Character Control with Neural Networks and Machine Learning

Daniel Holden
Animation Researcher, Ubisoft Montreal



3.75 km^2

1000 km^2

~15,000 Animations

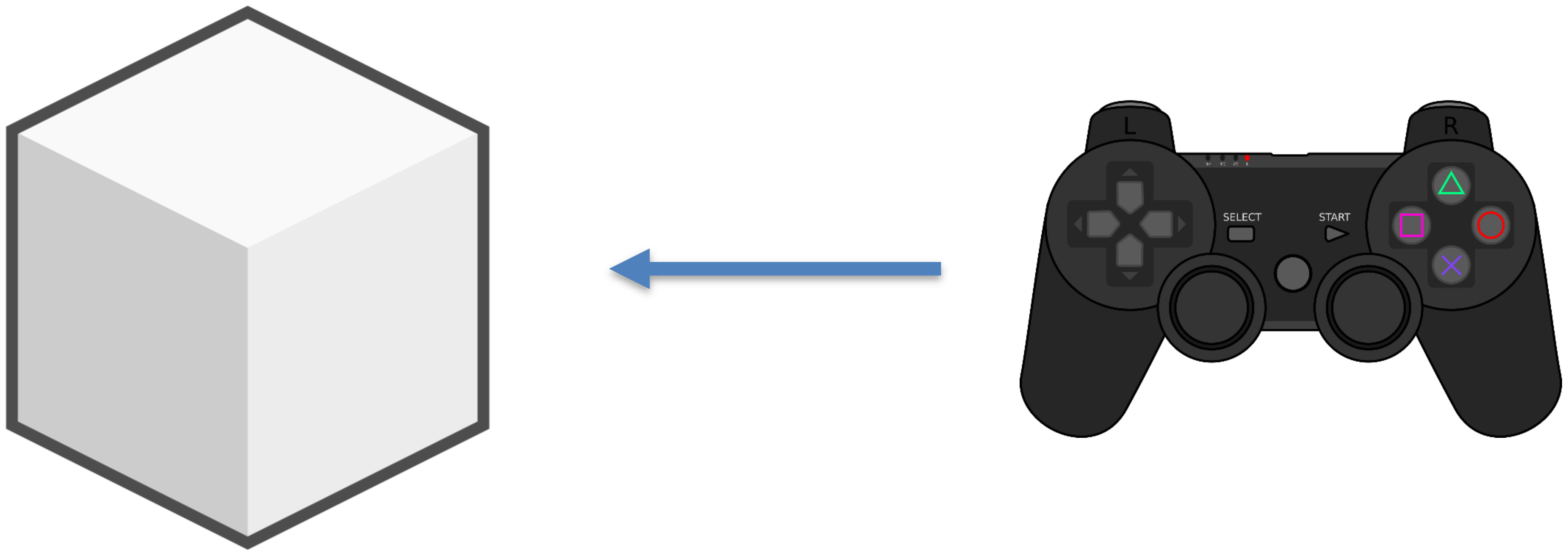
UBISOFT®

LA FORGE

laforge@ubisoft.com

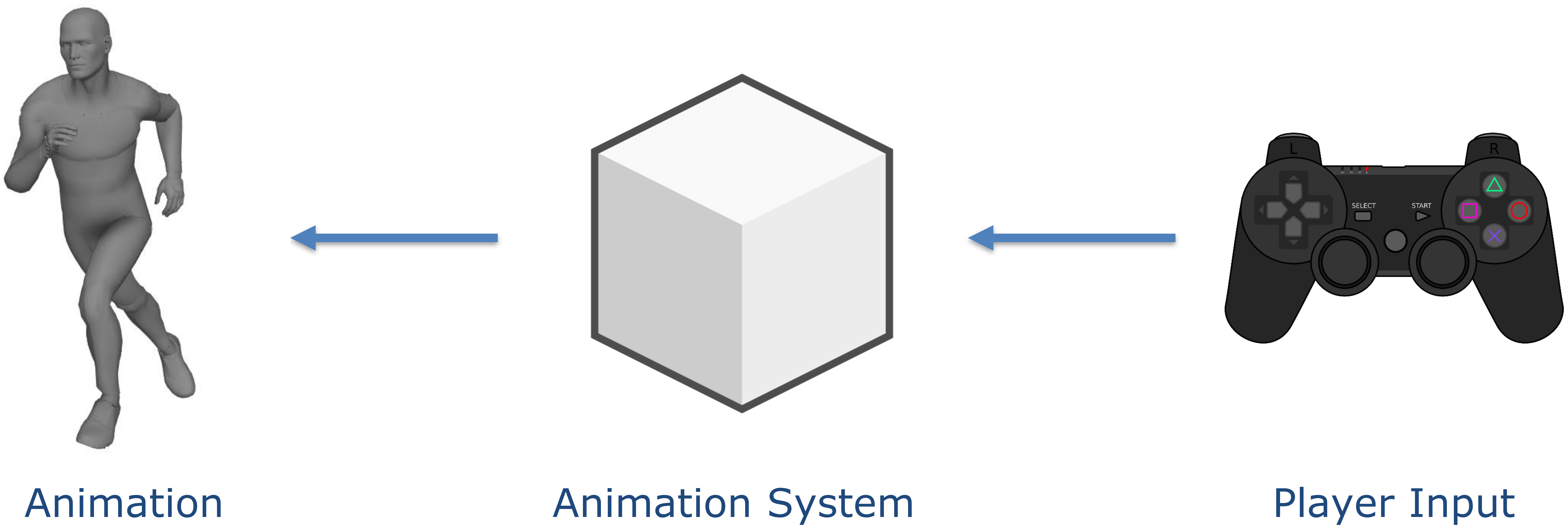
Background

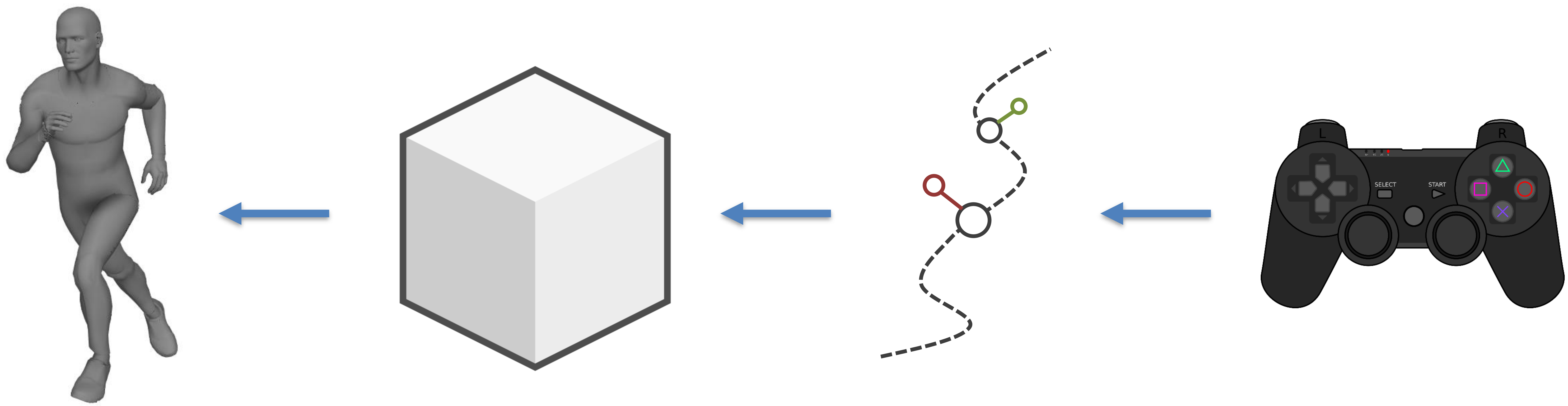
Player Input



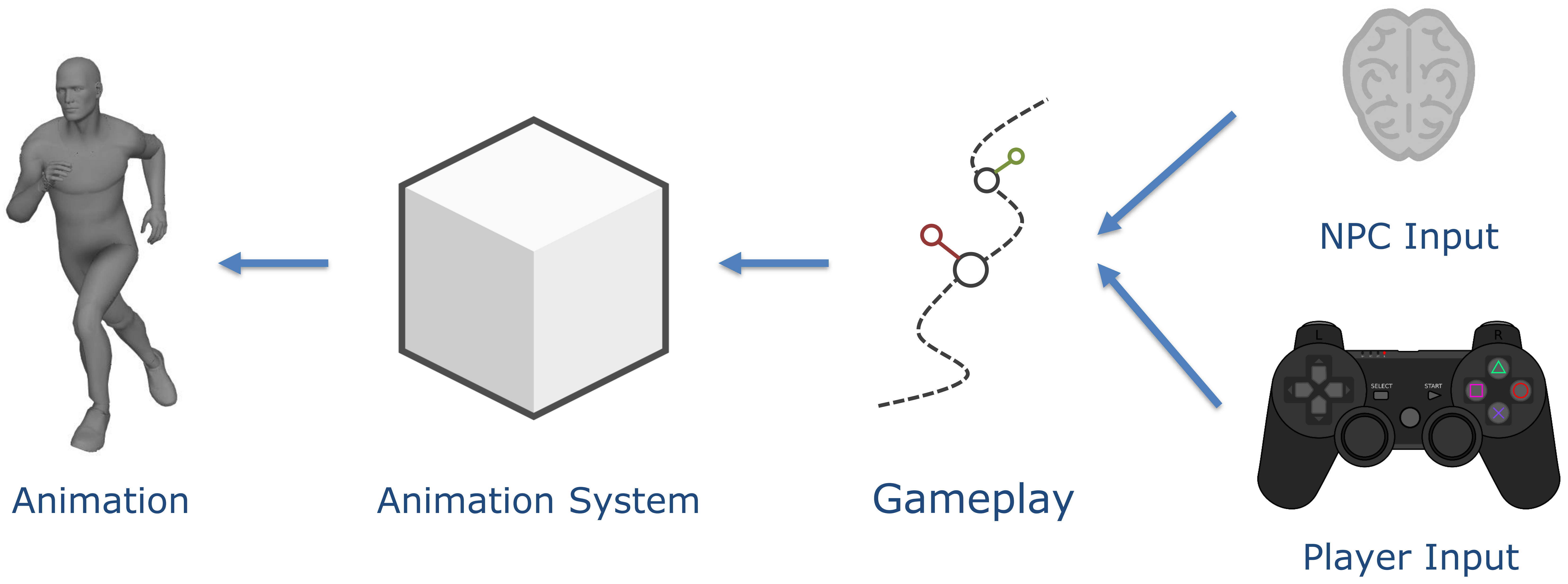
Animation System

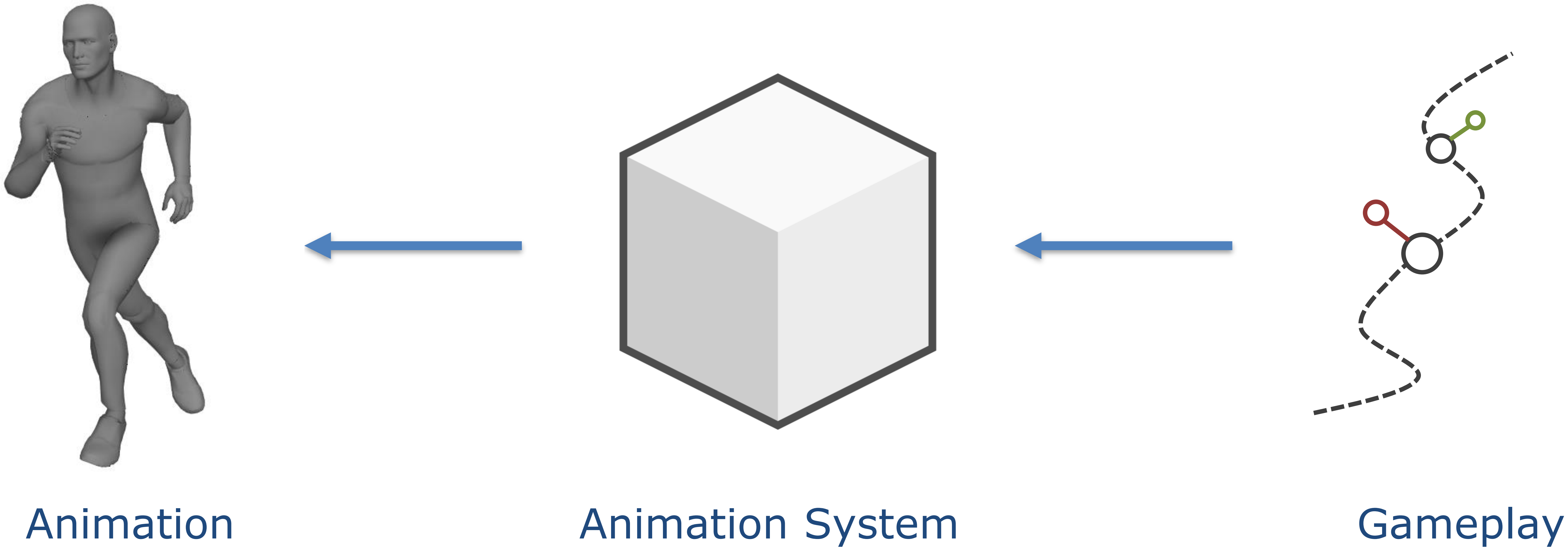
Player Input


Animation
Animation System
L
R
SELECT
START
Player Input

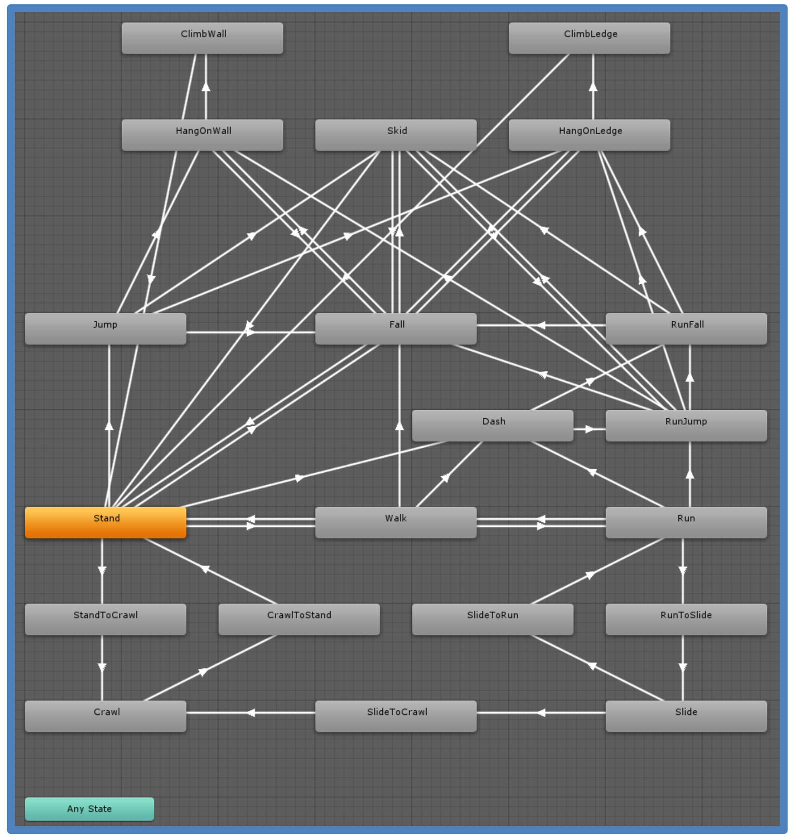


Animation Animation System Gameplay Player Input

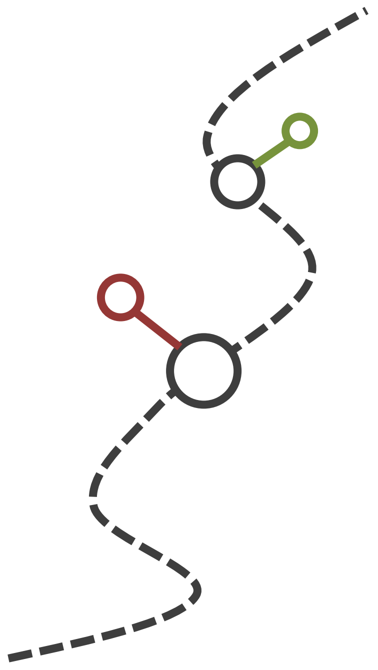

NPC Input
Animation
Animation System
Gameplay
L
R
SELECT
START
Player Input


Animation
Animation System
Gameplay

Animation

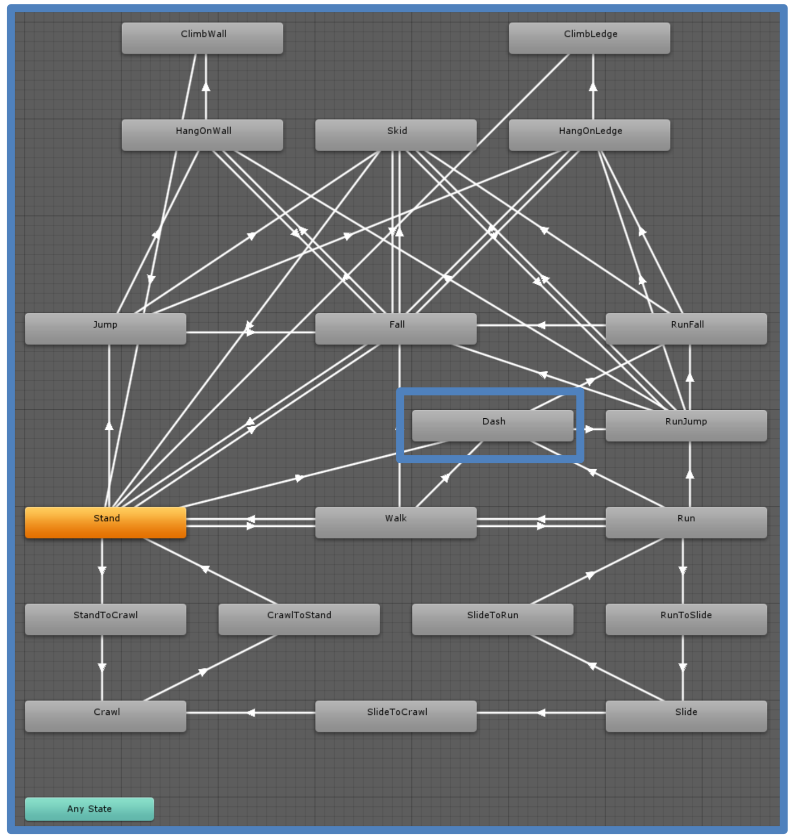


State Machine

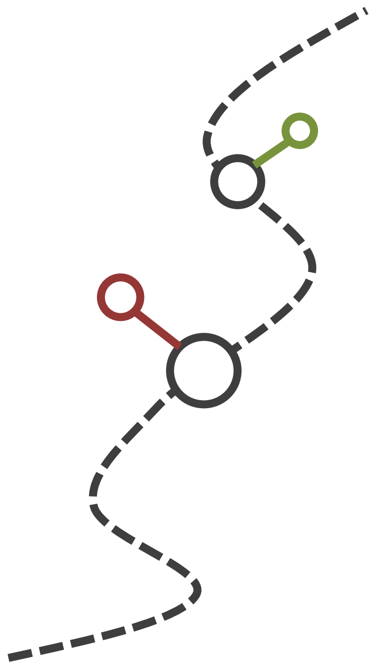


Gameplay

Animation

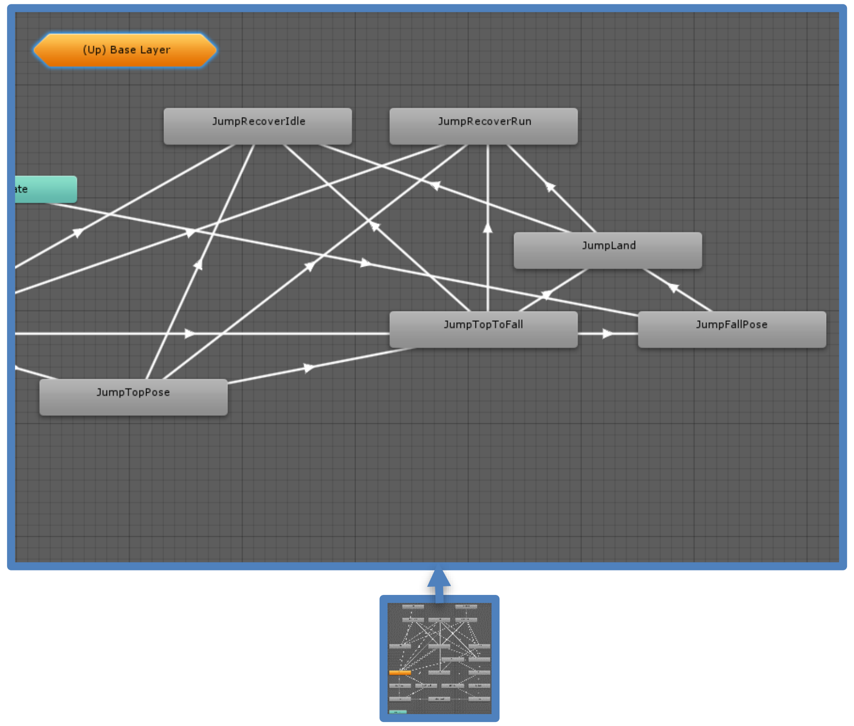


State Machine

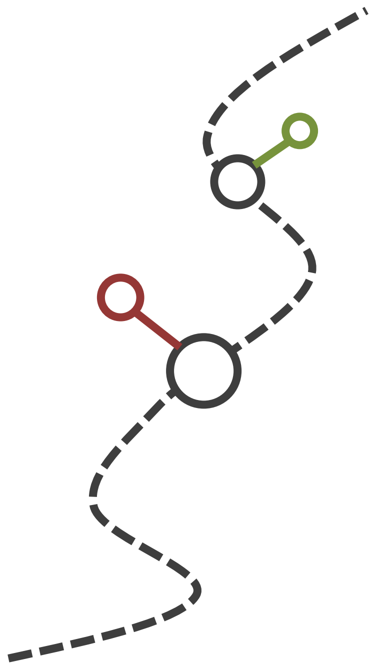


Gameplay

Animation

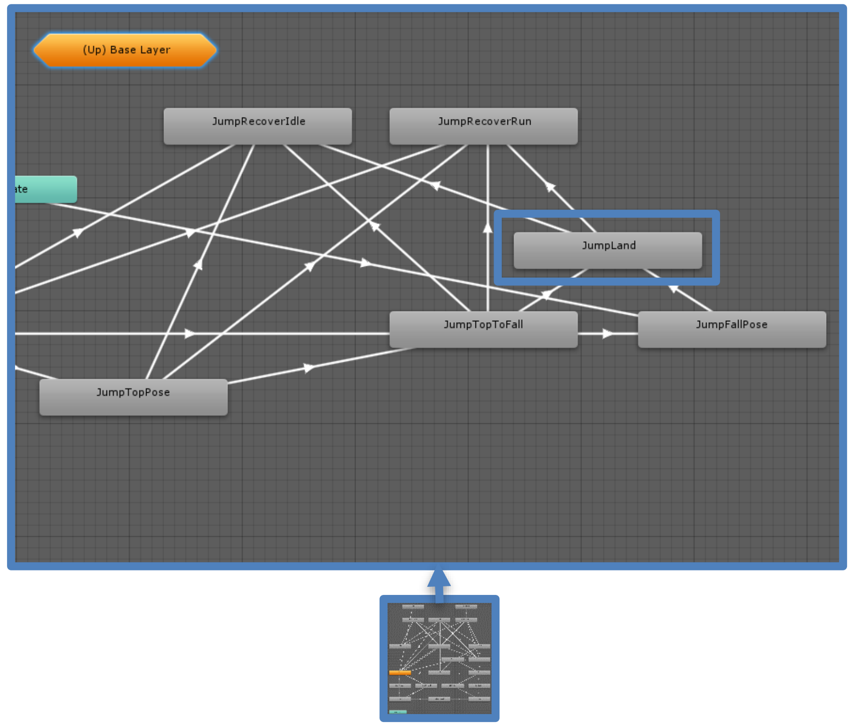


State Machine

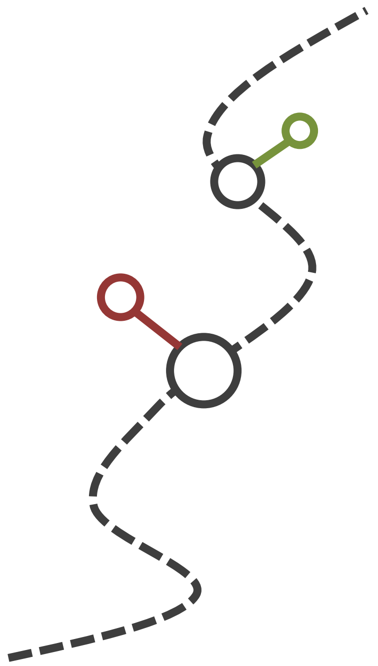


Gameplay

Animation

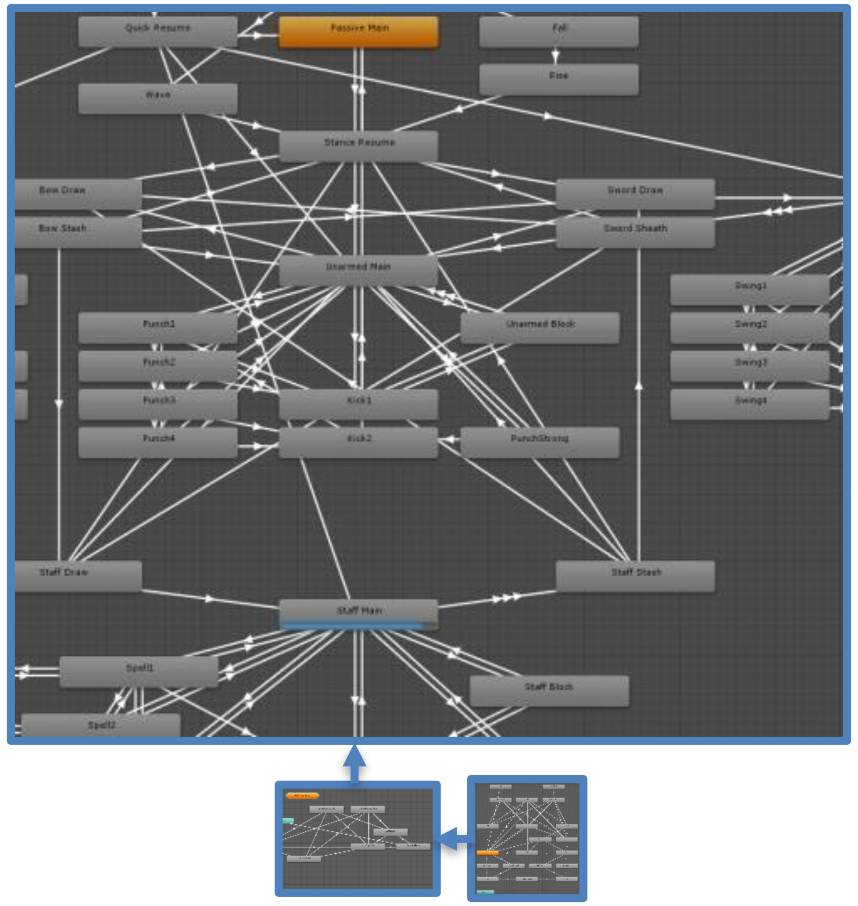


State Machine

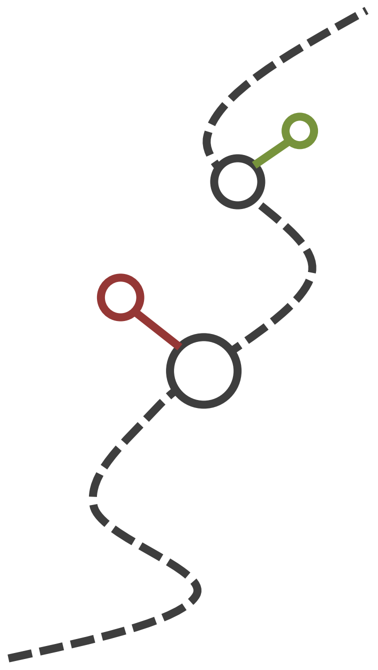


Gameplay

Animation

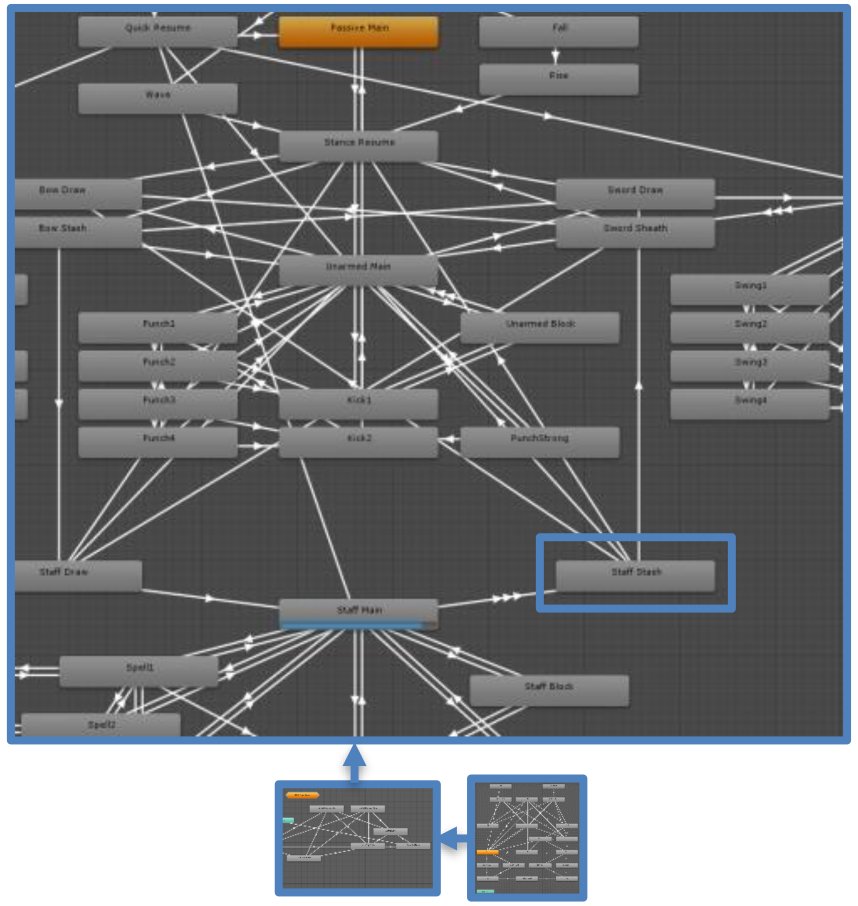


State Machine

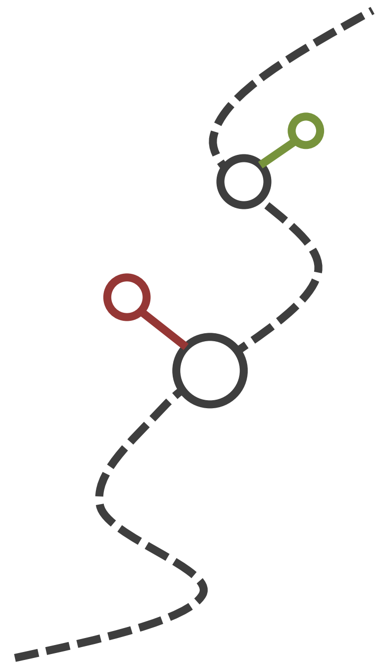


Gameplay

Animation

