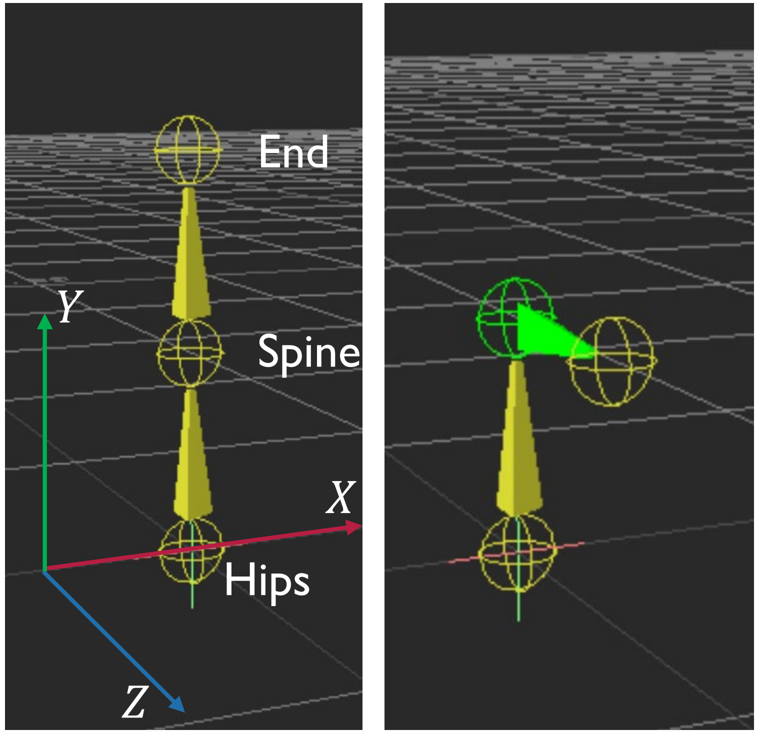
Coordinate system:Y-up and right-hand

Hierarchy:

- Joint names
- Parent-child relationship
- Constant translation and orientations for base pose
  (e.g. $T_0 = [0, 0, 0]^T, T_1 = [0, 10, 0]^T, T_2 = [0, 10, 0]^T,$
  $O_0 = O_1 = O_2 = I$)

How to represent a character animation?

- Consider a simple example



Coordinate system: Y-up and right-hand

Hierarchy:

- Joint names
- Parent-child relationship
- Constant translation and orientations for base pose (e.g. $T_0 = [0,0,0]^T, T_1 = [0,10,0]^T, T_2 = [0,10,0]^T,$ $O_0 = O_1 = O_2 = I$)

Motion:

- Joint translations and orientations at each time frame (e.g. $R_{x,j=J_1}(t_0) = 0^o, R_{x,j=J_1}(t_1) = 90^o$)

Outline

- Animation representation

  Data structure for hierarchical modeling

- File formats

  How to read/load files?

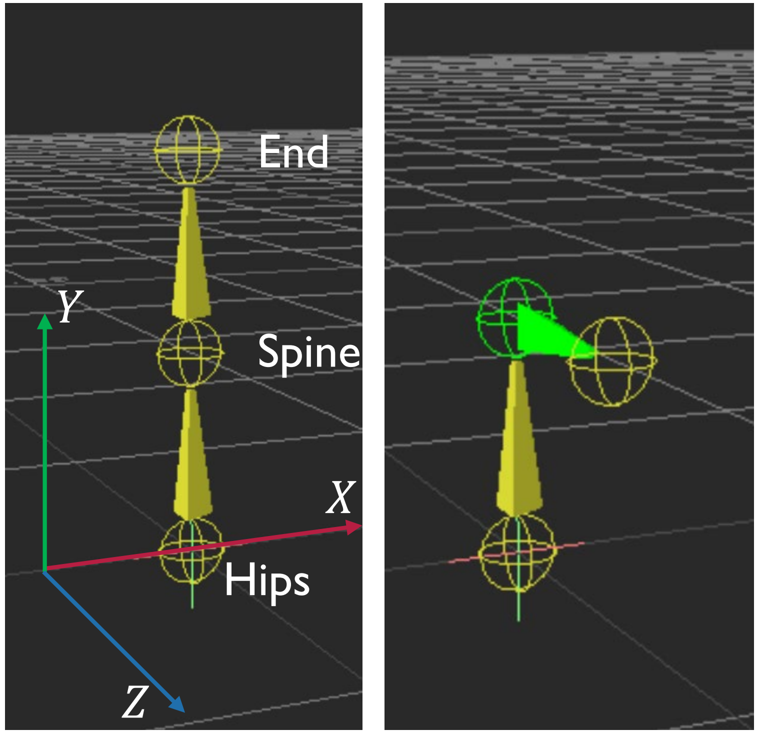
  Some popular formats: BVH, FBX, and ASF/AMC

- Introduction to start code

- A quick introduction to Unity and Motion Builder

Animation is represented by hierarchy and motion

- Consider a simple example



Coordinate system: Y-up and right-hand

Hierarchy:

