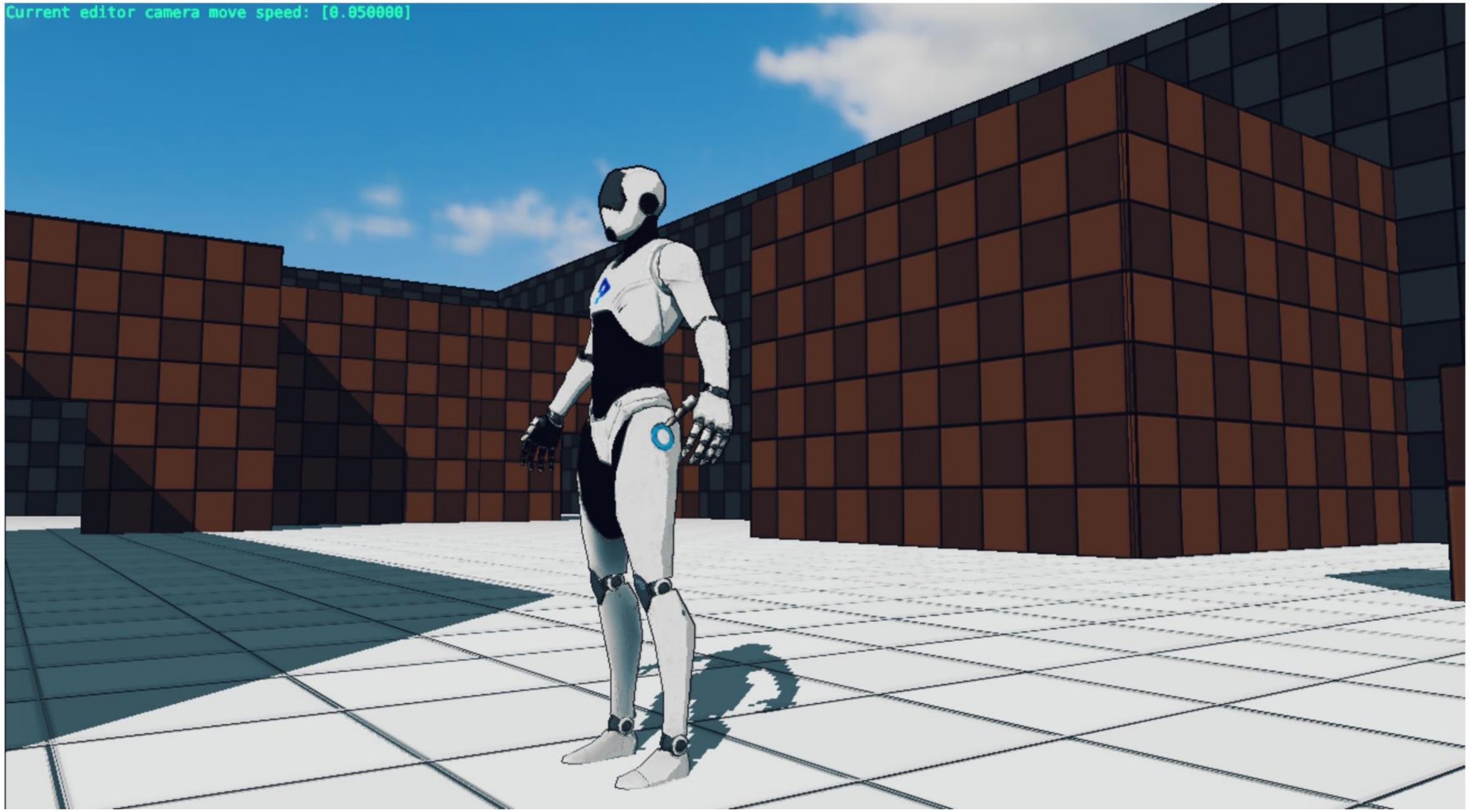
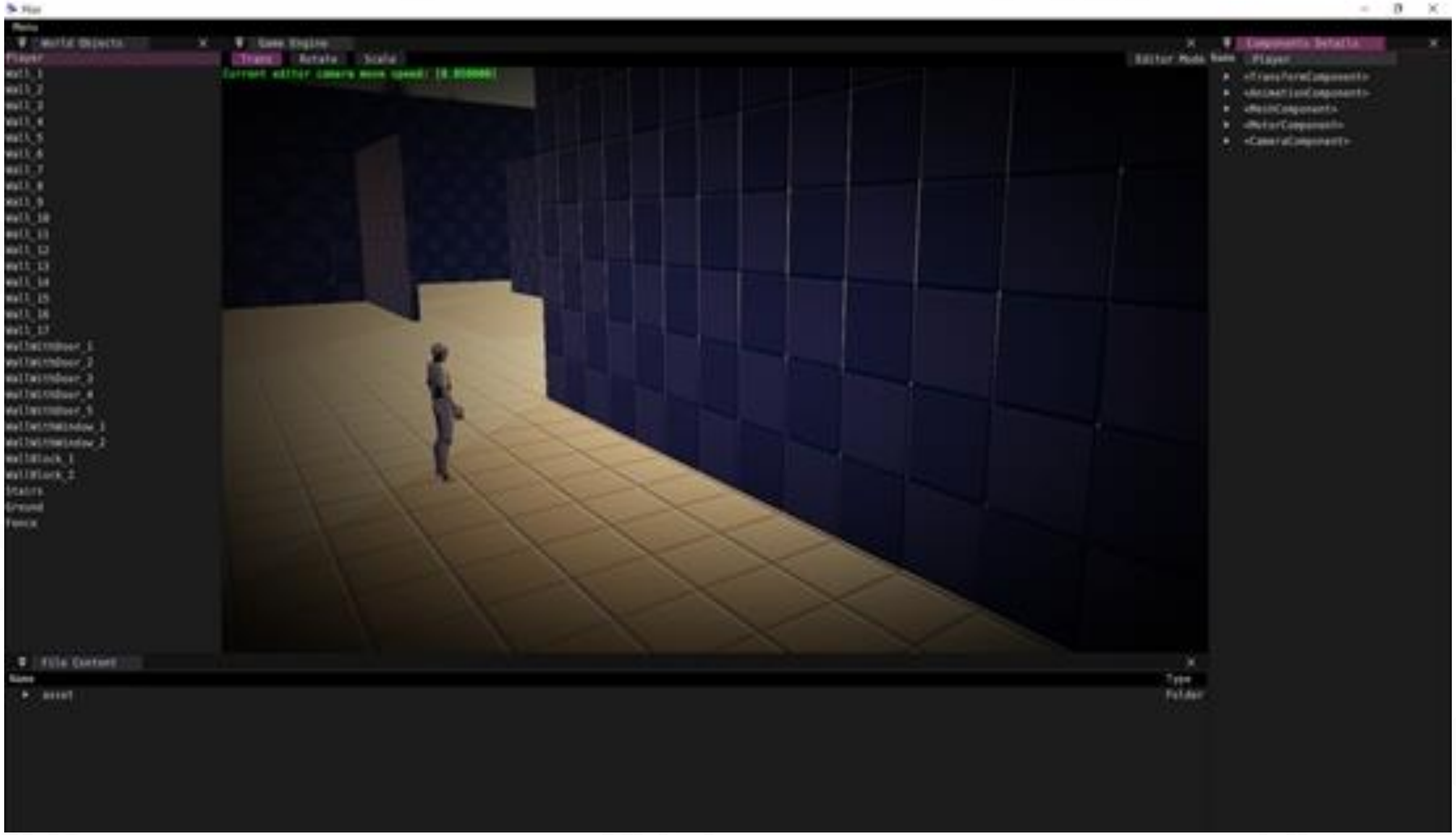
Cartoon Filter



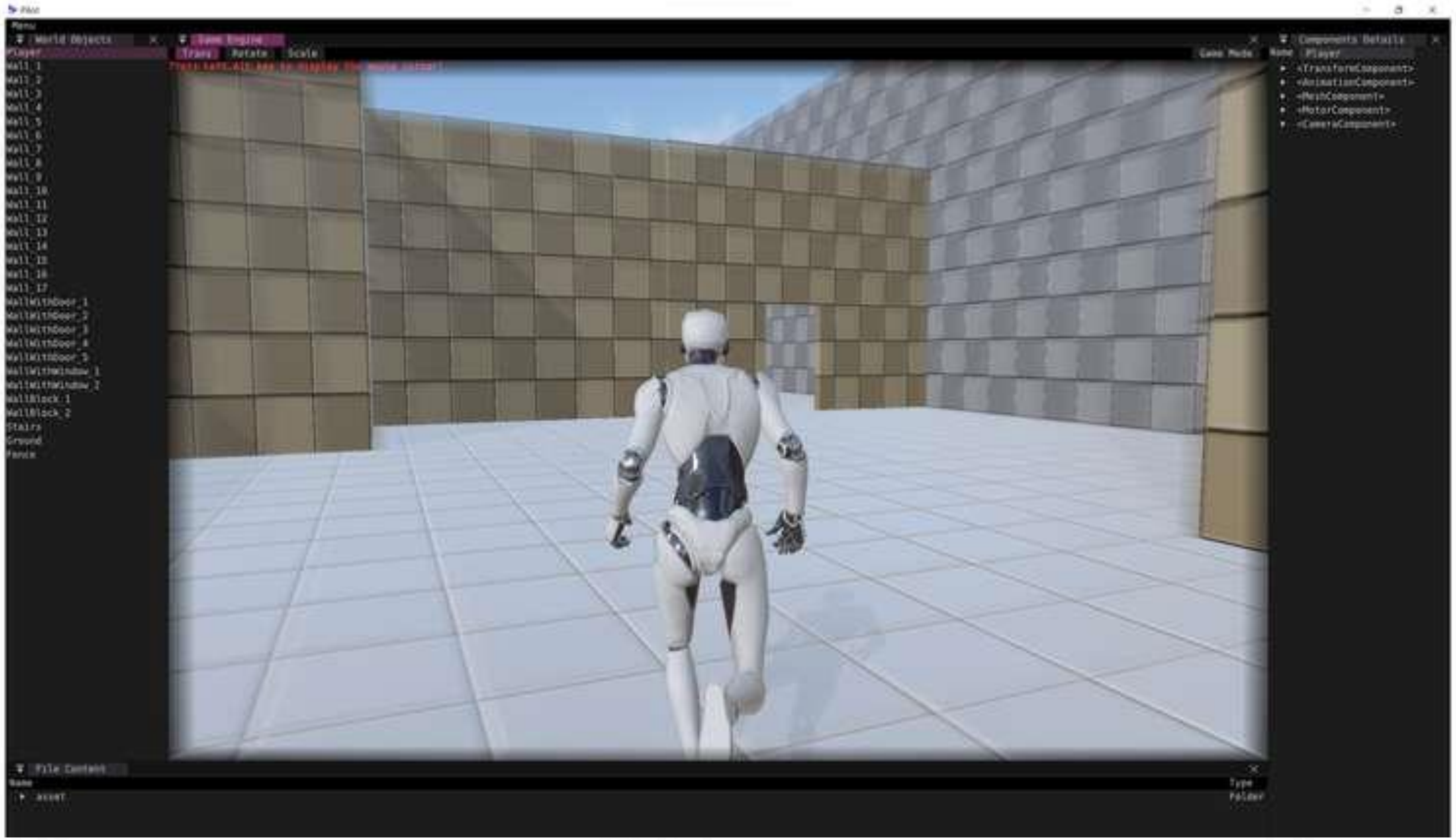
@Li Hypo

Vignette Filter



@li yanchen

Radial blur Effect



@L jay

Voice from Communities

- Pilot engine source code hard to read
  - Yes, we are working on refactoring to improve the architecture clearance
  - We will work on more wiki docs to go through after we done refactoring
- Vulkan API is hard to understand
  - Sorry, it's a problem. We focus on platform portability too much and ignore the learning curve
  - We will try to refactor a better RHI layer to hide the complexity of Vulkan API
- Need more extended reading materials to help understand the course
  - Yes, we agree. We will organize a list of reference papers on our engine website. It might take us a few weeks. Please give us more time
- Didn't submit homework 2 in time, can we open the submission again?
  - The due date of homework 2 is delayed to May 30th, 00:00

Thanks for cool names from communities

ONEOX Engine(@Jason), Brick Engine(@Kpure1000), Alkaid Engine(@核桃), Pi Engine(@王十一)

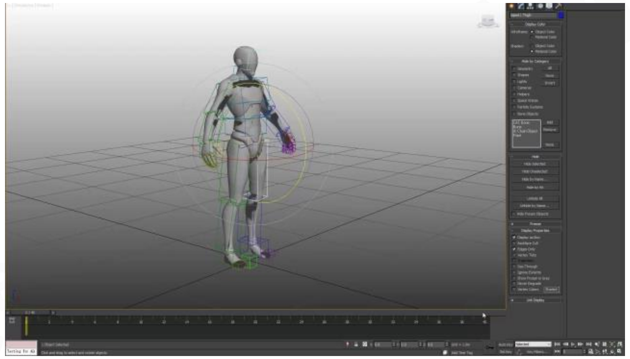
擎空一鹤排云上，引得诗情到碧霄(@Jiazi)

Lecture 09

Animation System

Advanced Animation Technology

How to Achieve the Animation Effect in Real Game?



simple animation

How?

complex animation in real game

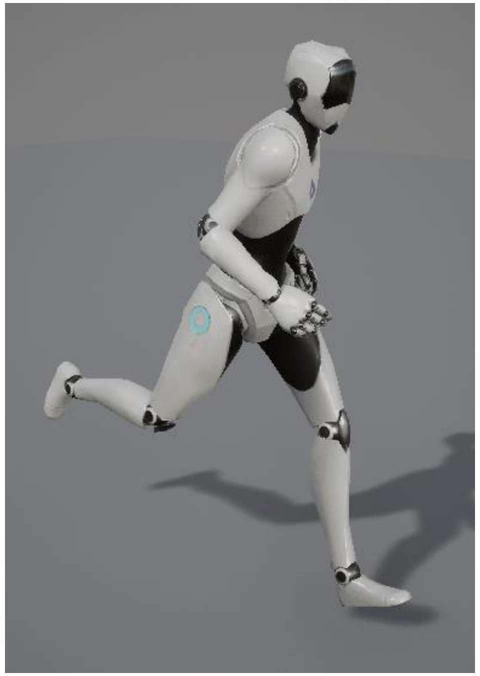
Animation Blending

- The term *animation blending* refers to any technique that allows more than one animation clip to contribute to the final pose of the character

Case: Walking to Running

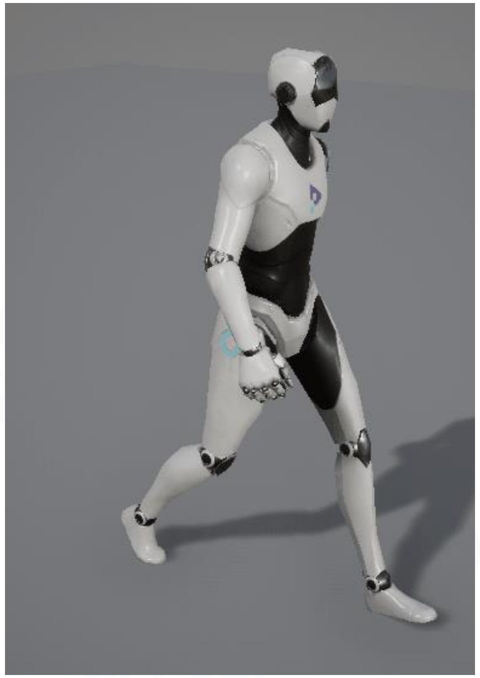
- Assume the character walks at 1.5m/s and runs at 3.0m/s in our game
- As the character's speed increase, we want to switch its animation from walking to running

walking clip



running clip

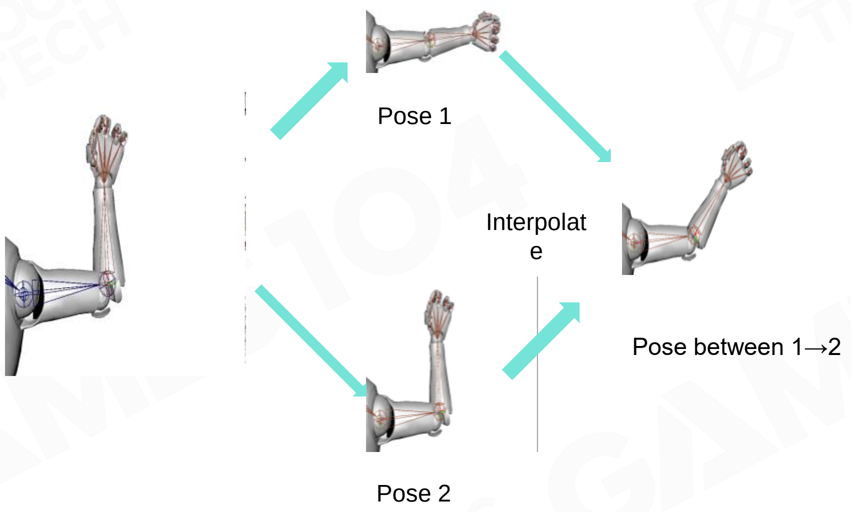
No blending



Blend Result

Math of Blending: LERP

Use LERP to get intermediate frame from poses of different clips
Weight is controlled by game parameters, i.e, character speed



LERP between key frames in one clip

Blending is LERP between poses of different clips

Calculate Blend Weight

$speed_{current}$: current speed
$speed_1$: speed of clip1
$speed_2$: speed of clip2
$weight_1$: calculated weight of clip1
$weight_2$: calculated weight of clip1

$$weight_1 = \frac{speed_{current} - speed_2}{speed_1 - speed_2}$$

$$weight_2 = \frac{speed_{current} - speed_1}{speed_2 - speed_1}$$

Blending is LERP between poses for different clips

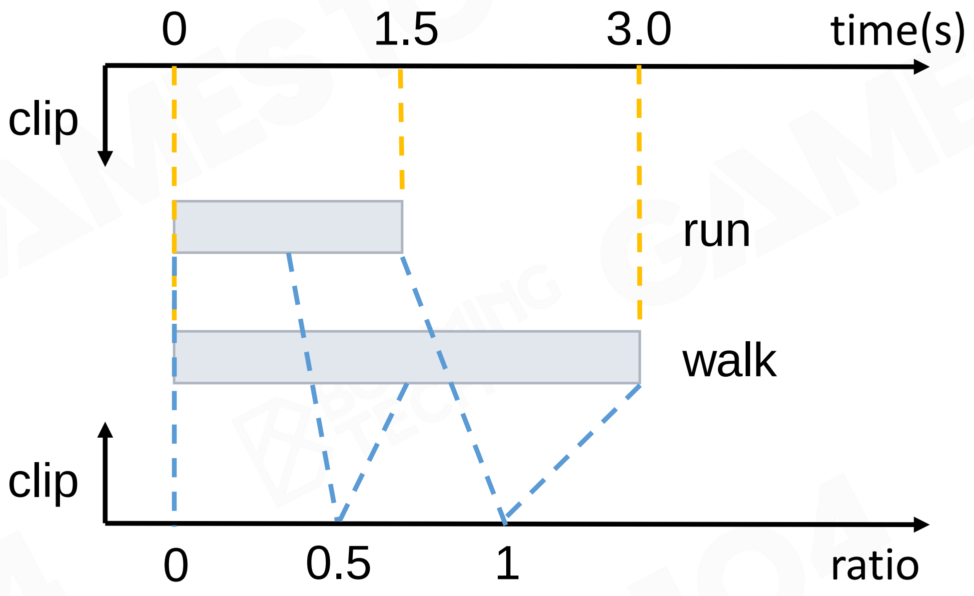
Align Blend Timeline

Align timeline of clips

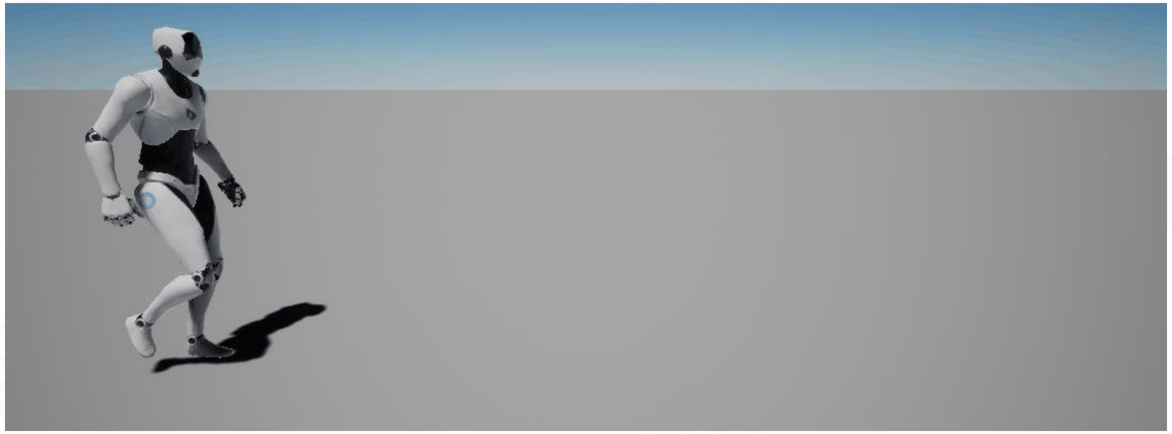
$length_{current}$: current length
Δt : delta time of current blend space
Δt_1 : delta time of clip 1
Δt_2 : delta time of clip 2

$$length_{current} = \frac{speed_1 * weight_1 * length_1 + speed_2 * weight_2 * length_2}{speed_{current}}$$

$$\Delta t_1 = \frac{length_1 * \Delta t}{length_{current}} \qquad \Delta t_2 = \frac{length_2 * \Delta t}{length_{current}}$$



walking normally



Sliding Step

Case: Walking to Running

In order to achieve the desired effect, we need a lot of animation clips with intermediate speeds.
Let the animators produce a whole bunch?

1.5m/s 1.9m/s 2.3m/s 2.7m/s 3.0m/s

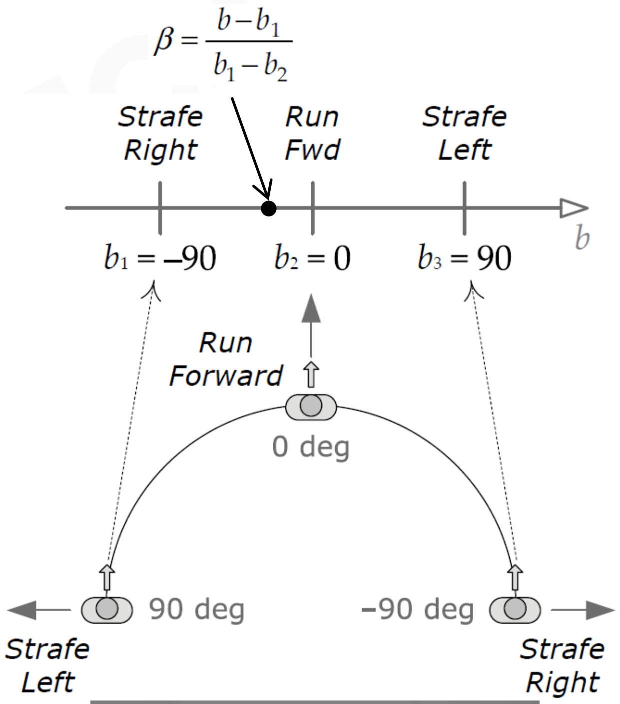
Blend Space

1D Blend Space: Directional Movement

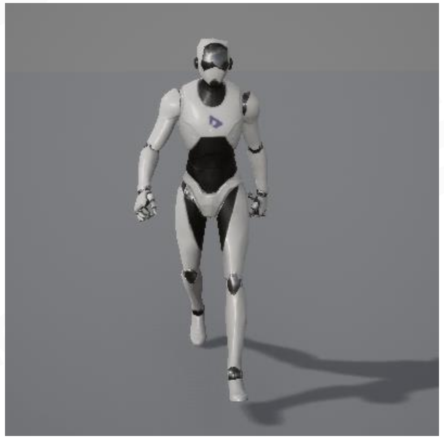
Players can move forward from multiple angles

We can blend any angle from three clips:
- Strafe Left clip
- Run Forward clip
- Strafe Right clip

The technique is called 1D Blend Space.




leftward


forward

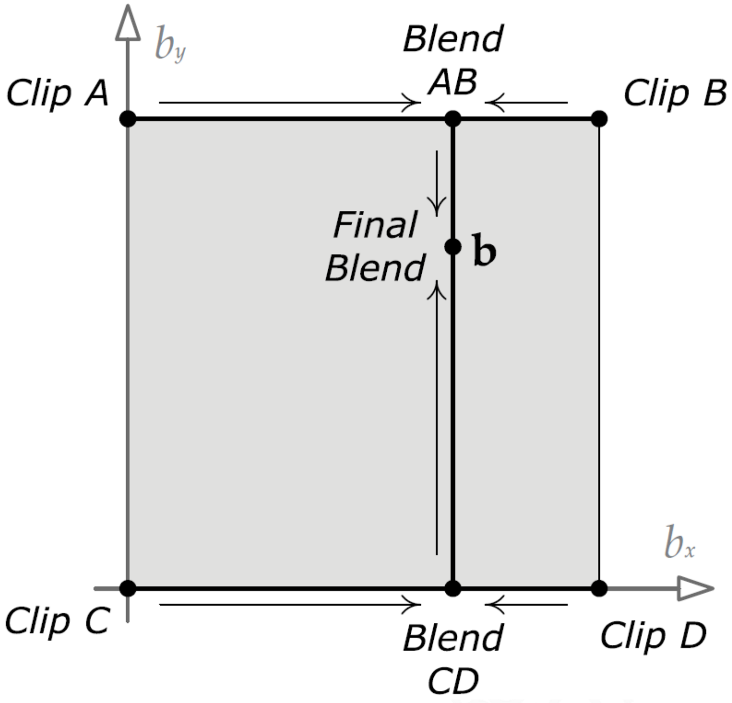

rightward


multi-directional movement

Directional Walking and Running

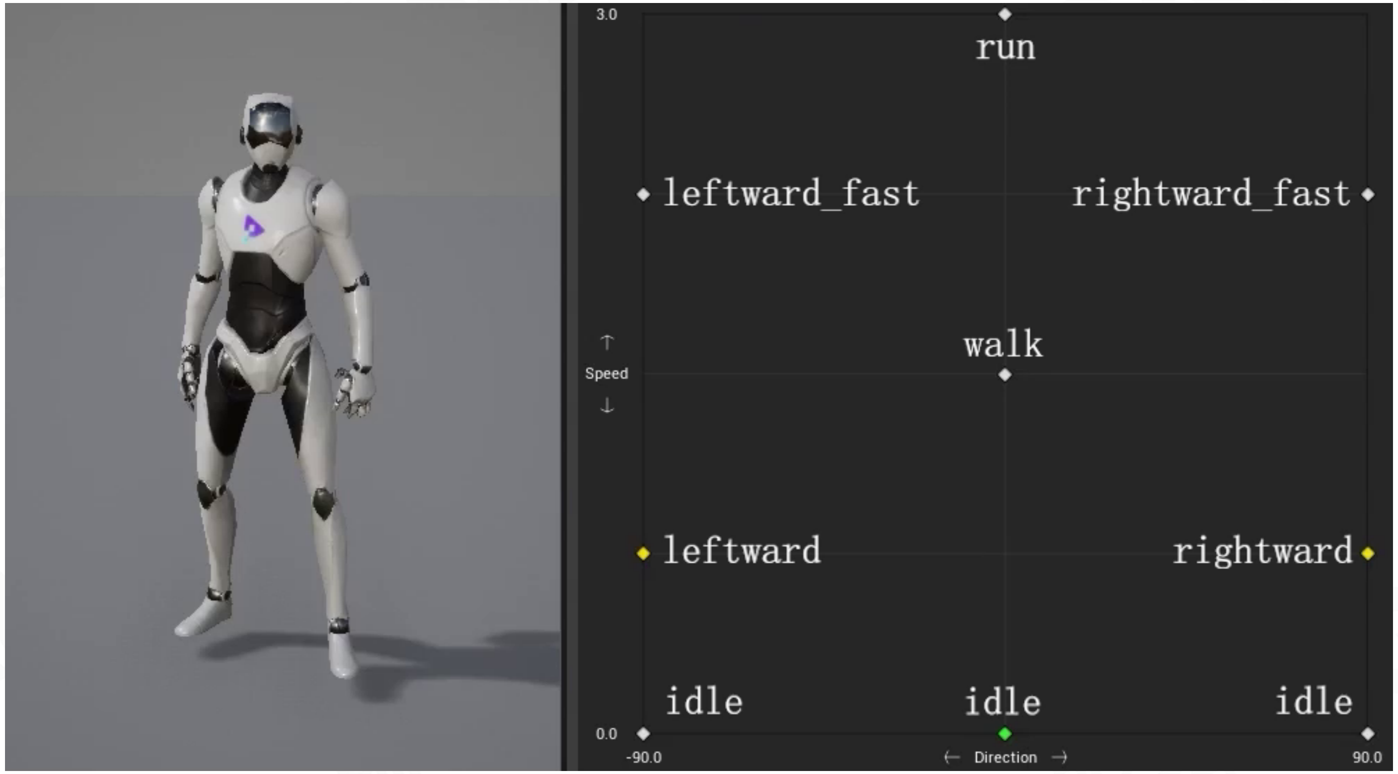
Players can change direction and speed at the same time