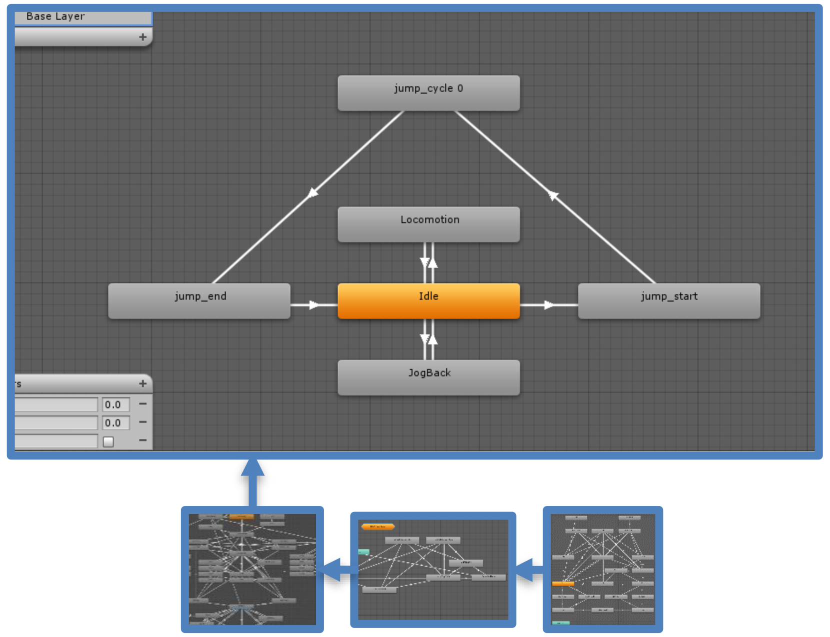


State Machine

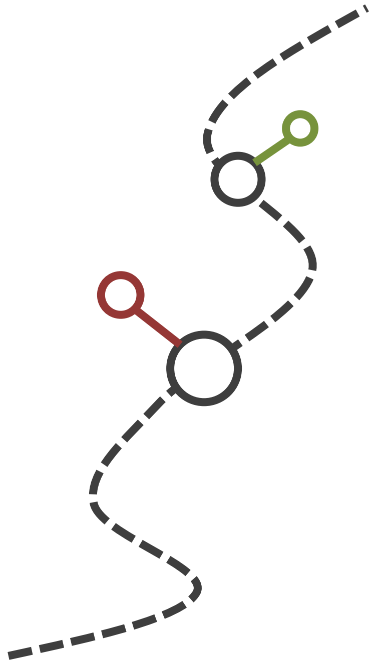


Gameplay

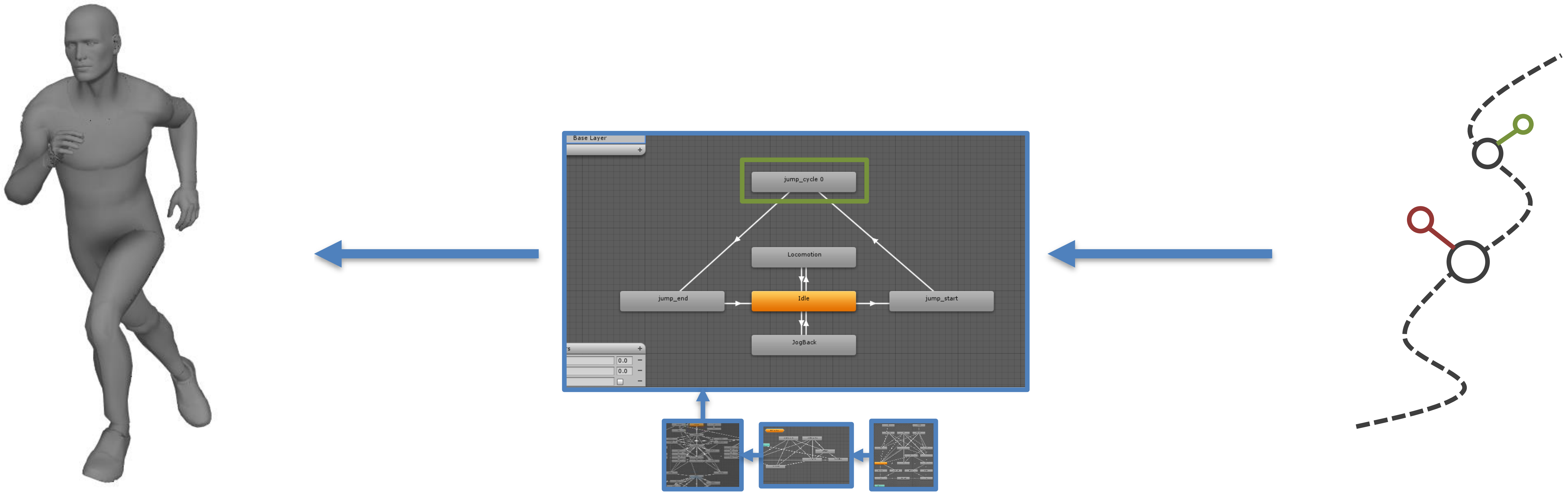
Animation



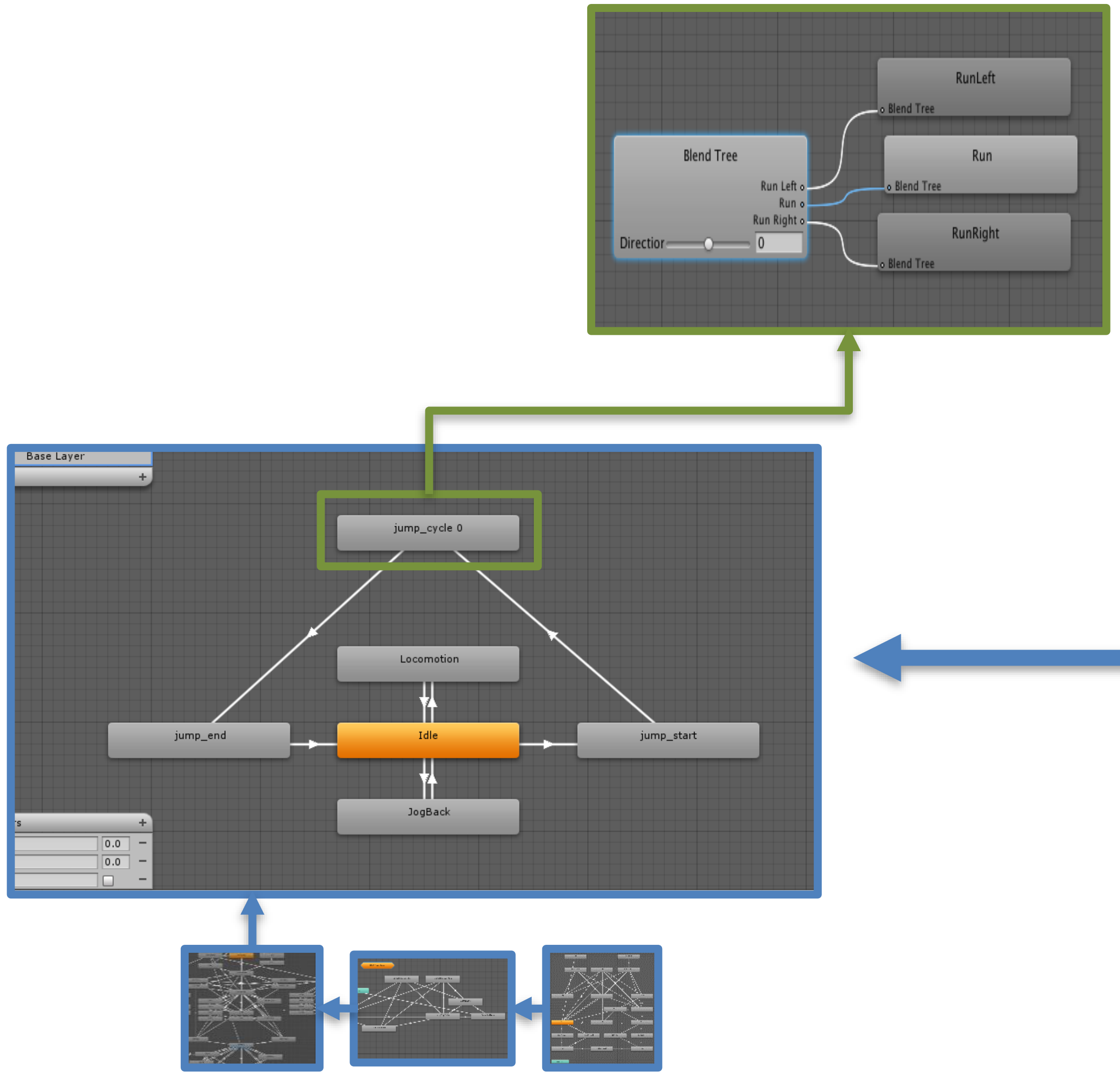
State Machine



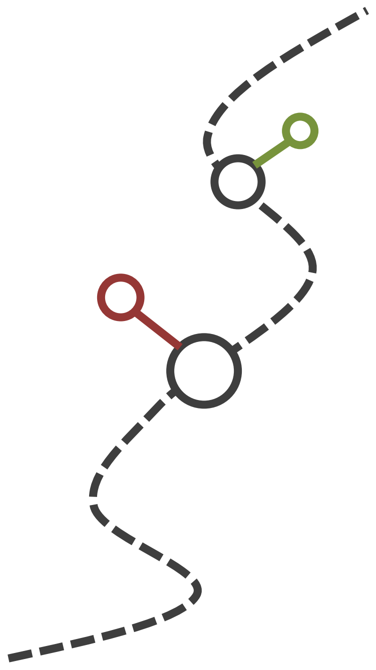
Gameplay



Animation State Machine Gameplay



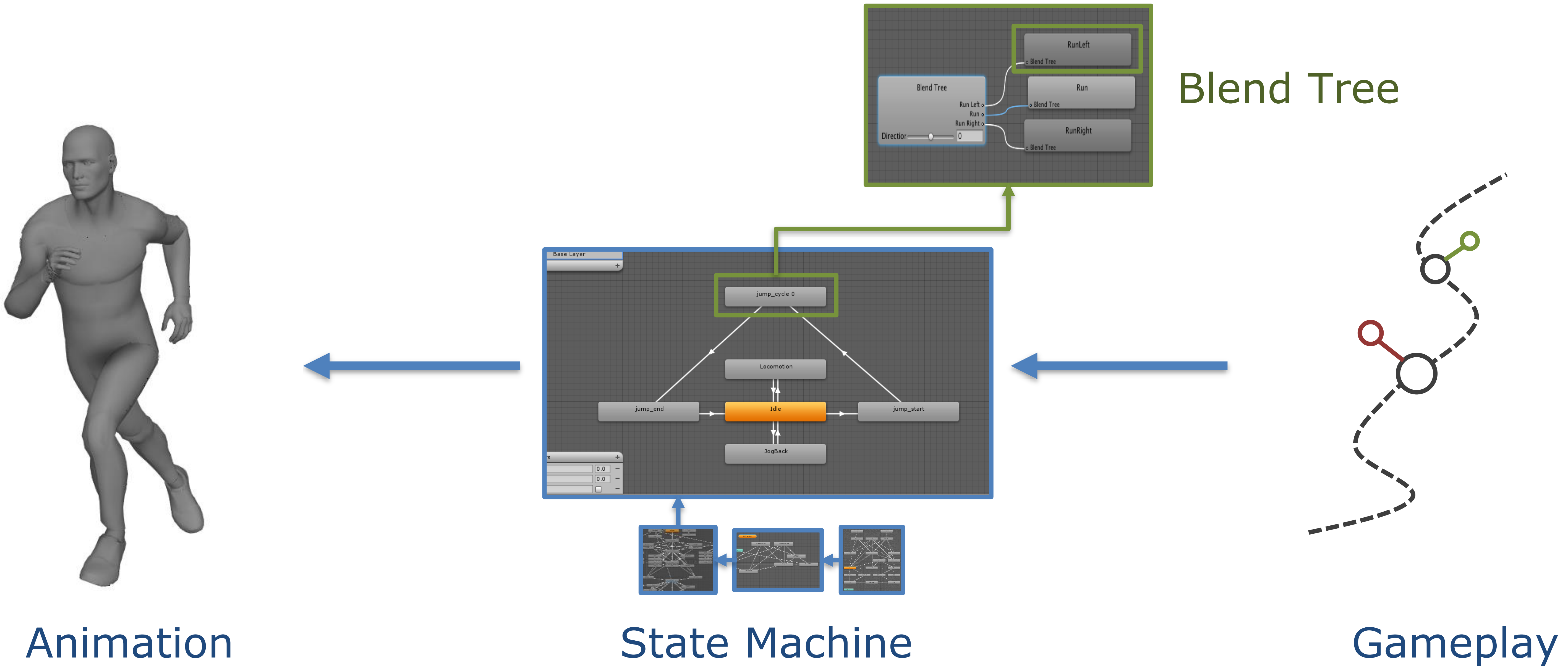
Blend Tree

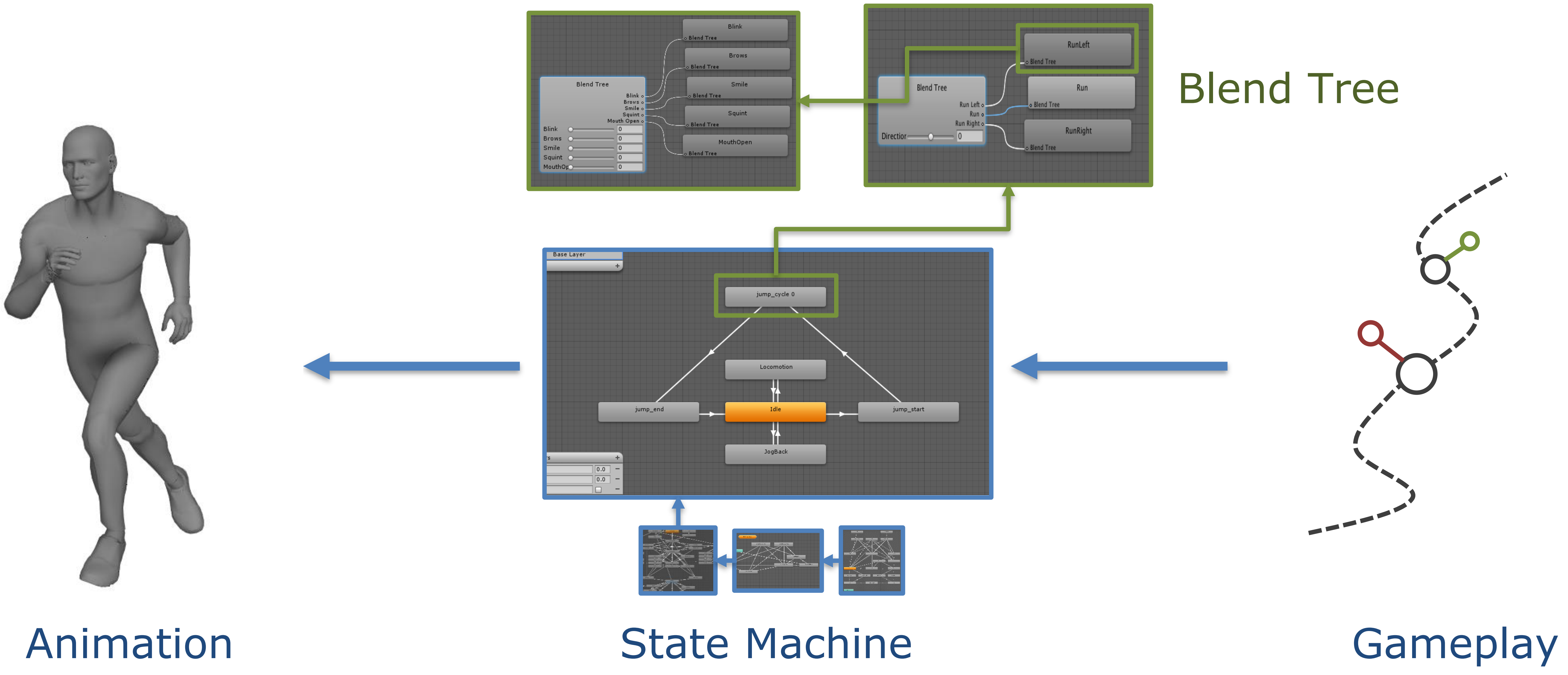


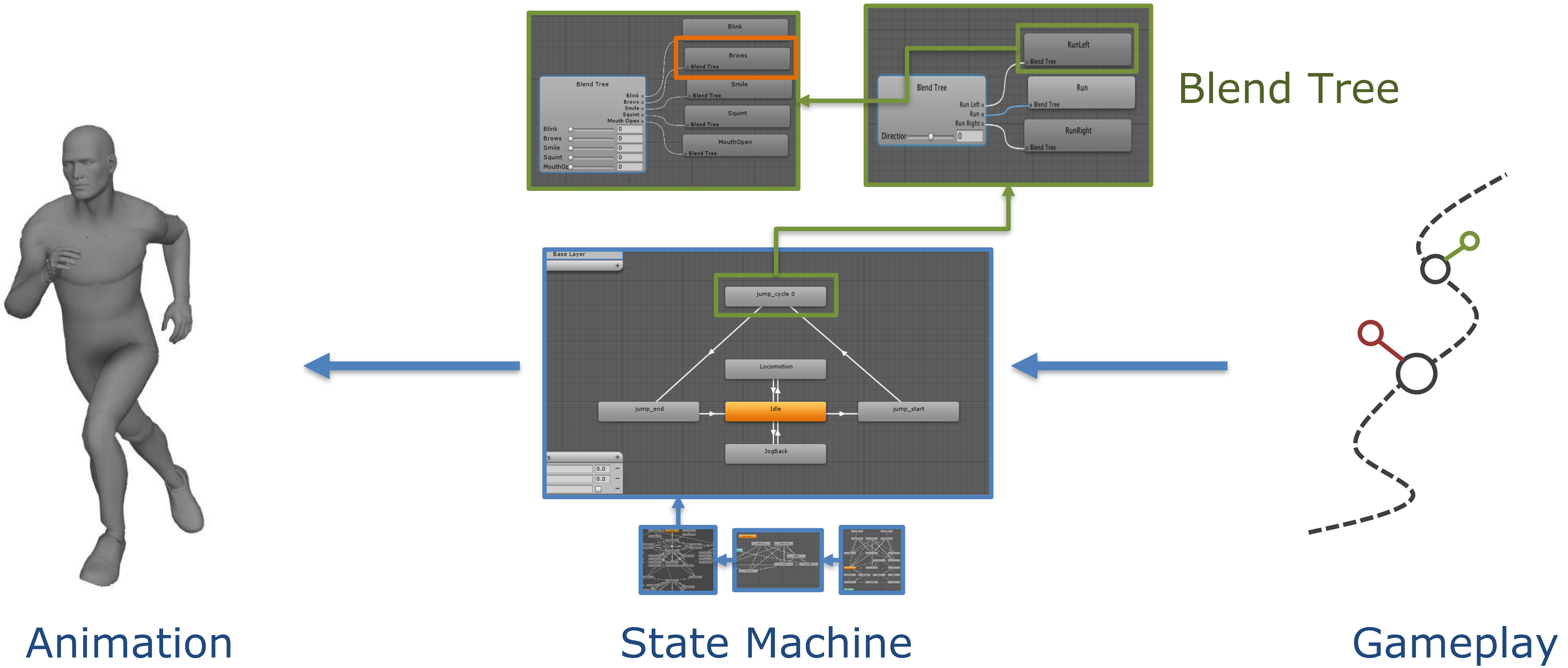
Animation

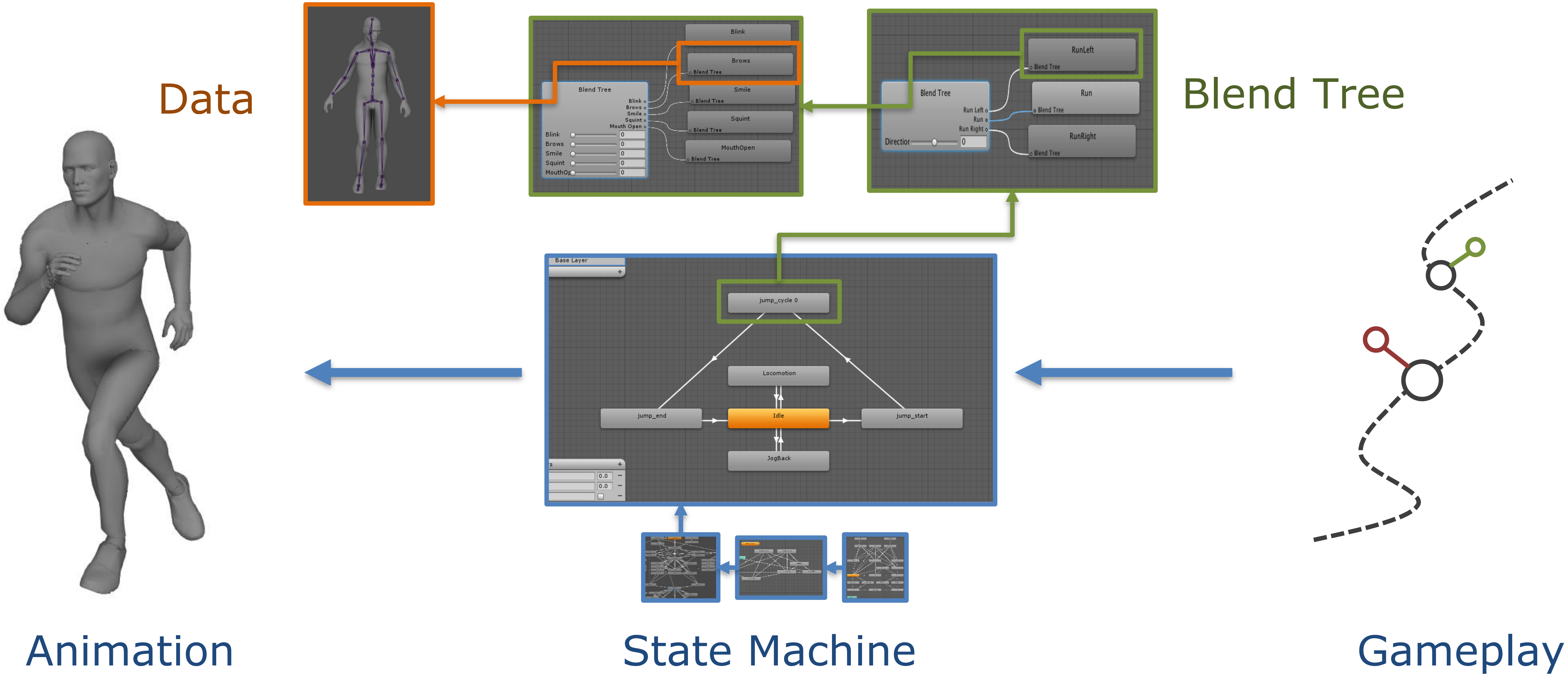
State Machine

Gameplay

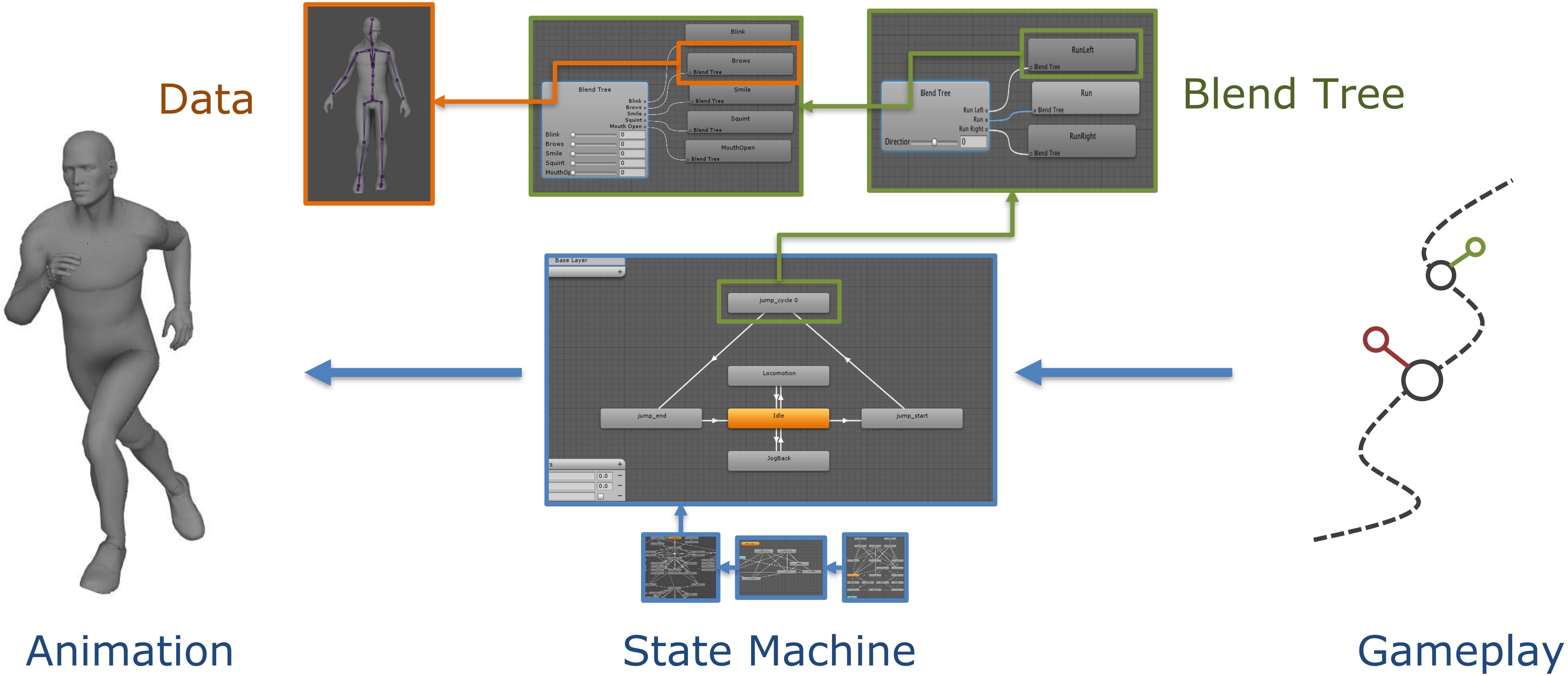

Blend Tree
RunLeft
Blend Tree
Run Left
Run
Run Right
Direction
0
Run
RunRight
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
State Machine
Gameplay


Blend Tree
Blink
Brows
Smile
Squint
MouthOpen
Blend Tree
RunLeft
Run
RunRight
Direction
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Base Layer
Blend Tree
Animation
State Machine
Gameplay

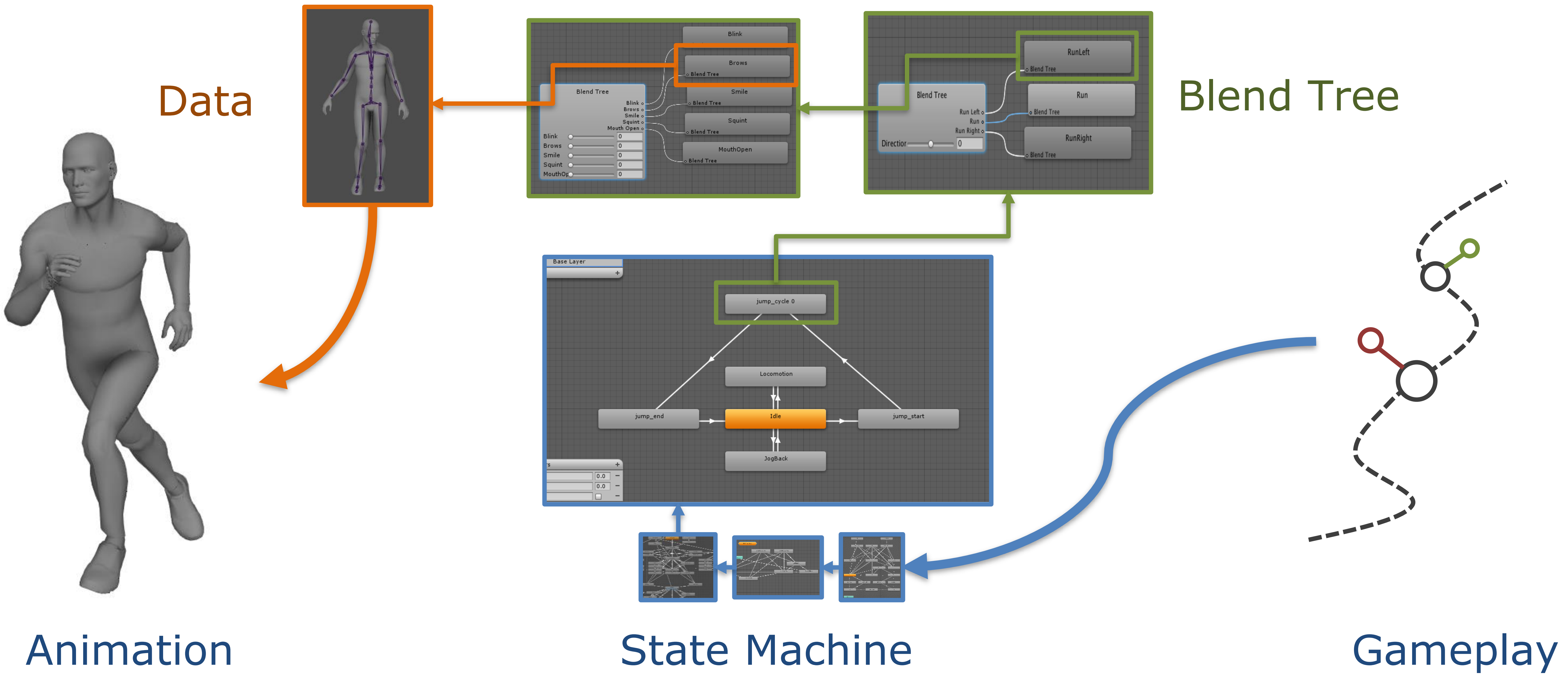

Blend Tree
Blink
Brows
Smile
Squint
MouthOpen
RunLeft
Run
RunRight
Direction
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
State Machine
Gameplay


Data
Blend Tree
Blink
Brows
Smile
Squint
MouthOpen
RunLeft
Run
RunRight
Direction
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
State Machine
Gameplay

asc6_rel2_char3_male_loco_slow_t45_to_idlebrk_wave_happy_v013_final.fbx



asc6_rel2_char3_male_loco_slow_t45_to_idlebrk_wave_happy_v013_final.fbx