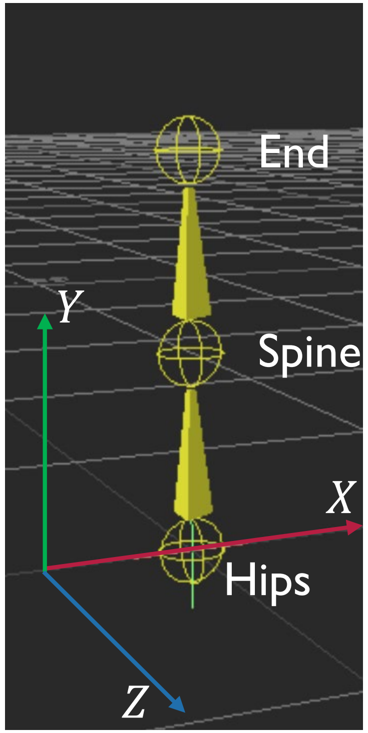
- Joint names
- Parent-child relationship
- Constant translation and orientations
  (e.g. $T_0 = [0,0,0]^T, T_1 = [0,10,0]^T, T_2 = [0,10,0]^T,$
  $O_0 = O_1 = O_2 = I$)

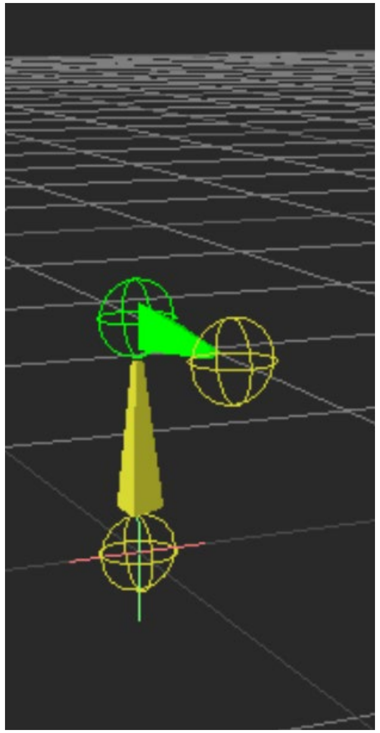
Motion:

- Joint translations and orientations at each time frame
  (e.g. $R_{x,j=J_1}(t_0) = 0^o, R_{x,j=J_1}(t_1) = 90^o$)

BVH files

- Consider a simple example

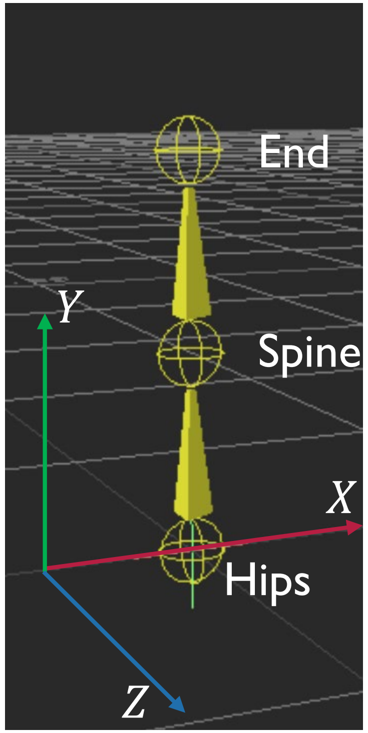


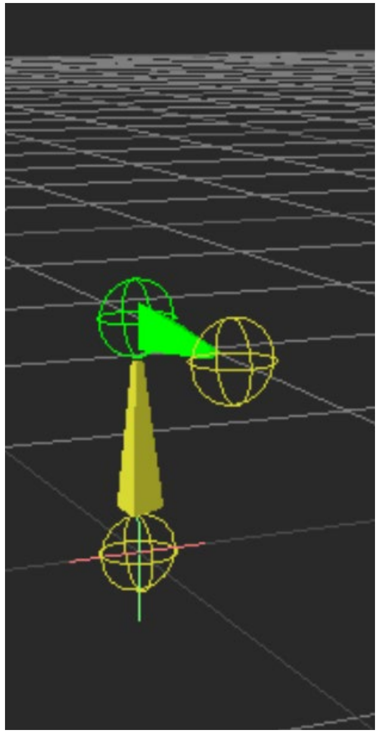


```
HIERARCHY
ROOT Hips
{
    OFFSET 0.000000 0.000000 0.000000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT Spine
    {
        OFFSET 0.000000 10.000000 0.000000
        CHANNELS 3 Zrotation Yrotation Xrotation
        End Site
        {
            OFFSET 0.000000 10.000000 0.000000
        }
    }
}
MOTION
Frames: 2
Frame Time: 0.033333
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 90
```