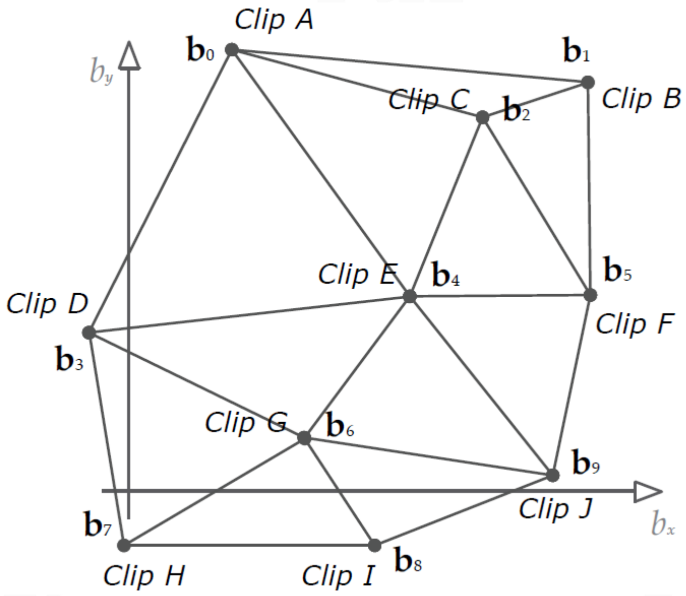
We simply place the two 1D Blend Spaces orthogonally and we get an 2D Blend Space



2D Blend Space

Since the movement speed in the lateral direction is lower in the forward direction, the character should enter the running state in a lower speed in the lateral direction





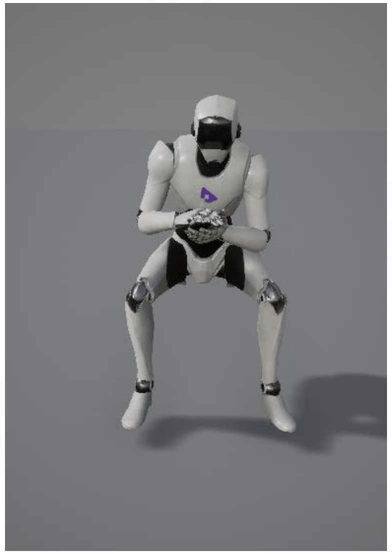
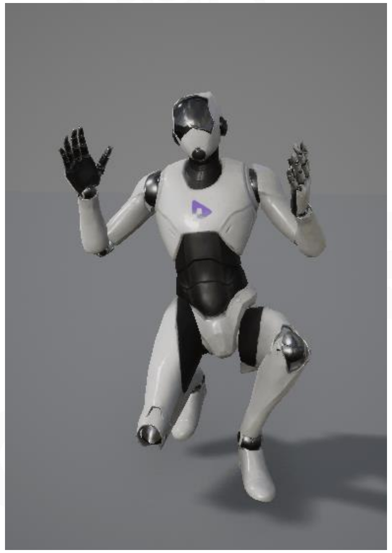
Delaunay Triangulation

Case: Applauding on Different Poses

There are multiple robots in different poses in the scene

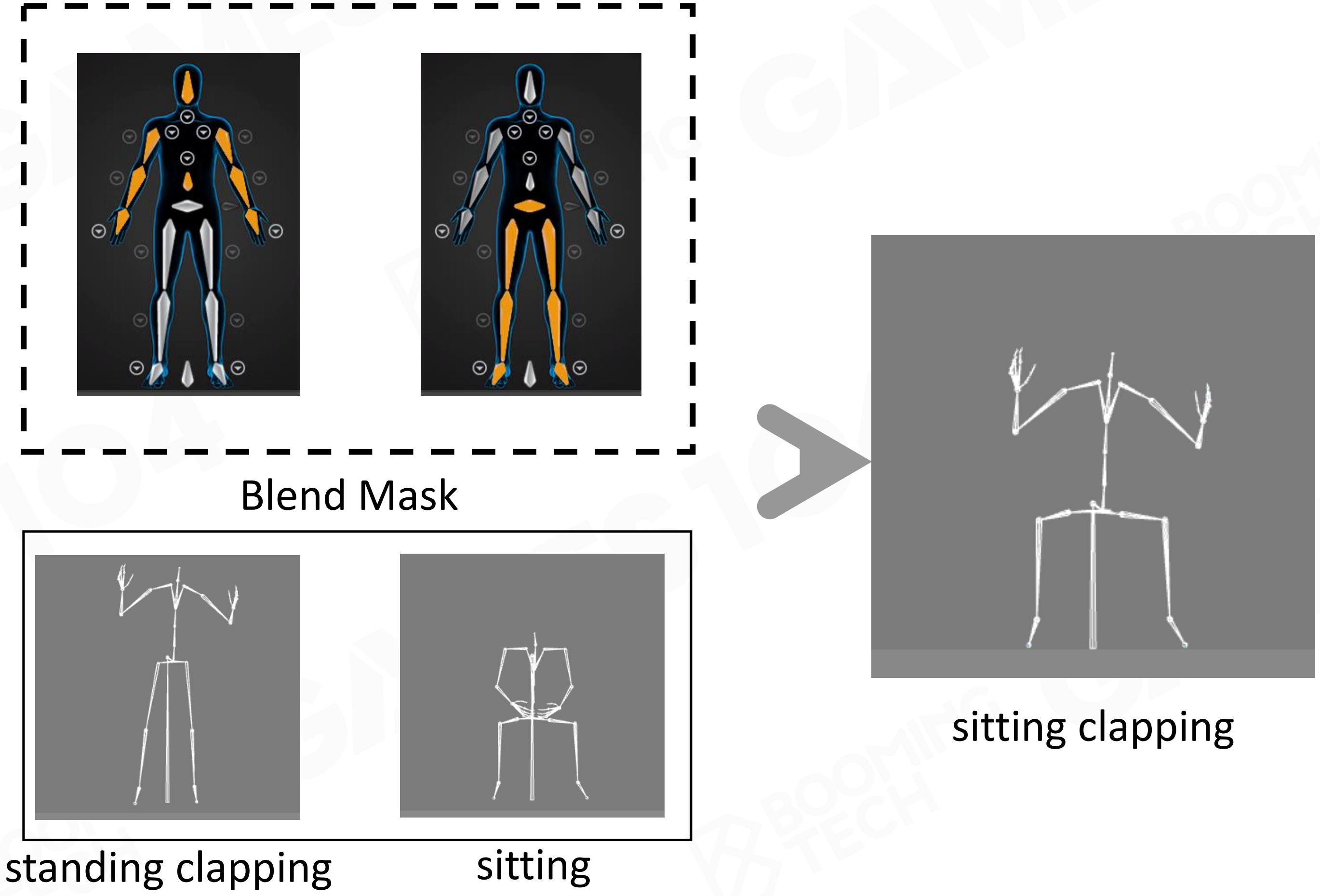
We need to make applause animations for various poses separately

Is it possible to make a single applauding animation that can be applied to all poses?

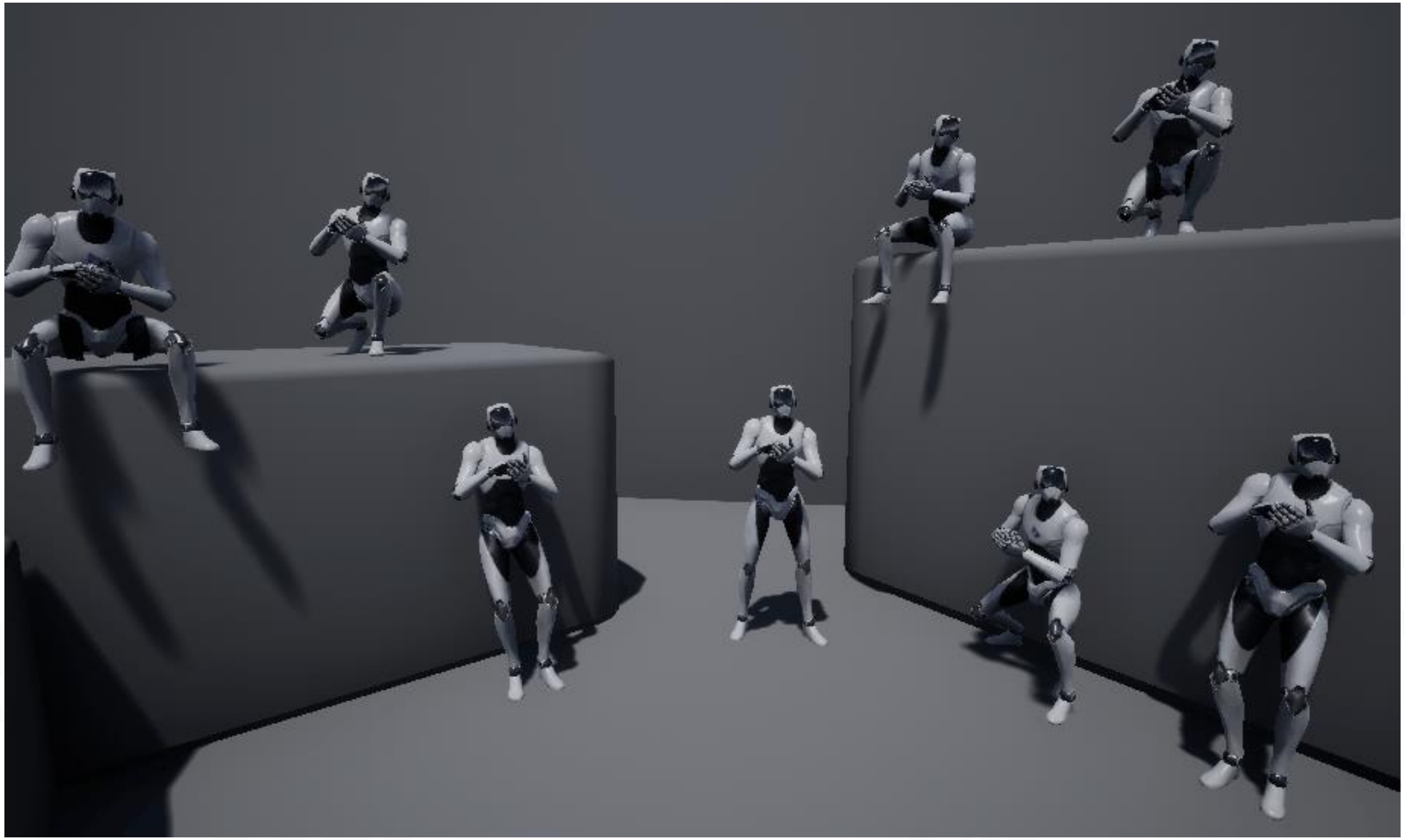





Skeleton Masked Blending

The set of all blend percentages for the entire skeleton $\{\beta_j\}|_{j=0}^{N-1}$ is sometimes called a blend mask b



Case: Warm Welcome from the Robots



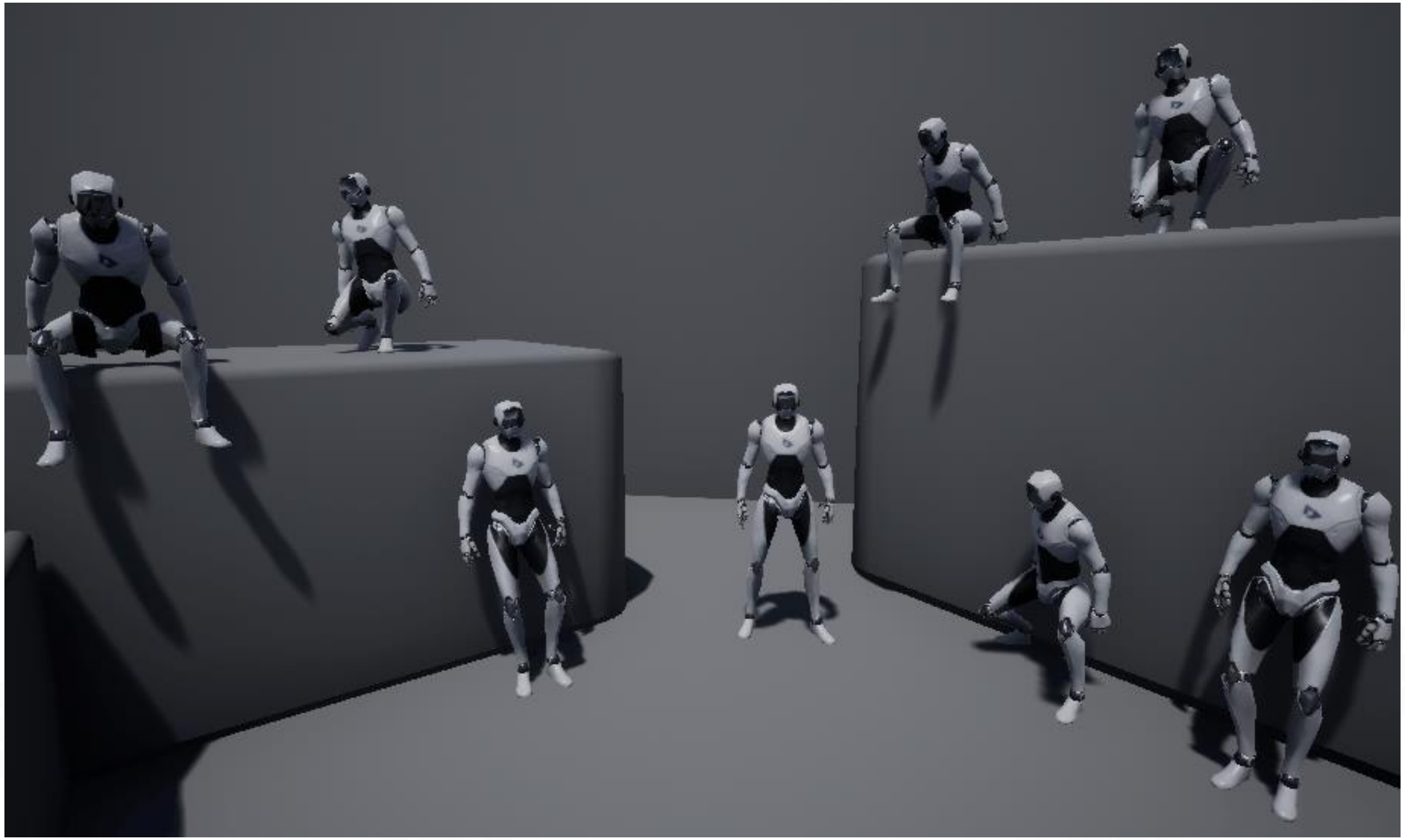
We will let robots applauding in different poses

Additive Blending

Add a difference clip into a regular clip to produce a new clip

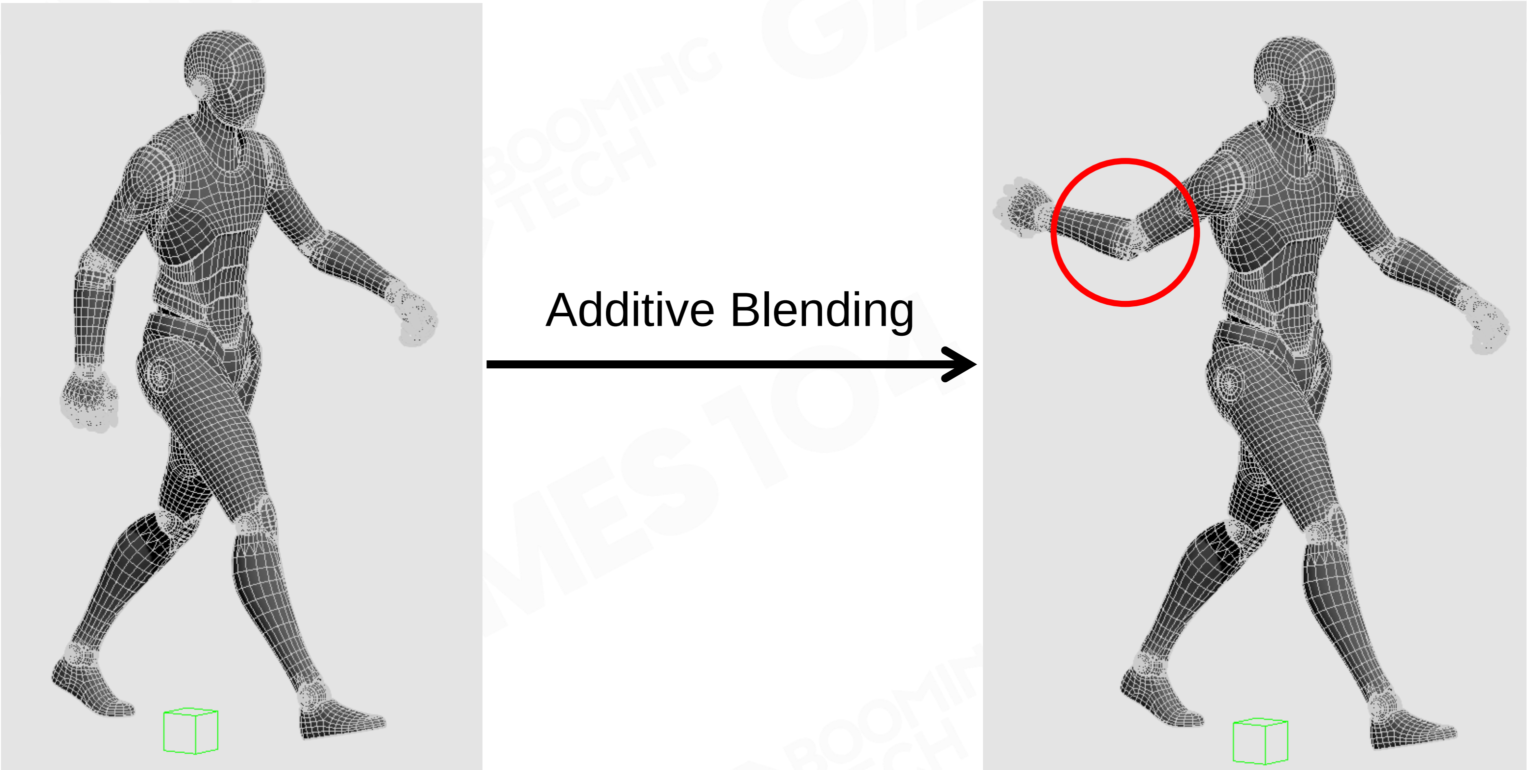
Additive Blending introduces a new kind of animation called a ***difference clip***, which represents the difference between two regular animation clips.
A difference clip can be added into a regular animation clip in order to produce interesting variations in the pose and movement of the character.

Nodding to Camera



Additive Blending - Abnormal Bone Results

- Additive blends are more likely to have abnormal bone results

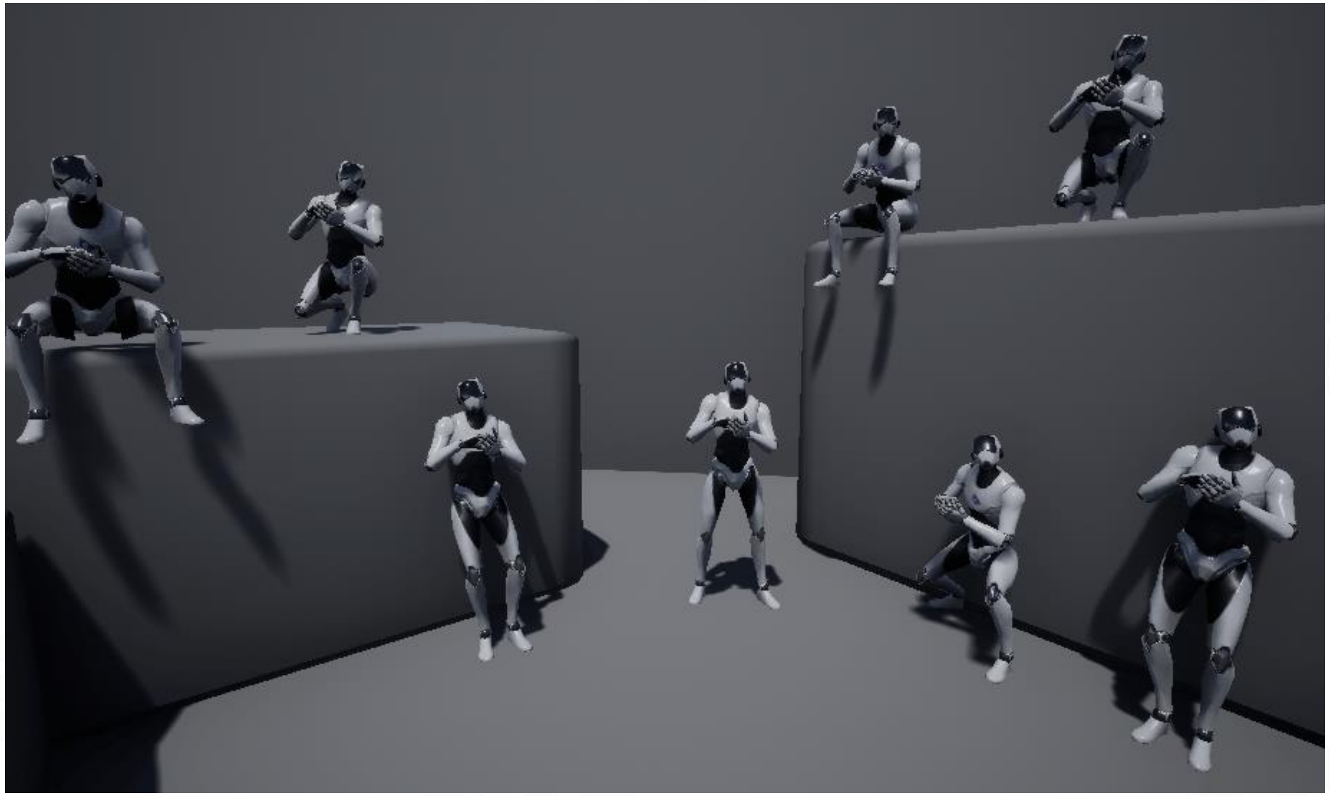


Origin

Target

Animation Blending Summary

- 1D Blend Space
  - Blend poses based on a single input value
- 2D Blend Space
  - Blend poses based on two input values
  - Triangular blend
- Masked Blending
- Additive Blending



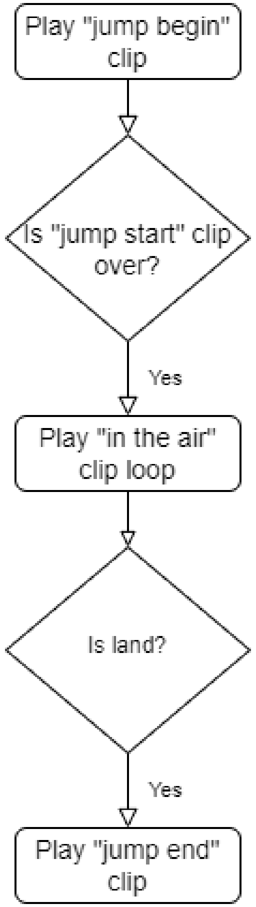
Action State Machine (ASM)

Case: Jumping

How to animate jump?

Blend Space is synchronous, but jump is stateful

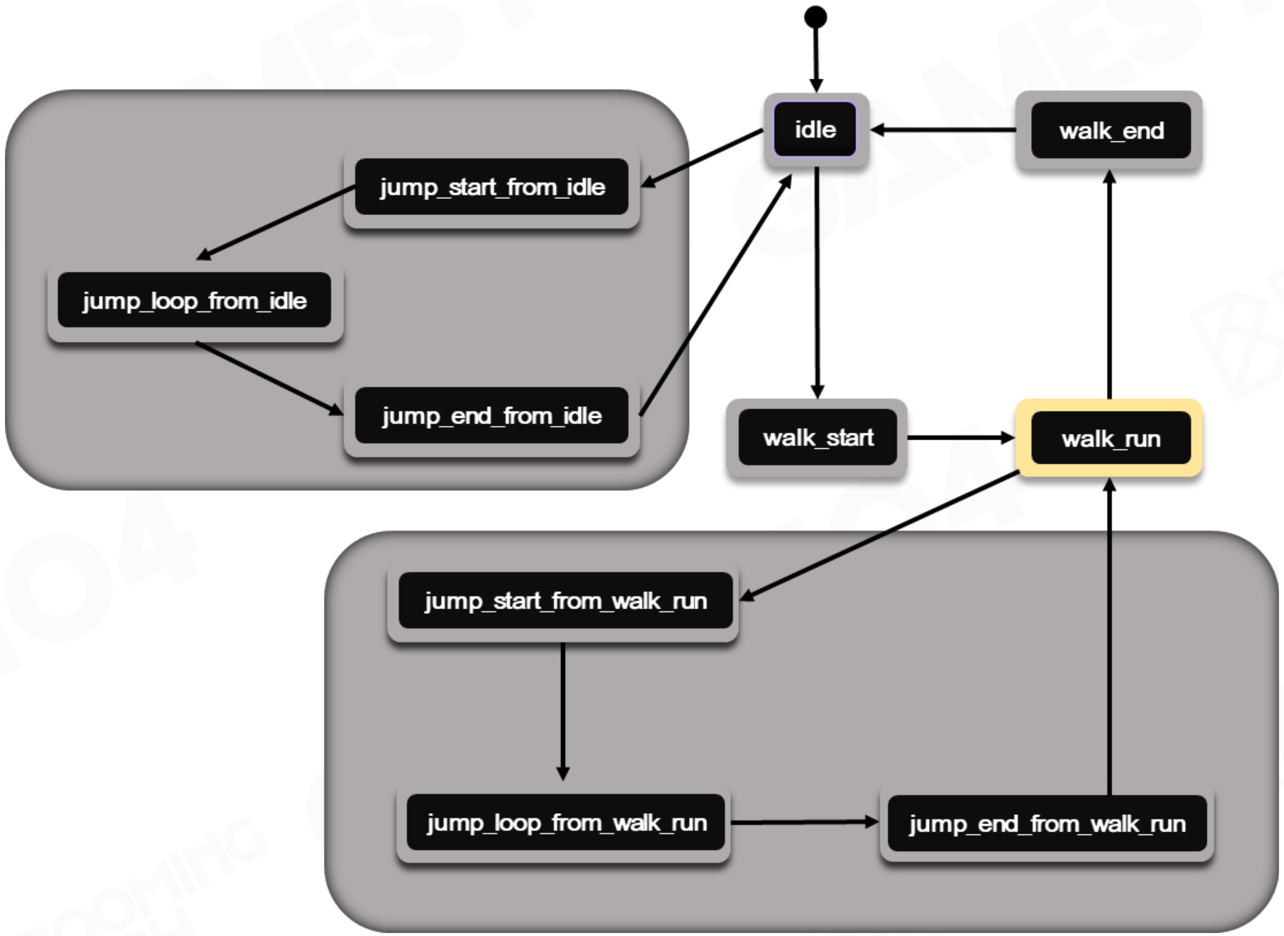
We usually model the jumping action via a finite state machine, commonly known as the Action State Machine (ASM)



ASM Definition

- ASM consists of nodes and transitions
- Node types
  - Blend space
  - Clip

```
class ActionStateMachineClipNode
{
    AnimationClip m_clip;
    bool m_is_loop;
};

class ActionStateMachineBlendSpaceNode
{
    BlendSpace m_blend_space;
    bool m_is_loop;
};
```