~15,000 Animations

~15,000	**Animations**
~5,000	**States**

~15,000	**Animations**
~5,000	**States**
~12	**Levels**

The Director

"We want the player character to be injured"

Option

Add a new “injured” state at every leaf

Option

Add a new “injured” state at every leaf

Issue

Some states don’t make sense when injured.

Option

Duplicate the graph and replace data with “injured” versions of the same animations.

Option

Duplicate the graph and replace data with "injured" versions of the same animations.

Issue

Some states have no injured data recorded.

Option

Hack something together for this case.

Option

Hack something together for this case.

Issue

Technical debt can build up quickly.

Some Time Later…

"Great! Now in the next scene the character is injured, tired, and blinded in one eye."

The Dream

Day 1

“We want the player character to be able to be injured”

Day 2

Limping around the motion capture studio

Day 3

Dragging and dropping the new data into the system

Day 4

Everything Just Works™

Day 5

"Great! Now in the next scene the character is injured, tired, and blinded in one eye."

Day 6

Getting blinded in one eye at the motion capture studio

Day 7

Dragging and dropping the new data into the system

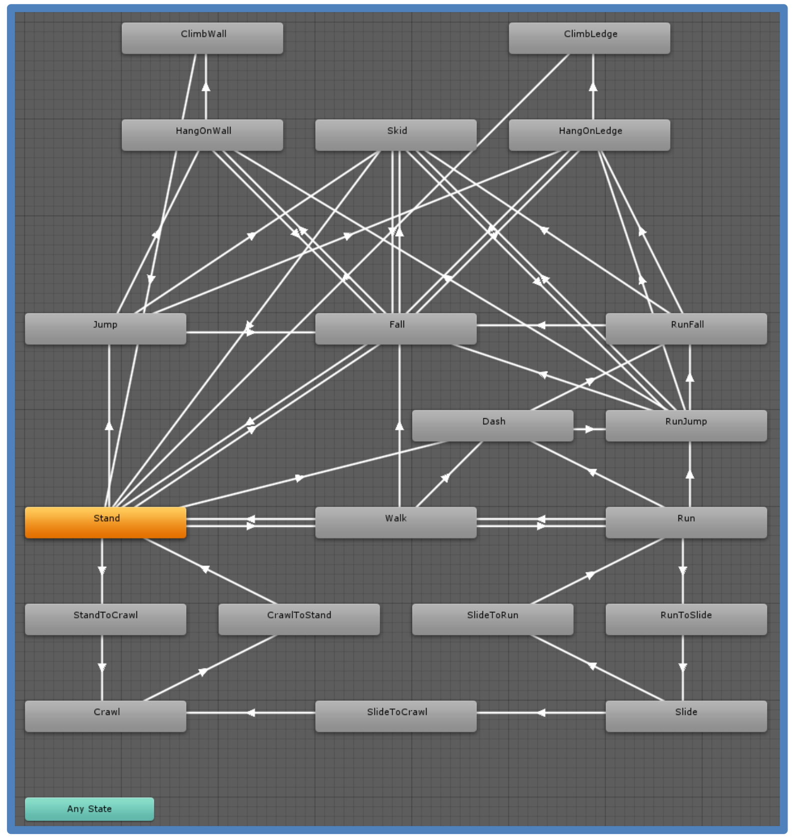
And so on…

Scalability

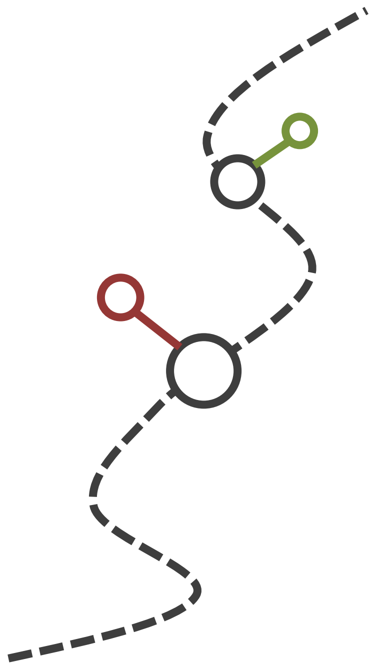
- Separate Data
- Specify Desired Variables
- Generalize Solution