BVH files

- Consider a simple example

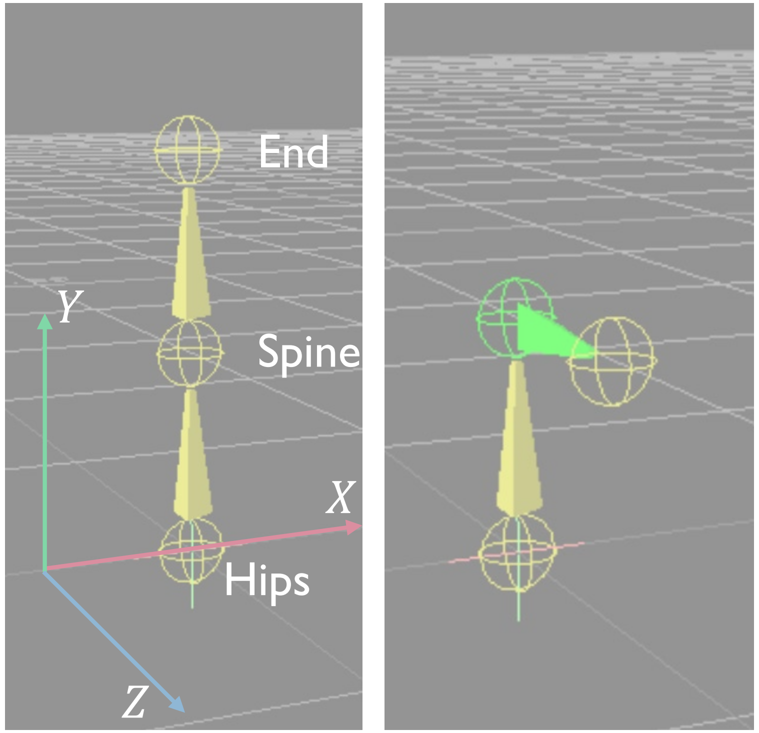




```
HIERARCHY
ROOT Hips
{
    OFFSET 0.000000 0.000000 0.000000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT Spine
    {
        OFFSET 0.000000 10.000000 0.000000
        CHANNELS 3 Zrotation Yrotation Xrotation
        End Site
        {
            OFFSET 0.000000 10.000000 0.000000
        }
    }
}
MOTION
Frames: 2
Frame Time: 0.033333
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 90
```

Channel order indicates the order of transformation

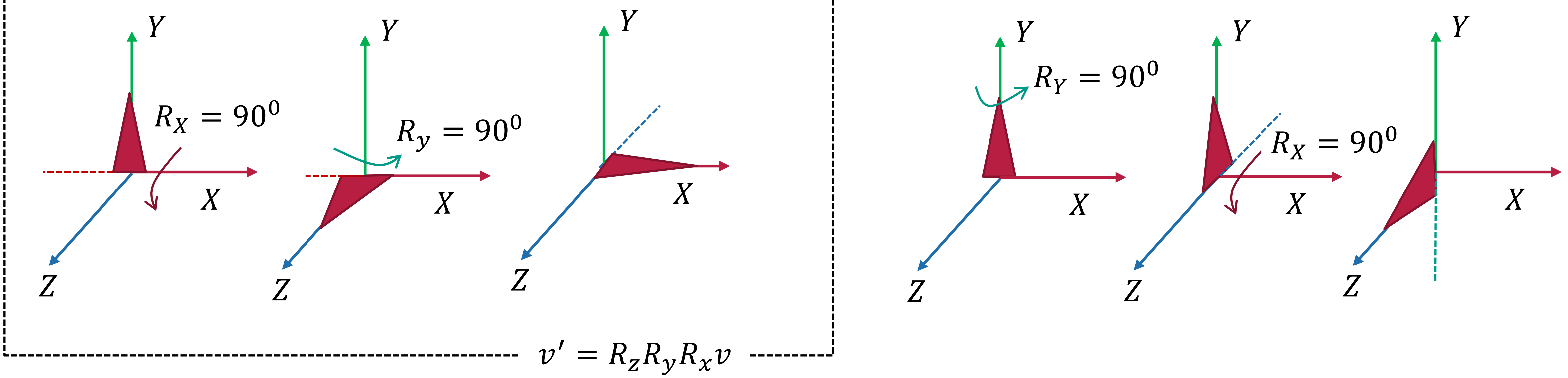
- $v' = R_z R_y R_x v$



```
HIERARCHY
ROOT Hips
{
    OFFSET 0.000000 0.000000 0.000000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT Spine
    {
        OFFSET 0.000000 10.000000 0.000000
        CHANNELS 3 Zrotation Yrotation Xrotation
        End Site
        {
            OFFSET 0.000000 10.000000 0.000000
        }
    }
}
MOTION
Frames: 2
Frame Time: 0.033333
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 90
```