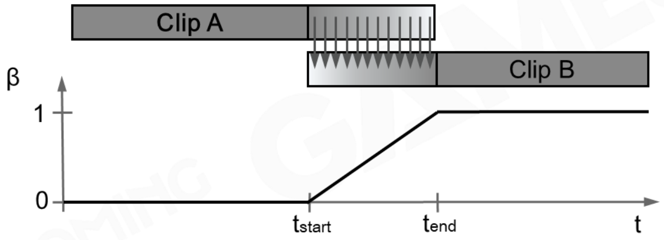
ASM Definition

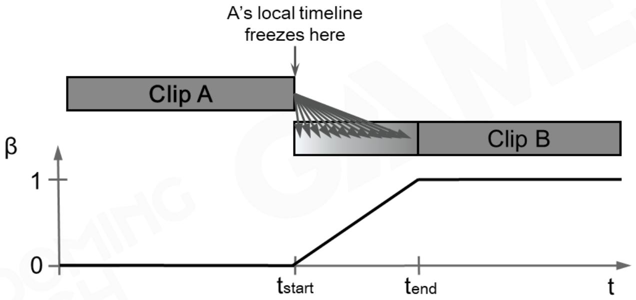
- Transition type
  - simply “pop” from one state to another
  - cross-fade from one state to the next
  - Special transitional states

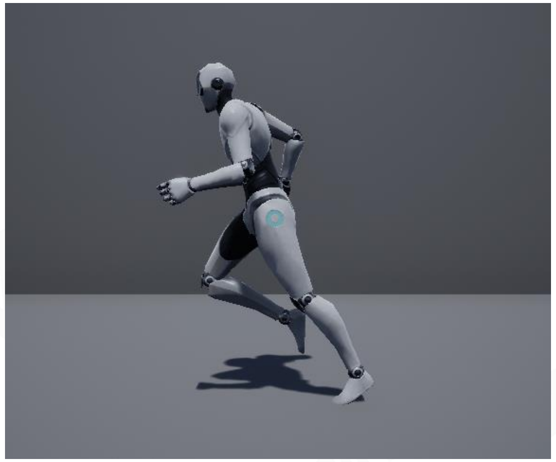
```
class ActionStateMachineTransition
{
    int     m_source_node_index;
    int     m_target_node_index;
};

class ActionStateMachineTransitionWithCrossFade
{
    int     m_source_node_index;
    int     m_target_node_index;
    float   m_duration;
    Curve   m_curve;
};
```

Cross Fades

Two common ways
- Smooth transition
  - restriction: the two clips must be looping animations, and their timelines must be synchronized
- Frozen transition







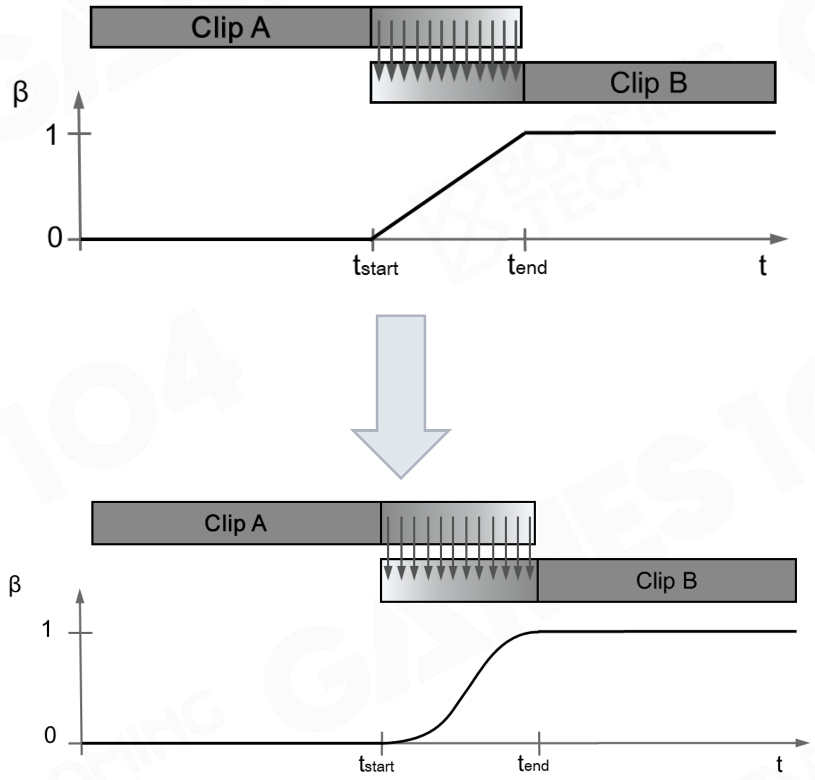
no cross fade

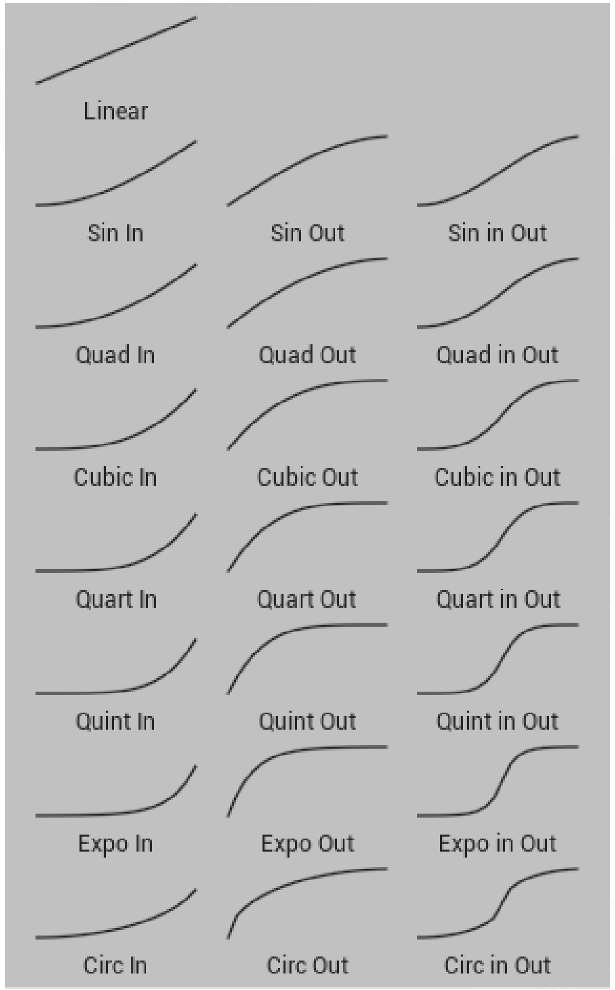
smooth transition

frozen transition

Cross Fades Curve

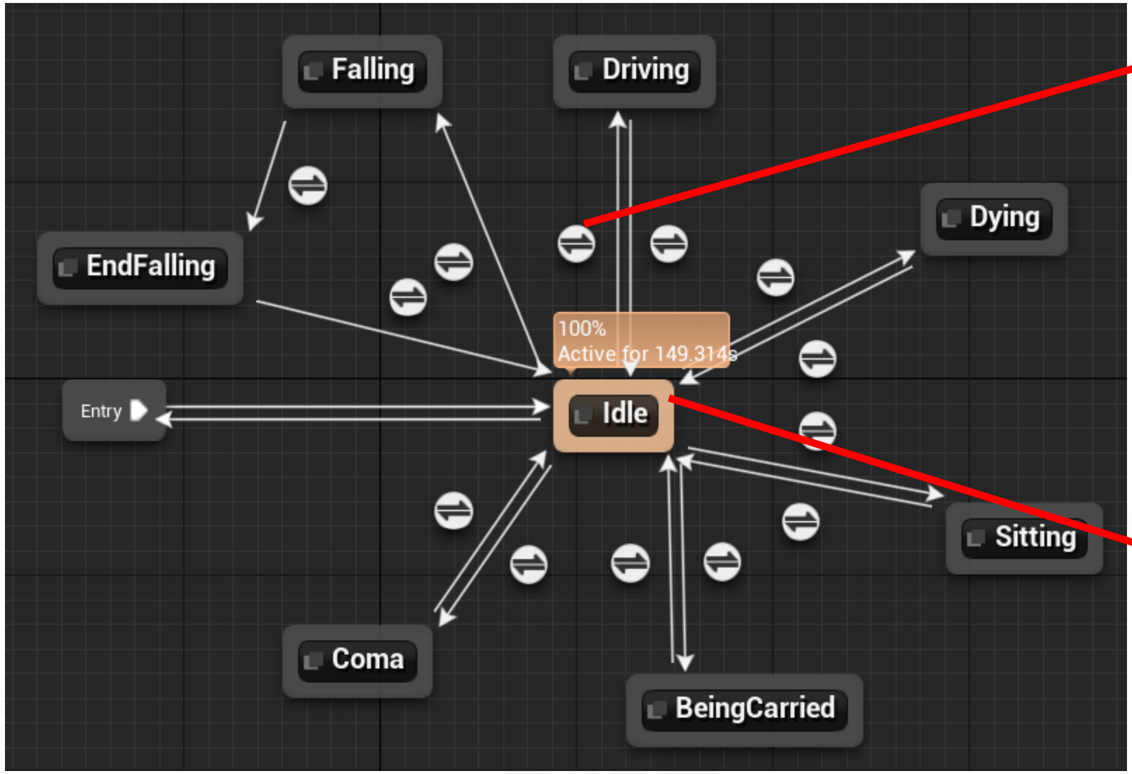
- Different cross fades curve could be used for different demands



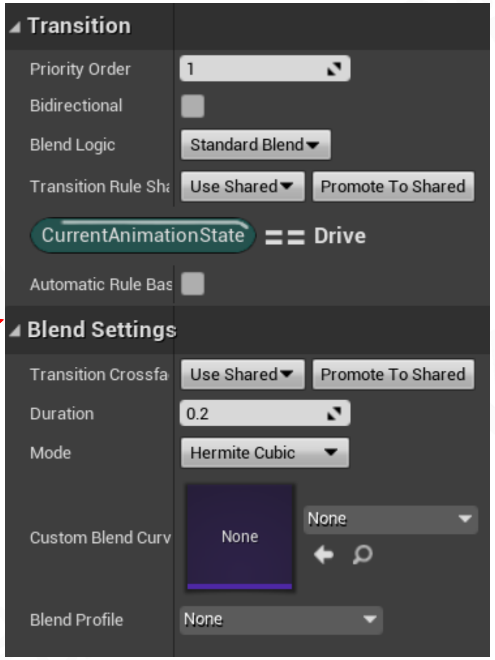


Animation State Machine in Unreal

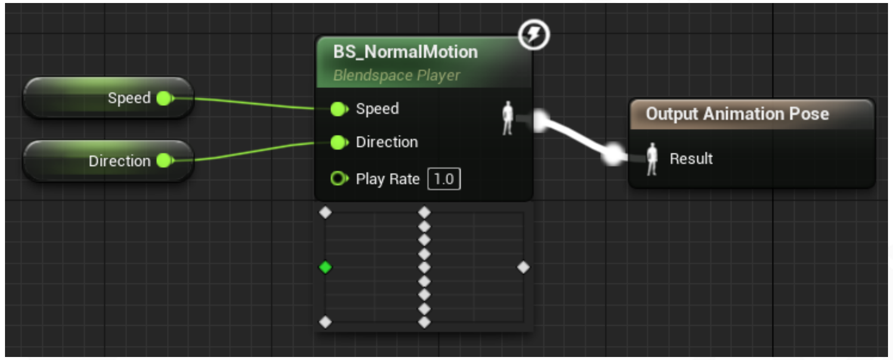
- State: a blueprint graph which outputs a pose
- Transition : control when to change state and how to blend (**Multi**)



Action State Machine

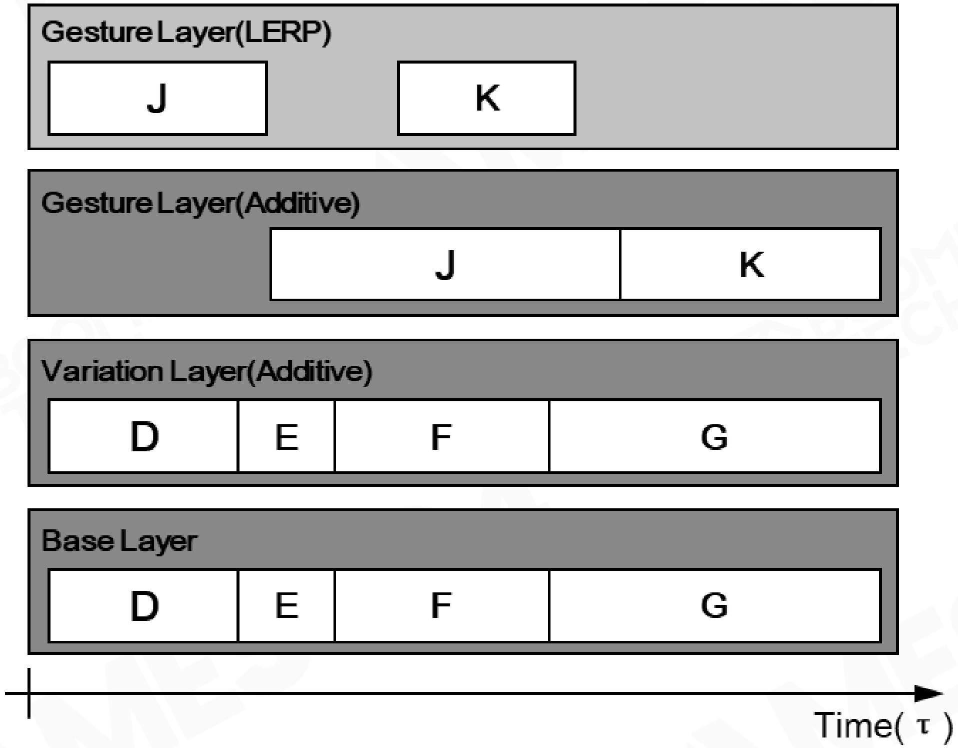


Transition Rule



State Node with Blend Space

Layered ASM



different parts of a character’s body to be doing different, independent or semi-independent actions simultaneously

Devil May Cry 5

Animation Blend Tree

Blend Tree

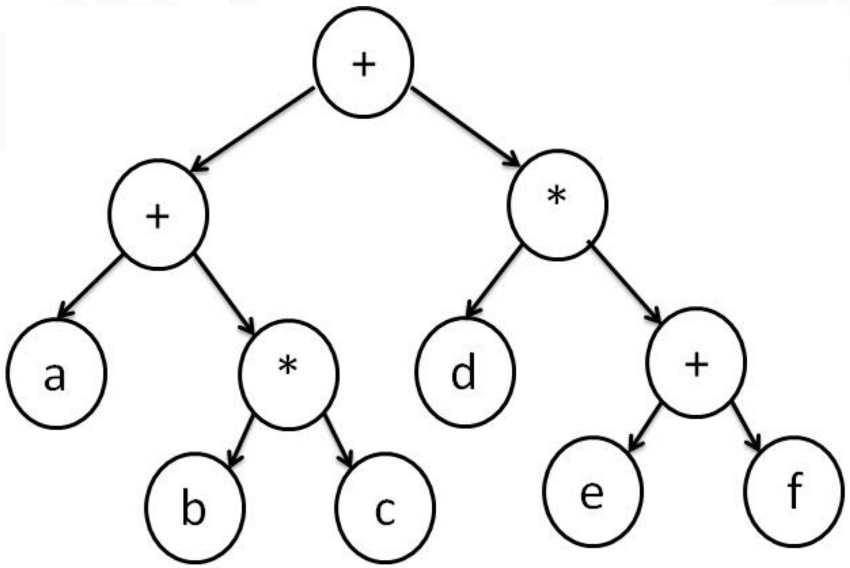
Structure layered ASMs and operations as a tree

- Inspired by expression tree
- Easy to understand for animators

For a blend tree

- Non-terminal nodes and terminal nodes (leaf nodes)
- The result of each non-terminal node is a pose

$$a + (b * c) + d * (e + f)$$



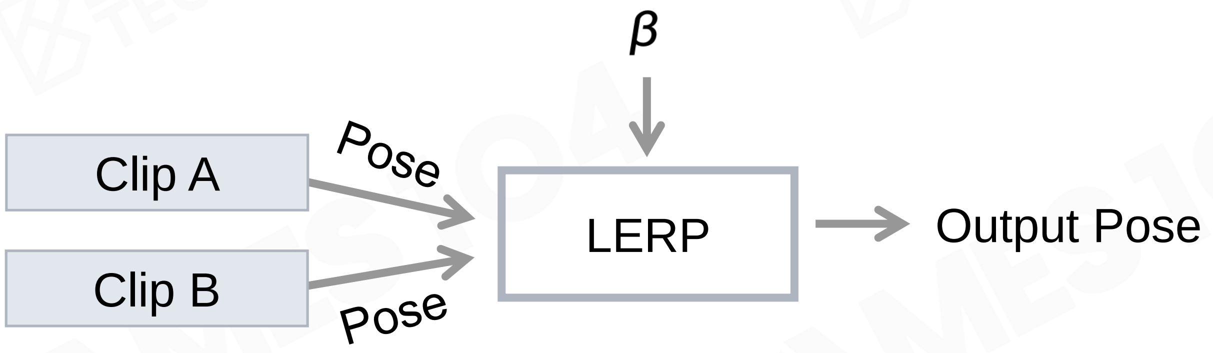
Expression Tree

LERP Blend Node

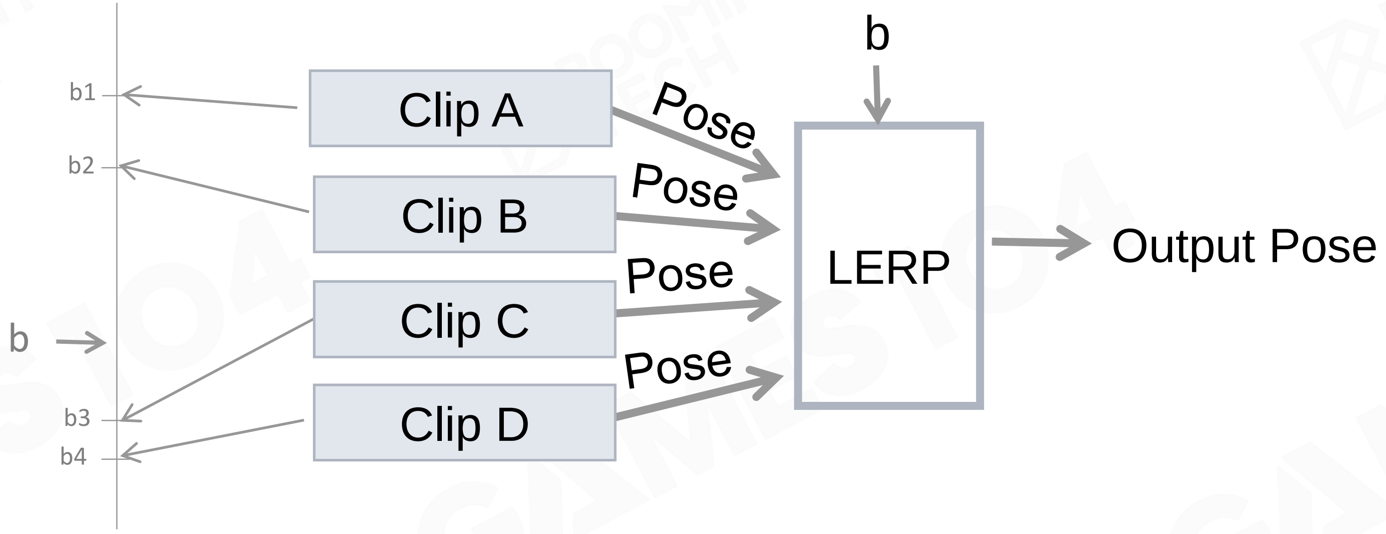
Binary LERP node

- Basic non-terminal node in blend tree
- LERP two input poses with weight β into one output pose

Usually extended to handle multiple inputs (e.g. Ternary/Quad LERP node)



Binary LERP node

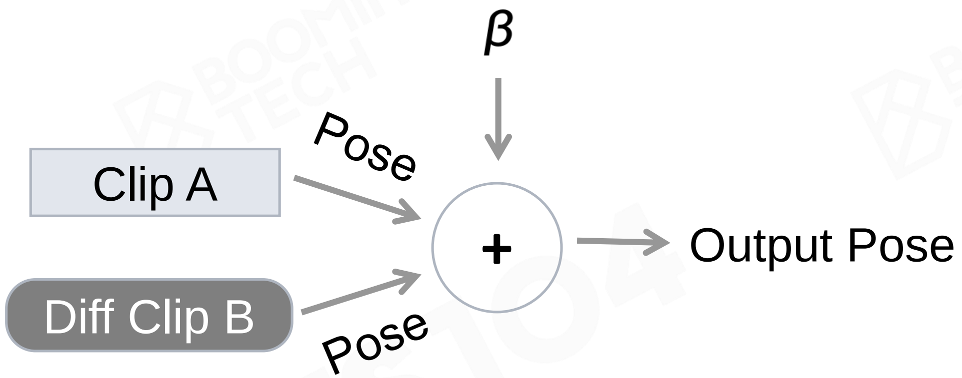


Extended LERP node

Additive Blend Node

Additive Blend node

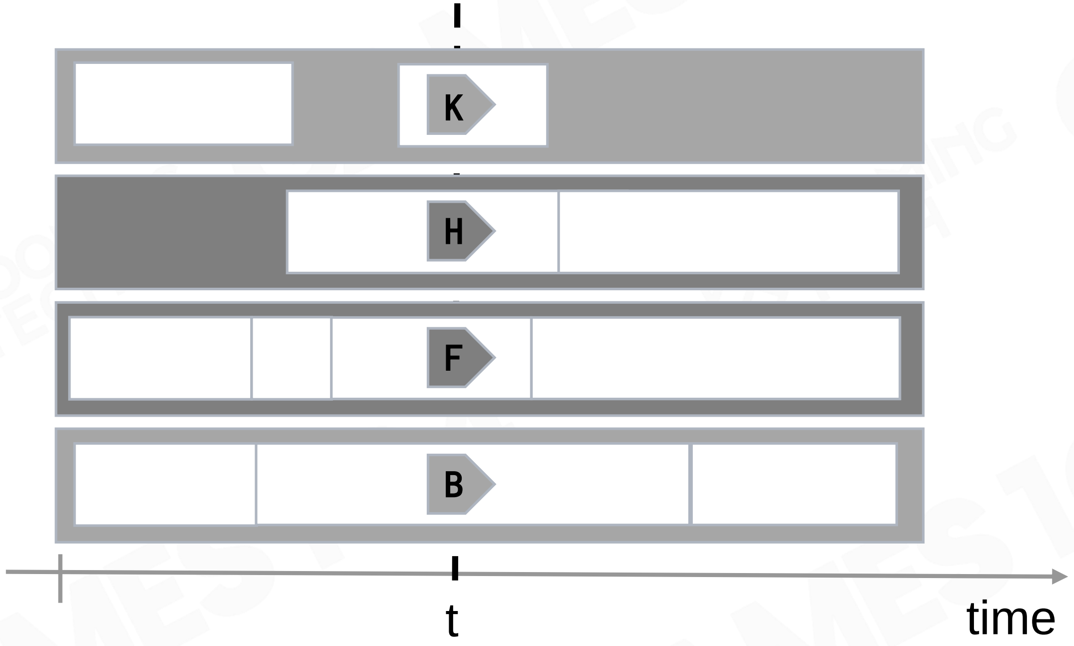
- Basic non-ternimal node in blend tree
- Add the second input pose (usually a difference one) into the first input pose by weight β



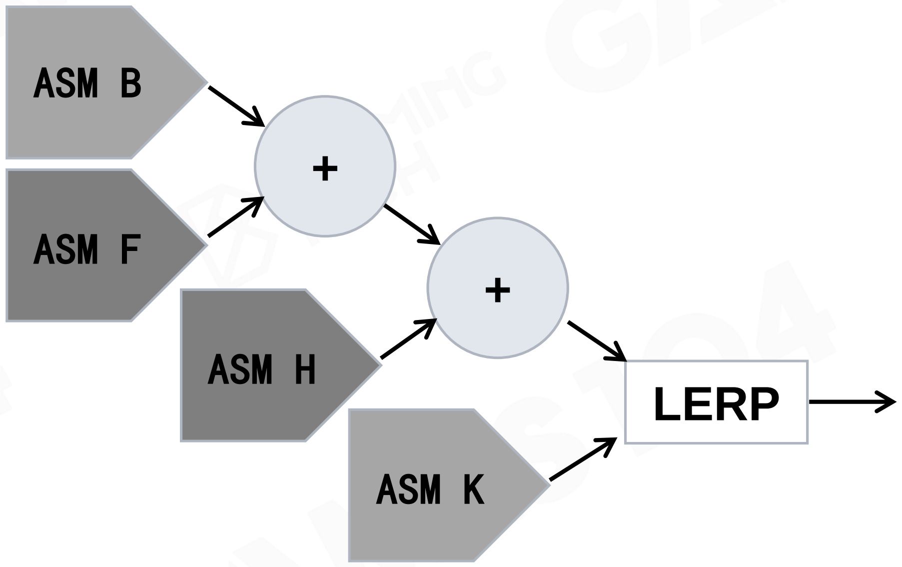
Additive Blend Node

Express Layered ASM in Blend Tree

Use a blend tree to describe the desired final pose of ASMs



Layered ASM at time t



Blend Tree

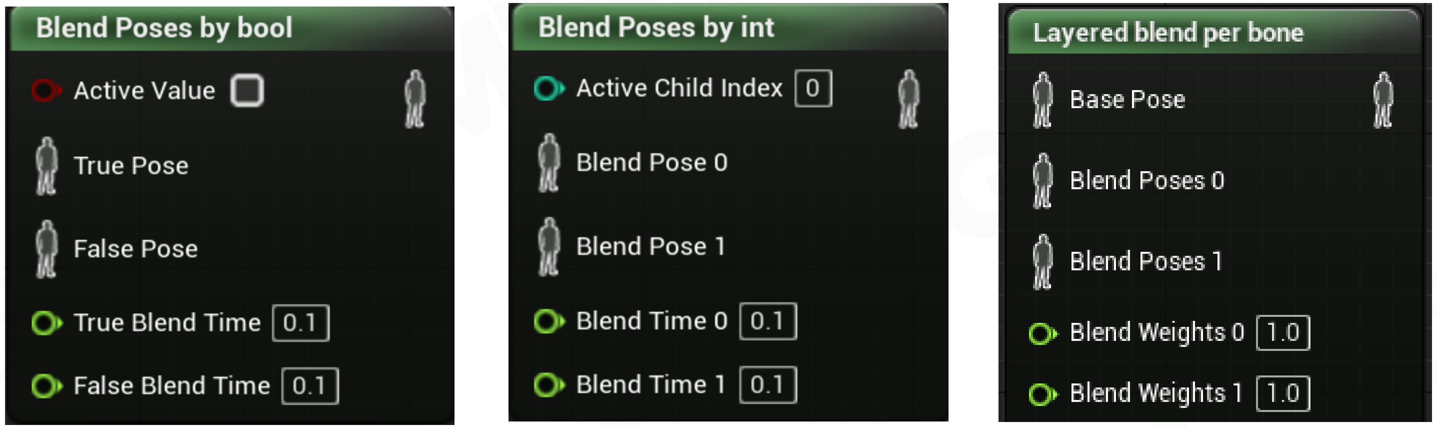
Blend Tree Nodes

Terminal node (Leaf Nodes)

- Clip
- Blend Space
- ASM

Non-terminal node (No-Leaf Nodes)

- Binary LERP blend node
- Ternary (triangular) LERP blend node
- Binary additive blend node

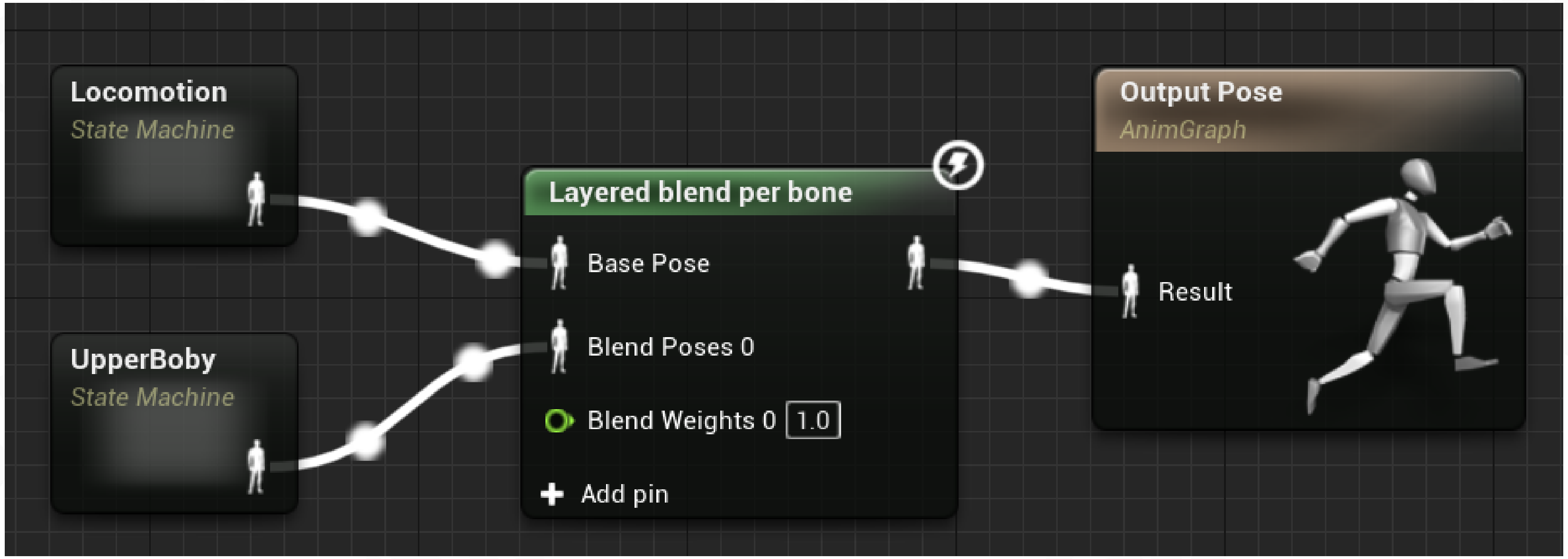


Nodes in UE4

Unreal Animation Blueprint

A blueprint graph which outputs a final pose

- Take clip poses or the results of ASMs as input
- Blend input poses by different methods



Blend Tree Control Parameters

node search

provide a way for higher-level code to find blend nodes in the tree

named variable

allow names to be assigned to the individual control parameters. The controlling code can look up a control parameter by name in order to adjust its value

control structure

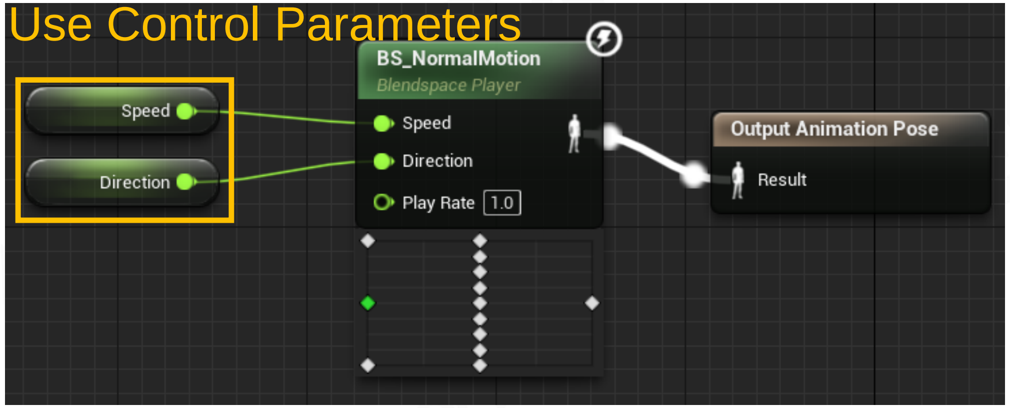
a simple data structure, contains all of the control parameters for the entire character. The nodes in the blend tree(s) are connected to particular control parameters

Animation blend tree is way more complicated that those classic notes (i,e, event nodes, calculation/logic nodes and special blending and flow control nodes).