Data Separation

Animation



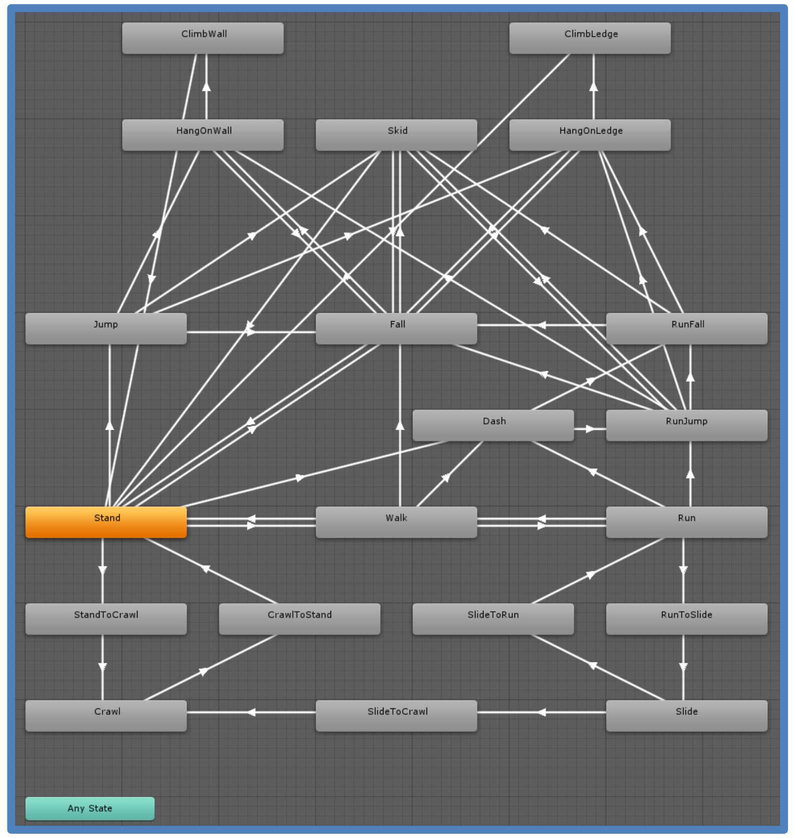
State Machine



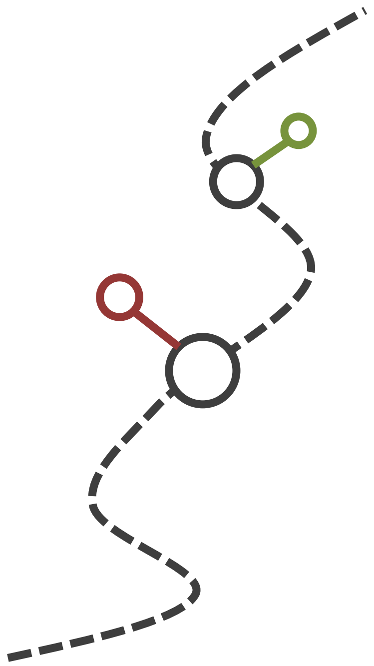
Gameplay

Animation

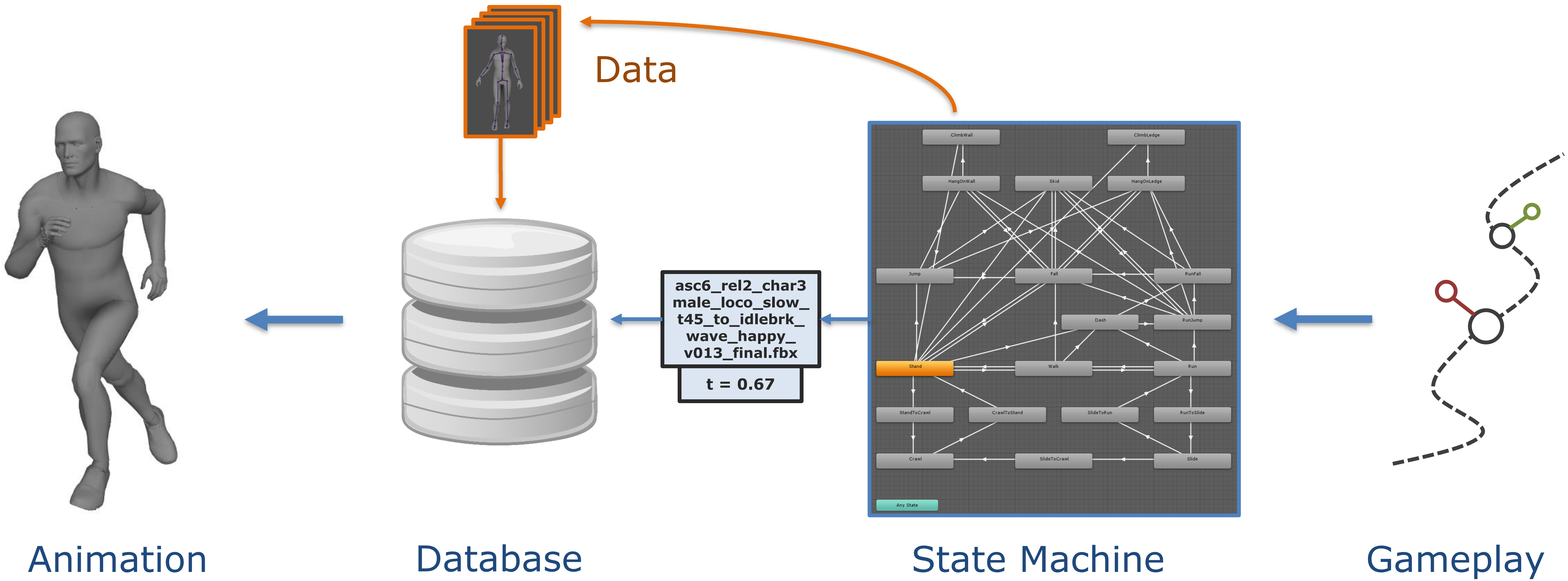
Database

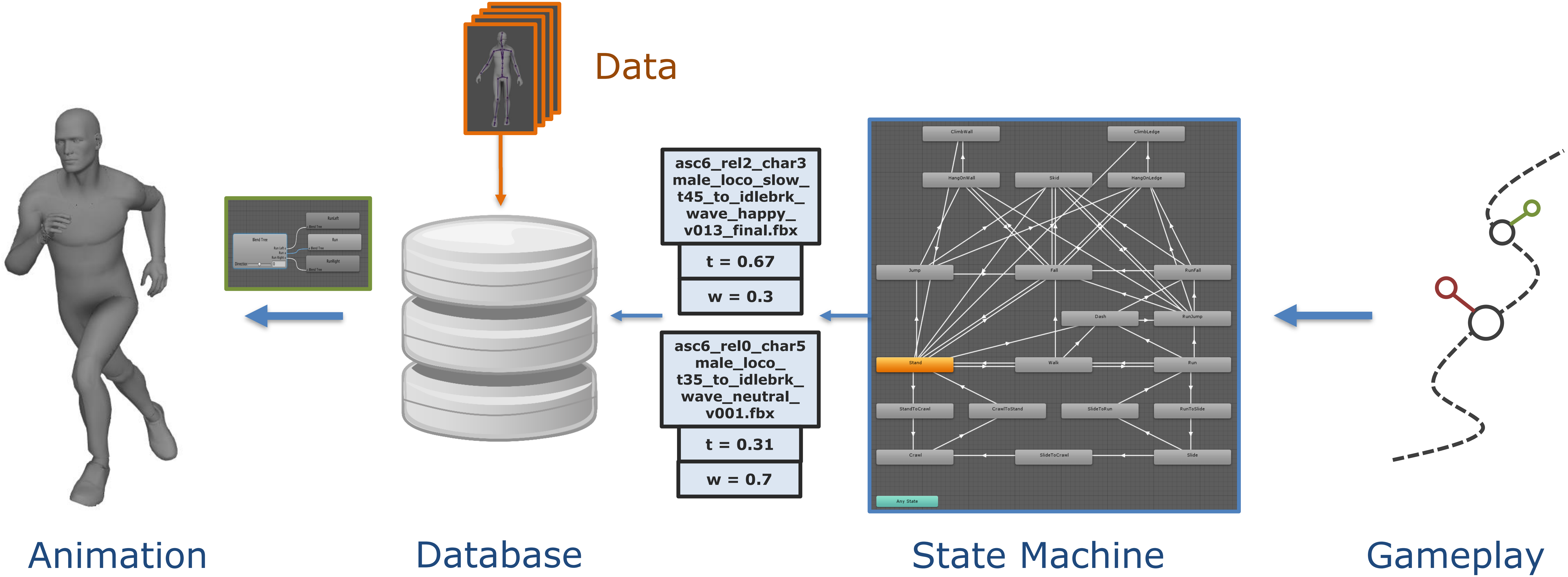


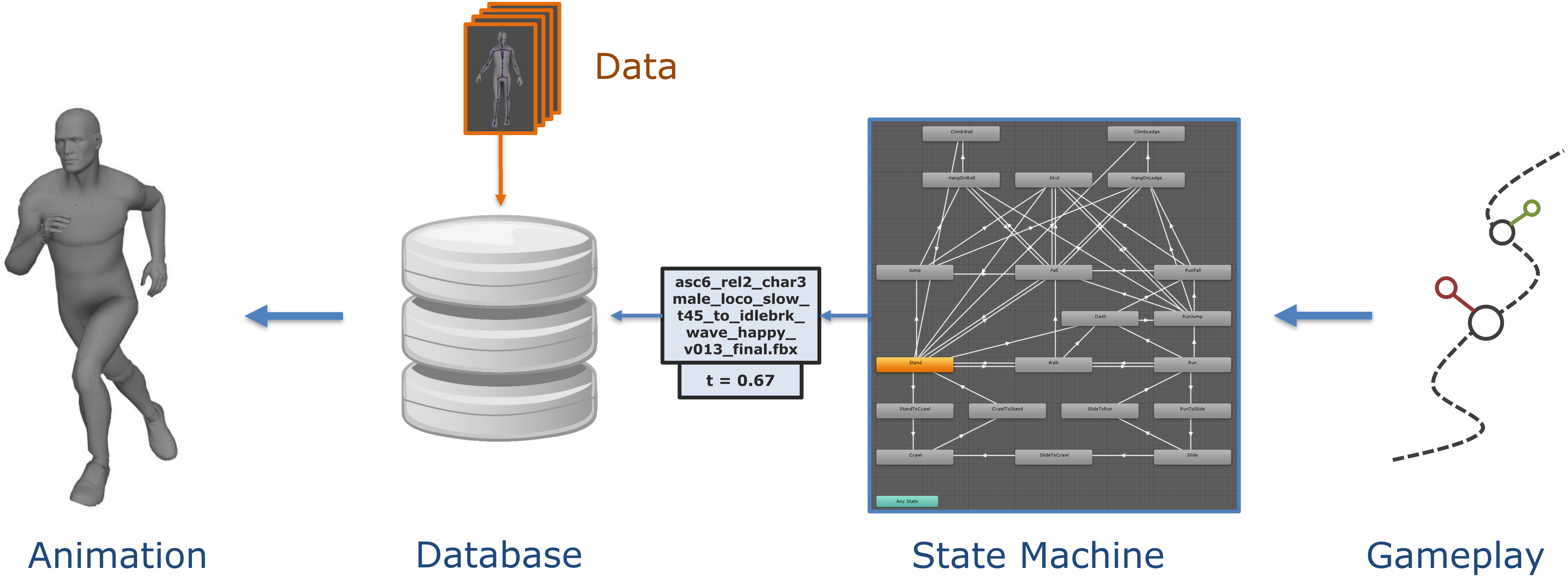
State Machine

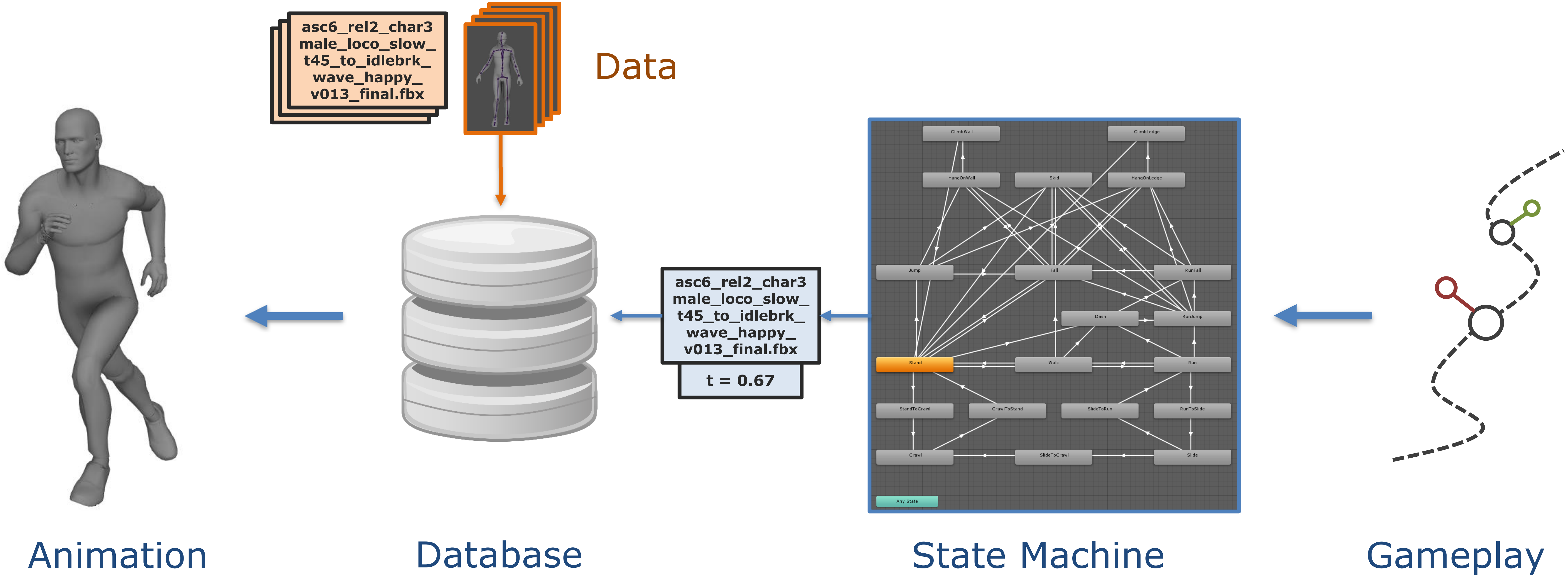


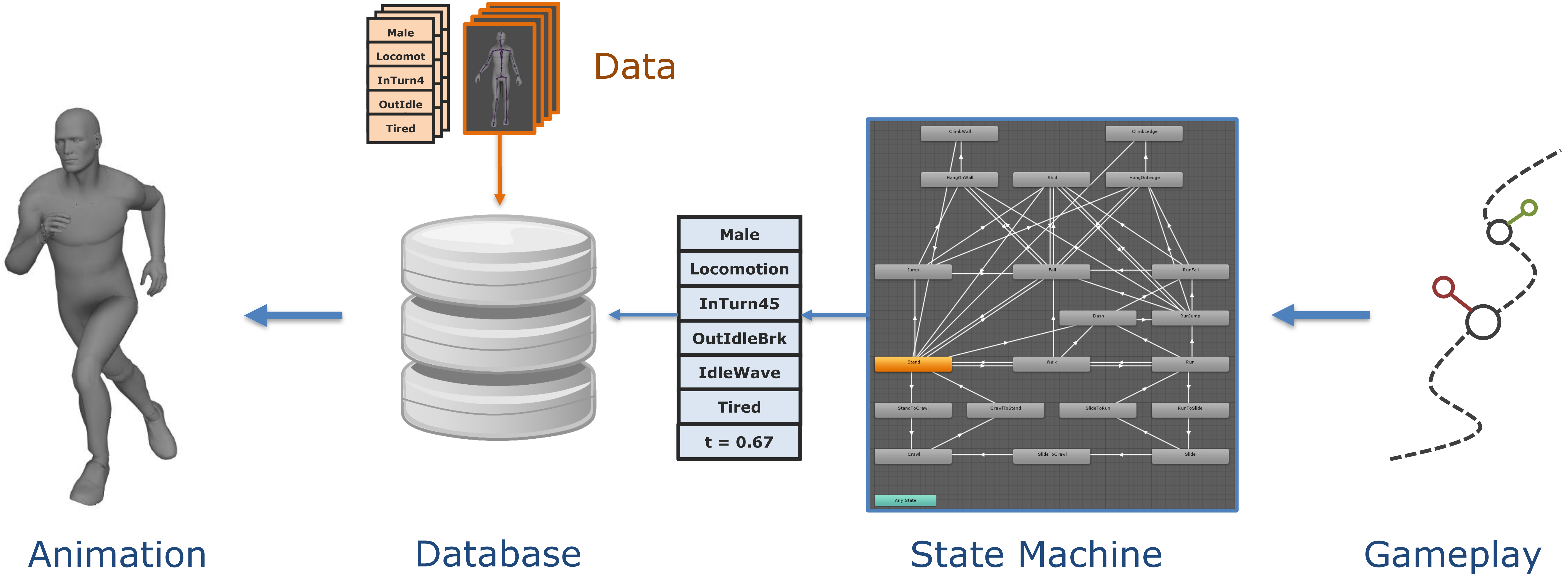
Gameplay

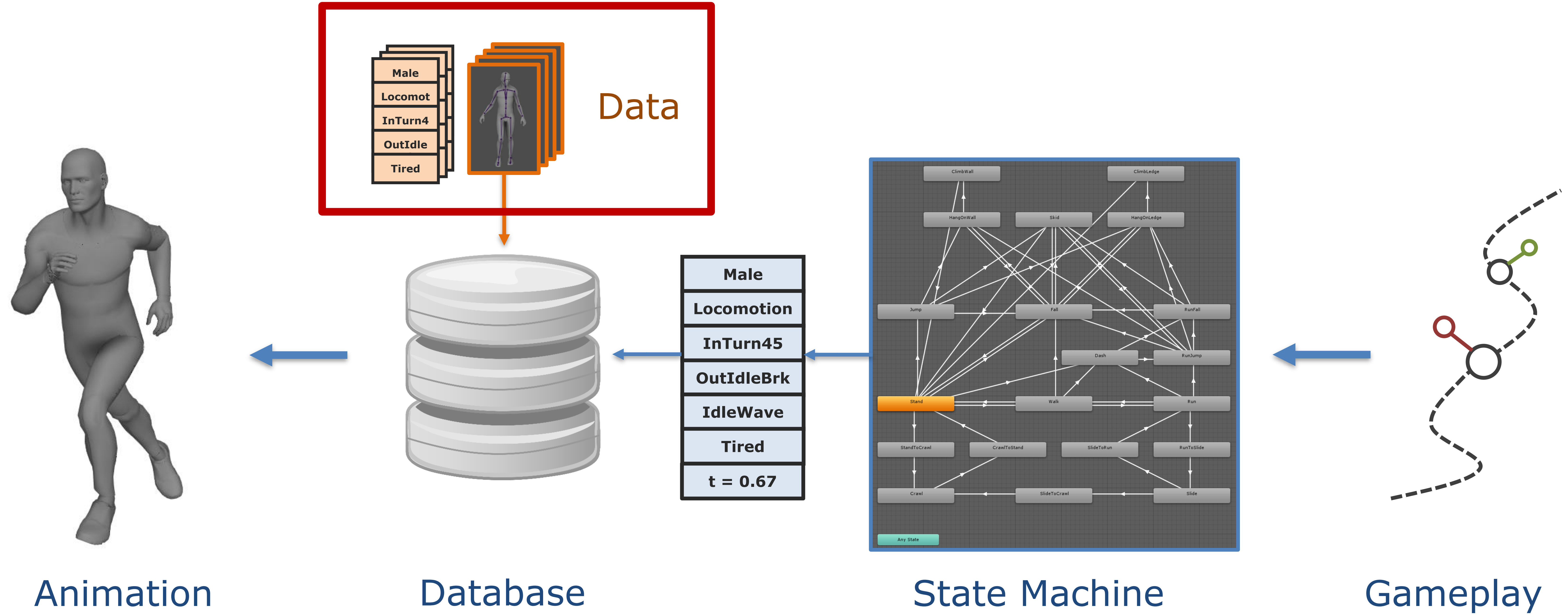

Data
asc6_rel2_char3
male_loco_slow_
t45_to_idlebrk_
wave_happy_
v013_final.fbx
t = 0.67
ClimbWall
ClimbLedge
HangOnWall
Skid
HangOnLedge
Jump
Fall
RunFall
Dash
RunJump
Stand
Walk
Run
StandToCrawl
CrawlToStand
SlideToRun
RunToSlide
Crawl
SlideToCrawl
Slide
Any State
Animation
Database
State Machine
Gameplay


Data
asc6_rel2_char3 male_loco_slow_ t45_to_idlebrk_ wave_happy_ v013_final.fbx
t = 0.67
w = 0.3
asc6_rel0_char5 male_loco_ t35_to_idlebrk_ wave_neutral_ v001.fbx
t = 0.31
w = 0.7
ClimbWall
ClimbLedge
HangOnWall
Skid
HangOnLedge
Jump
Fall
RunFall
Dash
RunJump
Stand
Walk
Run
StandToCrawl
CrawlToStand
SlideToRun
RunToSlide
Crawl
SlideToCrawl
Slide
Any State
Animation
Database
State Machine
Gameplay