Order of transformation is important to Euler angle

- Matrix multiplication is not commutative
- Example

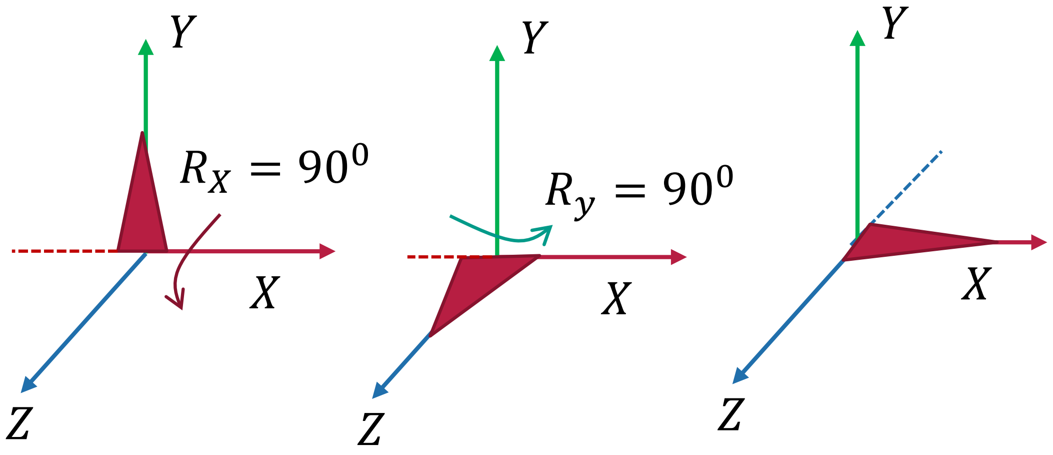


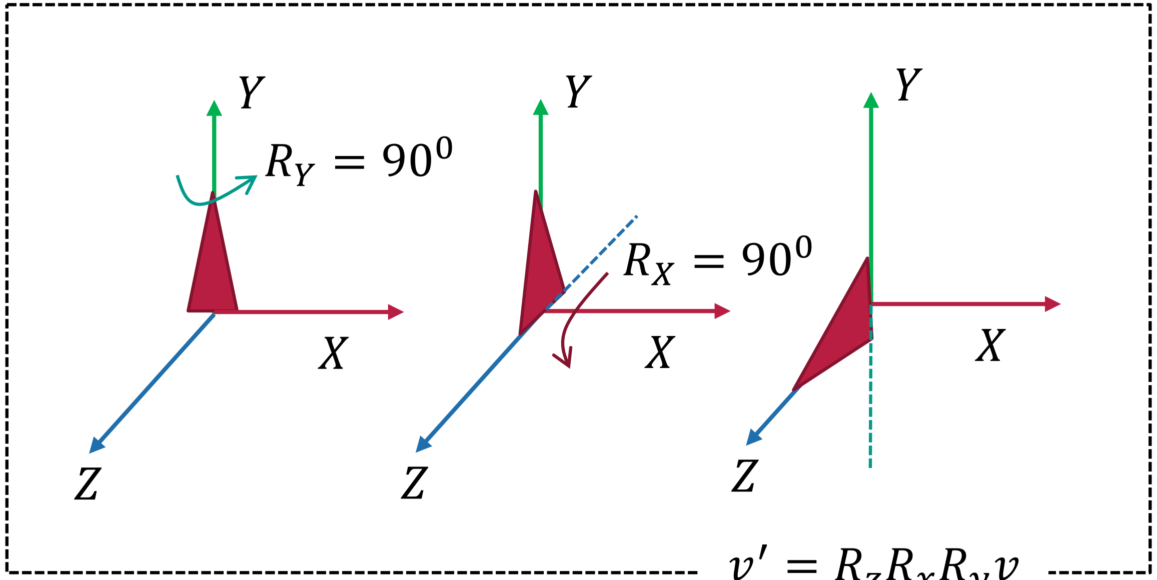
$v' = R_z R_y R_x v$

(Extrinsic rotations)

Order of transformation is important to Euler angle

- Matrix multiplication is not commutative
- Example





$v' = R_z R_x R_y v$

(Extrinsic rotations)

Other file formats

- FBX
  - A proprietary format owned by Autodesk.
  - Not human readable, needs to parse data using SDK (http://help.autodesk.com/view/FBX/2019/ENU/) or Maya scripts
  - Able to store more complete information, like pre and post rotations, mesh, lighting, skinning and et al.

- ASF/AMC
  - Similar to BVH but stores hierarchy (i.e. skeleton) and motion in separate files
  - ASF files store hierarchy
  - AMC files store motion

Outline

- Animation representation

  Data structure for hierarchical modeling

- File formats

  How to read/load files?

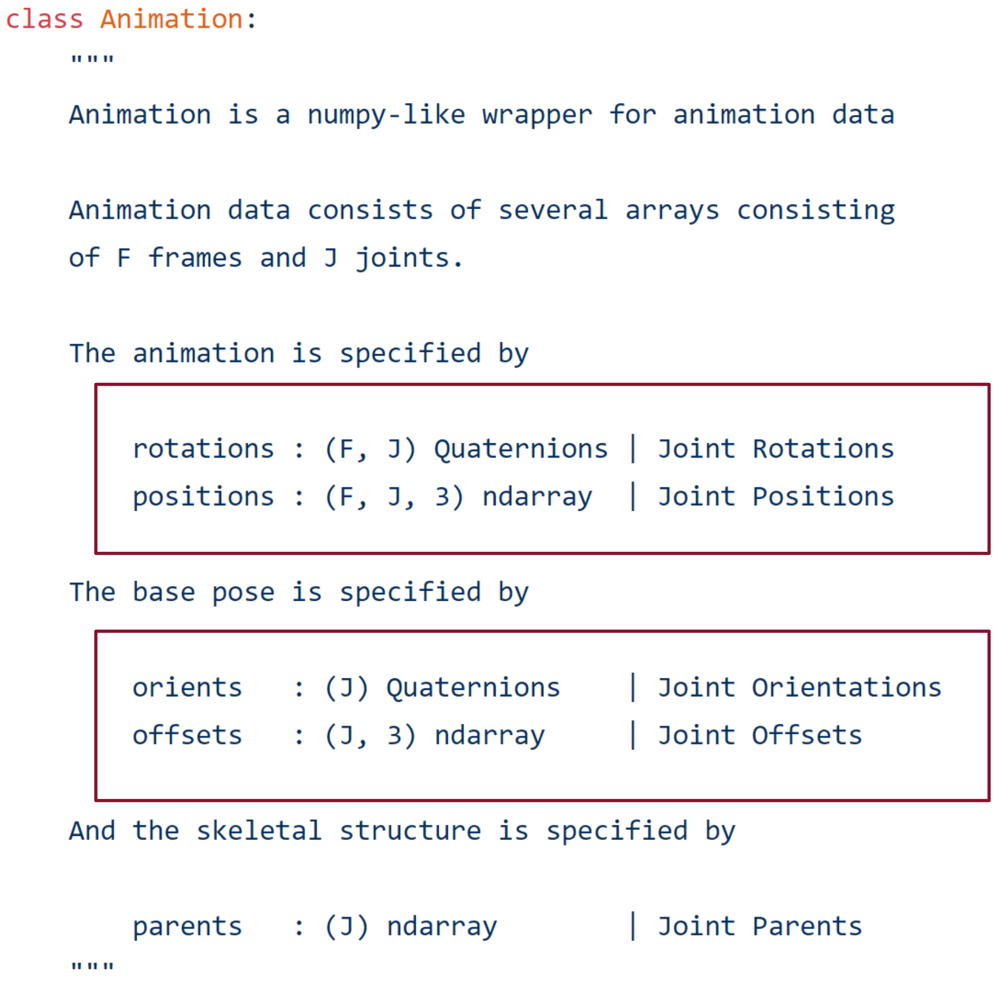
  Some popular formats: BVH, FBX, and ASF/AMC