Unreal Animation Blueprint Control

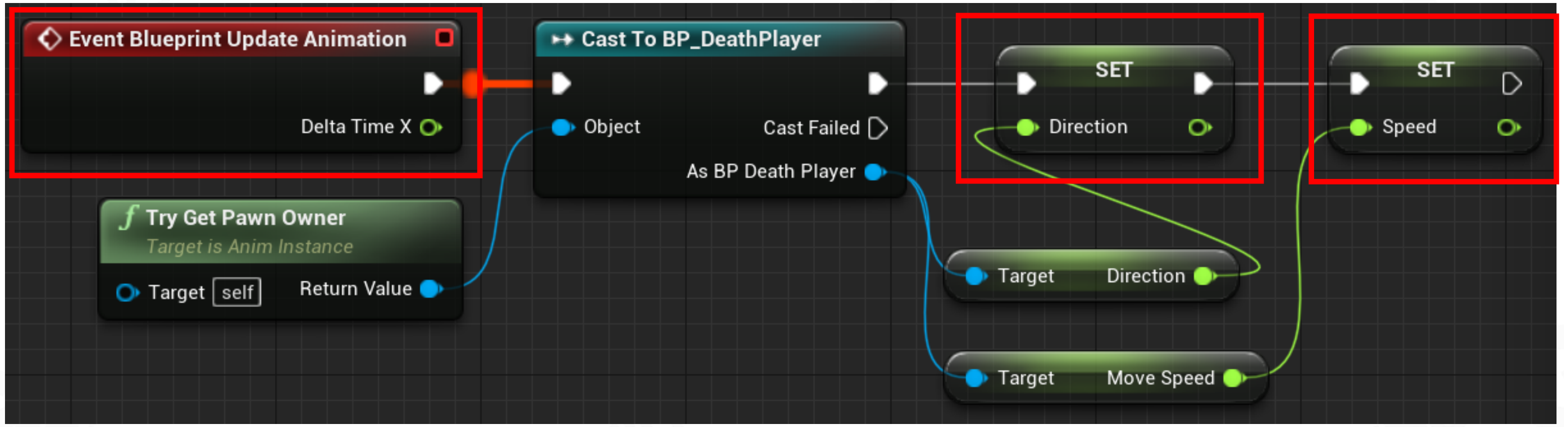
Named variables as members in animation blueprint

- Can be updated through blueprint
- Can be used anywhere inside the Blend Tree

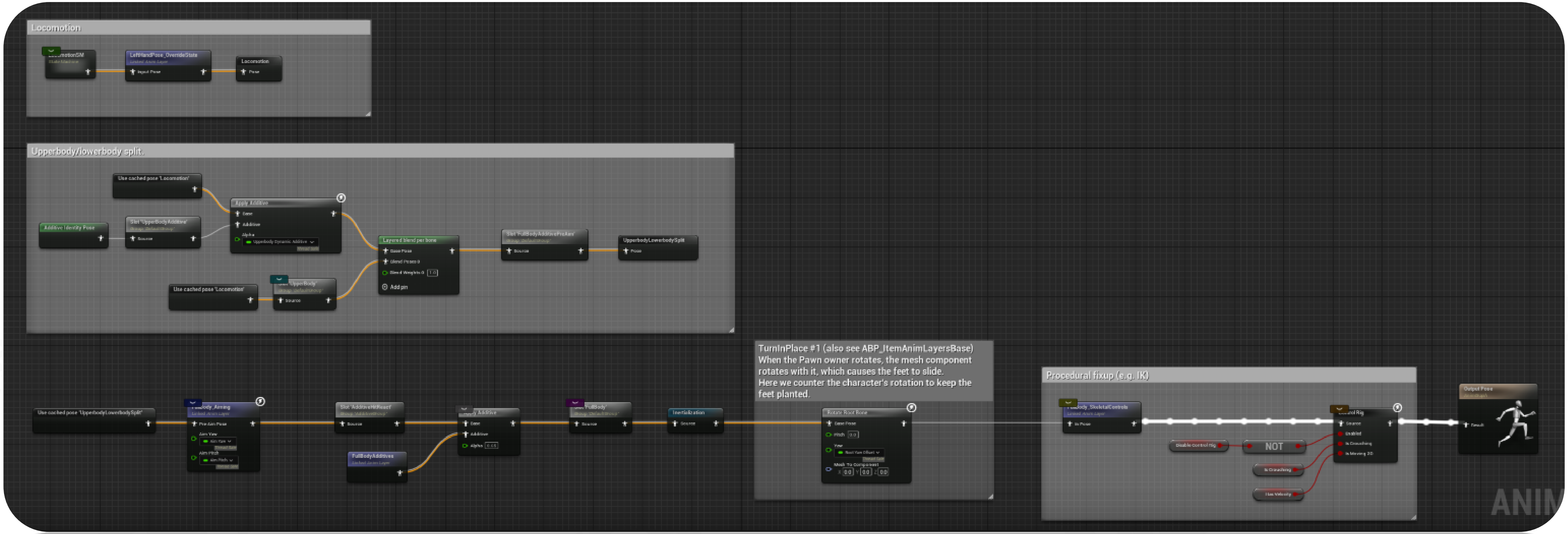


Called by event

Update Control Parameters



Unreal5 Animation Tree Sample



Inverse Kinematics (IK)

Basic Concepts

- **End-effector**

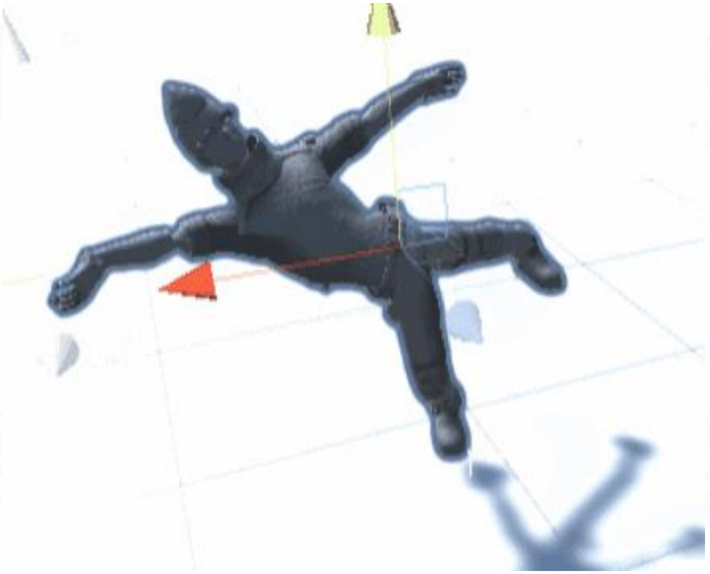
The bone which is expected to be moved to a desired position

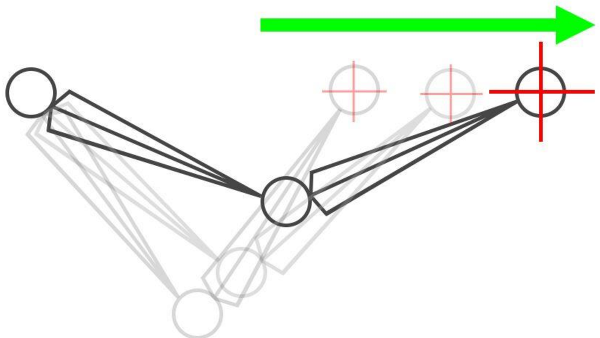
- **IK (Inverse Kinematics)**

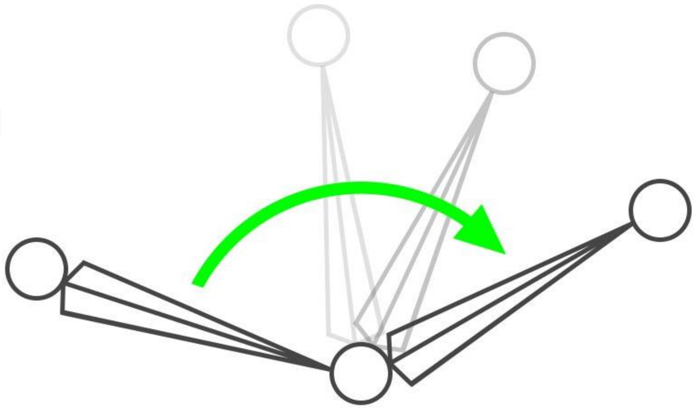
The use of kinematic equations to **determine the joint parameters** of a manipulator so that the end-effector moves to a desired position

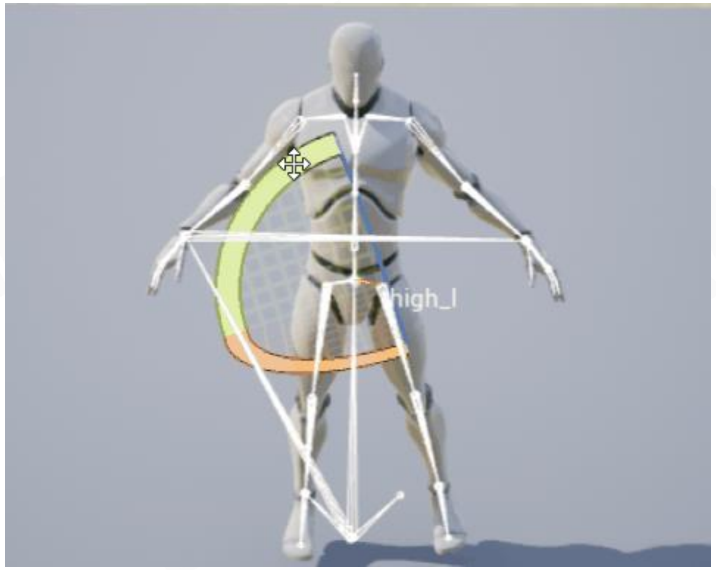
- **FK (Forward Kinematics)**

The use of the kinematics equations of a robot to **compute the position of the end-effectors** from specified values for the joint parameters

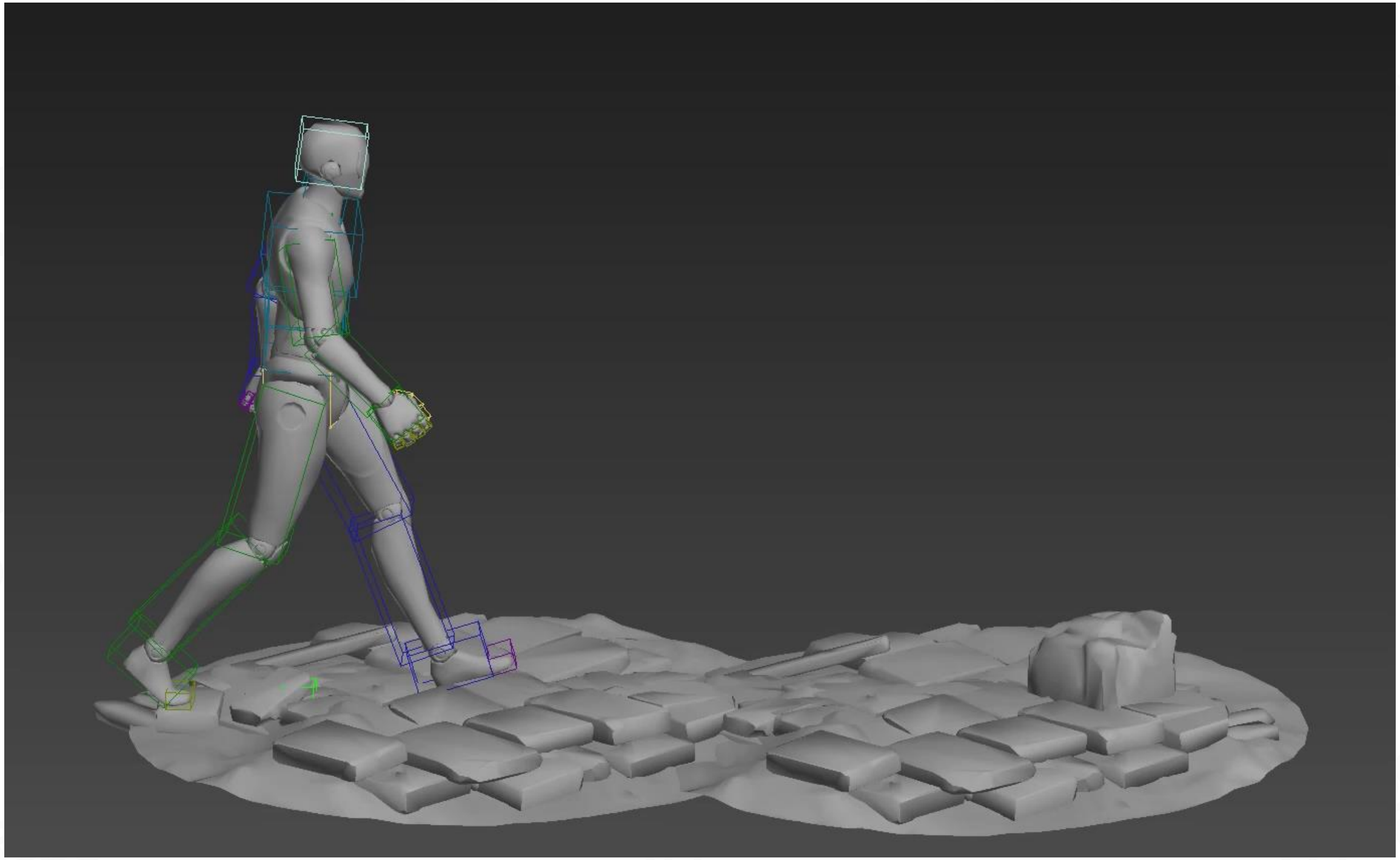




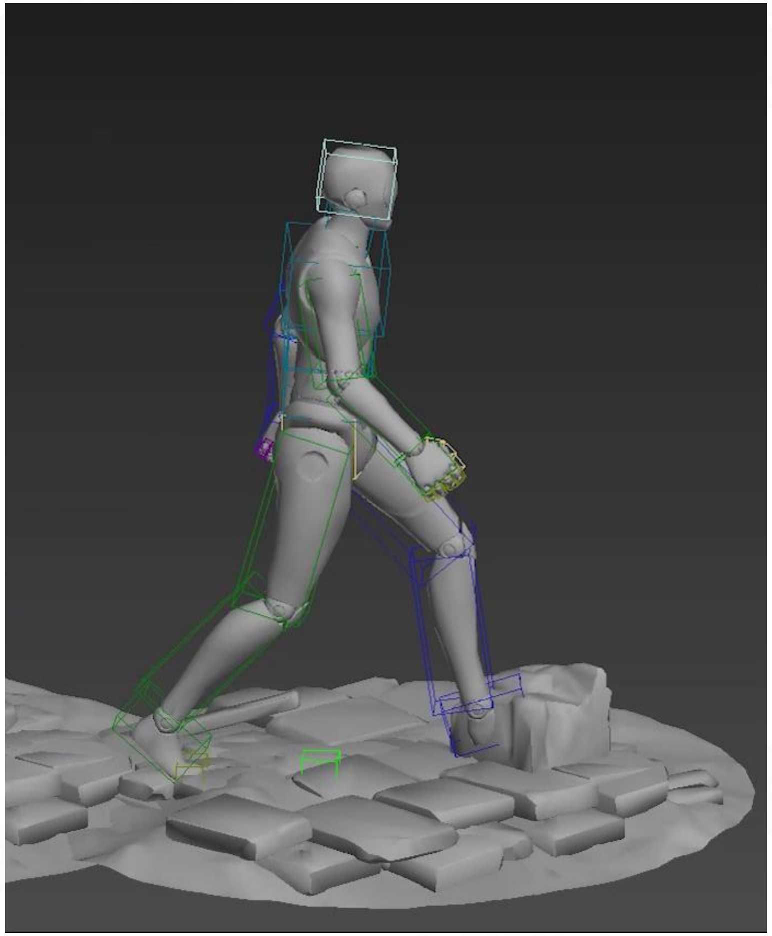




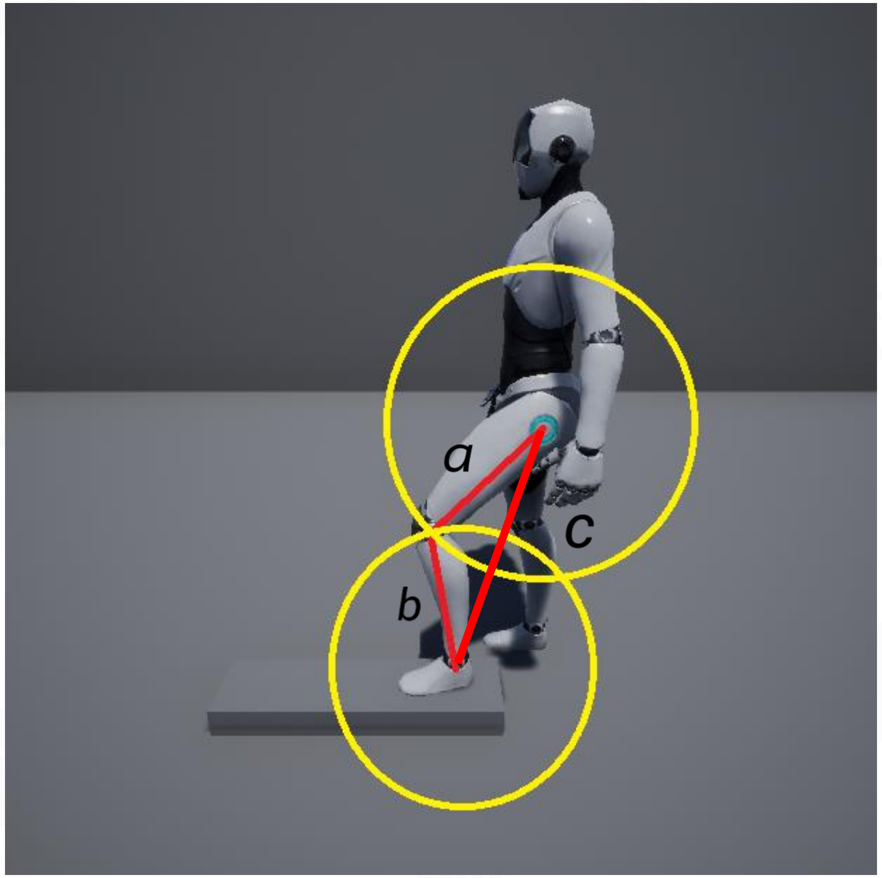
How to Touch the Ground?

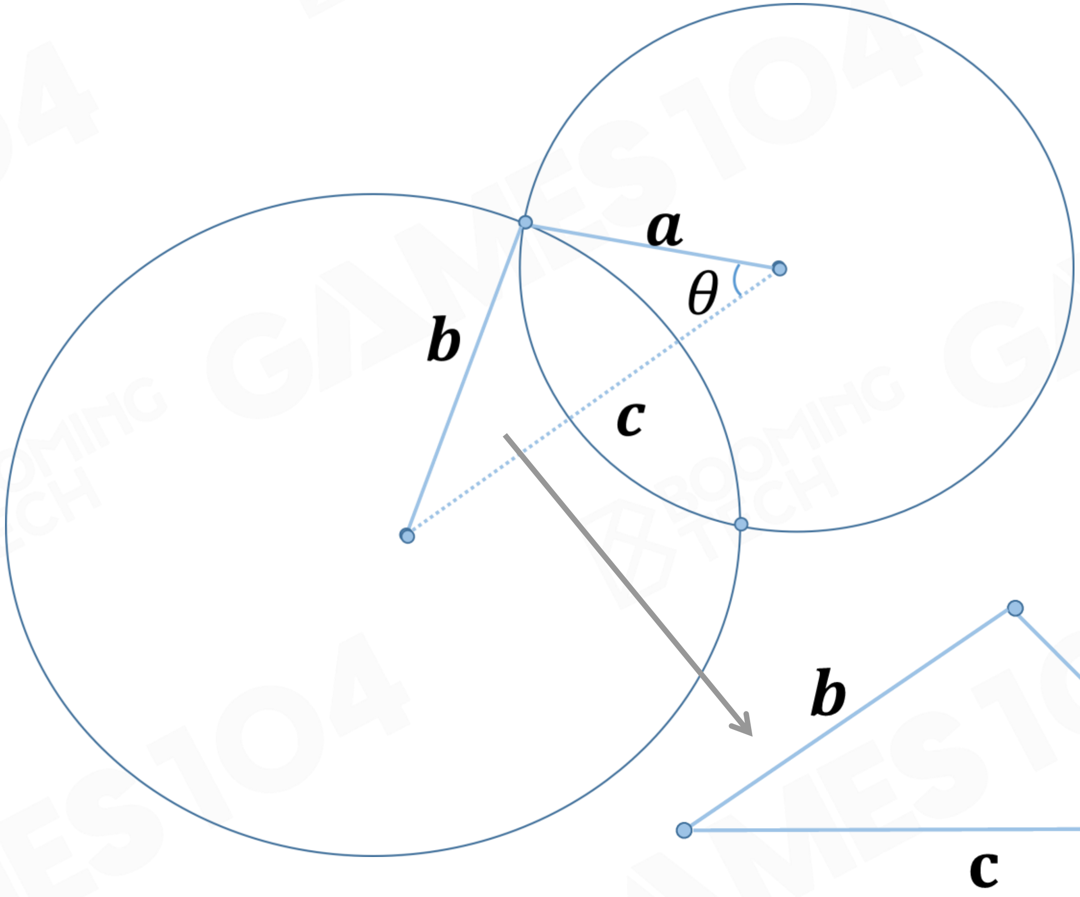


Intuitive Idea: Adjust Feet Position for Each Step



Two Bones IK

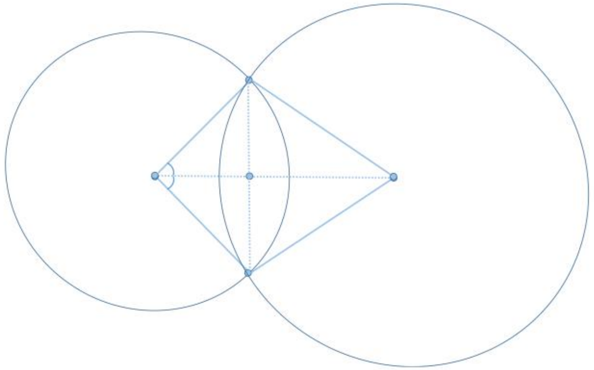




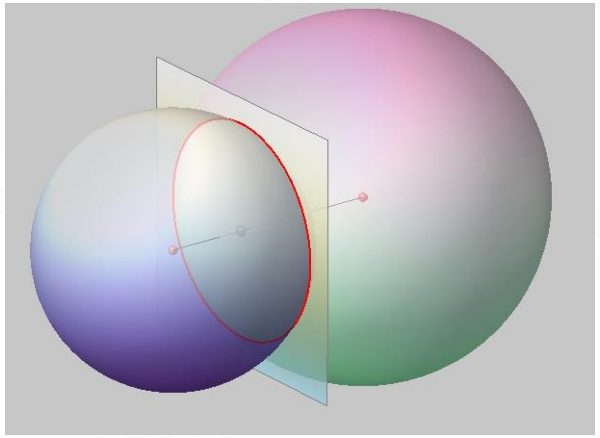
$$cos(\theta) = \frac{a^2 + c^2 - b^2}{2ac}$$