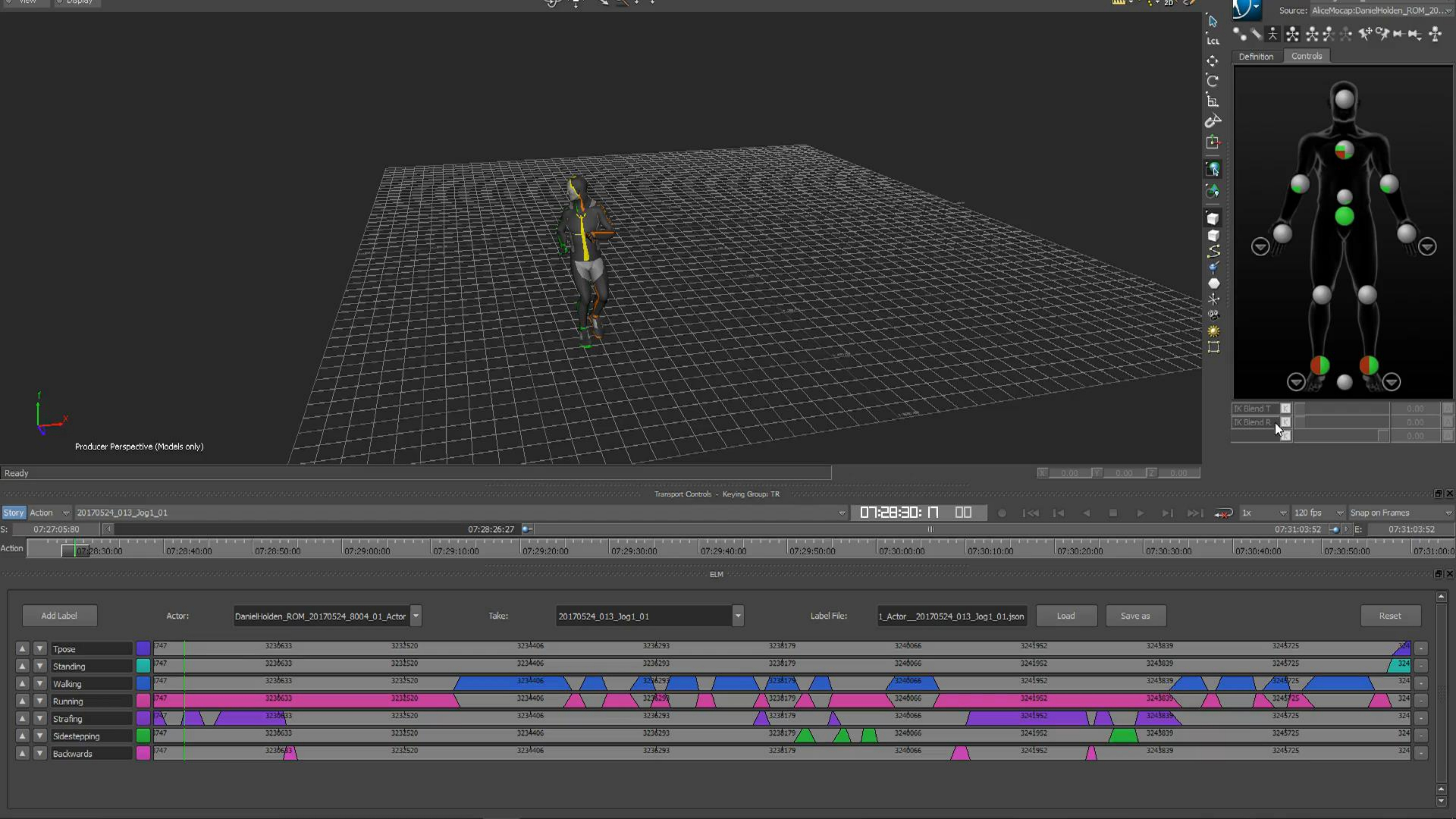

Data
asc6_rel2_char3
male_loco_slow_
t45_to_idlebrk_
wave_happy_
v013_final.fbx
t = 0.67
Animation
Database
State Machine
Gameplay


asc6_rel2_char3
male_loco_slow_
t45_to_idlebrk_
wave_happy_
v013_final.fbx
Data
ClimbWall
ClimbLedge
HangOnWall
Skid
HangOnLedge
Jump
Fall
RunFall
Dash
RunJump
Stand
Walk
Run
StandToCrawl
CrawlToStand
SlideToRun
RunToSlide
Crawl
SlideToCrawl
Slide
Any State
asc6_rel2_char3
male_loco_slow_
t45_to_idlebrk_
wave_happy_
v013_final.fbx
t = 0.67
Animation
Database
State Machine
Gameplay


Male
Locomot
InTurn4
OutIdle
Tired
Data
Male
Locomotion
InTurn45
OutIdleBrk
IdleWave
Tired
t = 0.67
ClimbWall
ClimbLedge
HangOnWall
Skid
HangOnLedge
Jump
Fall
RunFall
Dash
RunJump
Stand
Walk
Run
StandToCrawl
CrawlToStand
SlideToRun
RunToSlide
Crawl
SlideToCrawl
Slide
Any State
Animation
Database
State Machine
Gameplay



Male
Locomot
InTurn4
OutIdle
Tired
Data
Male
Locomotion
InTurn45
OutIdleBrk
IdleWave
Tired
t = 0.67
ClimbWall
ClimbLedge
HangOnWall
Skid
HangOnLedge
Jump
Fall
RunFall
Dash
RunJump
Stand
Walk
Run
StandToCrawl
CrawlToStand
SlideToRun
RunToSlide
Crawl
SlideToCrawl
Slide
Any State
Animation
Database
State Machine
Gameplay

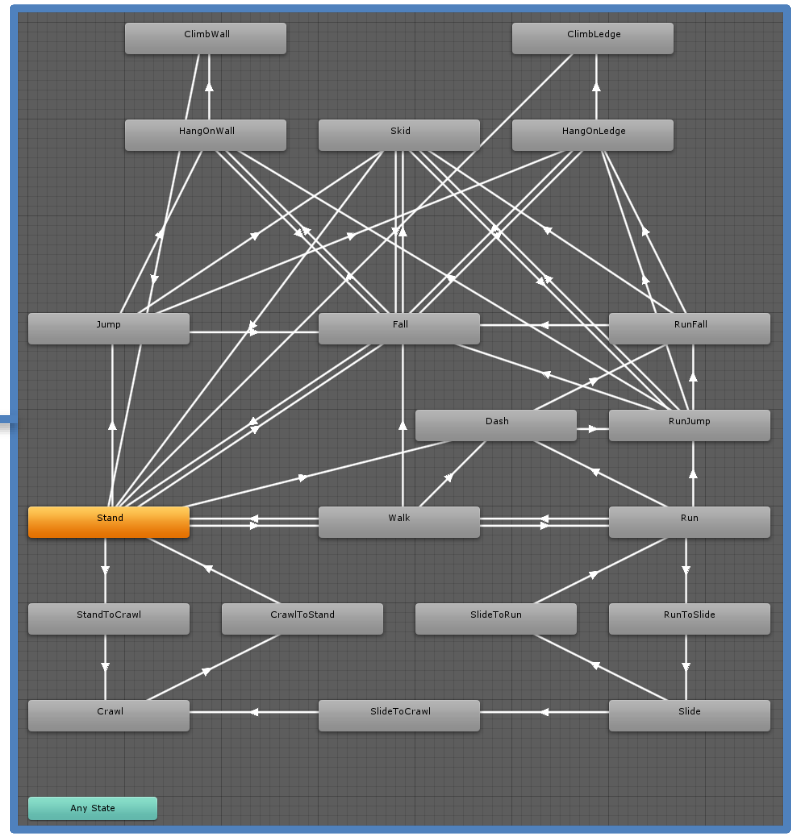


Source: AliceMocap:DanielHolden_ROM_20...
Definition
Controls
IK Blend T
IK Blend R
Producer Perspective (Models only)
Ready
Transport Controls - Keying Group: TR
Story
Action
20170524_013_Jog1_01
07:28:30:17
1x
120 fps
Snap on Frames
07:27:05:80
07:28:26:27
07:31:03:52
ELM
Add Label
Actor:
DanielHolden_ROM_20170524_8004_01_Actor
Take:
20170524_013_Jog1_01
Label File:
1_Actor__20170524_013_Jog1_01.json
Load
Save as
Reset
Tpose
Standing
Walking
Running
Strafing
Sidestepping
Backwards

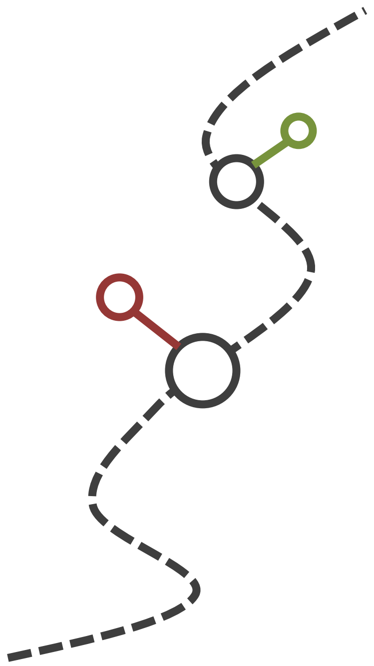
Animation

Database

Male
Locomotion
InTurn45
OutIdleBrk
IdleWave
Tired
t = 0.67



State Machine



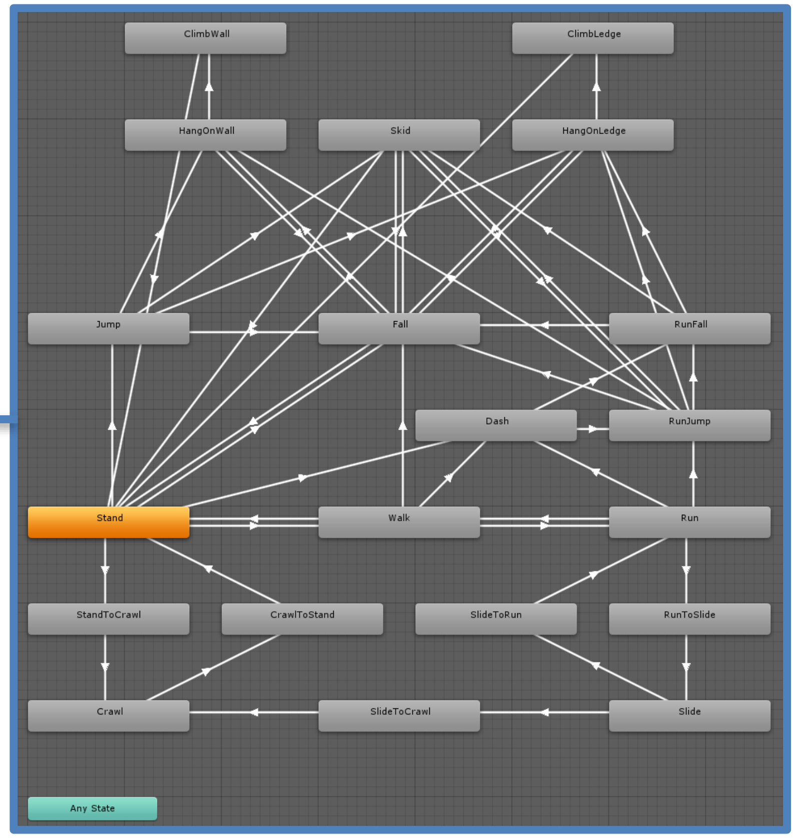
Gameplay

Swap out for new characters

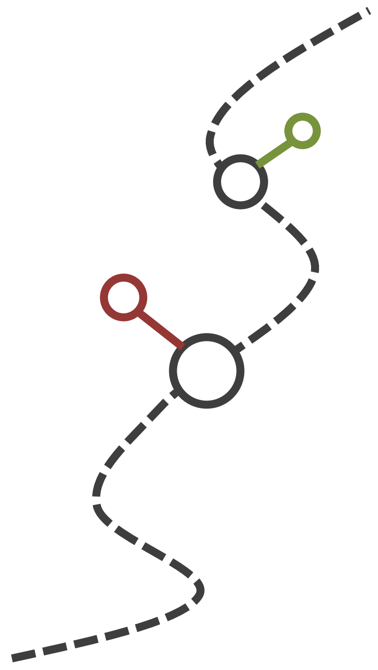
Animation

Database

Male
Locomotion
InTurn45
OutIdleBrk
IdleWave
Tired
t = 0.67



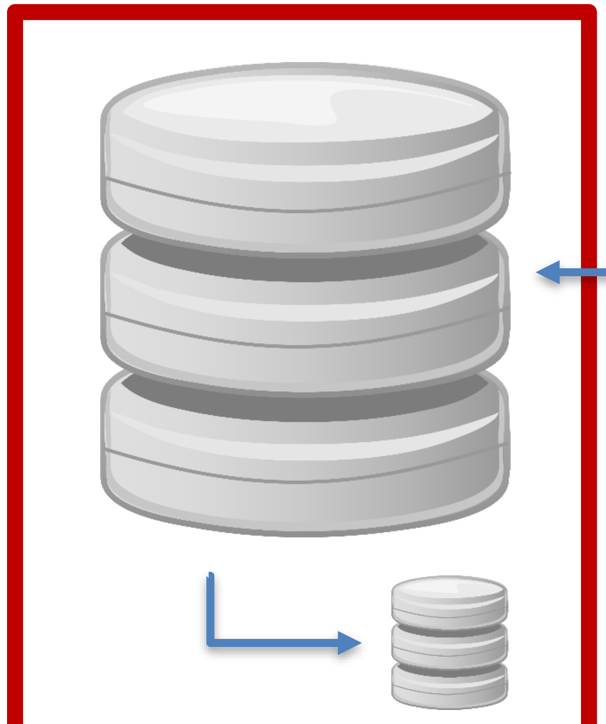
State Machine



Gameplay

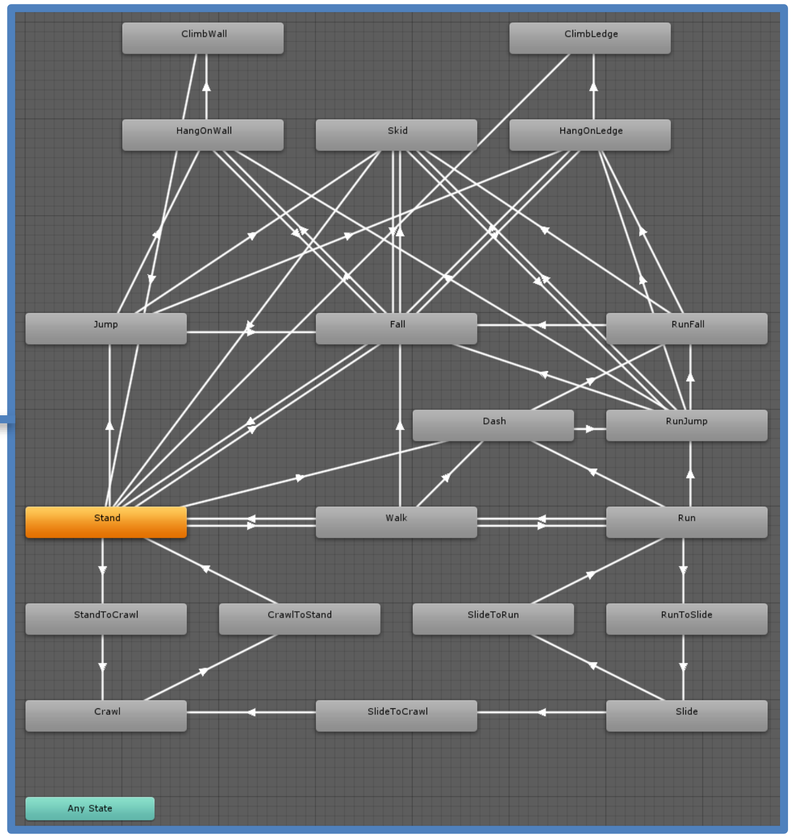
Fallback for missing data

Animation

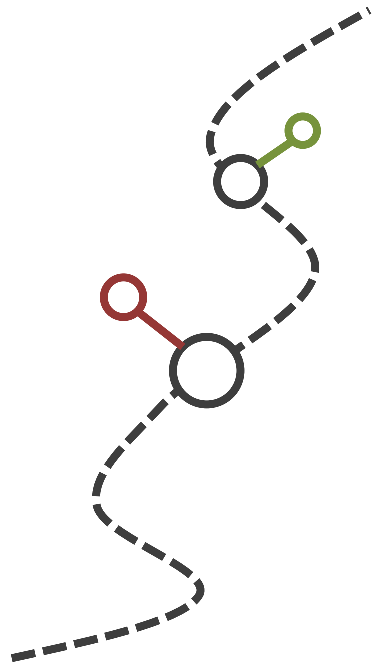


Database

Male
Locomotion
InTurn45
OutIdleBrk
IdleWave
Tired
t = 0.67



State Machine



Gameplay

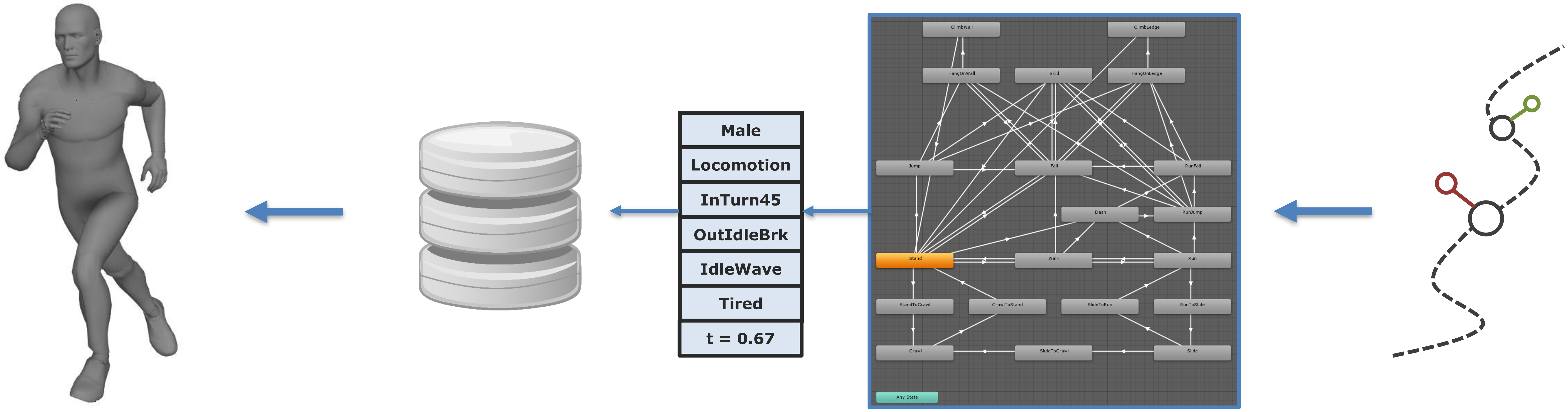
Assets $\longrightarrow$ Databases

Filenames → Tags

Separate Motion Retrieval

- **Separate Data** ✓
- **Specify Desired Variables**
- **Generalize Solution**

Desired Variables



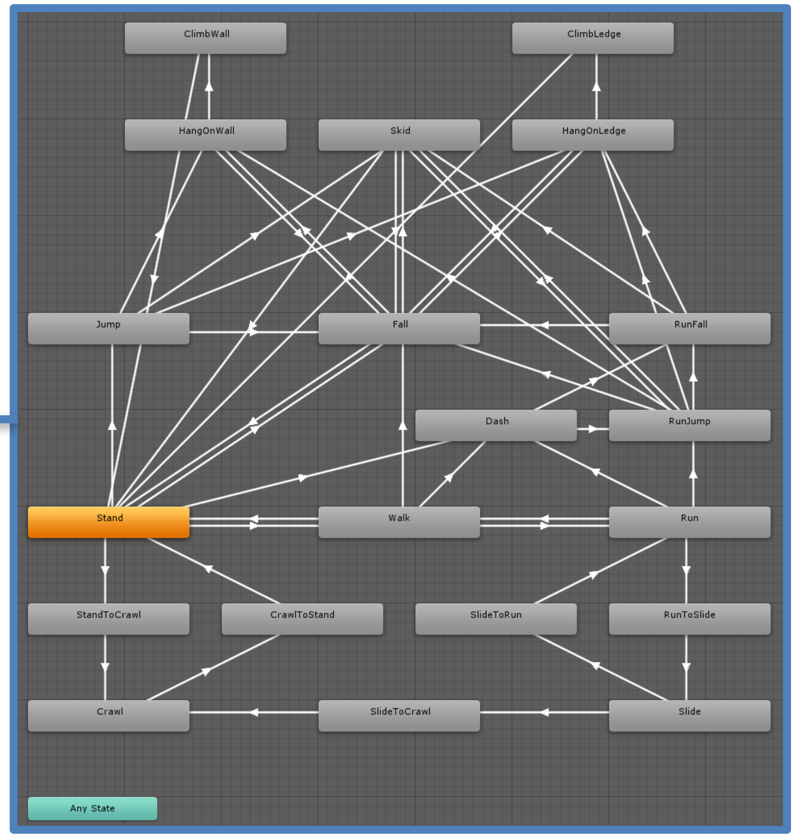
Animation Database State Machine Gameplay

Mix of Gameplay and Animation

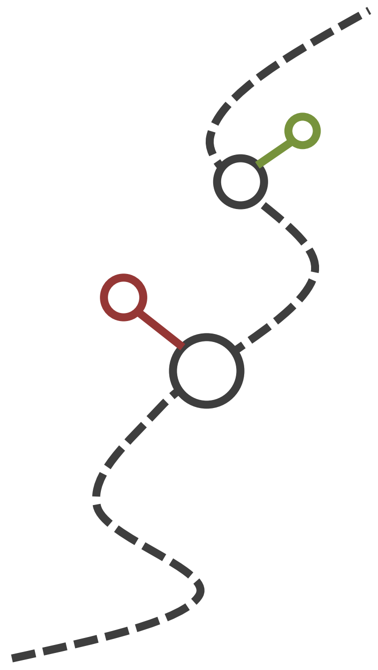
Animation

Database

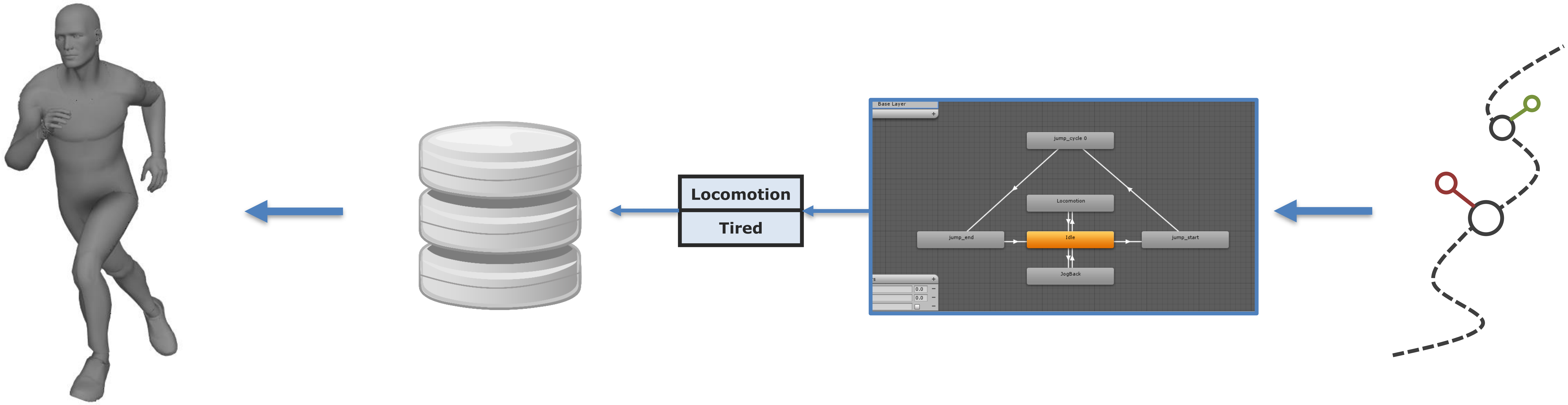
Male
Locomotion
InTurn45
OutIdleBrk
IdleWave
Tired
t = 0.67