- Introduction to start code

  https://github.com/yangyanzhe/assignment1

- A quick introduction to Unity and Motion Builder

Animation class



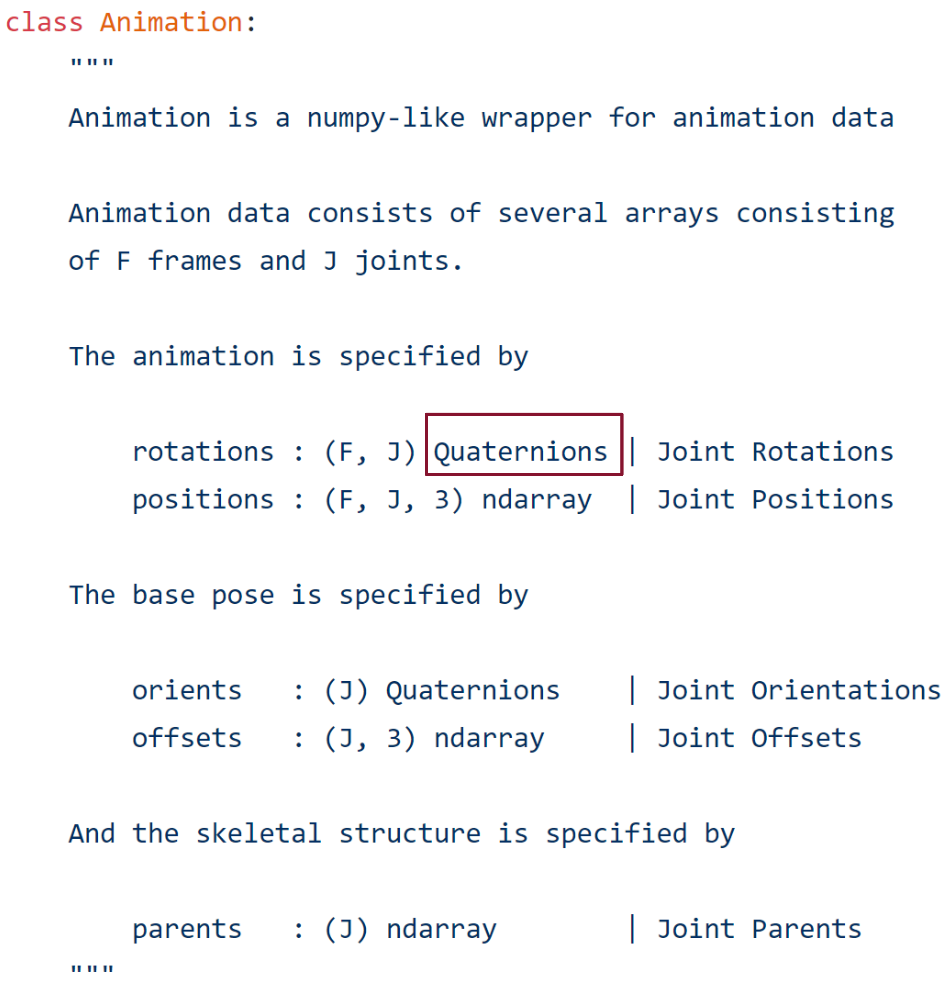
```
class Animation:
    """
    Animation is a numpy-like wrapper for animation data

    Animation data consists of several arrays consisting
    of F frames and J joints.

    The animation is specified by

        rotations : (F, J) Quaternions | Joint Rotations
        positions : (F, J, 3) ndarray  | Joint Positions

    The base pose is specified by

        orients   : (J) Quaternions    | Joint Orientations
        offsets   : (J, 3) ndarray     | Joint Offsets

    And the skeletal structure is specified by

        parents   : (J) ndarray        | Joint Parents
    """
```