Two Bones IK

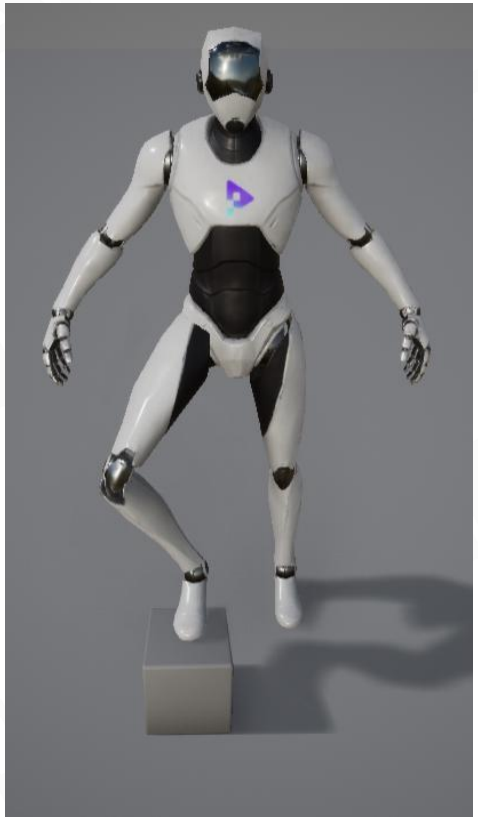
- 3D space

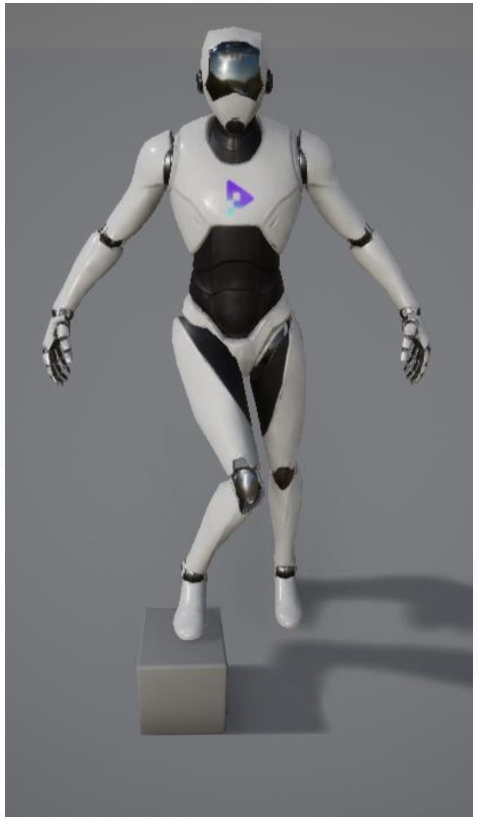


2D



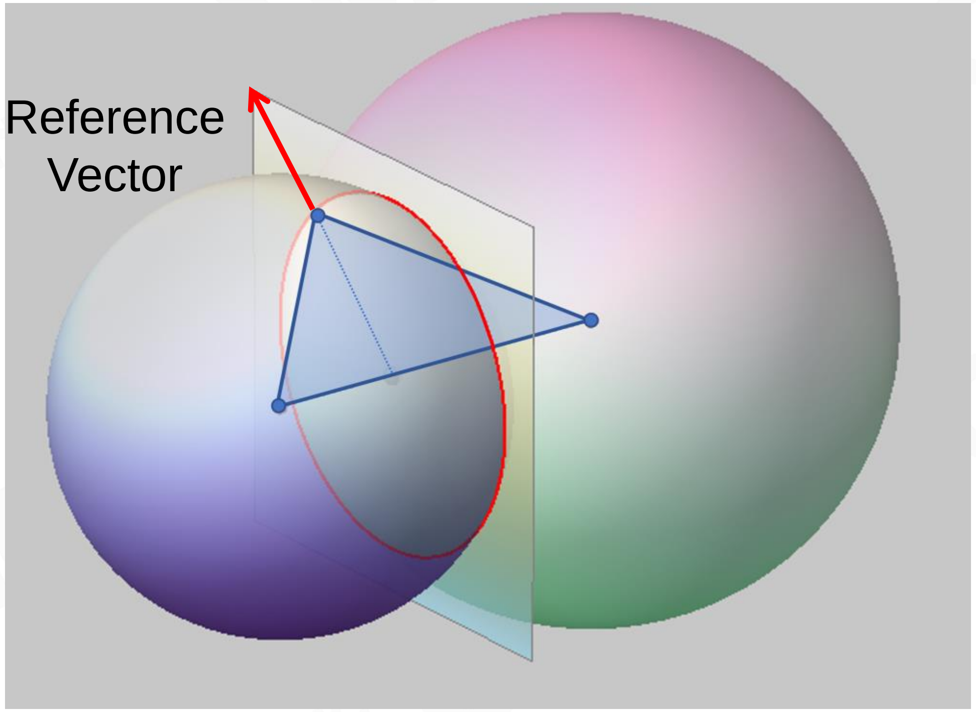
3D



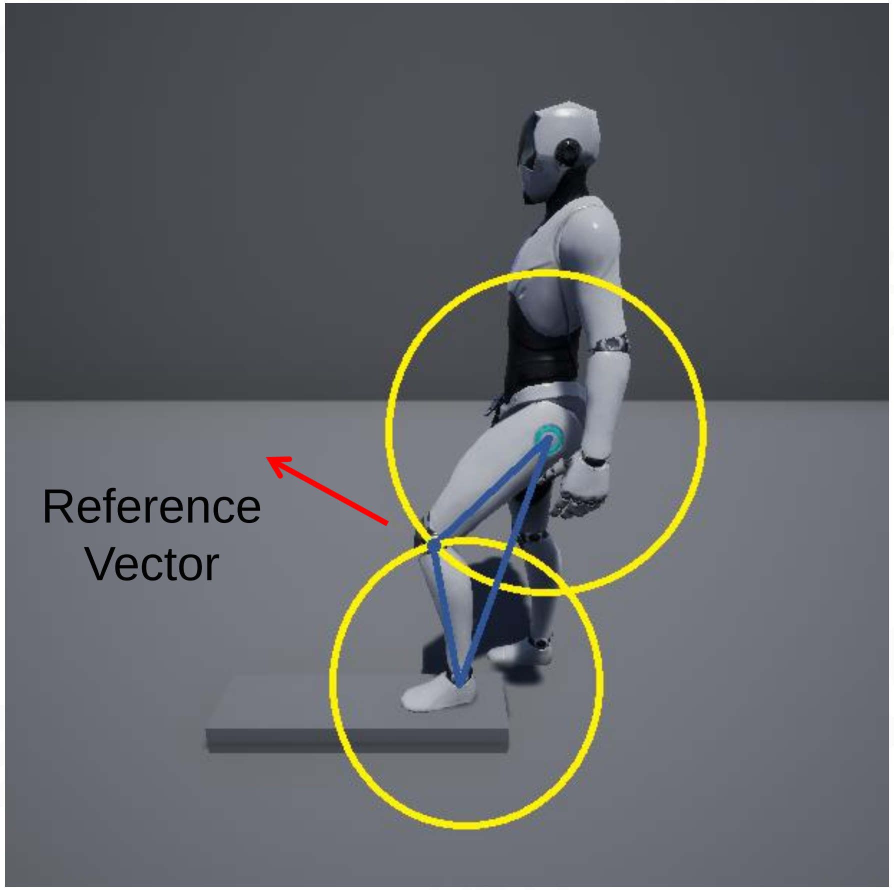


Two Bones IK

- 3D space
- Determine the final pose by a reference vector



3D View

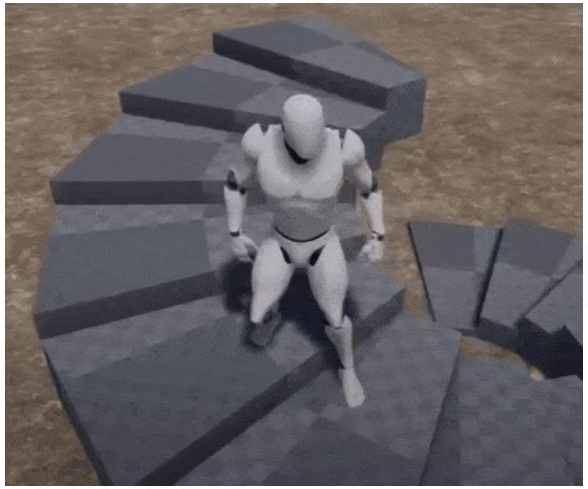


2D View of the Final Plane

More Complicated IK Scenarios

Look At IK

Hand IK

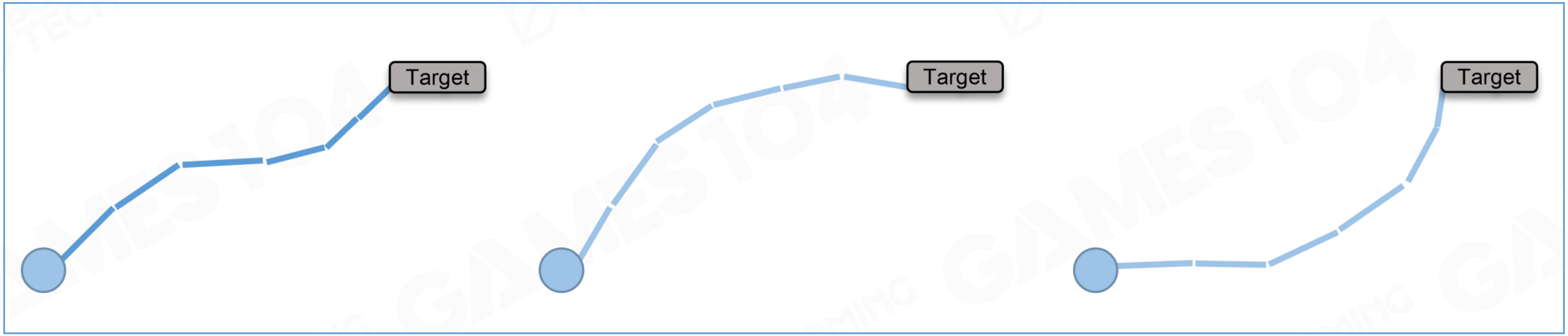


Foot IK

Full Body IK

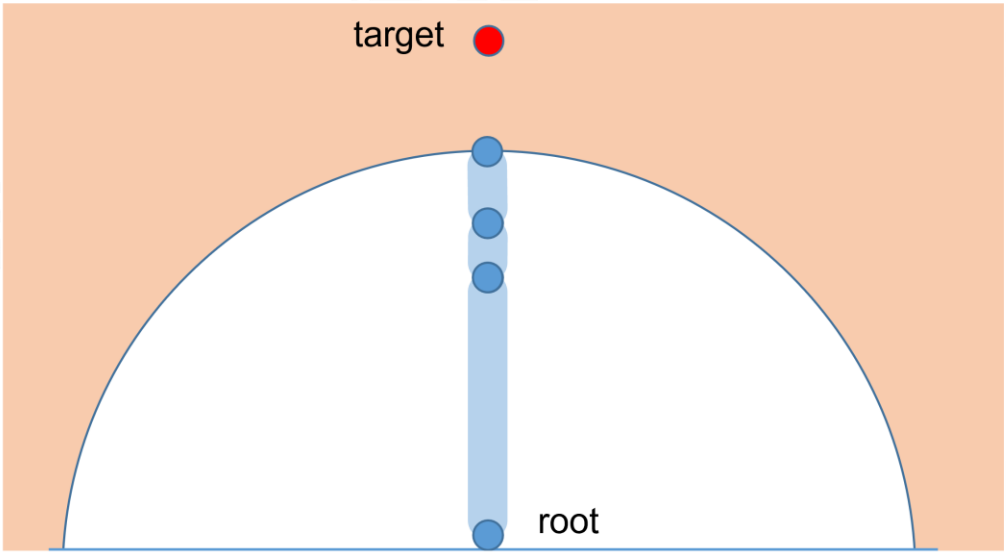
Complexity of Multi-Joint IK Solving

- Computation cost: high dimension non-linear function solving in real-time
- May have multiple solutions, unique solution or no solution

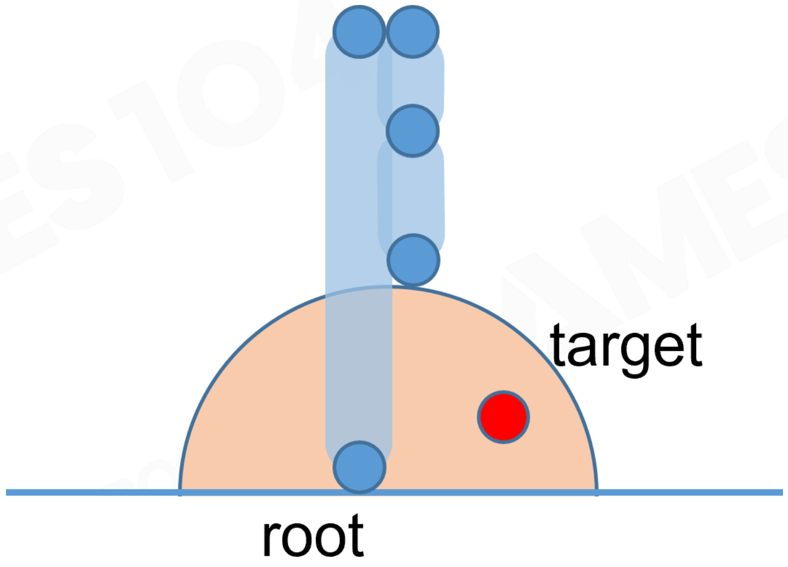


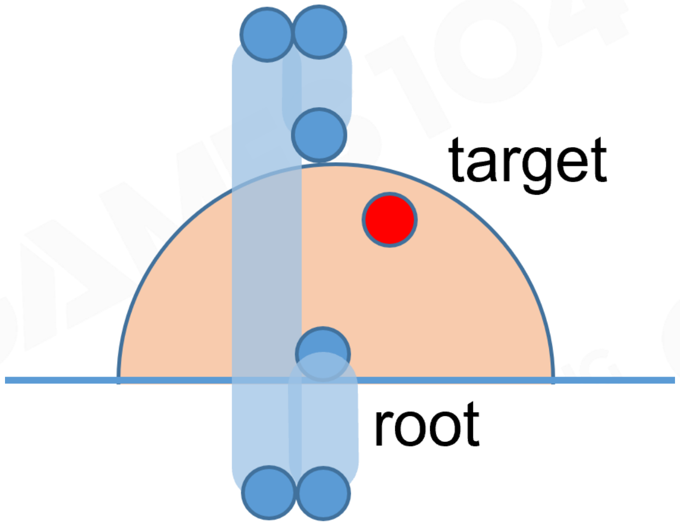
Multiple Solution

Check Reachability of the Target

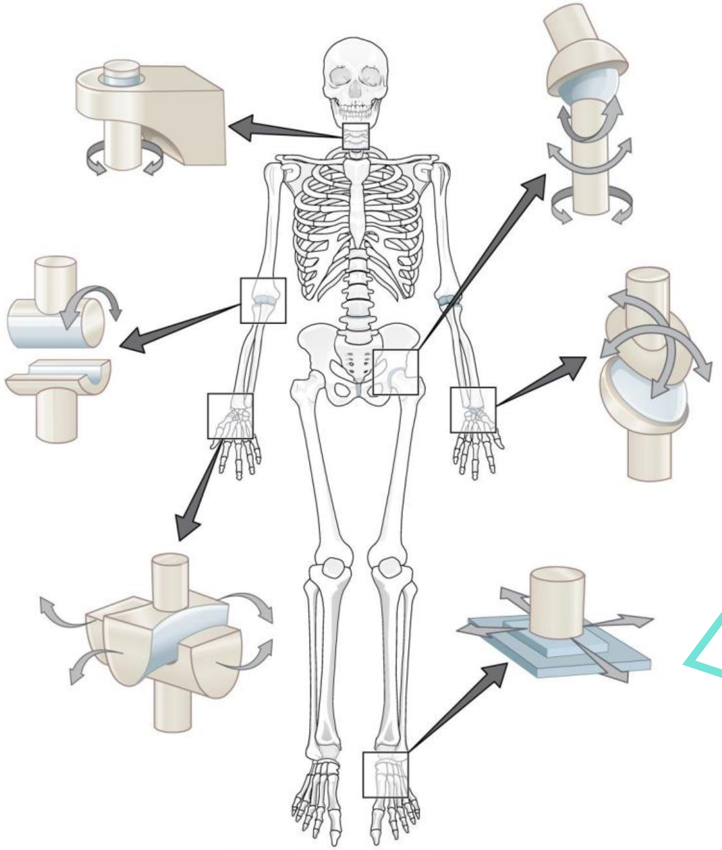


or

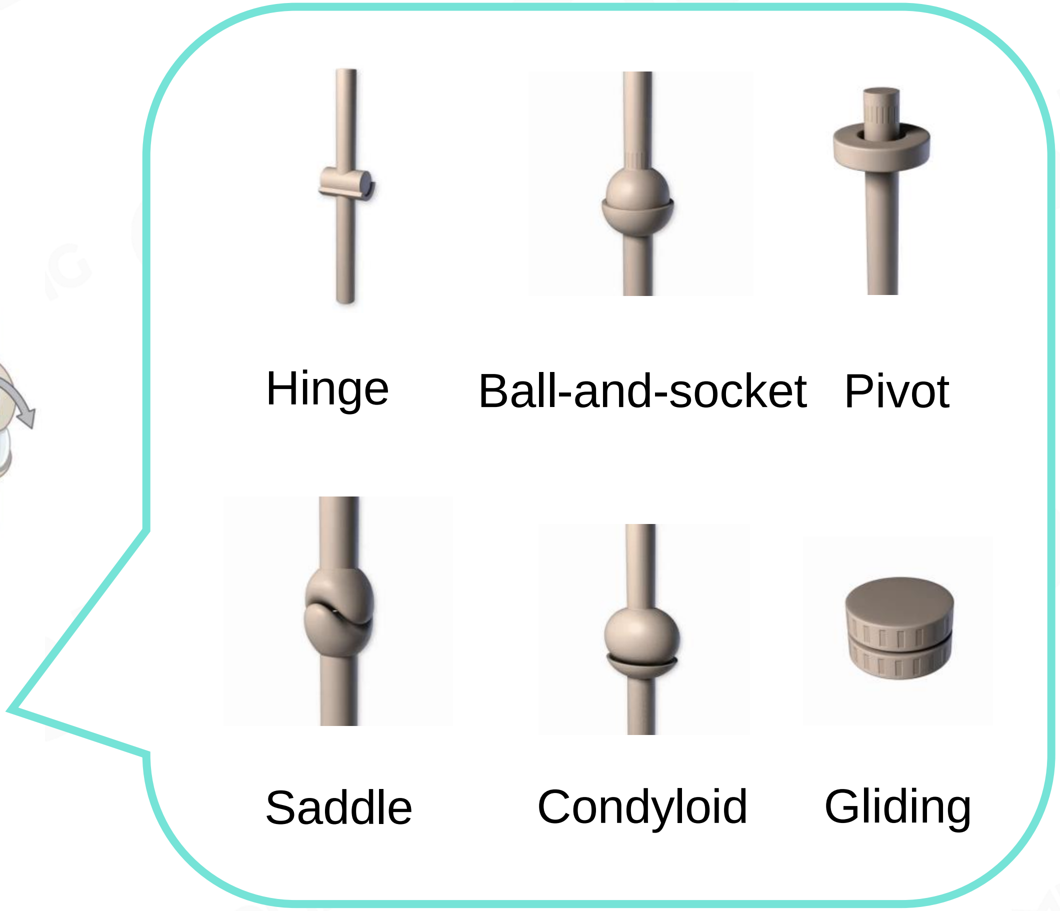




Constraints of Joints



Constraints of Human Skeleton



Need Treat Constraints Seriously

Heuristics Algorithm

Why

- Too many joints + constraints, difficult to solve with analysis method

Basic Idea

Designed to solve problem in faster and more efficient fashion by sacrificing optimality, accuracy, precision, or completeness for speed

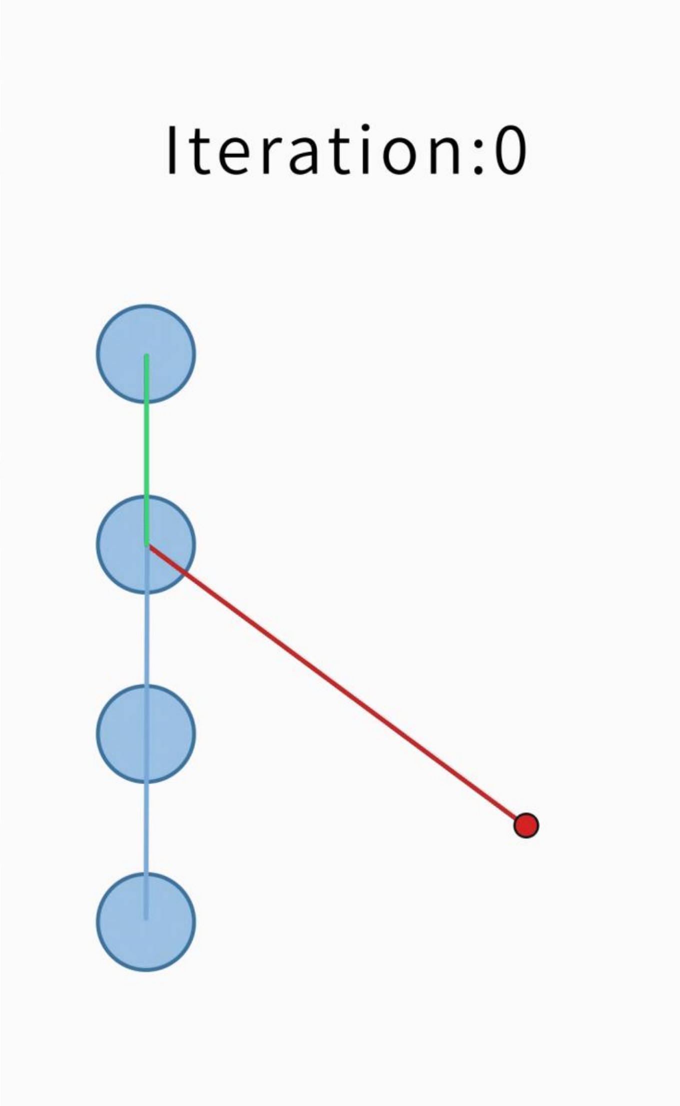
- Approximation
- Global optimality is not guaranteed
- Iteration is usually used with a maximum limit

CCD (Cyclic Coordinate Decent)

Principle
- From joint-to-joint, rotates the end-effector as close as possible to the target, solves IK problem in orientation space

Reachability
- Algorithm can stop after certain number of iterations to avoid unreachable target problem

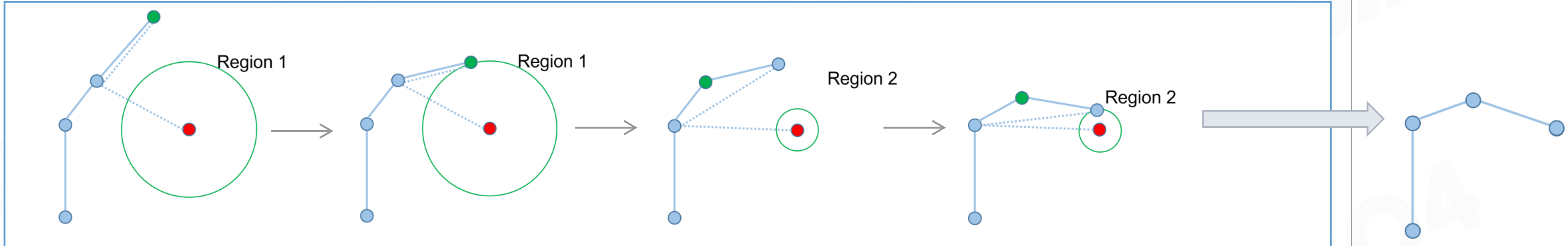
Contraints
- Angular limits is allowed, by checking after each iteration



Optimized CCD (1/2)

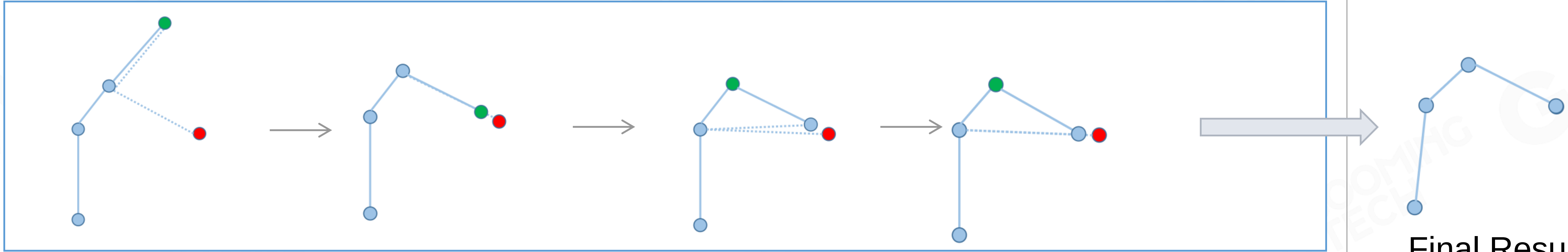
Add tolerance regions to each bone's goal

- Each bone stops rotating and moves onto the next bone within tolerance region
- Helps to produce poses that are less rigid and more comfortable looking



CCD with tolerance region

Final Result



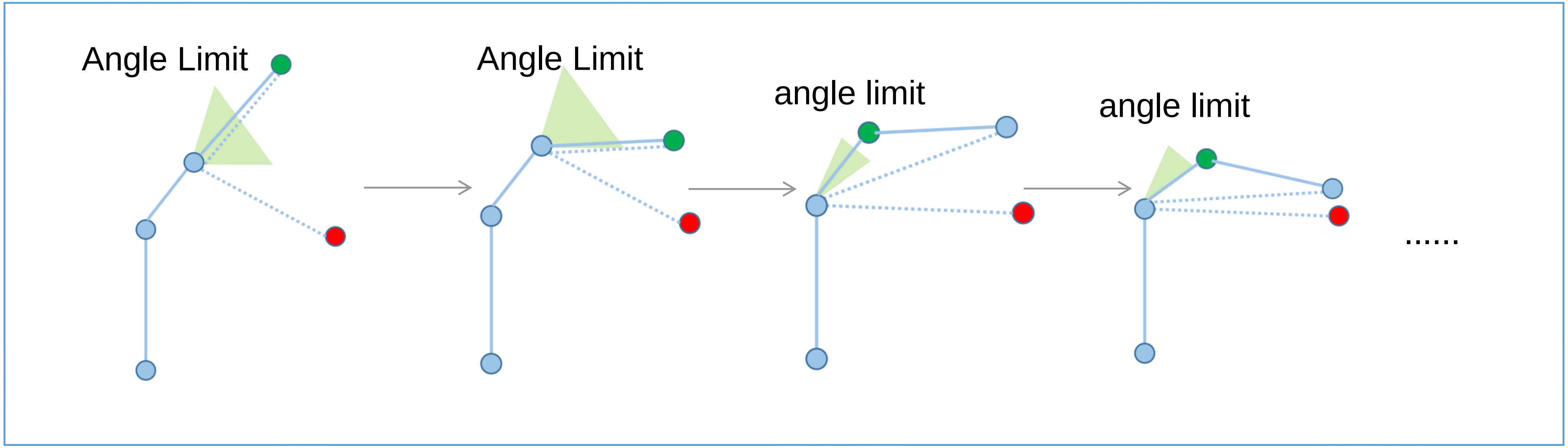
Normal CCD

Final Result

Optimized CCD (2/2)

Use under-damped angle scaling

- Each joint moves only a small amount toward the goal and distributes the movement across multiple bones
- Produce less abrupt joint changes and more smooth and casual poses for character movement



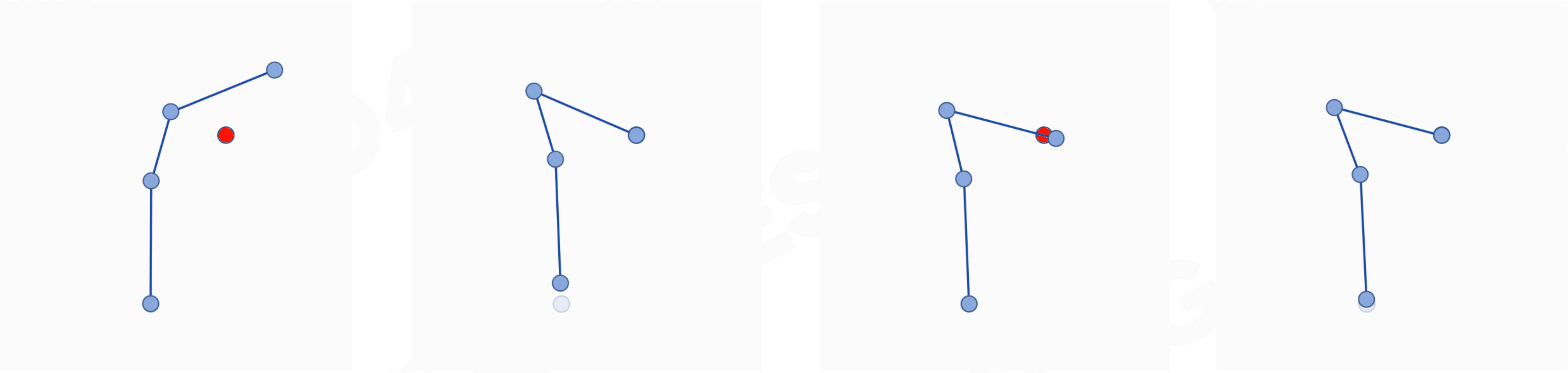
FABRIK (Forward And Backward Reaching Inverse Kinematics)