State Machine



Gameplay



Animation

Database

Simplified State Machine

Gameplay

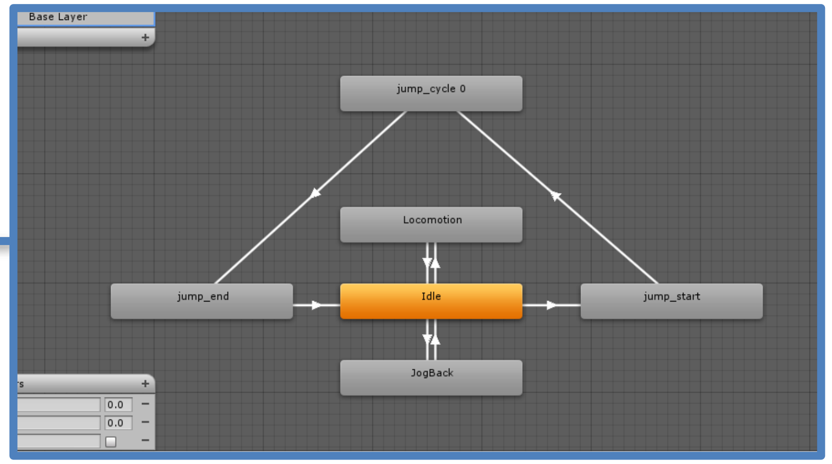
Gameplay Only

Animation

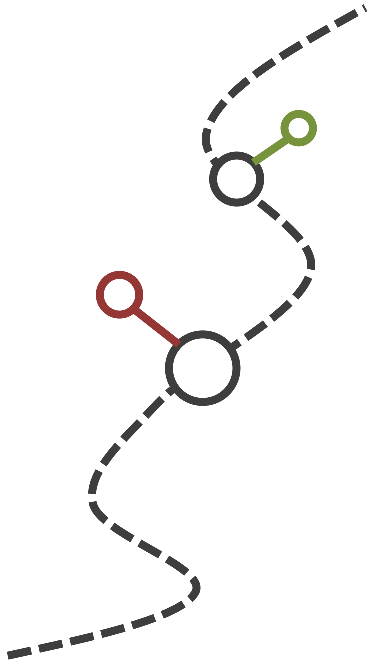
Database

Locomotion

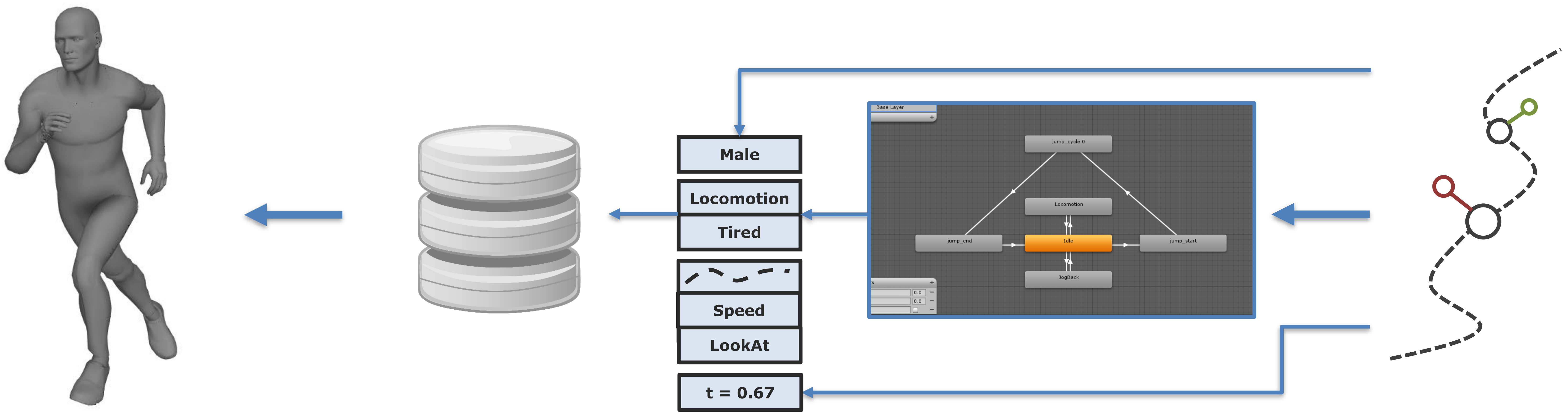
Tired



Simplified State Machine



Gameplay

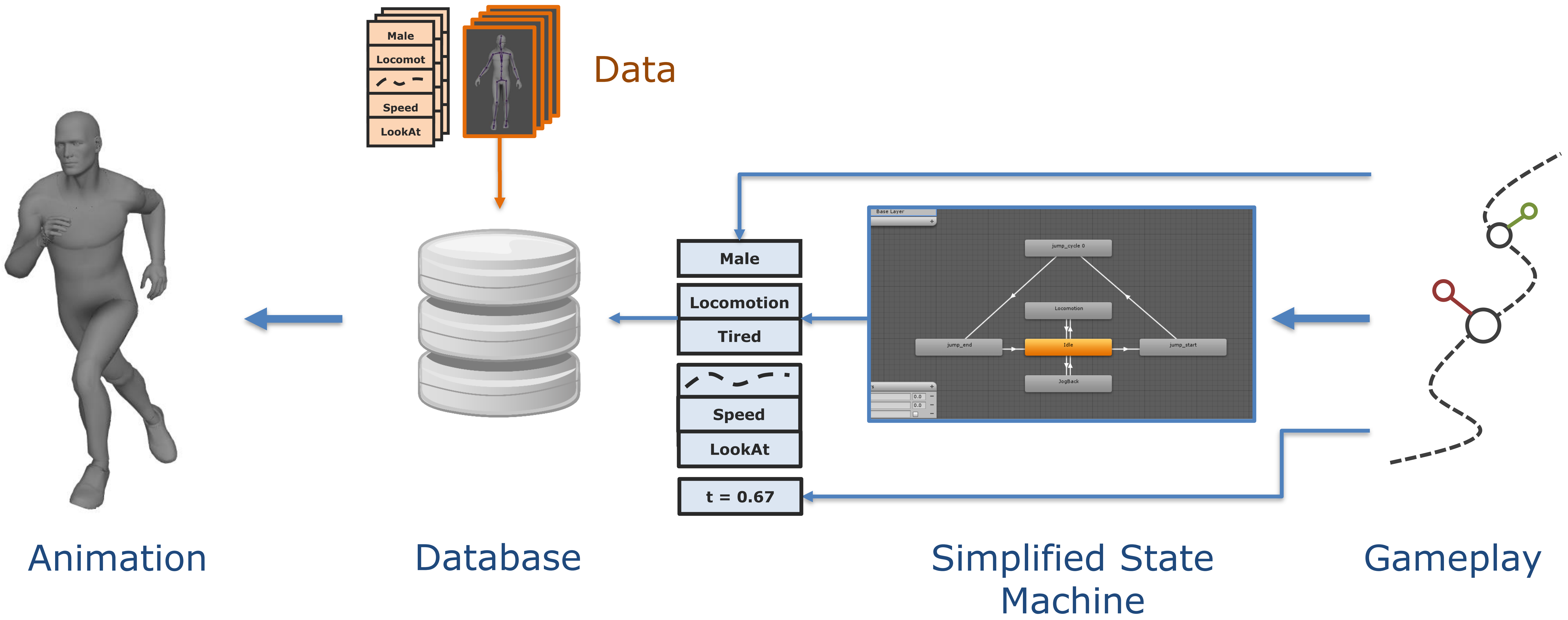


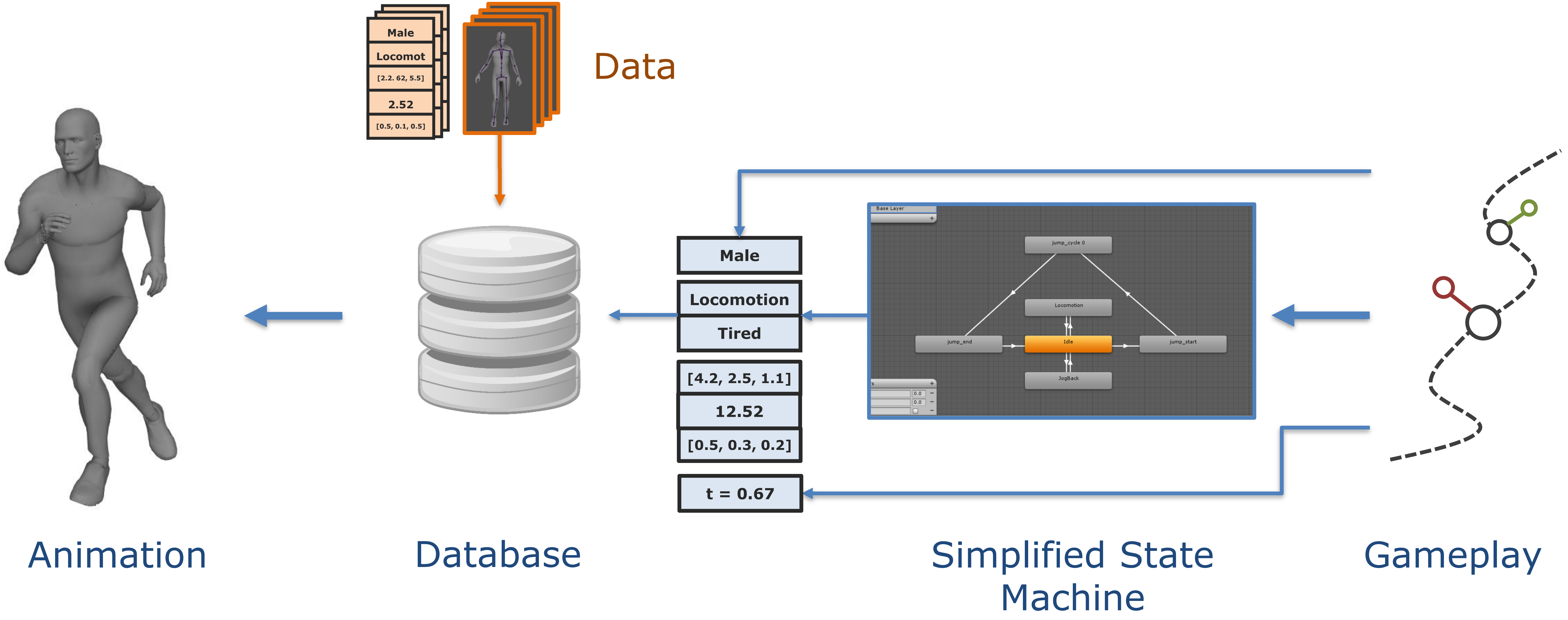
Animation

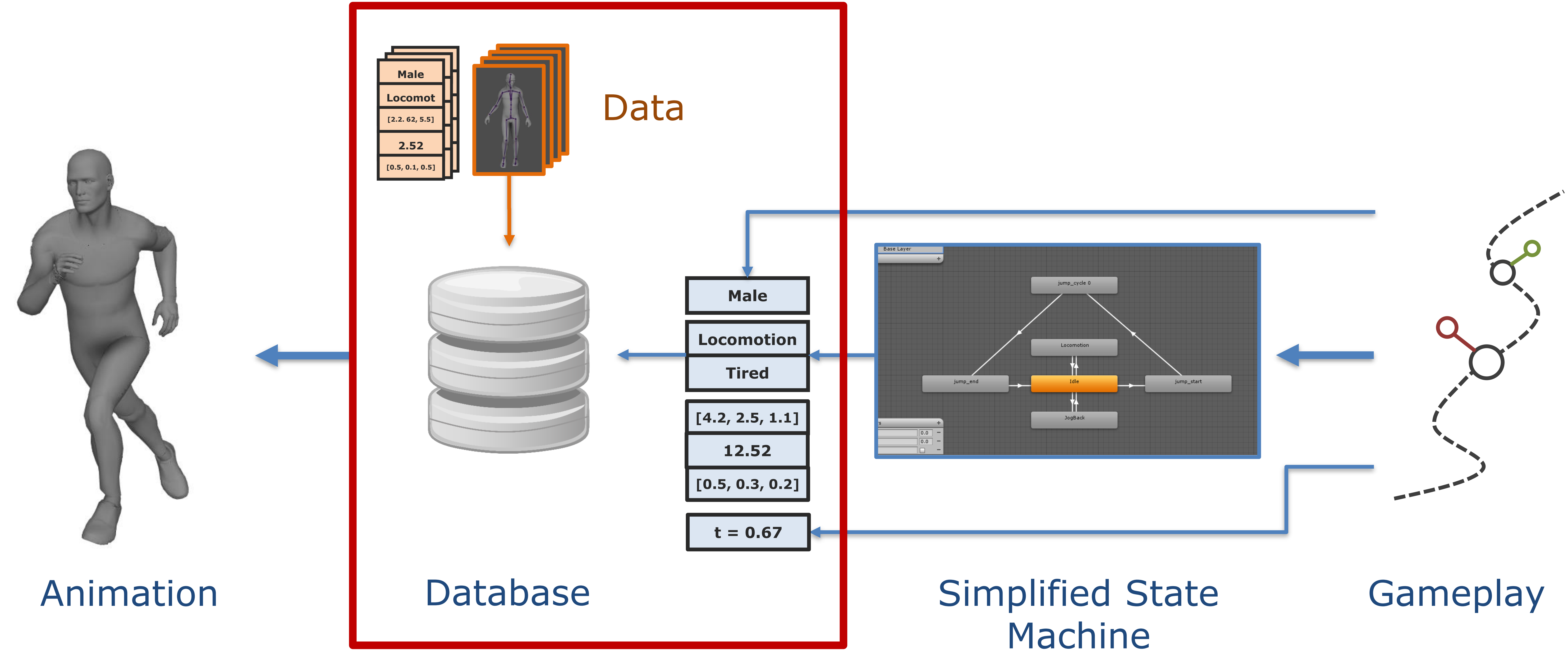
Database

Simplified State Machine

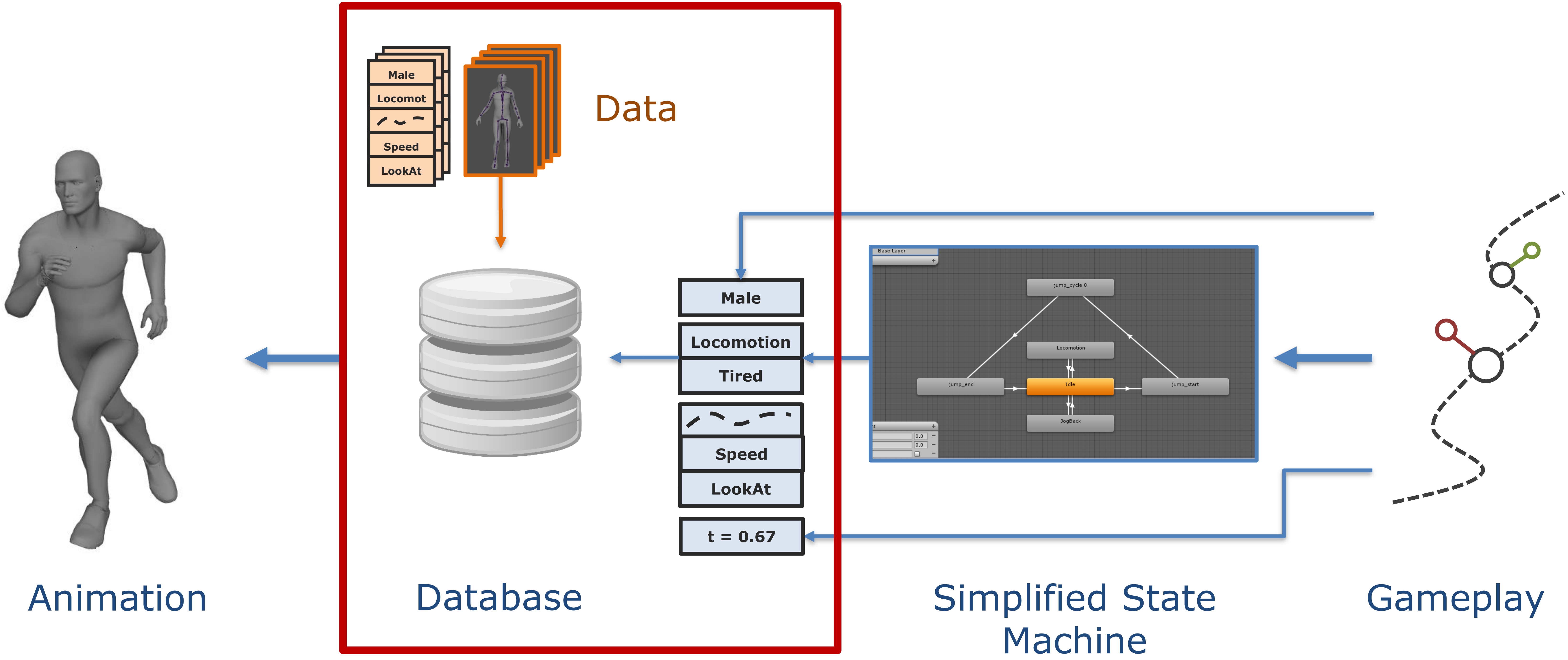
Gameplay

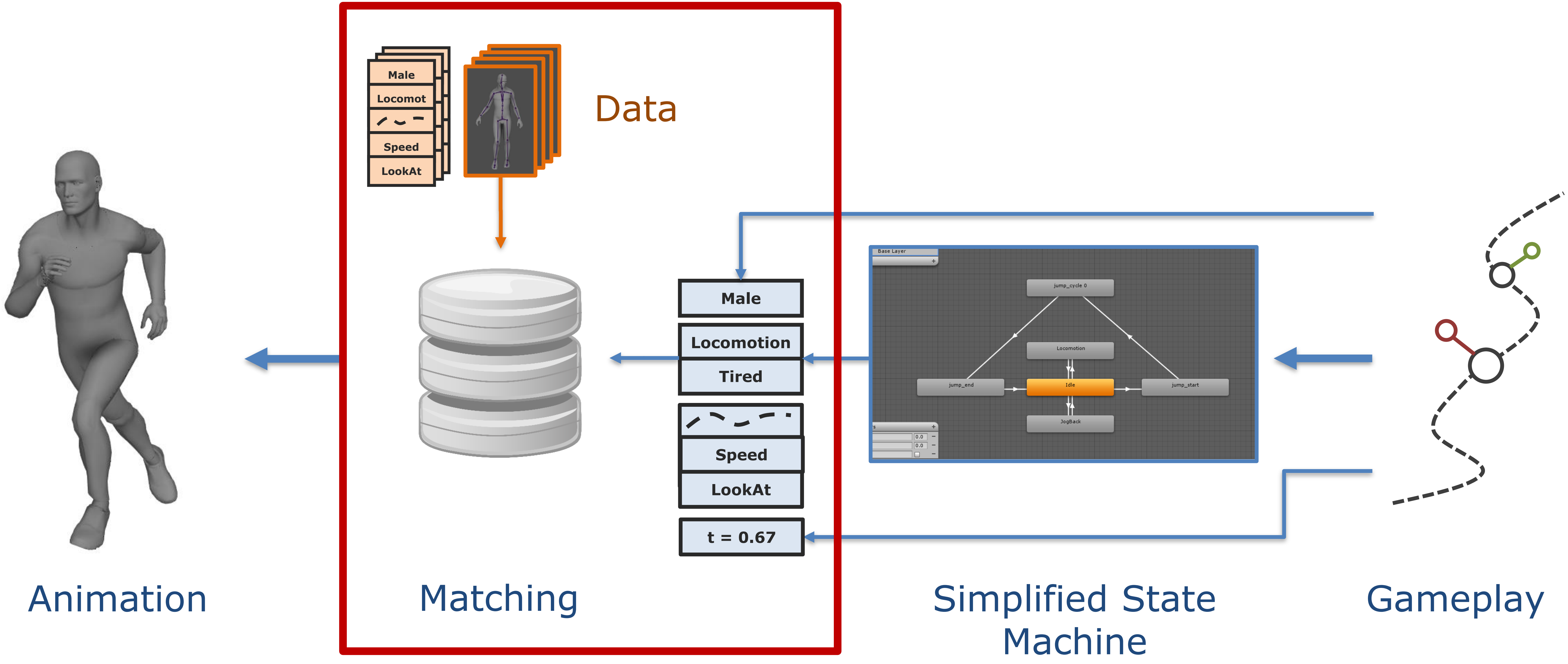

Male
Locomot
Speed
LookAt
Data
Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
Database
Simplified State Machine
Gameplay

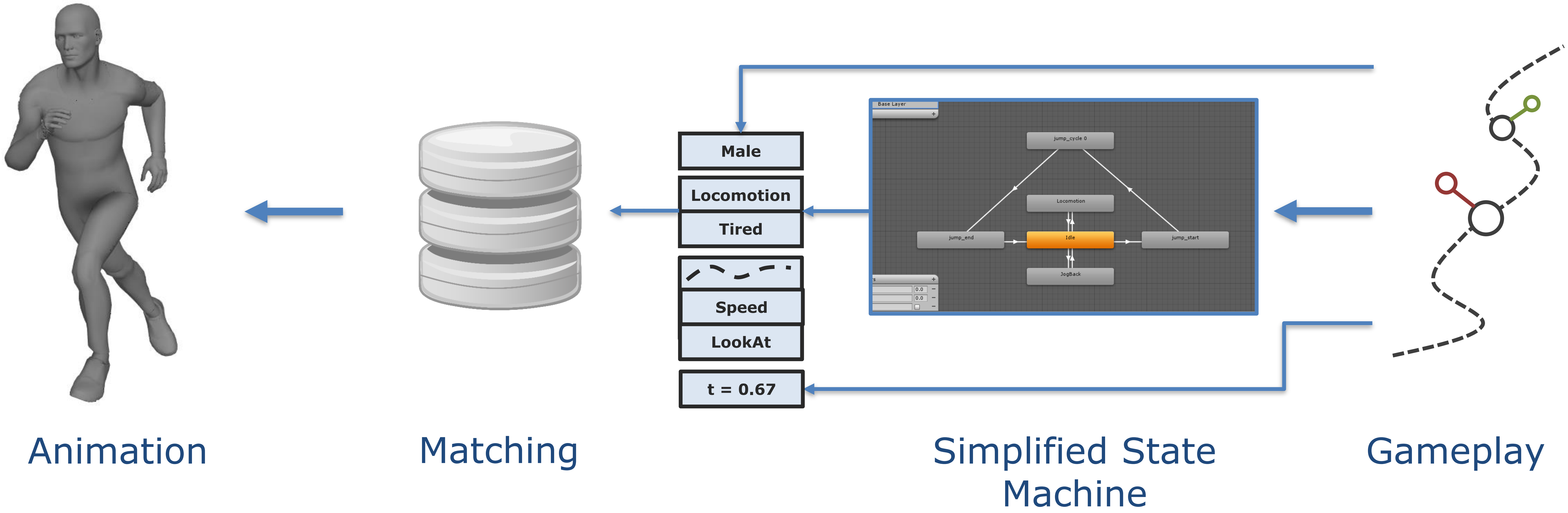

Male
Locomot
[2.2. 62, 5.5]
2.52
[0.5, 0.1, 0.5]
Data
Male
Locomotion
Tired
[4.2, 2.5, 1.1]
12.52
[0.5, 0.3, 0.2]
t = 0.67
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
Database
Simplified State Machine
Gameplay