Motion

Hierarchy / Skeleton

We can assume orients is identity transformation in this assignment

Angle representation: Quaternions

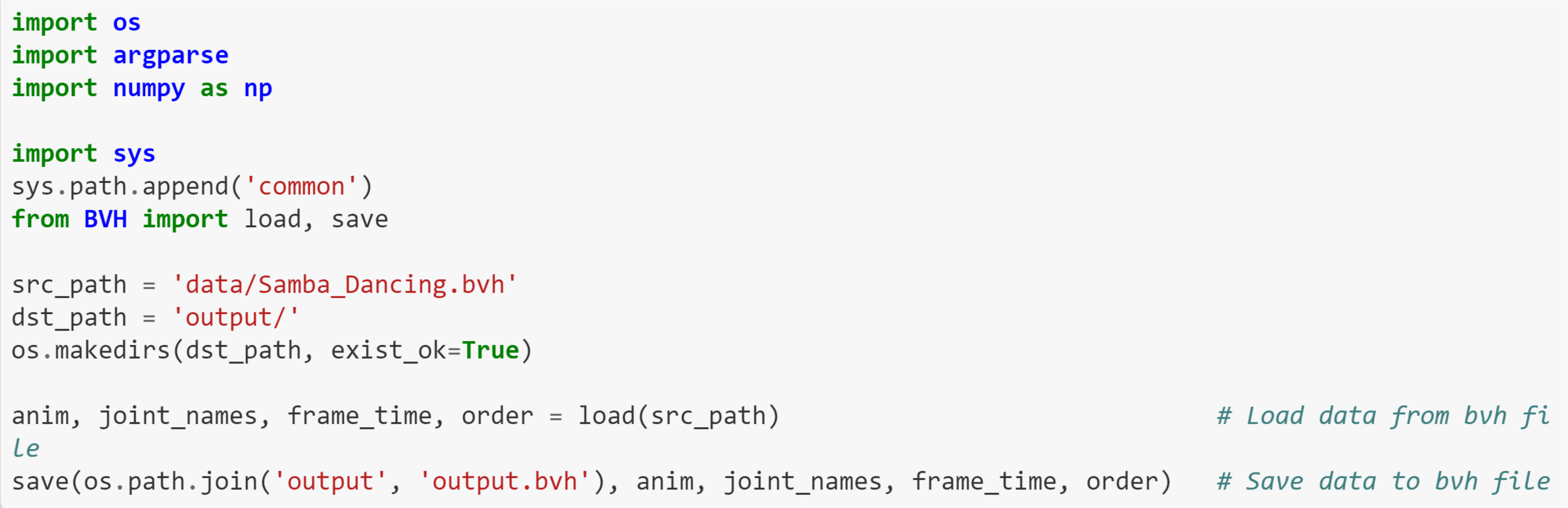


```
class Animation:
    """
    Animation is a numpy-like wrapper for animation data

    Animation data consists of several arrays consisting
    of F frames and J joints.

    The animation is specified by

        rotations : (F, J) Quaternions | Joint Rotations
        positions : (F, J, 3) ndarray  | Joint Positions

    The base pose is specified by

        orients   : (J) Quaternions    | Joint Orientations
        offsets   : (J, 3) ndarray     | Joint Offsets

    And the skeletal structure is specified by

        parents   : (J) ndarray        | Joint Parents
    """
```

Call “euler” function to get euler angle in radius

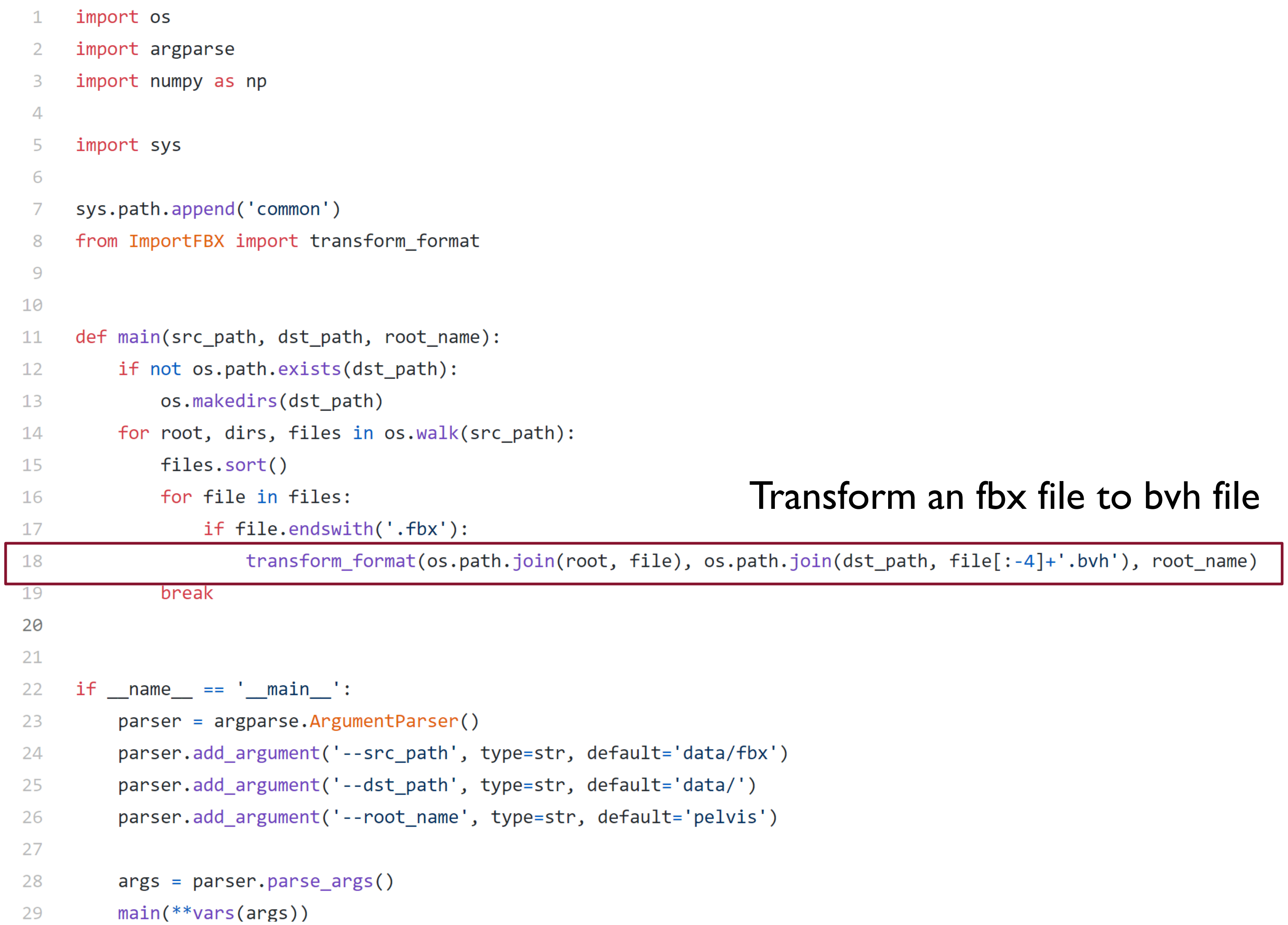
anim = Animation(…)
euler_angle = anim.rotations.euler()