Principle

- Instead of orientation space, solves IK problem in position space

Reachability

- Algorithm can stop after certain number of iterations to avoid unreachable target problem

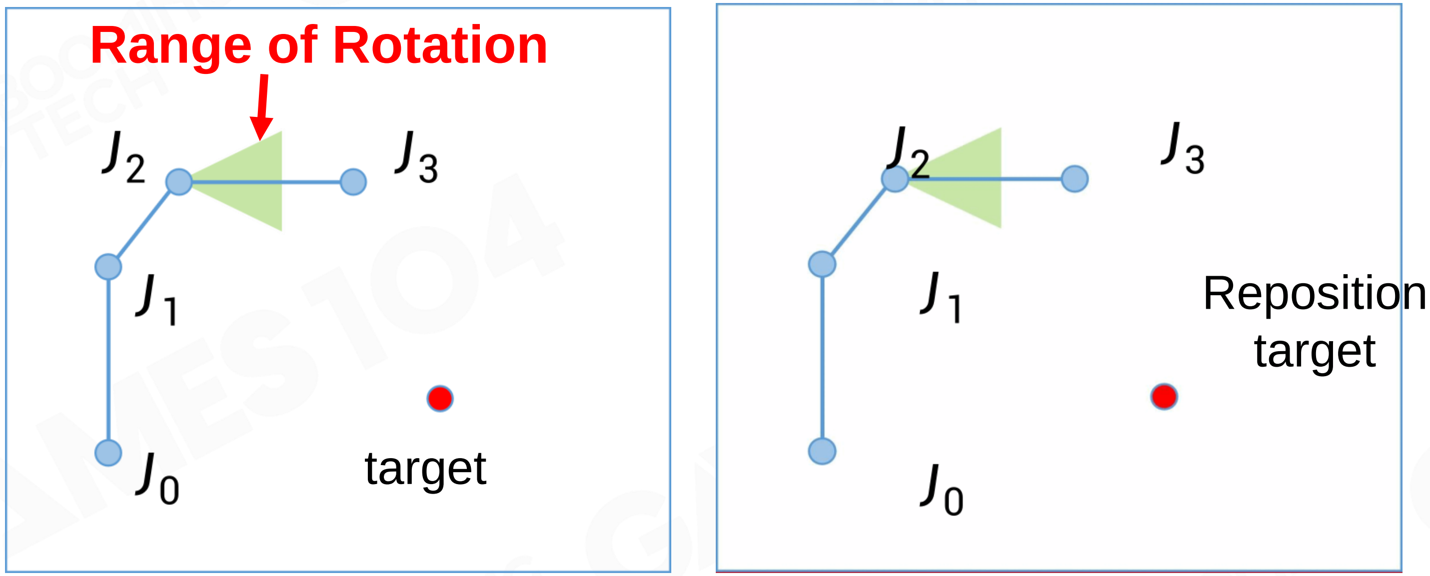


Forward(1) Backward(2) Forward(3) Backward(4)

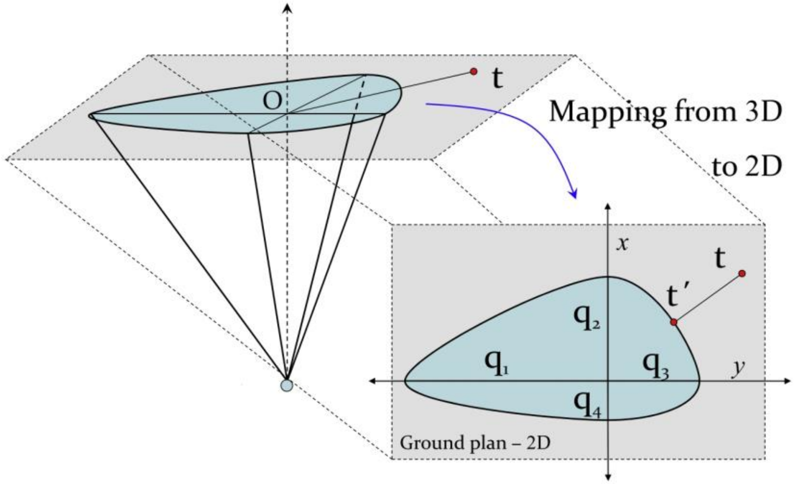
FABRIKF with Constraints

Re-positioning

- Joint restrictions can be enforced at each step by taking the resultant orientation and forcing it to stay with in the valid range



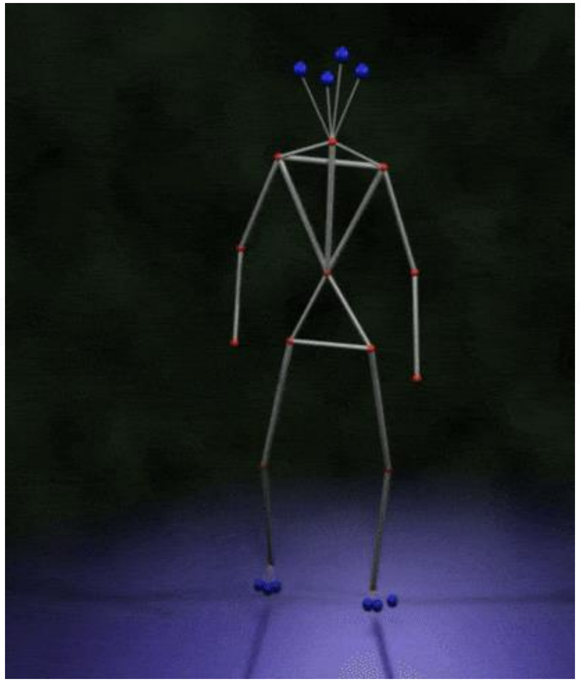
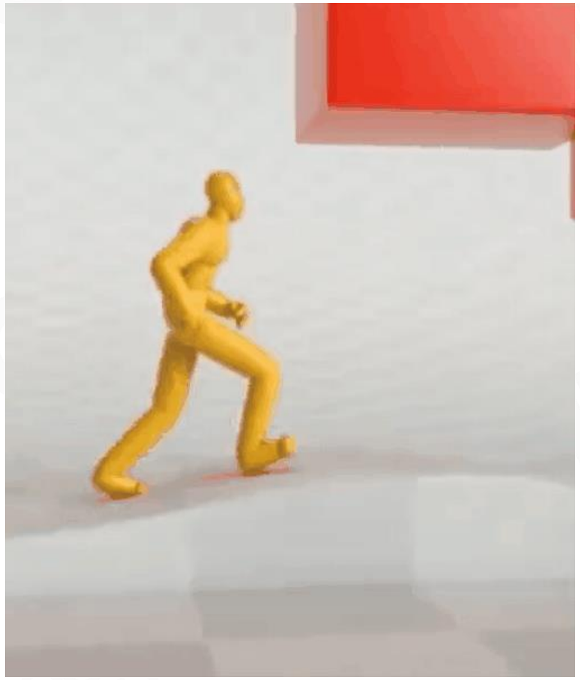
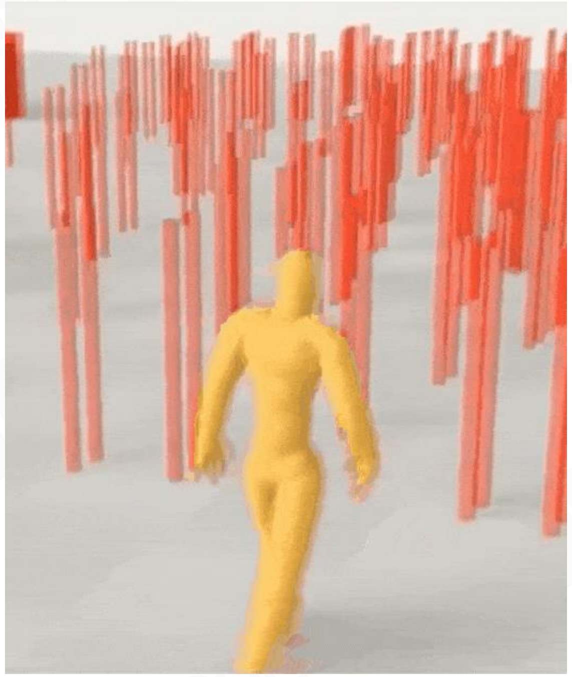
Find reposition target in 2D



3D constraints

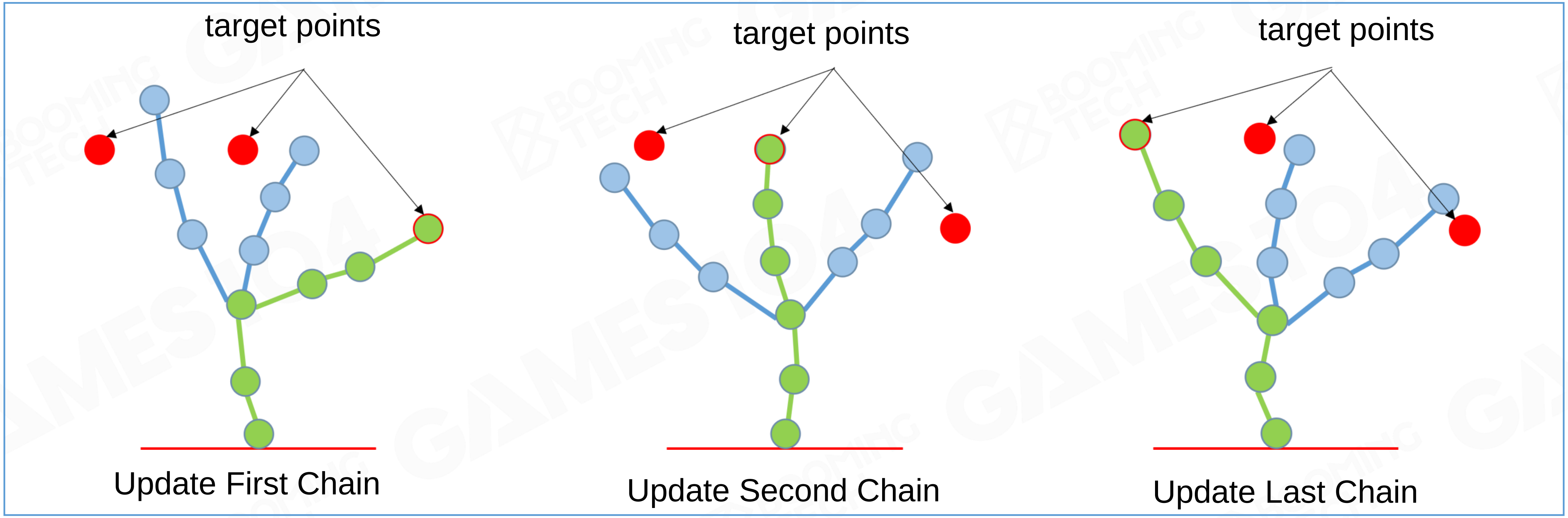
Multiple End-Effectors

- May result in conflict between goals, which can not be achieved simultaneously
- May use a priority or a weighted approach

IK with Multiple End-Effectors

- If a shared bone needs to be moved, the end-effector that is updated last will get priority and the other bones will be pulled away



Jacobian Matrix

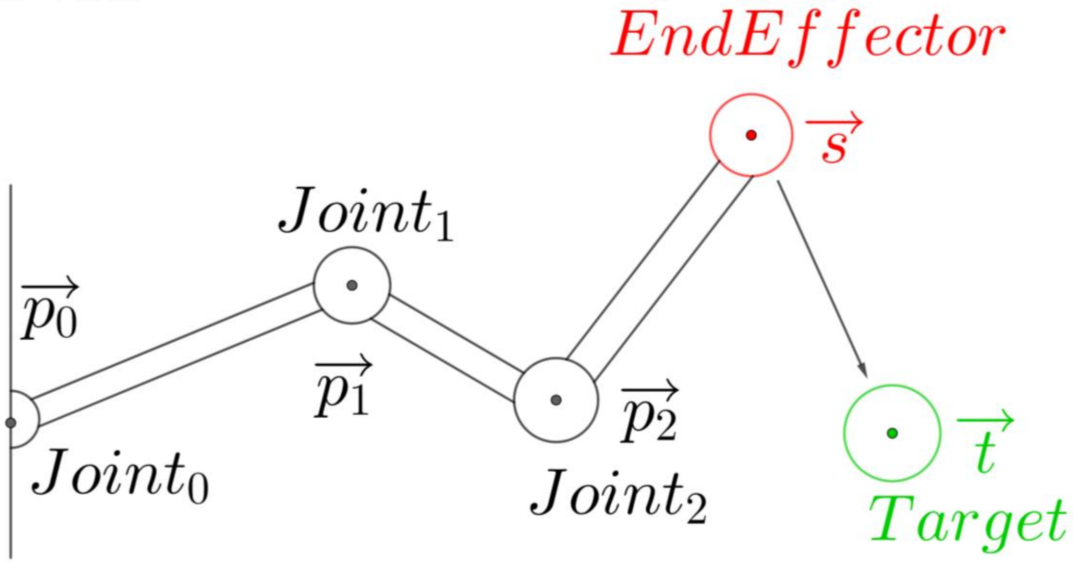
In vector calculus, the Jacobian Matrix of a vector-valued function of several variables is the matrix of all its first-order partial derivatives

Suppose:

$$\vec{f}(\vec{x}) = \begin{bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_m(\vec{x}) \end{bmatrix} \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

then, the Jacobian Matrix of $\vec{f}(\vec{x})$:

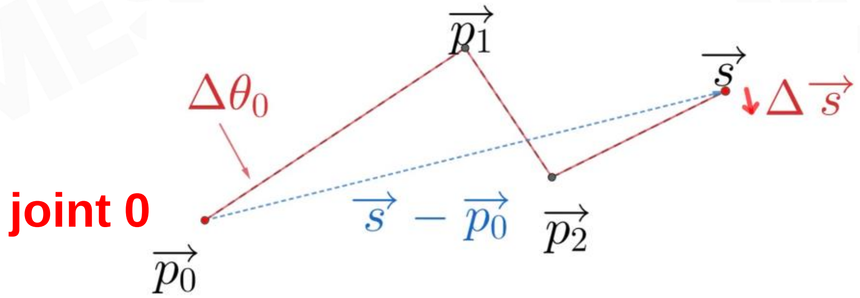
$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$



Using Jacobian Matrix to Present Joint Rotations

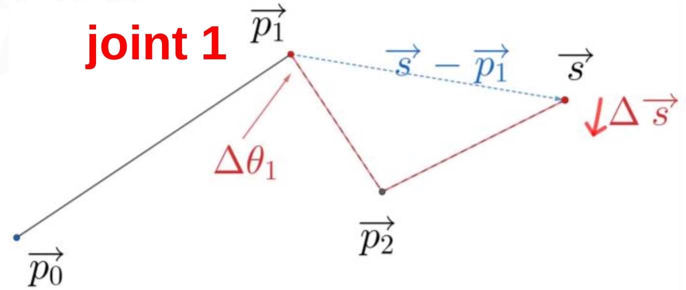
Only rotate joint 0:

$$\Delta\vec{s} = \vec{u_0} \times (\vec{s} - \vec{p_0})\Delta\theta_0$$



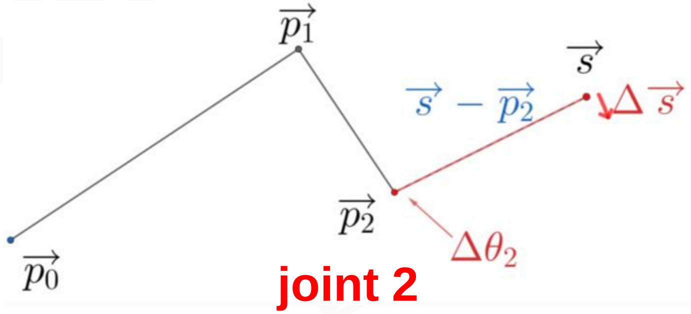
Only rotate joint 1:

$$\Delta\vec{s} = \vec{u_1} \times (\vec{s} - \vec{p_1})\Delta\theta_1$$



Only rotate joint 2:

$$\Delta\vec{s} = \vec{u_2} \times (\vec{s} - \vec{p_2})\Delta\theta_2$$



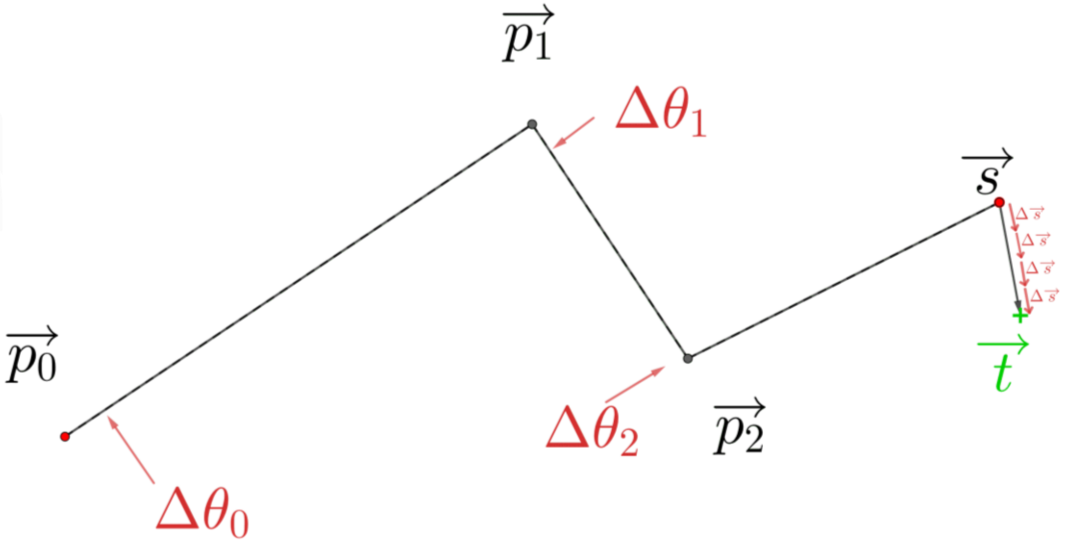
Jacobian Matrix with Multiple End-effectors

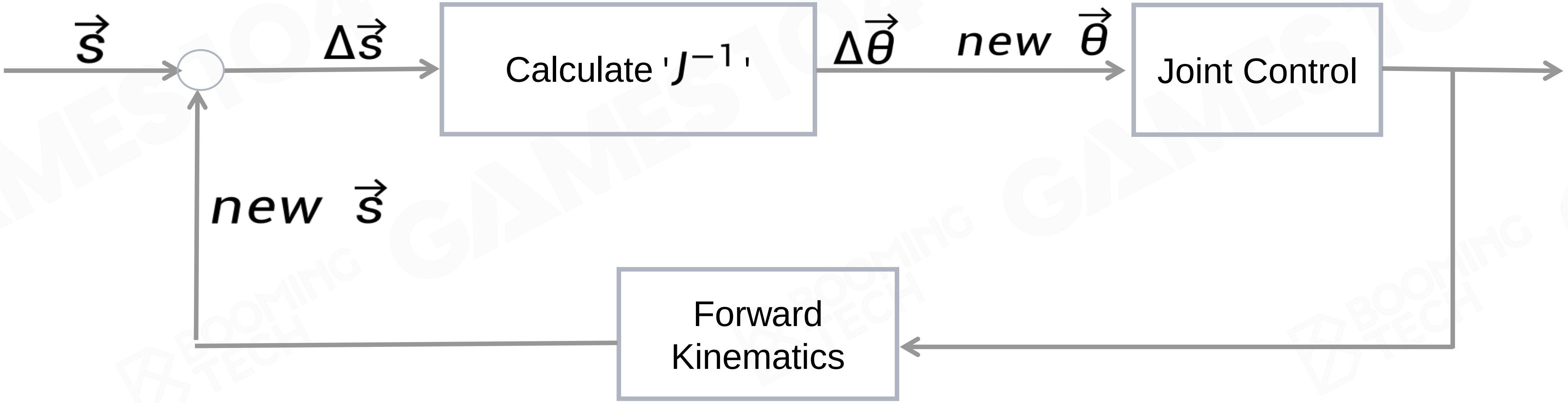
$$J = \begin{bmatrix} \frac{\partial \overrightarrow{s_1}}{\partial \theta_1} & \frac{\partial \overrightarrow{s_1}}{\partial \theta_2} & \cdots & \frac{\partial \overrightarrow{s_1}}{\partial \theta_n} \\ \frac{\partial \overrightarrow{s_2}}{\partial \theta_1} & \frac{\partial \overrightarrow{s_2}}{\partial \theta_2} & \cdots & \frac{\partial \overrightarrow{s_2}}{\partial \theta_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \overrightarrow{s_m}}{\partial \theta_1} & \frac{\partial \overrightarrow{s_m}}{\partial \theta_2} & \cdots & \frac{\partial \overrightarrow{s_m}}{\partial \theta_n} \end{bmatrix}$$

m : the number of end-effectors

n : the number of joints

Approaching to Target Step by Step





Other IK Solutions

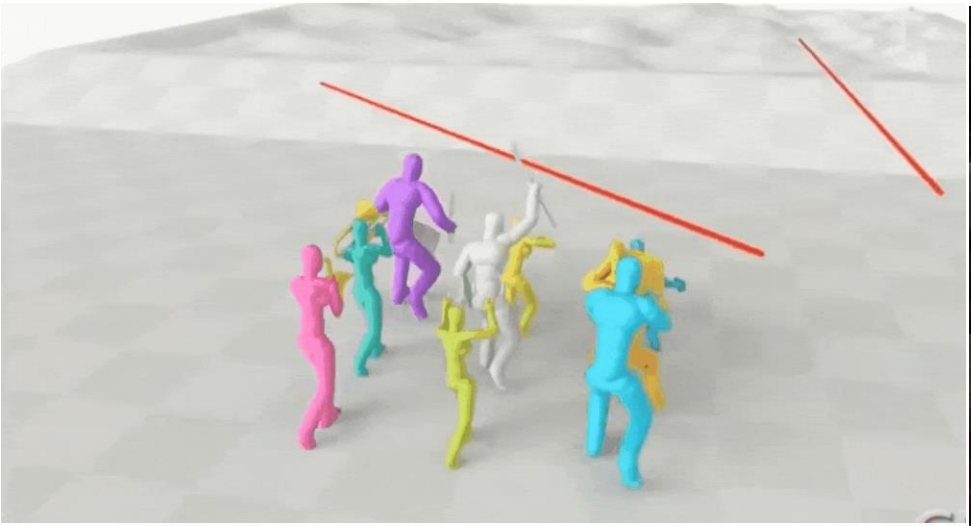
Physics-based Method
- More natural
- Usually need lots of computation if no optimization

PBD(Position Based Dynamics)
- Different from traditional physics-based method
- Better visual performance
- Lower computational cost

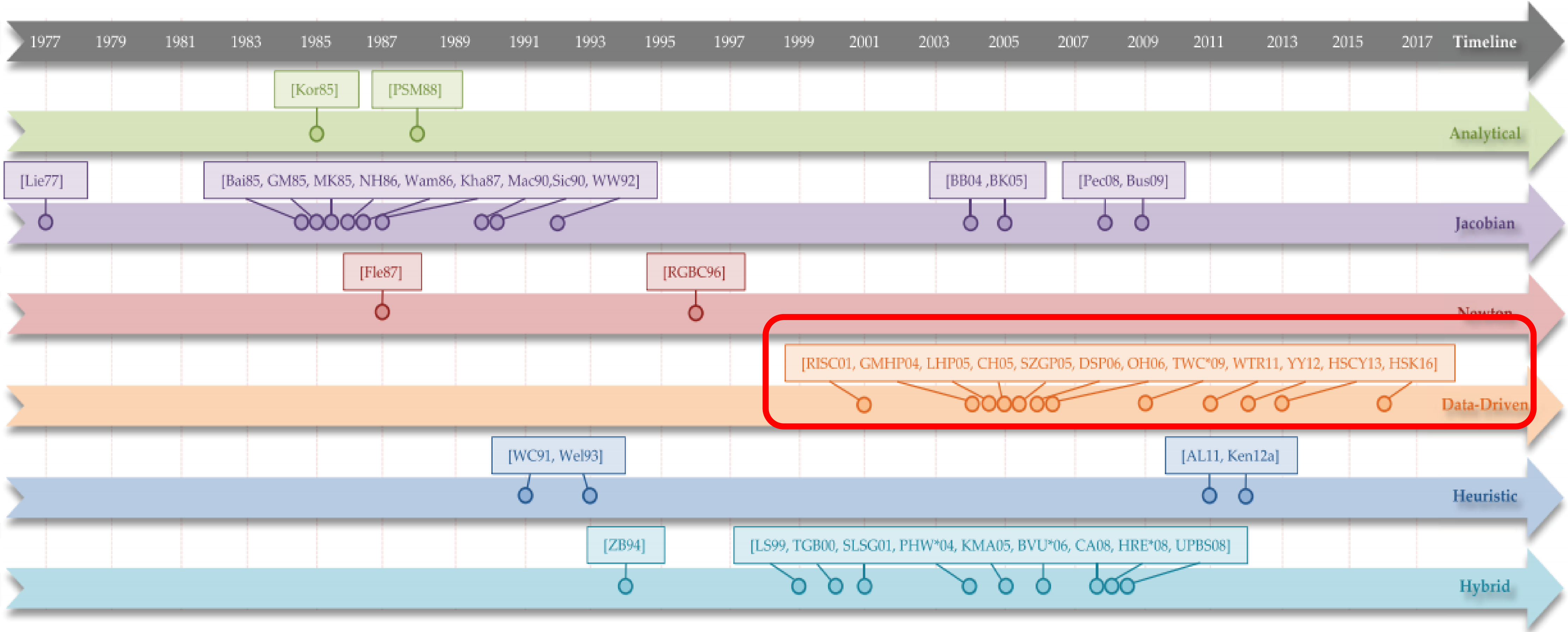
Fullbody IK in UE5
- XPBD（Extended PBD）

IK is still Challenge

- **Self collision avoidance**
- **IK with predication during moving**
- **Natural human behavior**
    - Data-driven and deep learning