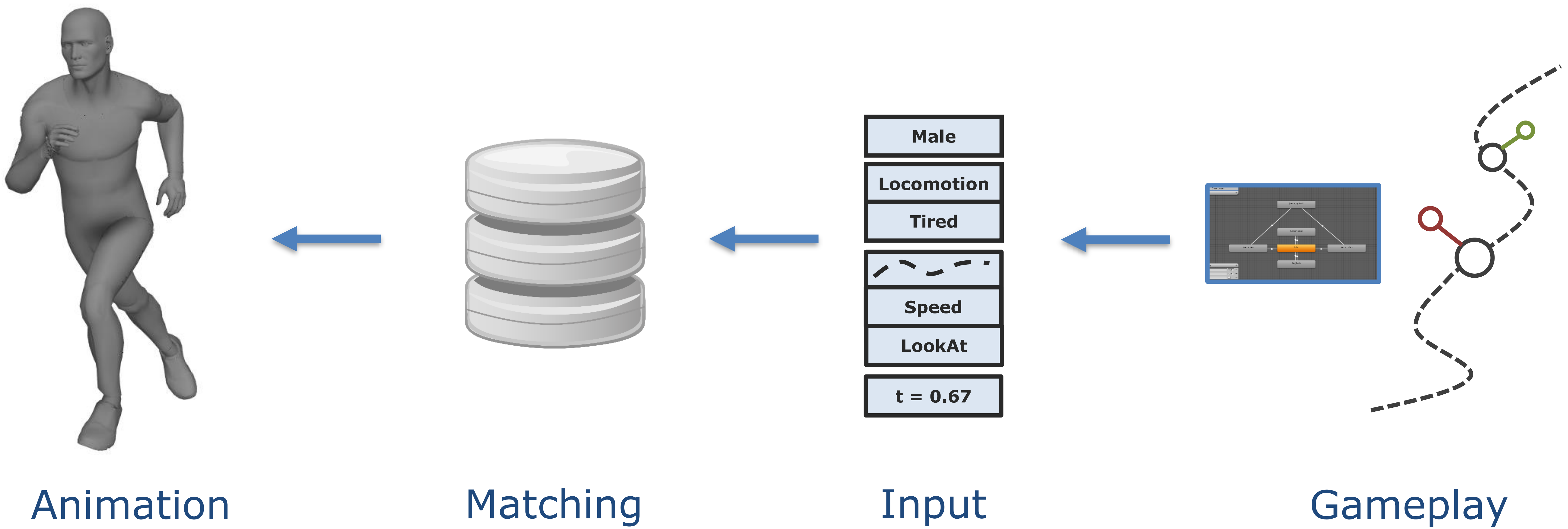

Male
Locomot
[2.2. 62, 5.5]
2.52
[0.5, 0.1, 0.5]
Data
Male
Locomotion
Tired
[4.2, 2.5, 1.1]
12.52
[0.5, 0.3, 0.2]
t = 0.67
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
Database
Simplified State Machine
Gameplay

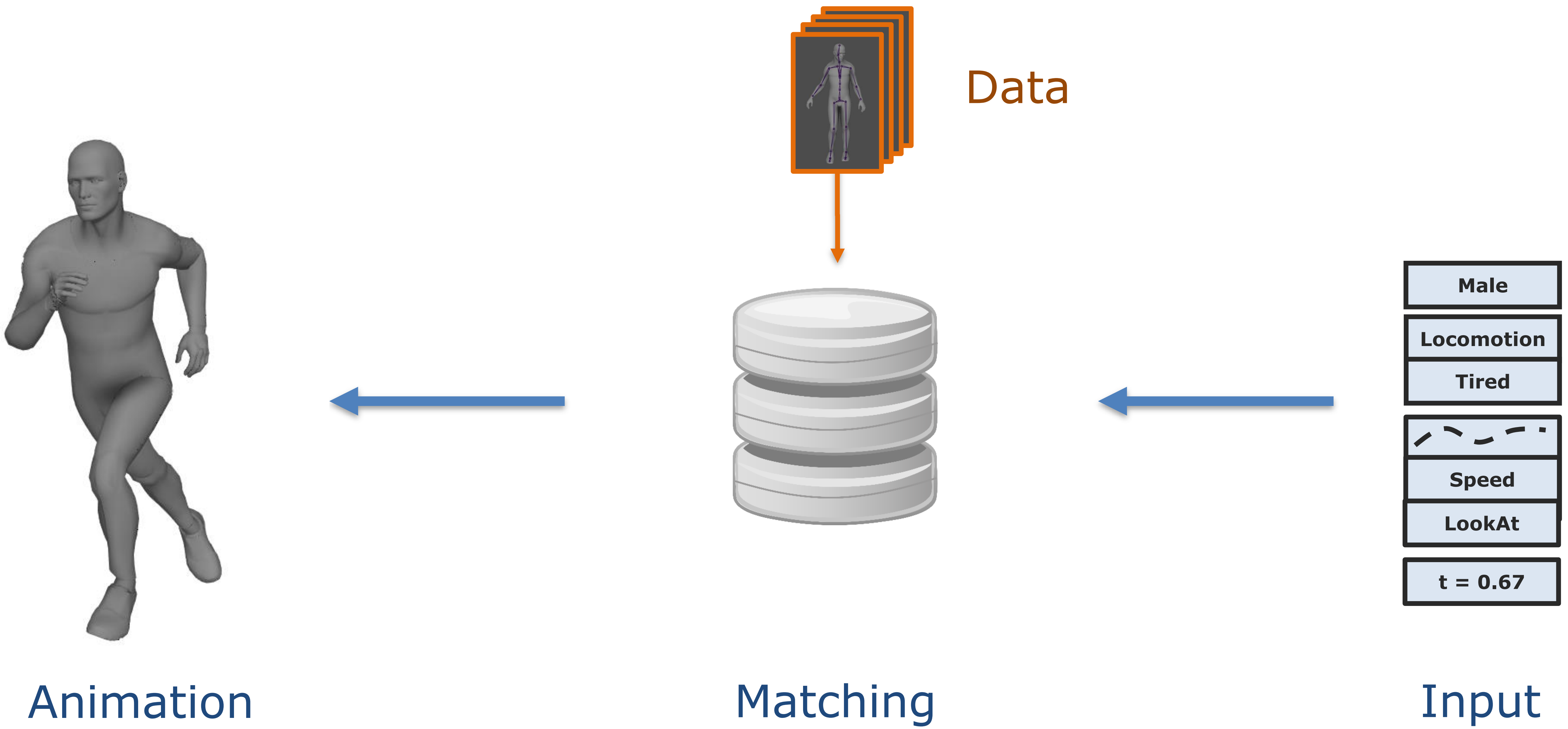
1. Filter out clips where the discrete tags don’t match.
2. Return the clip with the nearest numerical match.

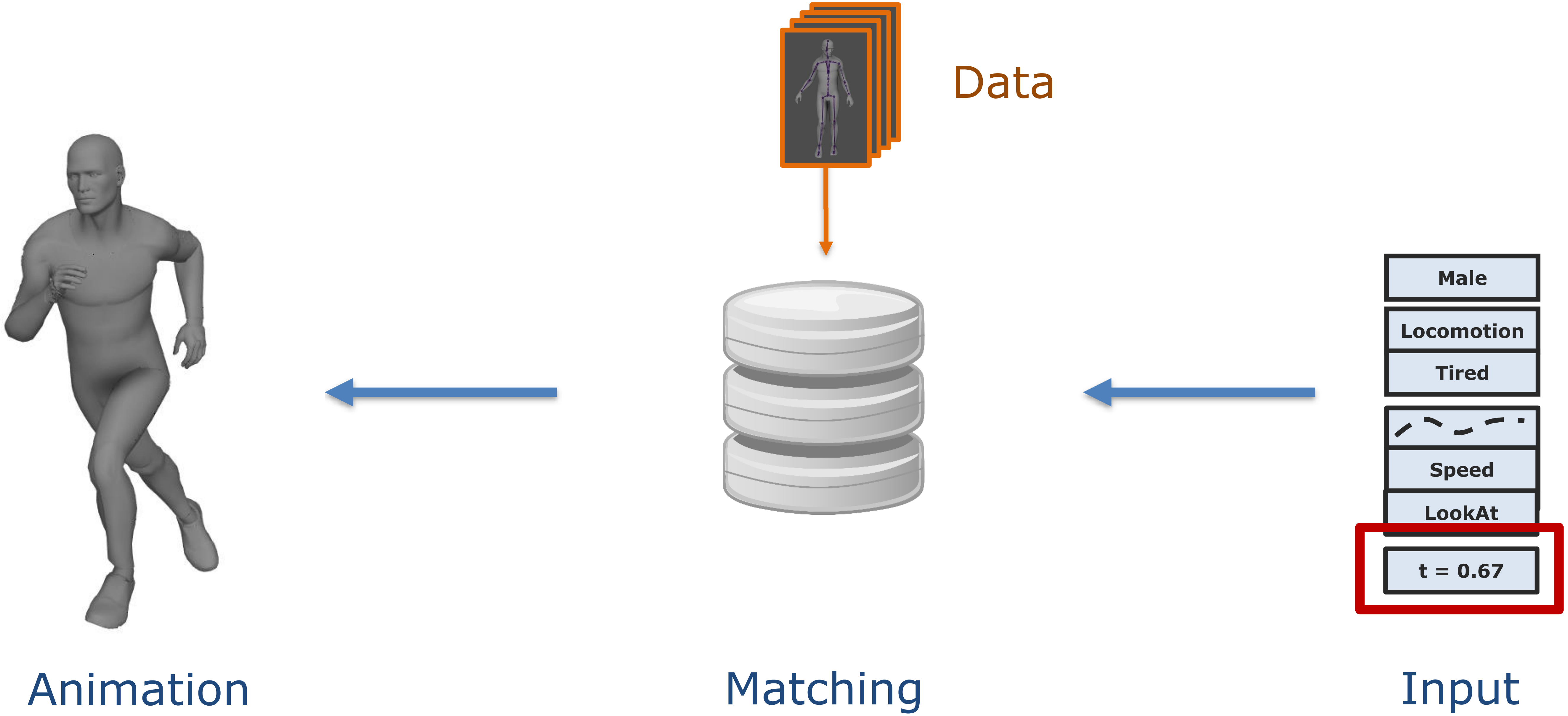

Male
Locomot
Speed
LookAt
Data
Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
Database
Simplified State Machine
Gameplay

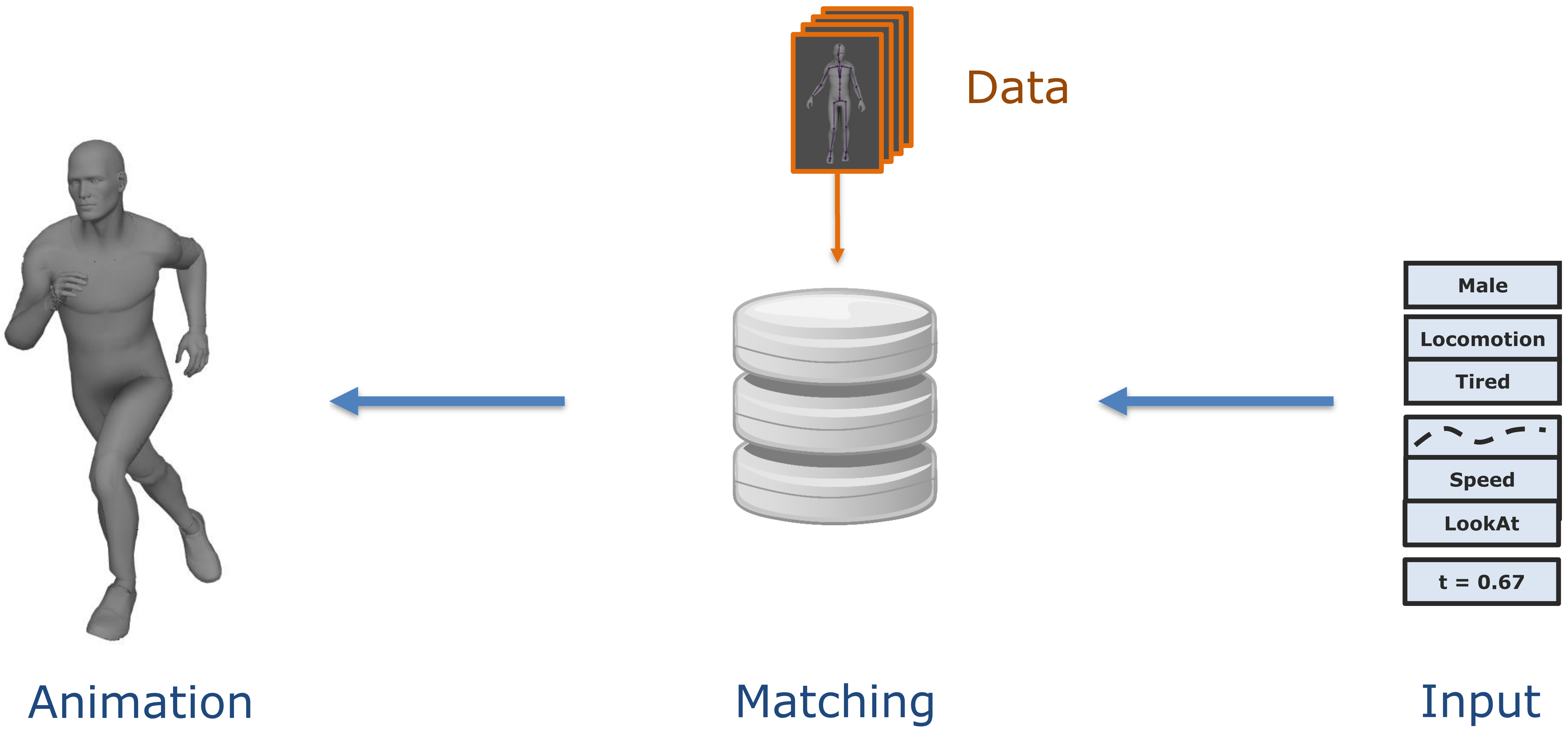

Male
Locomot
Speed
LookAt
Data
Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
Matching
Simplified State
Machine
Gameplay

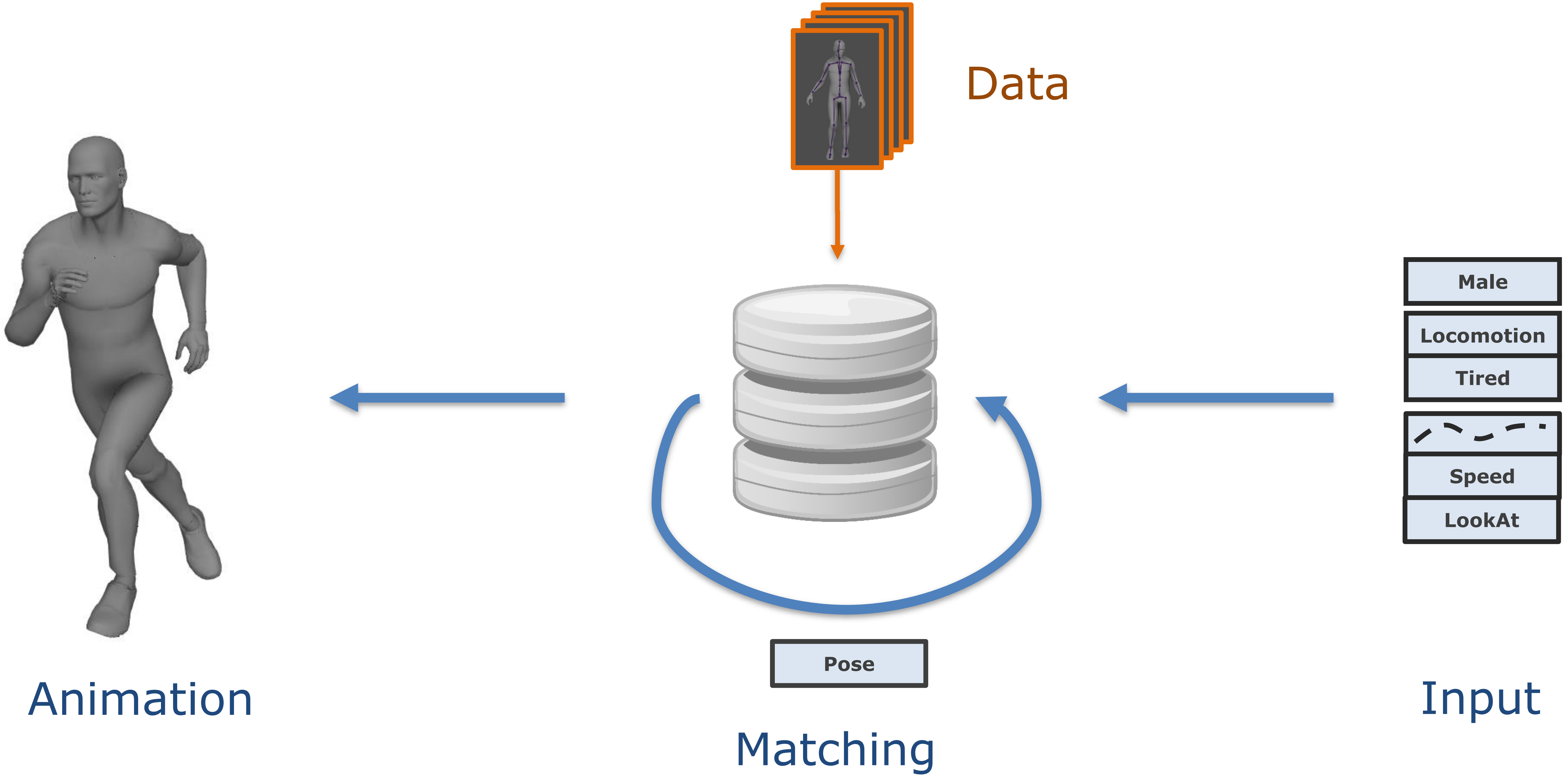

Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Base Layer
jump_cycle 0
Locomotion
jump_end
Idle
jump_start
JogBack
Animation
Matching
Simplified State Machine
Gameplay


Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Animation
Matching
Input
Gameplay


Data
Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Animation
Matching
Input

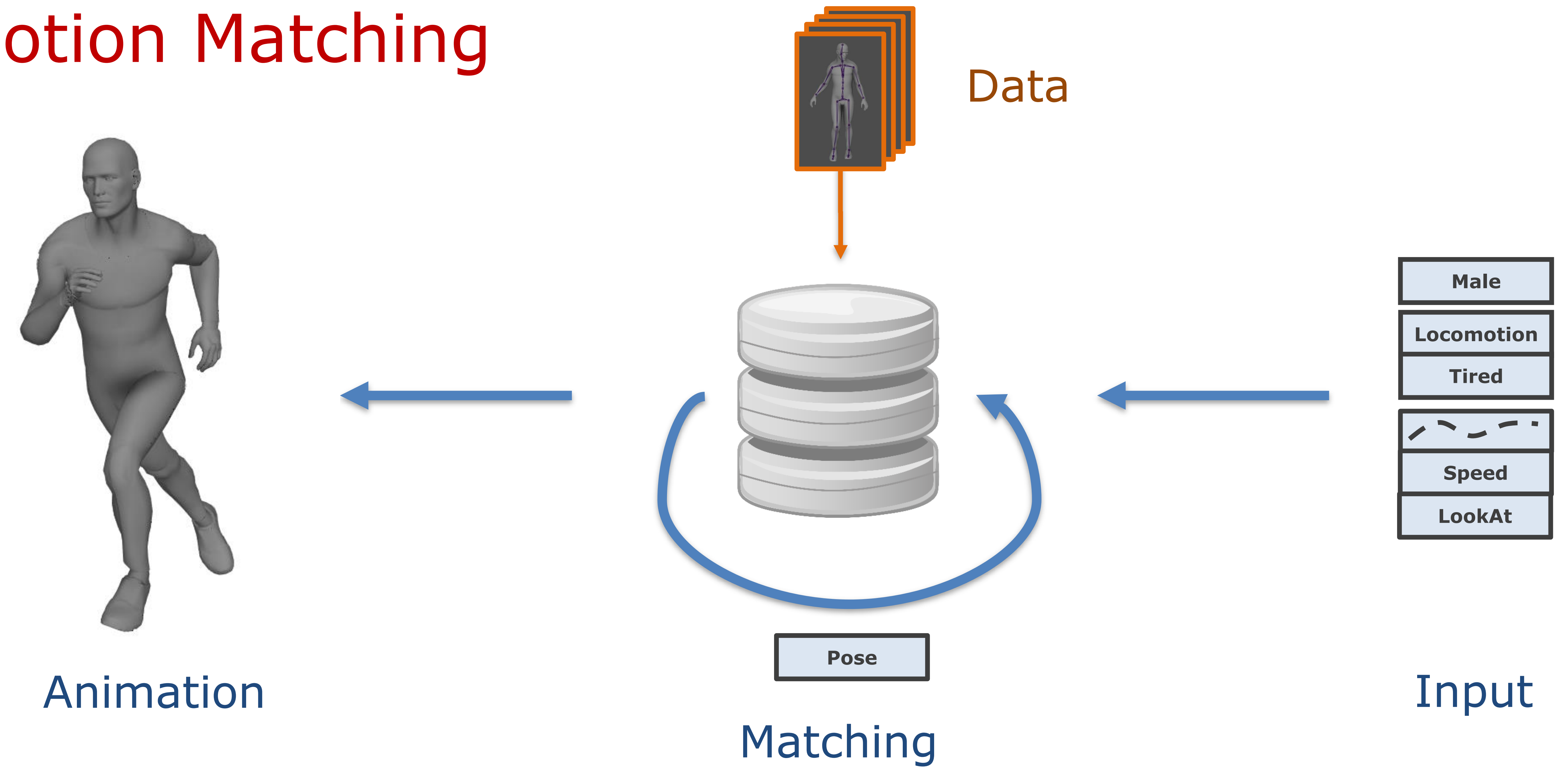

Data
Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Animation
Matching
Input