Load and save animation for bvh files



```
import os
import argparse
import numpy as np

import sys
sys.path.append('common')
from BVH import load, save

src_path = 'data/Samba_Dancing.bvh'
dst_path = 'output/'
os.makedirs(dst_path, exist_ok=True)

anim, joint_names, frame_time, order = load(src_path)                              # Load data from bvh fi
le
save(os.path.join('output', 'output.bvh'), anim, joint_names, frame_time, order)   # Save data to bvh file
```

Format transformer: FBX to BVH



```python
import os
import argparse
import numpy as np

import sys

sys.path.append('common')
from ImportFBX import transform_format


def main(src_path, dst_path, root_name):
    if not os.path.exists(dst_path):
        os.makedirs(dst_path)
    for root, dirs, files in os.walk(src_path):
        files.sort()
        for file in files:
            if file.endswith('.fbx'):
                transform_format(os.path.join(root, file), os.path.join(dst_path, file[:-4]+'.bvh'), root_name)
        break


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--src_path', type=str, default='data/fbx')
    parser.add_argument('--dst_path', type=str, default='data/')
    parser.add_argument('--root_name', type=str, default='pelvis')

    args = parser.parse_args()
    main(**vars(args))
```

Transform an fbx file to bvh file

Outline

- Animation representation

  Data structure for hierarchical modeling

- File formats

  How to read/load files?

  Some popular formats: BVH, FBX, and ASF/AMC

- Introduction to start code

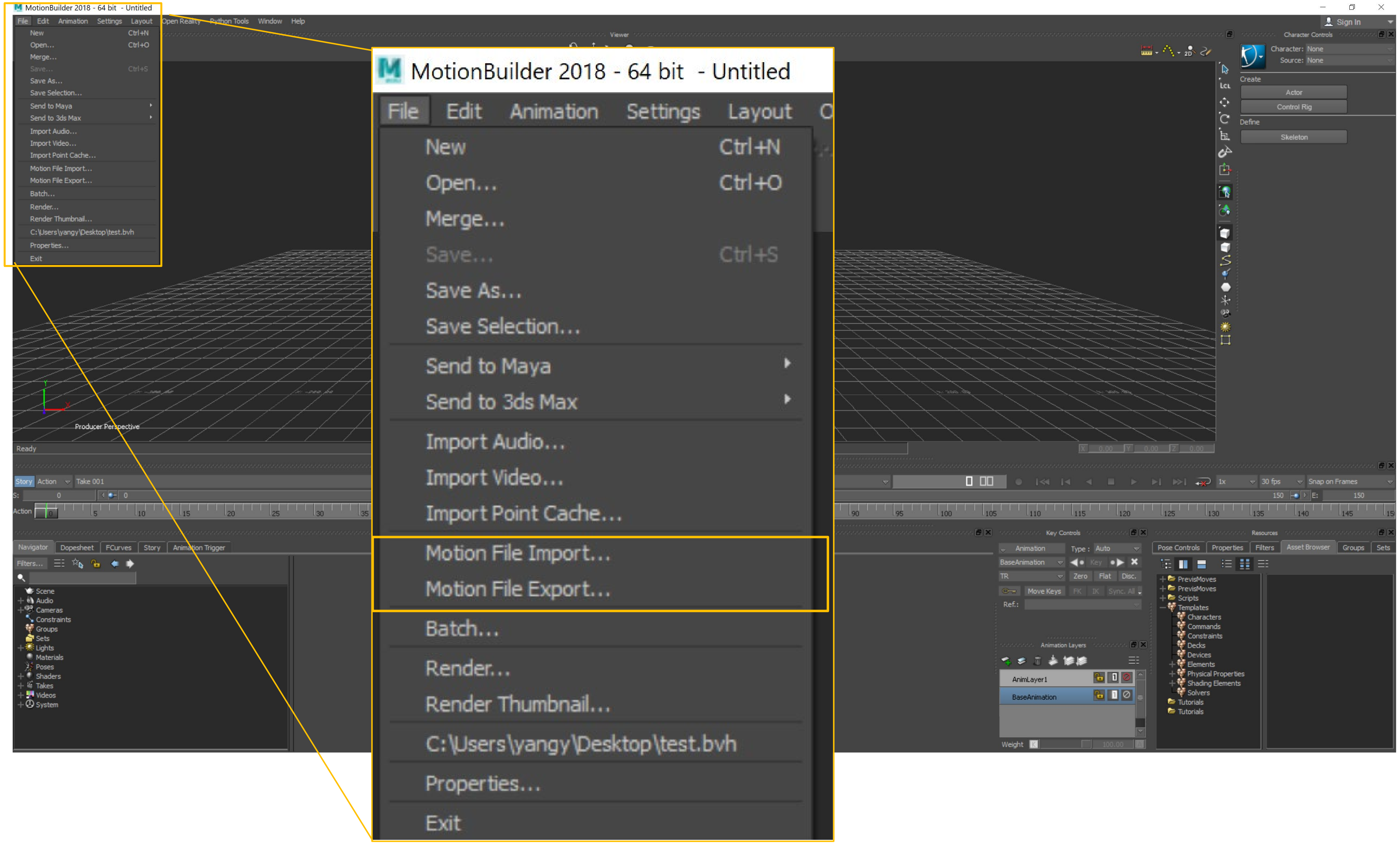
  https://github.com/yangyanzhe/assignment1