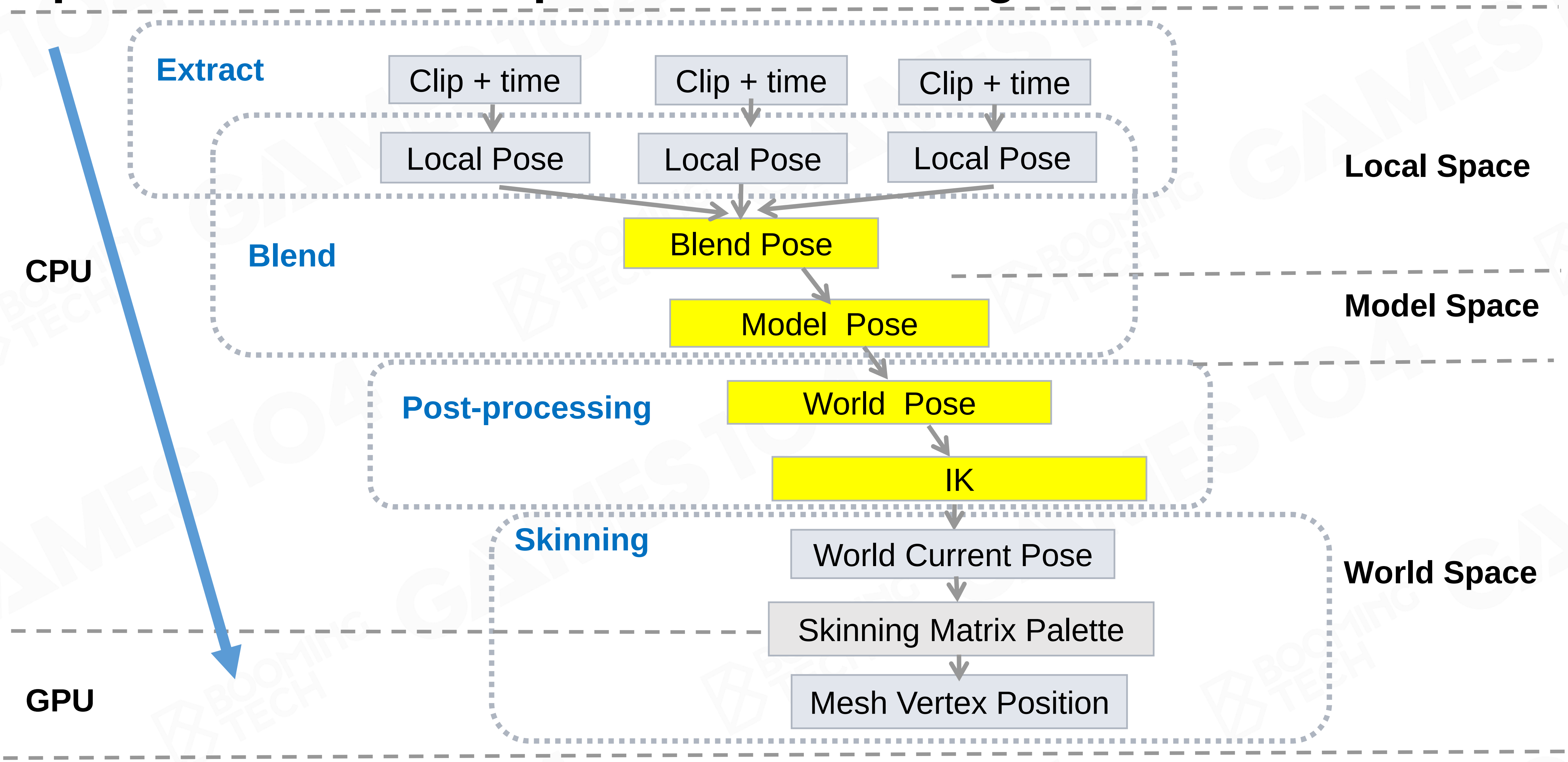


IK Hot Research Areas



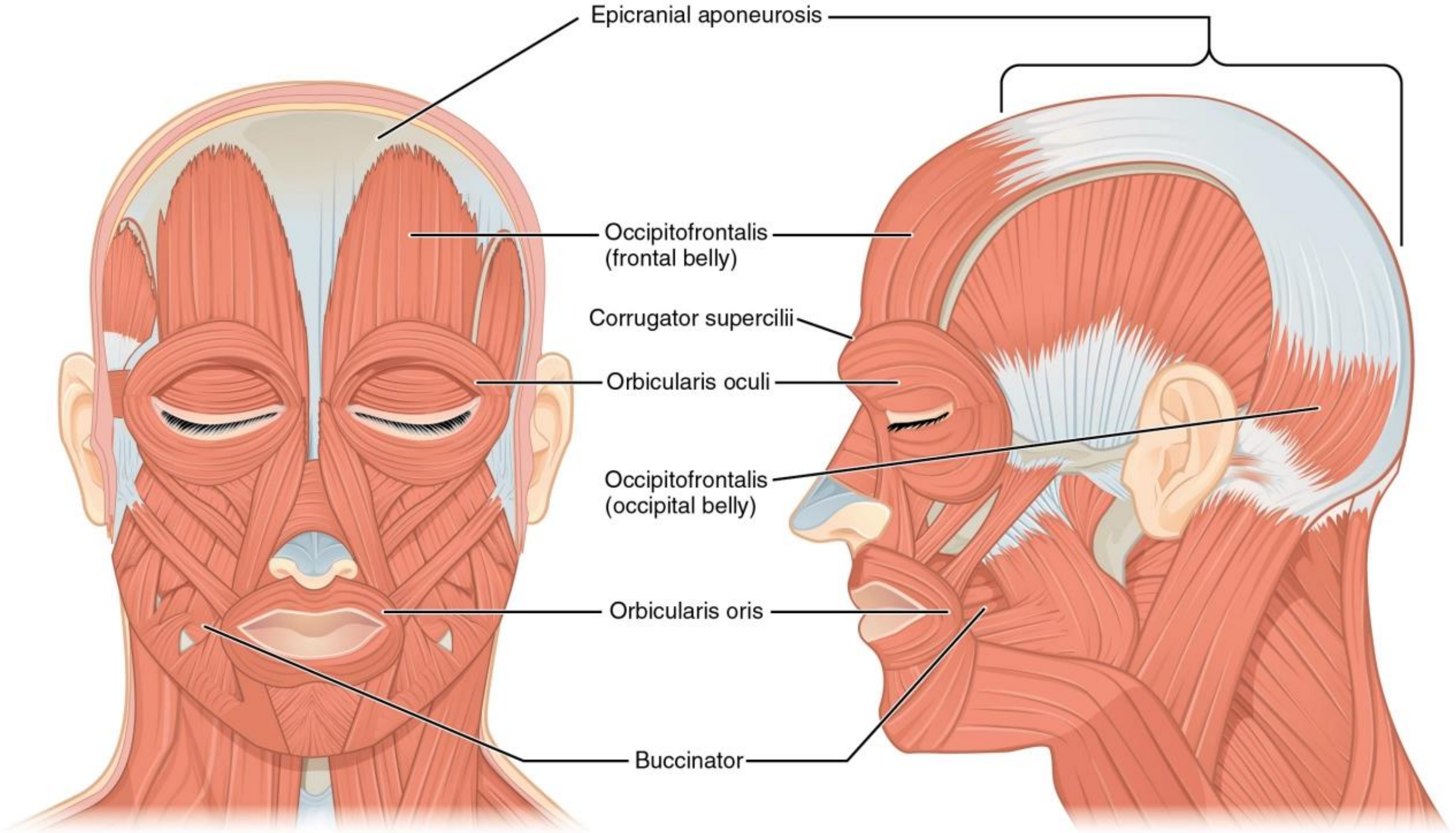
From *Inverse Kinematics Techniques in Computer Graphics: A Survey*

Updated Animation Pipeline with Blending and IK



Facial Animation

Face is Driven by Complex Muscle System



Facial Muscles

- 43 Muscles
- Variant shape, strength and movement
- Work together to make expressions

High Precision Requirements

Minor change makes difference:
- Voluntary / Forced
- Natural / Intentional
- Sometimes shows quite opposite expressions

'In Love' Stare to 'Hate You' Stare

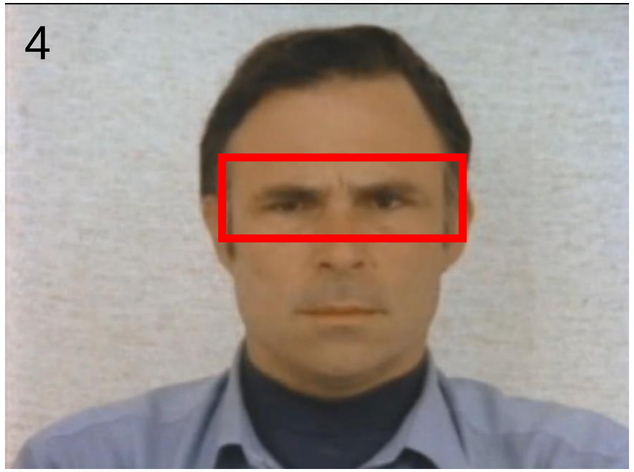
Facial Action Coding System

Facial Action Coding System (FACS) is a system to taxonomize human facial movements by their appearance on the face.

Part of 46 basic movements are named action units(AU)

Action Units Combination

An expression can be considered as a combination of some of the basic movements



Brow Lowerer

+

Upper Lid Raiser

=

‘4+5’ Combination

28 Core Action Units

- Apple Inc. extracted the 28 core AUs
- 23 Symmetric AUs are divided into two basic actions
- Basic actions set varies accoring to the animation production requirements

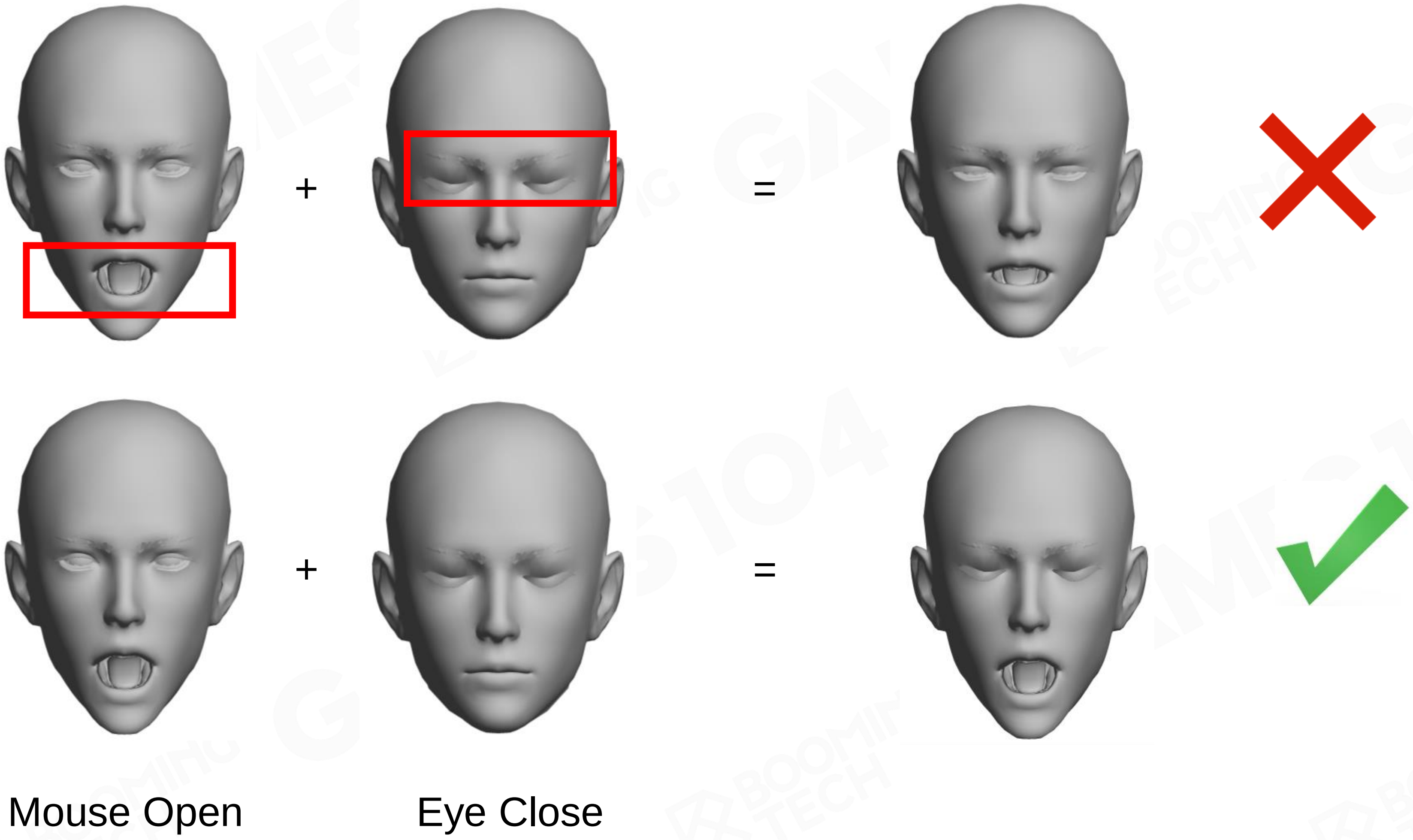
Neutral face	Lip corner depressor
Inner brow raiser	Lower lip depressor
Outer brow raiser	Chin raiser
Brow lowerer	Lip pucker
Upper lid raiser	Tongue show
Cheek raiser	Lip stretcher
Lid tightener	Neck tightener
Lips toward each other	Lip funneler
Nose wrinker	Lip tightener
Upper lip raiser	Lip pressor
Nasolabial deepener	Lips part
Lip conner puller	Jaw drop
Sharp lip puller	Mouth stretch
Dimpler	Lip suck

28 Core AUs

Key Pose Blending

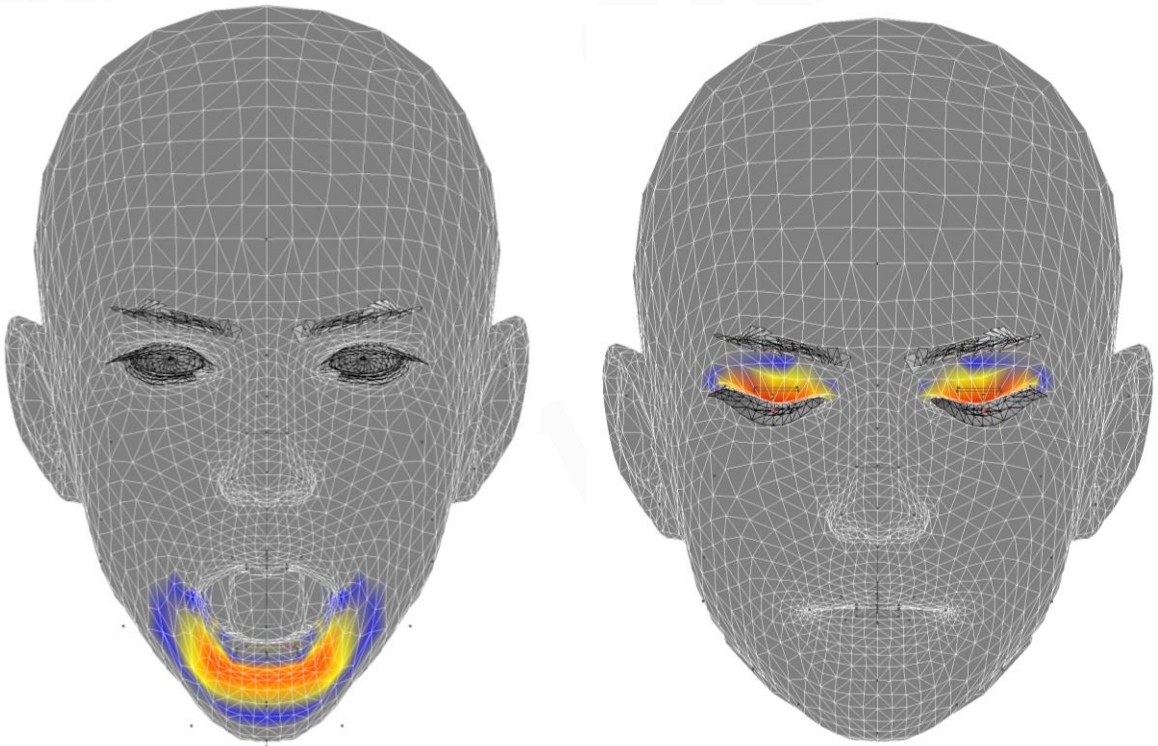
A set of key poses (a variation on per-vertex animation)

Problems of Simple Blending



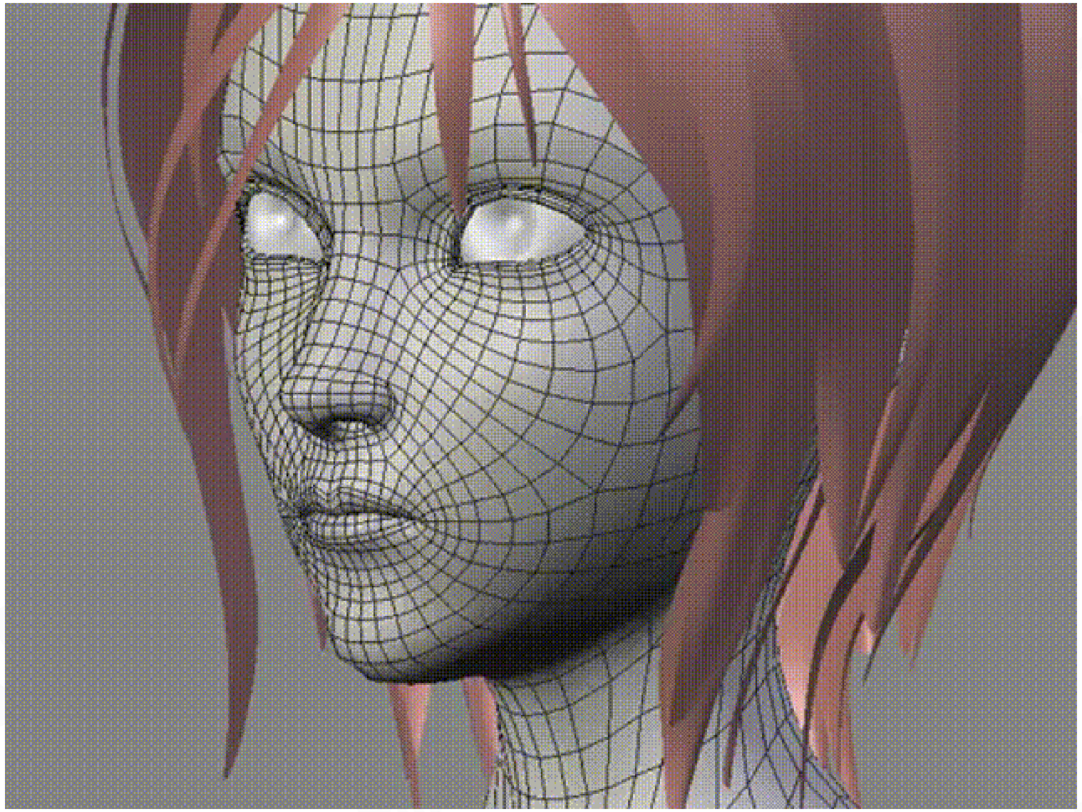
FACS In Morph Target Animation

- Create AU key poses only store vertexes different from the neutral pose (Additive Blending)

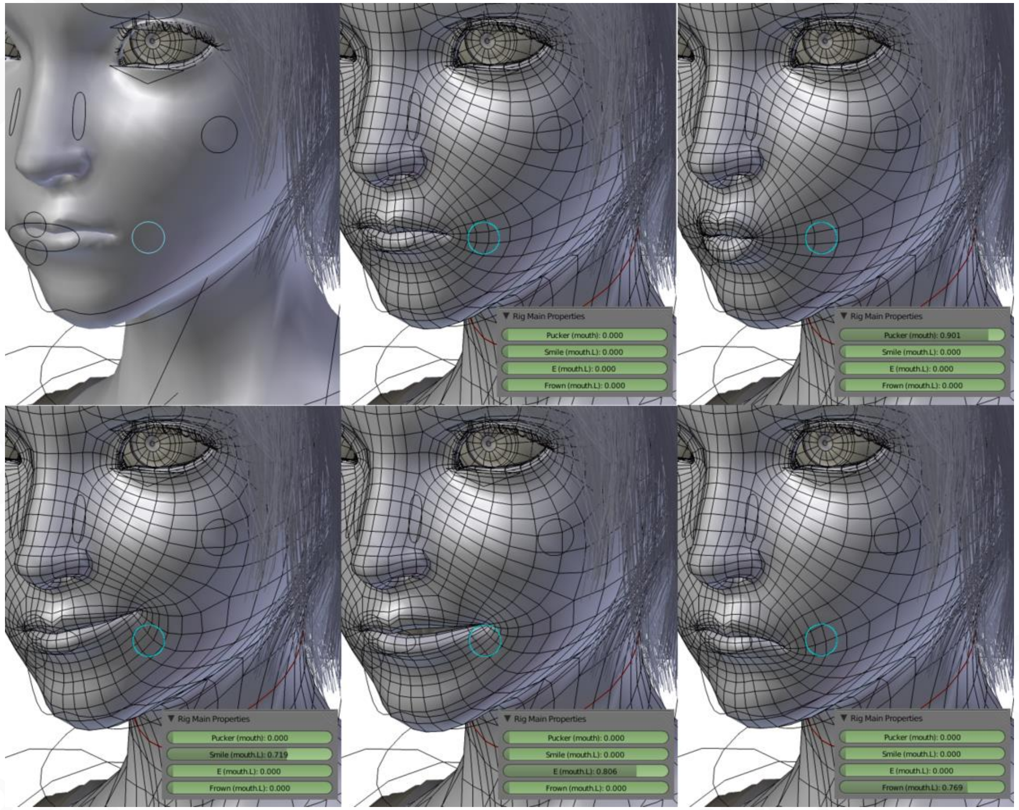


Vertex offset from neutral face

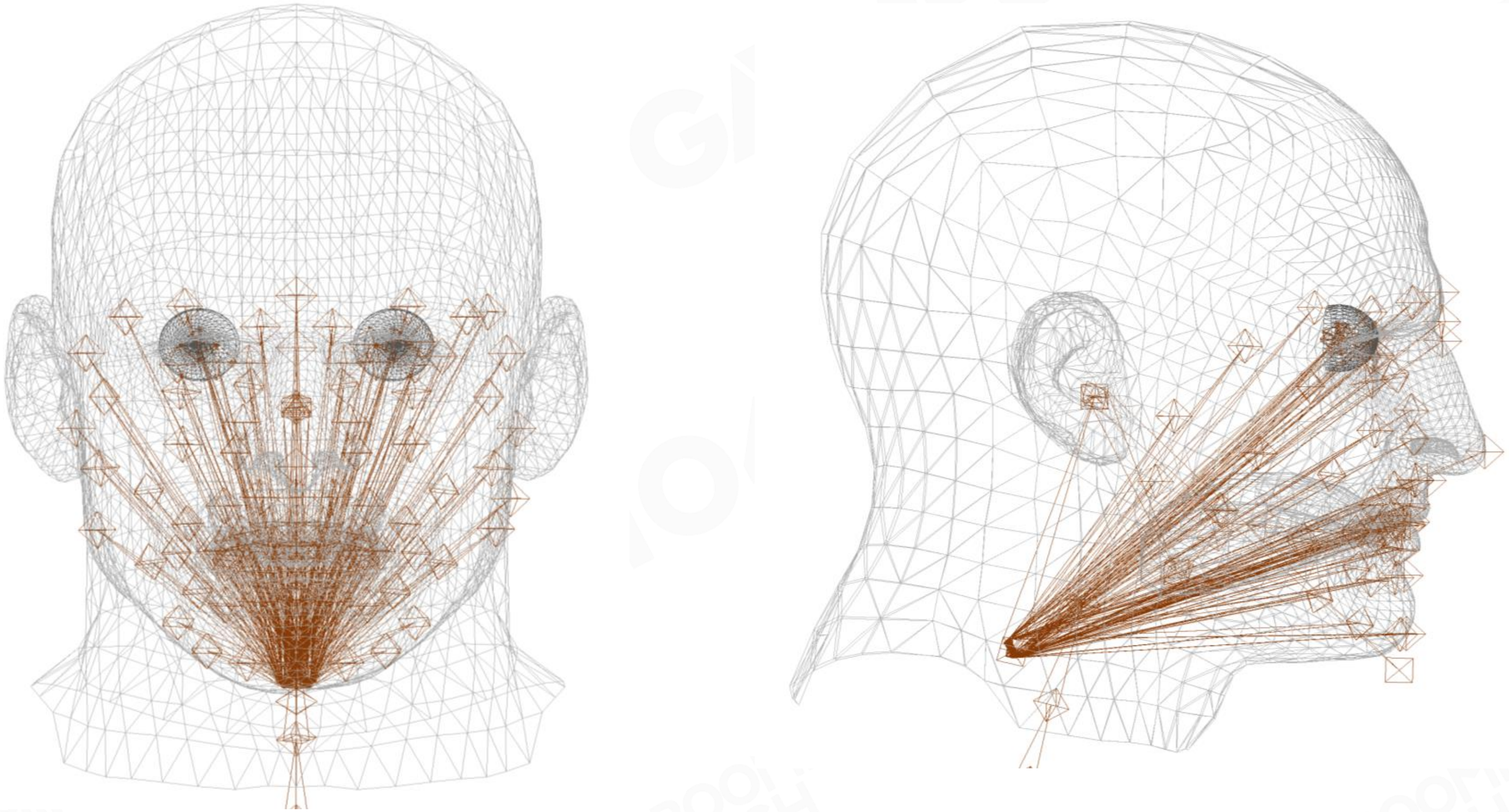
Morph Target Animation



Facial animation by morphing among key poses



Key Poses



Complex Facial Skeleton

UV Texture Facial Animation

- Using a series of texture maps applied to a simple head shape

Animal Crossing: New Horizons

The Legend of Zelda: Breath of the Wild

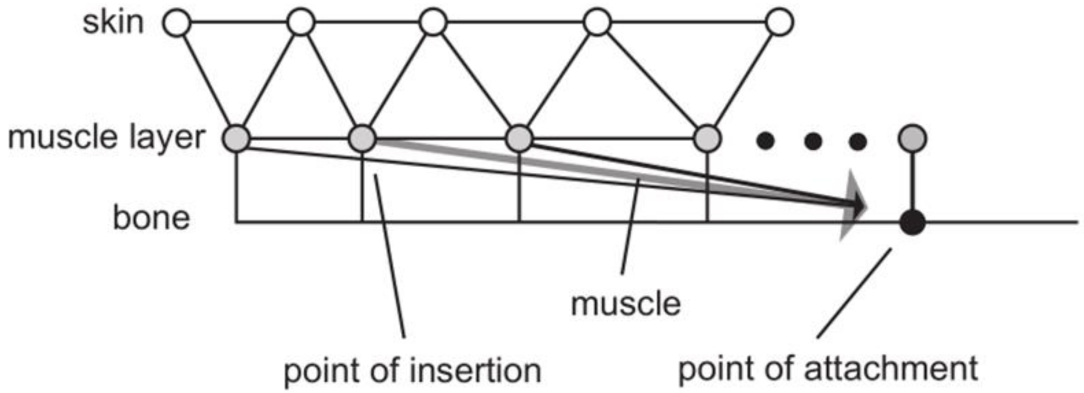
Muscle Model Animation

In the reliance on a physical basis, more precise, but more sophisticated

- Muscle controls most part of the face
- 3 Layers: Skin Layer, Muscle Layer, Bone Layer
- The point of insertion will move an amount determined by the muscle

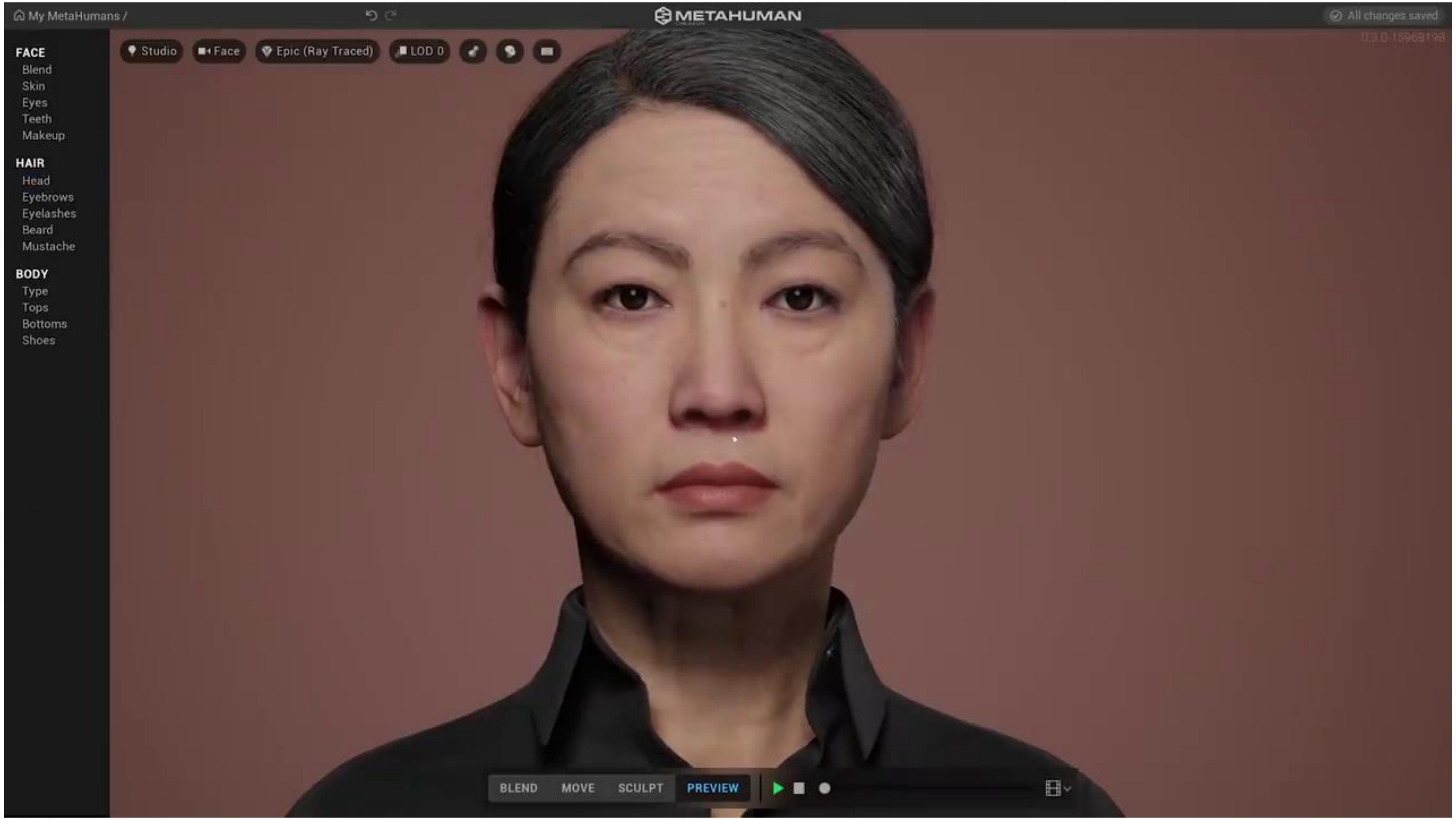
The model used for the skin will dictate how the area around the insertion point muscle reacts

Muscle Spreads Over The Face



Cross Section of The Muscle Model

Metahuman



My MetaHumans /
METAHUMAN
All changes saved
Studio
Face
Epic (Ray Traced)
LOD 0
FACE
Blend
Skin
Eyes
Teeth
Makeup
HAIR
Head
Eyebrows
Eyelashes
Beard
Mustache
BODY
Type
Tops
Bottoms
Shoes
BLEND
MOVE
SCULPT
PREVIEW

Animation Retargeting

Share Animation Among Characters

- Allow animations to be reused between characters (save animator's work)
- Adapt motion captured animations to different characters (reduce the cost)

Terminology

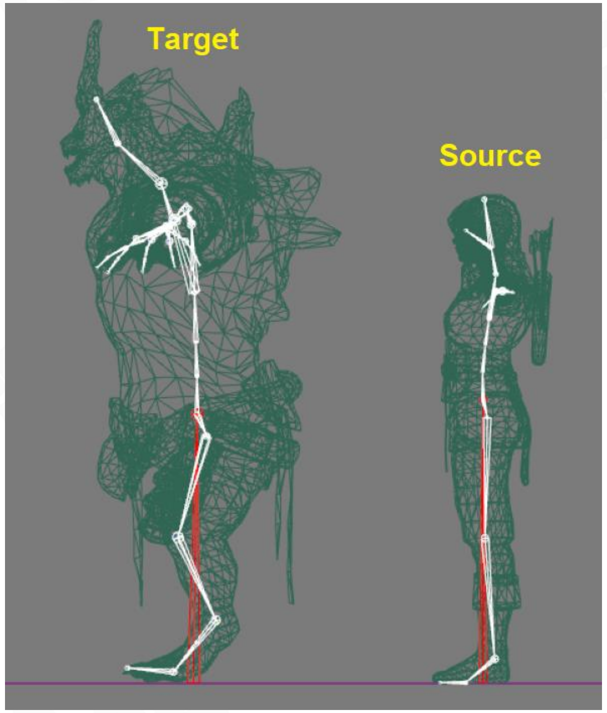
Source Character

Target Character

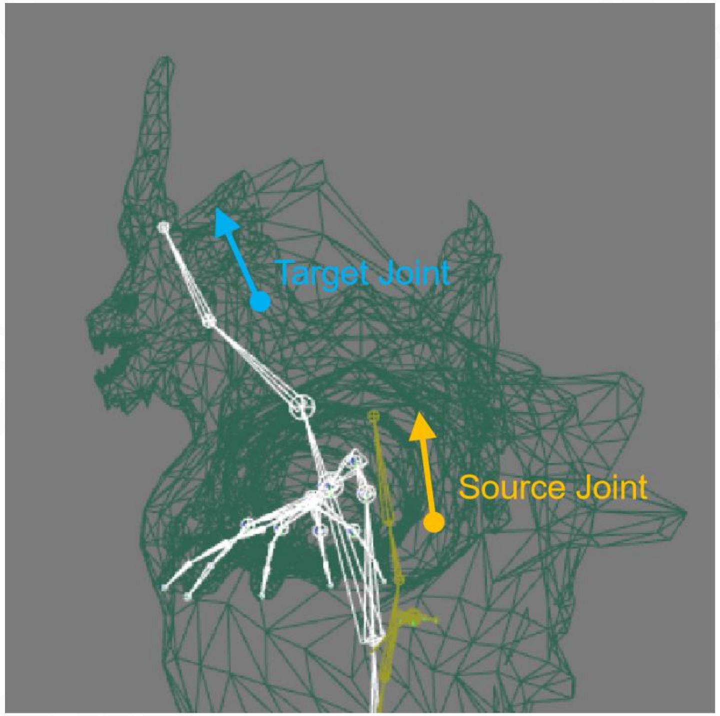
Source Animation

Target Animation

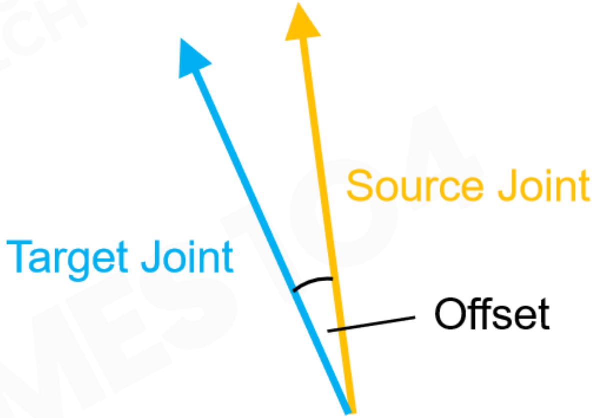
Ignore Offset Between Source and Target Joints



Source vs. Target at Retarget Pose

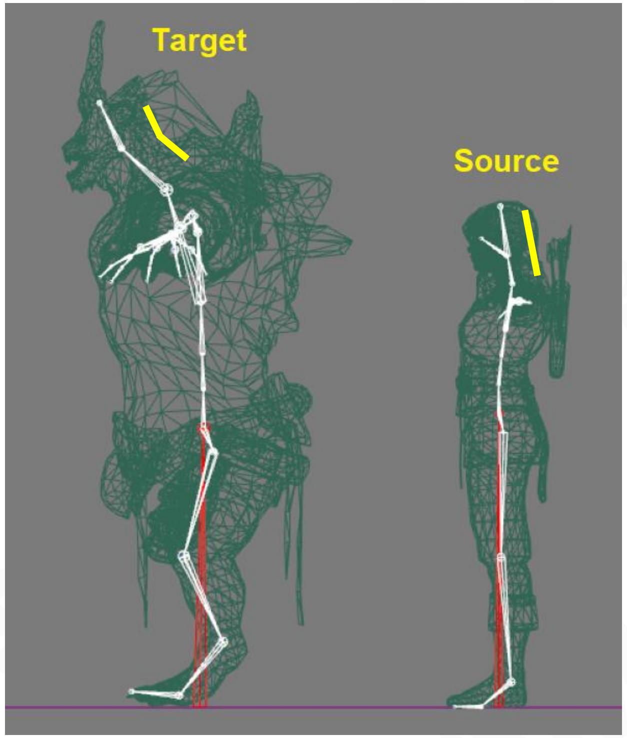


(Source skeleton in yellow)

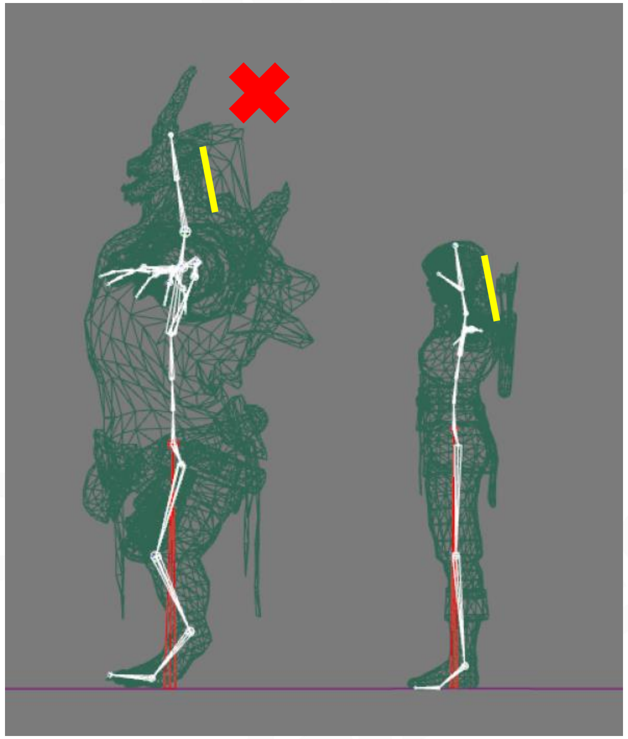


The Offset in Retarget Pose

Keep Orientation in Different Binding Pose



Target vs. Source at retarget pose



The target looks weird

Process Tracks

Handle animation tracks respectively

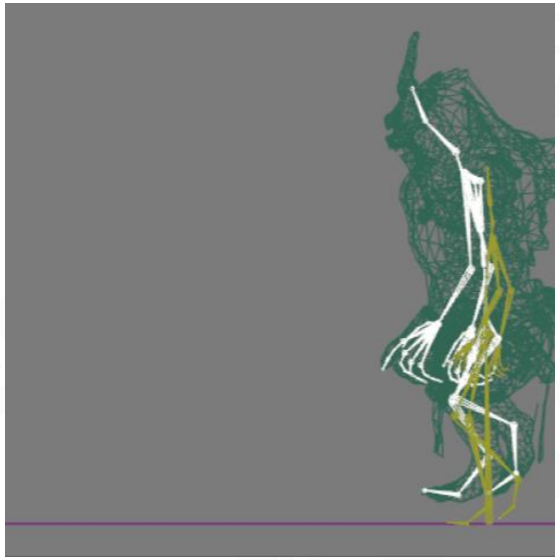
- Rotation track comes from source animation
  - Keep joint orientation in animation
- Translation track comes from target skeleton
  - Keep the proportion of target skeleton
- Scale track comes from source animation
  - Keep the scale in animation

Target animation with retargeting (Source animation in yellow)

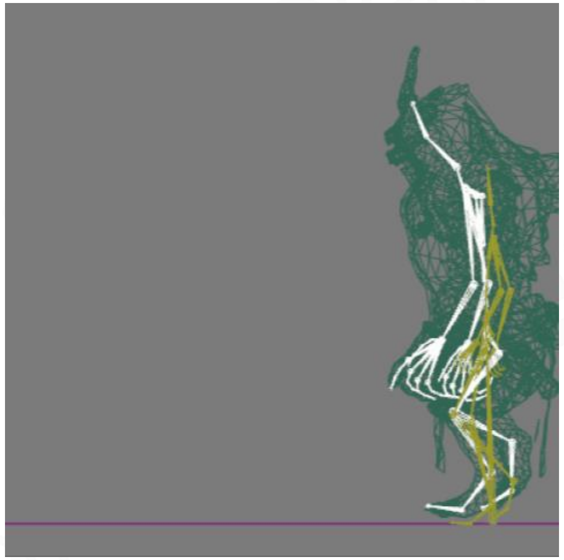
Align Movement by Pelvis Height

The movement of the character

- Usually controlled by displacement curve or motor system at runtime
  - Displacement Curve is extracted from the pelvis pose in animation
- Needs to be scaled by the proportion of the pelvis

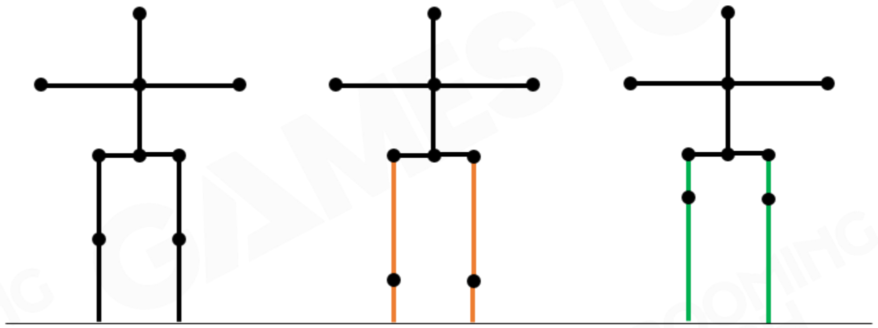


Hanging feet without movement scale
(Source animation in yellow)

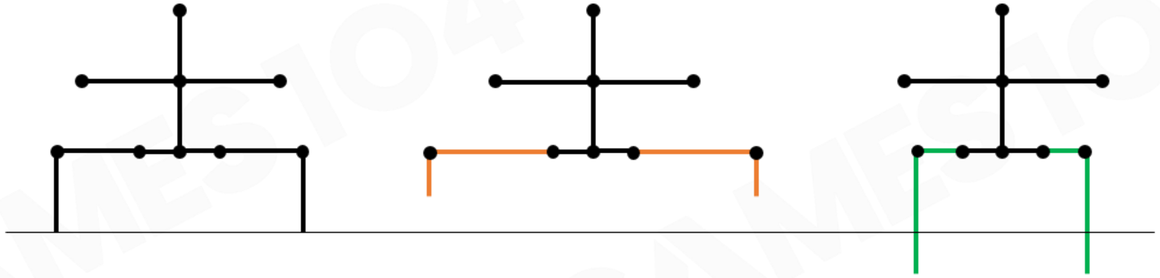


Problem eased with movement scale
(Source animation in yellow)

Lock Feet by IK after Retargeting



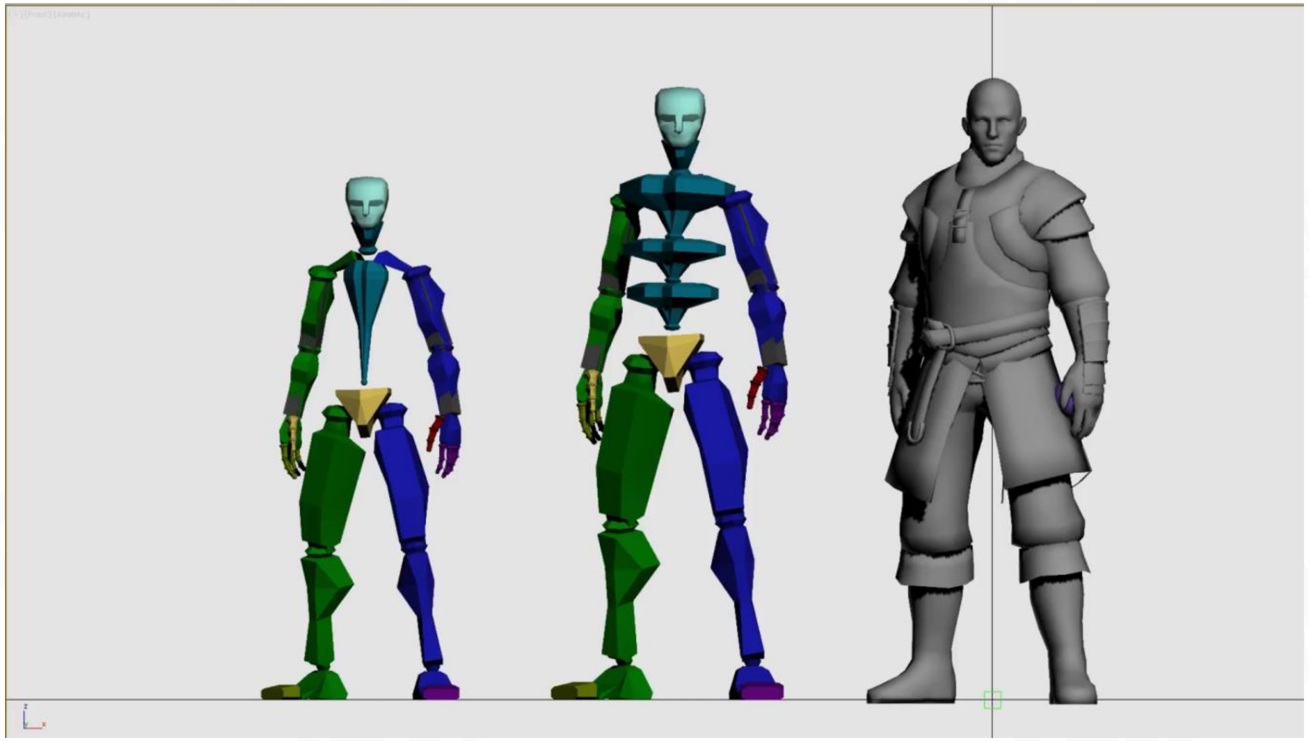
Source Skeleton(left) vs. Target Skeleton
with longer thigh(middle) or longer calf(right)



If the thigh is horizontal (left), longer thigh
results in hanging feet (middle) while longer
calf results in penetration (right)

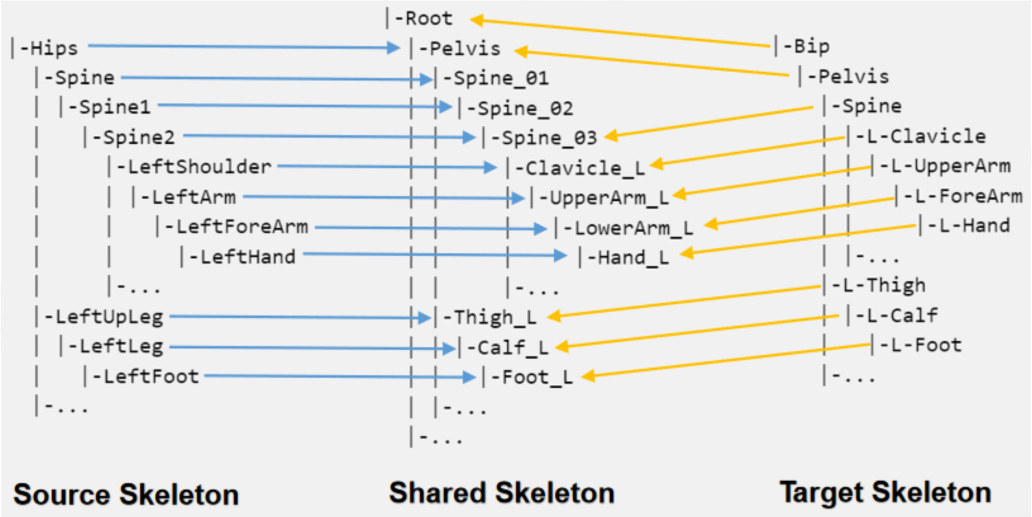
Target animation with foot IK

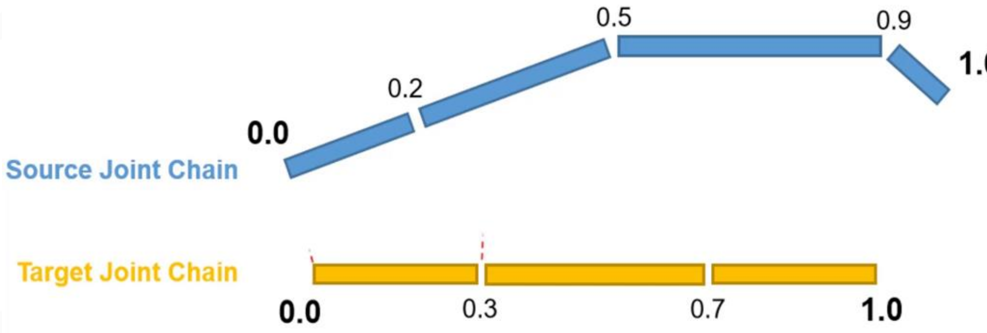
Retargeting with Different Skeleton Hierarchy



Source Skeleton with 1 spine (left) vs.
Target Skeleton with 3 spines (center)

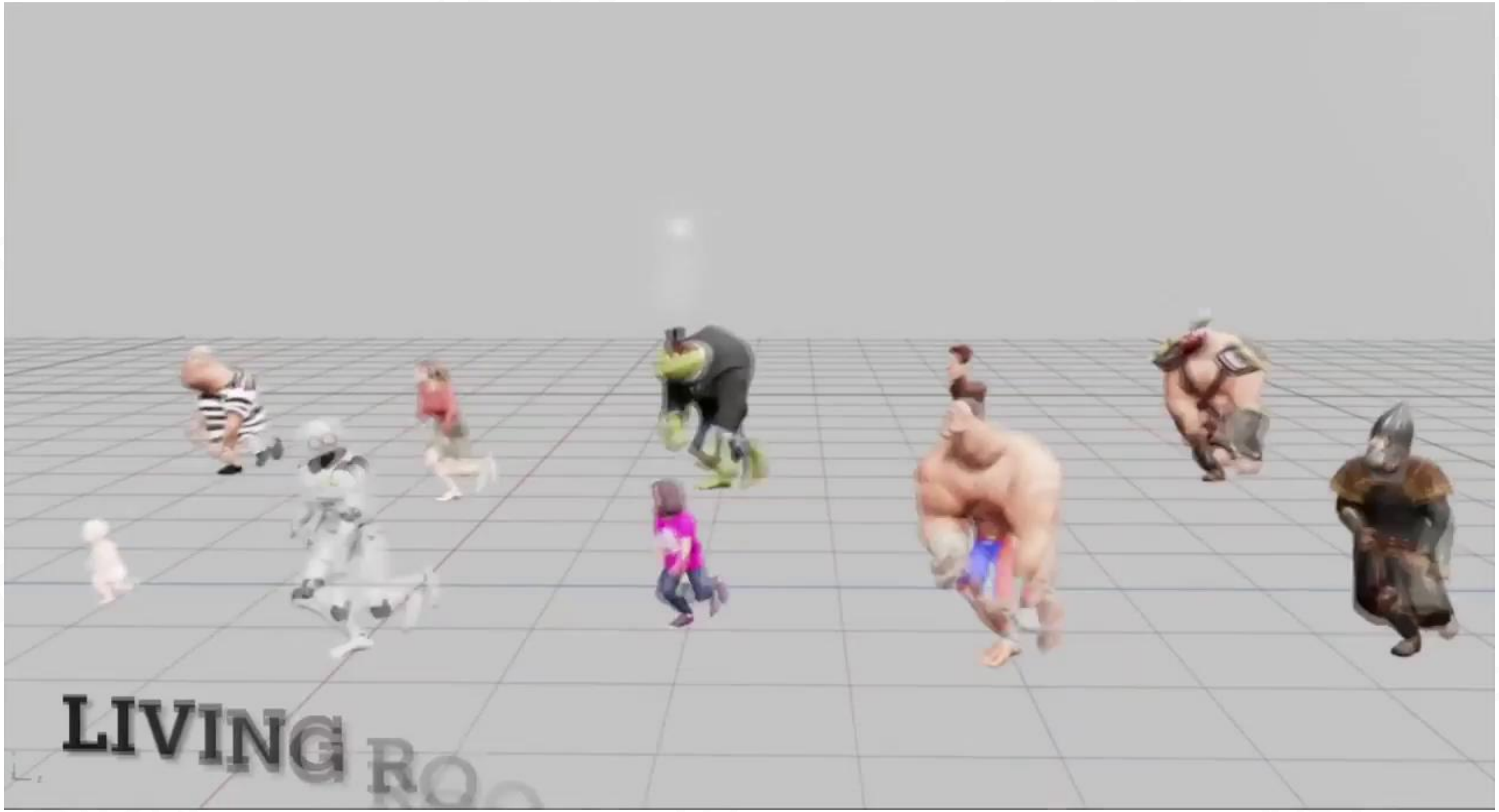
Easy Solution





The solution in Omniverse

Retargeting Animation in Omniverse



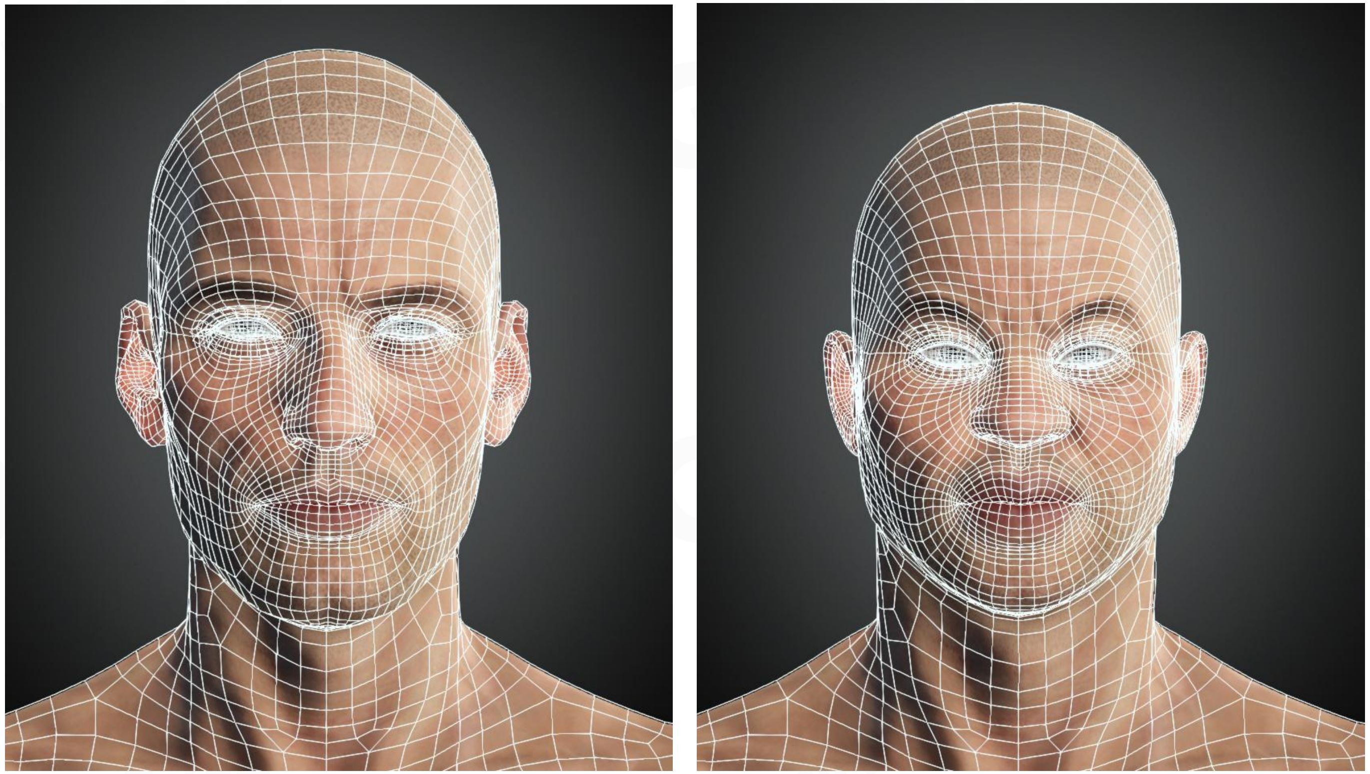
Unresolved Problems of Retargeting

- Self mesh penetration
- Self contact constrains (eg. the hands when clap)
- The balance of the target character

Self mesh penetration

Hands do not contact when clapping

Morph Animation Retargeting



Different face sharing the same topology

Morph Animation Retargeting Problem

Eye cannot be fully closed

Cannot close

Move vertex

Vertex moved

Smoothed

Take Away

- Controlled animation blending system is the key to animate character according to game play
- Inverse Kinematics help character’s animation adapt to environment constrains
- Facial expression can be encoded in Action Units in FACS
- Morph target animation is well applied in facial animation
- Retarget can help reuse skeleton animation and facial animations among characters

Pilot Engine V0.0.4 Releasing – May 17

Refactoring
- Level
- GObject/Component
- Editor Framework
- Style following Wiki documentation

Bugfixes
- Fixed errors in rendering subpass dependency
- Fixed overlapped button and cursor twinkling

Optimizations
- Optimized camera rotation control in high resolution
- Optimized AMD and NVIDIA graphic device race when initializing Vulkan
- Optimized editor camera controlling

Contributors

hyv1001, boooooommmmmm, and 6 other contributors

Lecture 09 Contributor

- 一将
- 喵小君
- 灰灰
- 蓑笠翁
- 小老弟
- 建辉
- Hoya
- 爵爷
- Jason
- 砚书
- BOOK
- MANDY
- 乐酱
- 灰灰
- 金大壮
- Leon
- 梨叔
- Shine
- 浩洋
- Judy
- 乐酱
- QIUU
- C佬
- 阿乐
- 靓仔
- CC
- 大喷
- 大金

Q&A

Enjoy ;) Coding

Course Wechat

Follow us for further information

Please note that all videos and images and other media are cited from the Internet for demonstration only.