- A quick introduction to Unity and Motion Builder

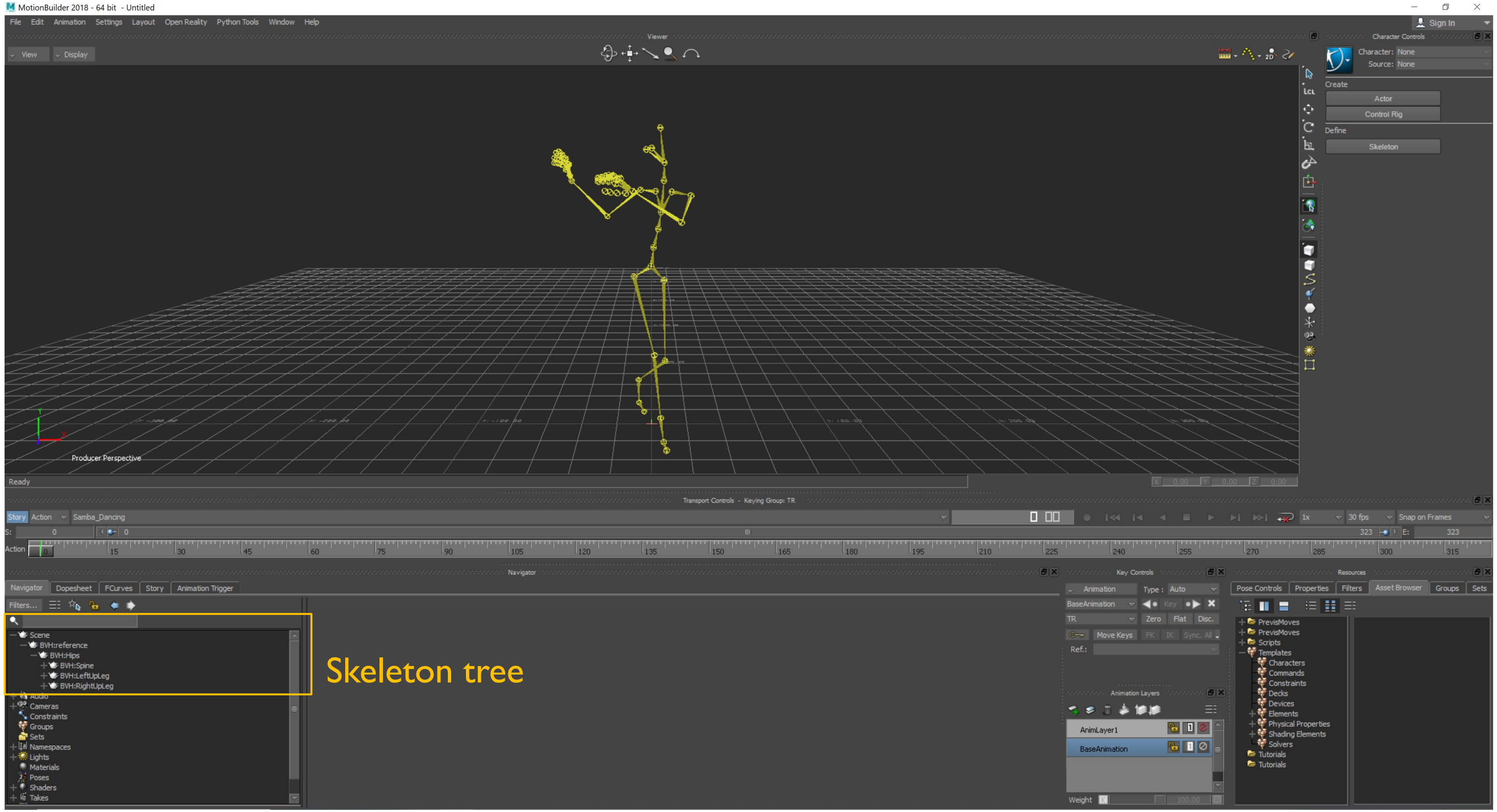
Unity

- Download unity here: https://unity3d.com/get-unity/download
- Live demo

Motion Builder: Load data



Motion Builder: Check data



Skeleton tree