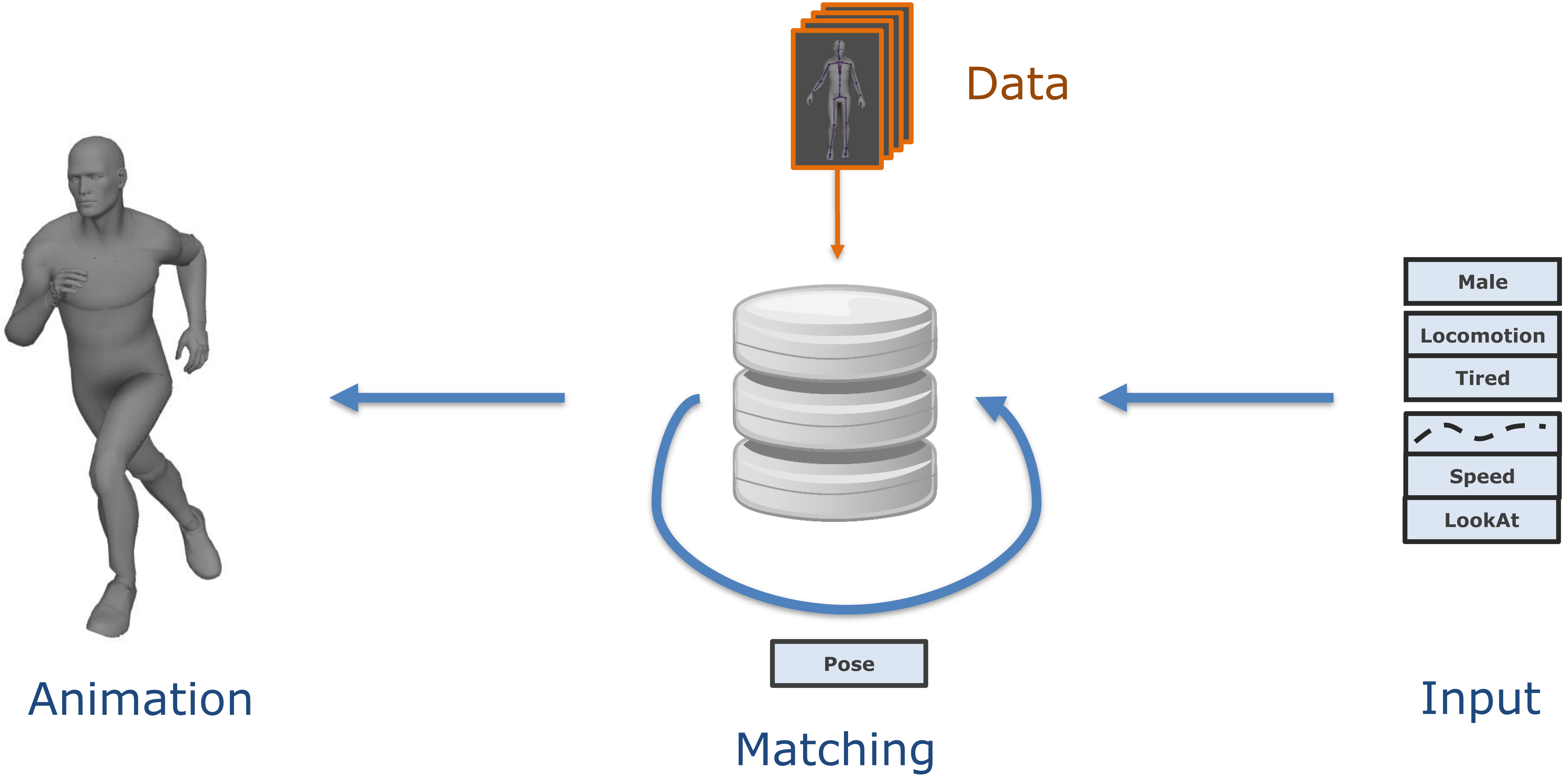

Data
Male
Locomotion
Tired
Speed
LookAt
t = 0.67
Animation
Matching
Input



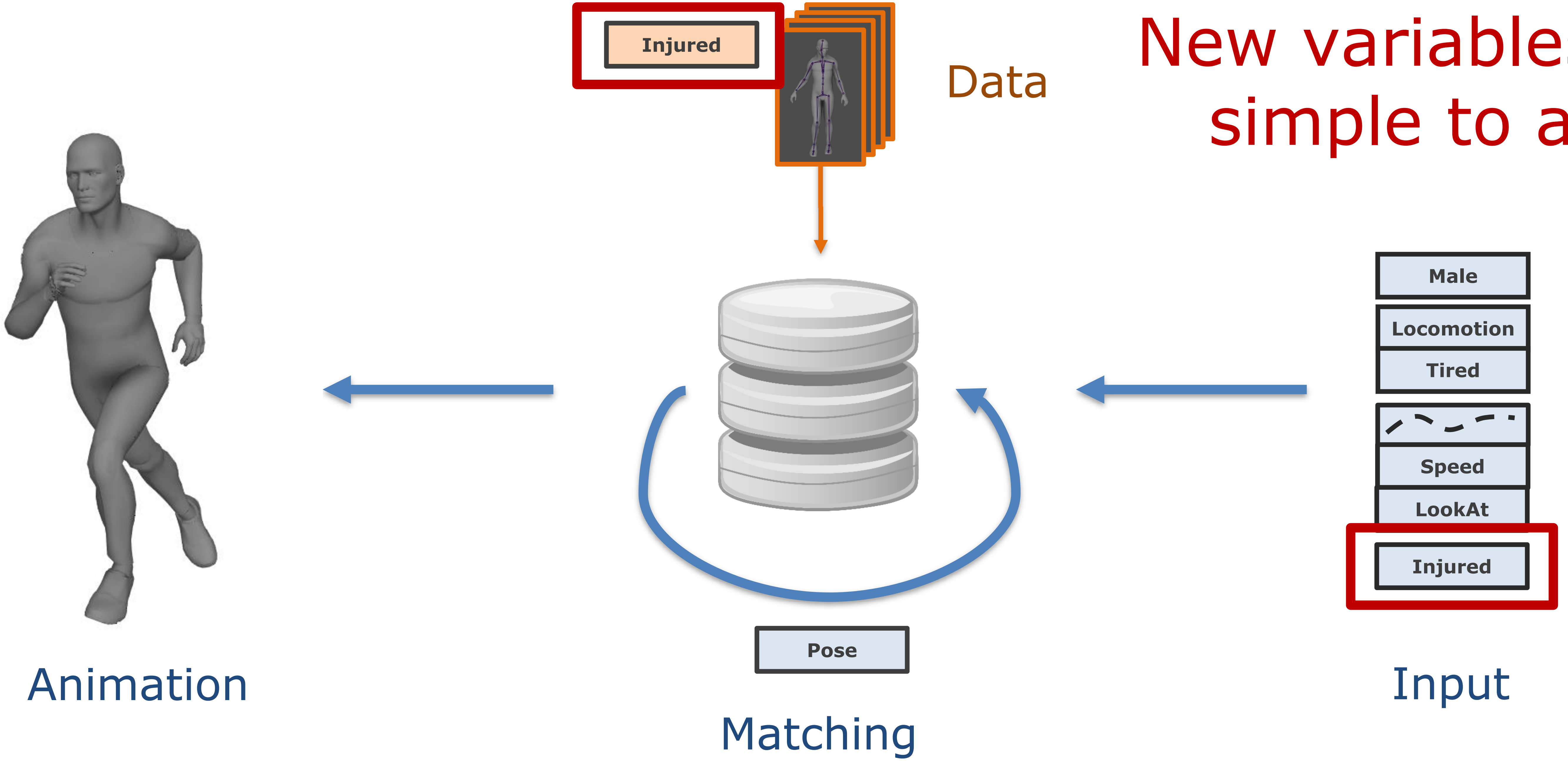
Data
Male
Locomotion
Tired
Speed
LookAt
Pose
Animation
Matching
Input

Motion Matching




Data
Male
Locomotion
Tired
Speed
LookAt
Pose
Animation
Matching
Input

New variables are simple to add



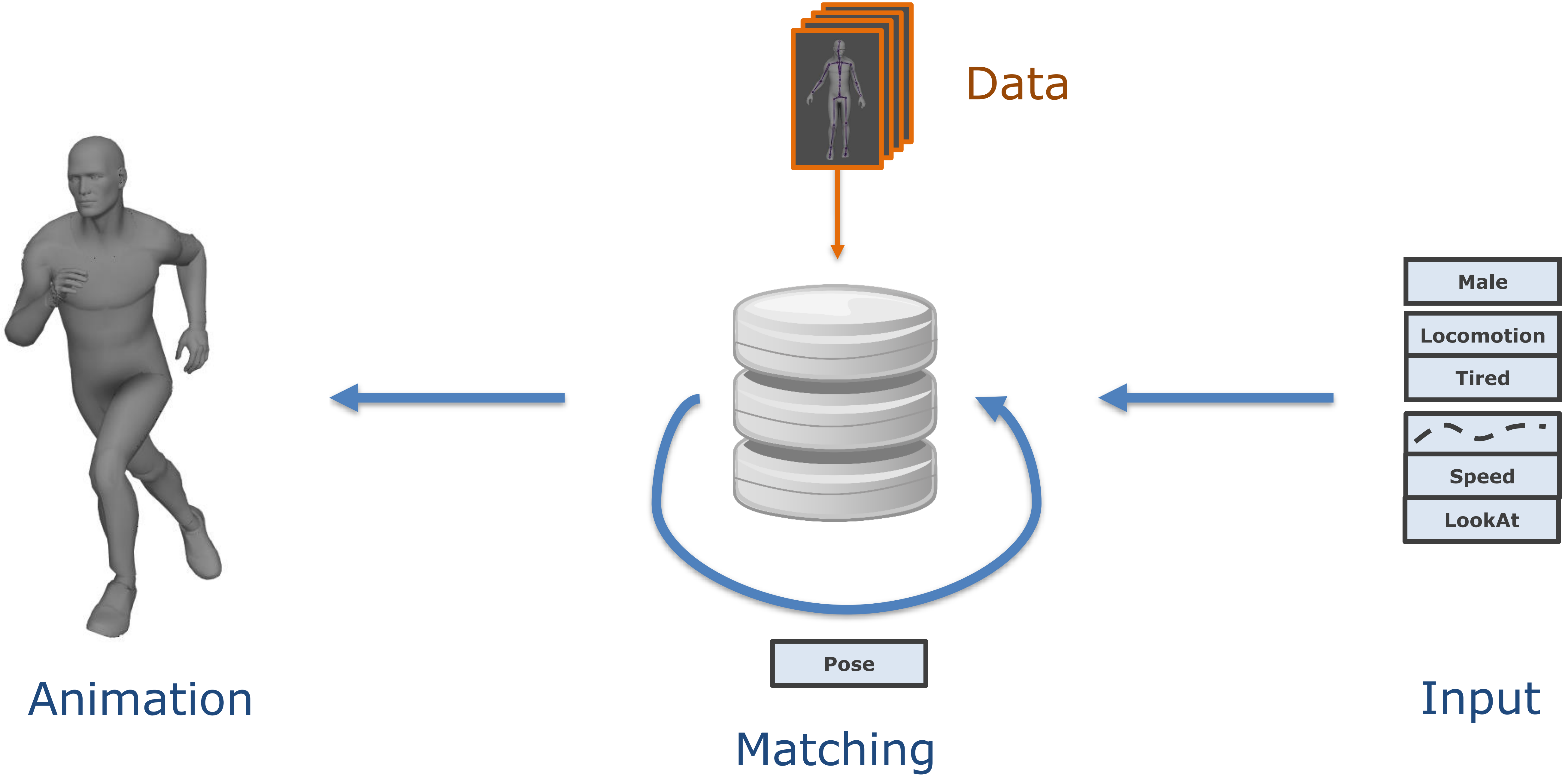
States → Variables

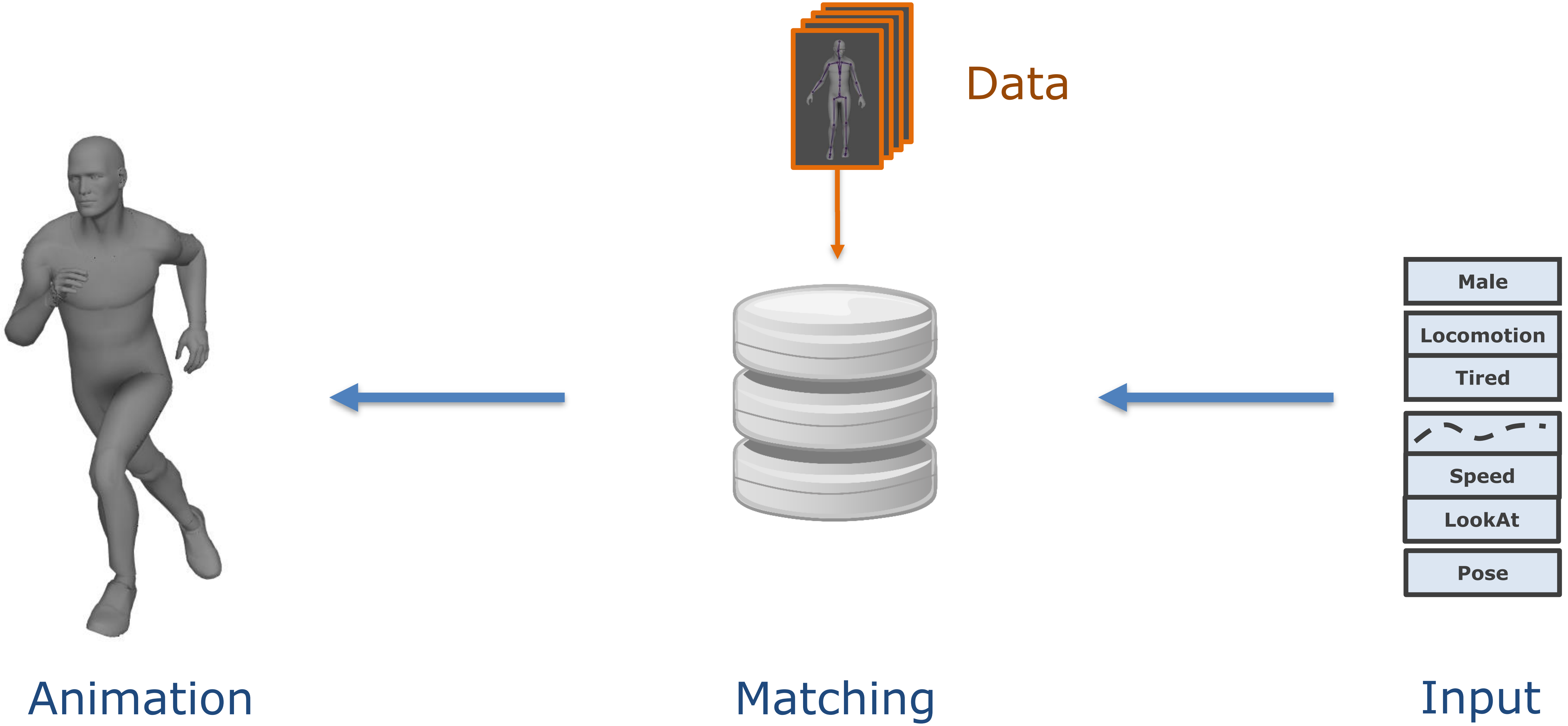
Querying → Matching

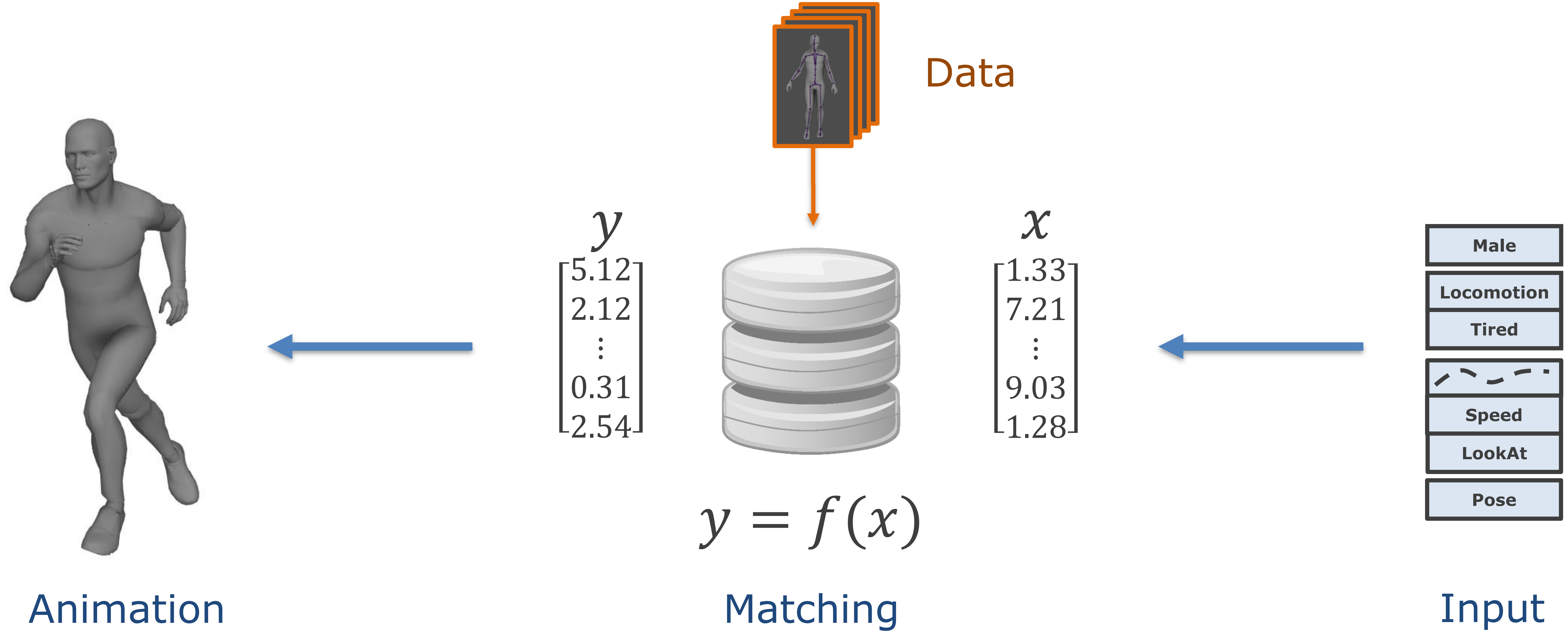
Annotate Variables in Data

- Separate Data ✓
- Specify Desired Variables ✓
- Generalize Solution

Generalize Solution


Data
Male
Locomotion
Tired
Speed
LookAt
Pose
Animation
Matching
Input


Data
Male
Locomotion
Tired
Speed
LookAt
Pose
Animation
Matching
Input


Data
y
5.12
2.12
⋮
0.31
2.54
x
1.33
7.21
⋮
9.03
1.28
Male
Locomotion
Tired
Speed
LookAt
Pose
y = f(x)
Animation
Matching
Input

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{array}{r} \text{Female} \\ \text{Male} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} \boxed{\begin{matrix} 0 \\ 1 \end{matrix}} \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{array}{r} \\ \\ \text{Attack} \\ \text{Idle} \\ \text{Locomotion} \\ \text{Falling} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} 0 \\ 1 \\ \boxed{0} \\ \boxed{0} \\ \boxed{1} \\ \boxed{0} \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

- Male
- Locomotion
- Tired
- Speed
- LookAt
- Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{array}{r l} & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix} \end{array}$$

Neutral
Tired
Drunk

Input

Male

Locomotion
Tired

Speed
LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ \boxed{\begin{matrix} 0.23 \\ 0.43 \\ 0.12 \end{matrix}} \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Goal Position

Input

- Male
- Locomotion
- Tired
- Speed
- LookAt
- Pose

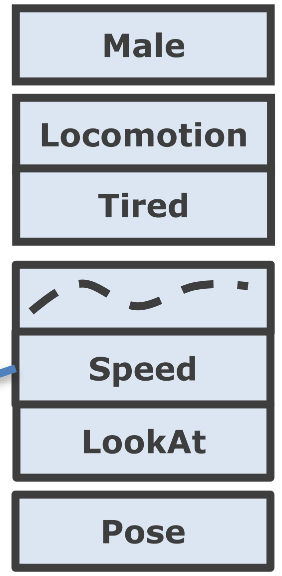
Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{array}{r} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \text{Goal Speed} \\ \\ \\ \\ \\ \end{array} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ \boxed{2.43} \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input



Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ \boxed{\begin{matrix} 0.91 \\ 0.39 \\ 0.01 \end{matrix}} \\ \vdots \end{bmatrix}$$

Goal Look At

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Pose

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

y

$$\begin{bmatrix} \boxed{\begin{matrix} 1.32 \\ 5.23 \\ 0.67 \end{matrix}} \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

First Joint Position

$$y = f(x)$$

x

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

First Joint Rotation

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ \boxed{\begin{matrix} 2.75 \\ 0.73 \\ 8.54 \end{matrix}} \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

Second Joint Position

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

Second Joint Rotation

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

Final Joint Rotation

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

- Male
- Locomotion
- Tired
- Speed
- LookAt
- Pose

Animation

$$y = \begin{bmatrix} 1.32 \\ 5.23 \\ 0.67 \\ 0.62 \\ 9.54 \\ 3.45 \\ 2.75 \\ 0.73 \\ 8.54 \\ 8.23 \\ 8.53 \\ 6.54 \\ \vdots \\ 7.34 \\ 4.23 \\ 6.23 \end{bmatrix}$$

$$y = f(x)$$

$$x = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0.23 \\ 0.43 \\ 0.12 \\ 2.43 \\ 0.91 \\ 0.39 \\ 0.01 \\ \vdots \end{bmatrix}$$

Input

Male

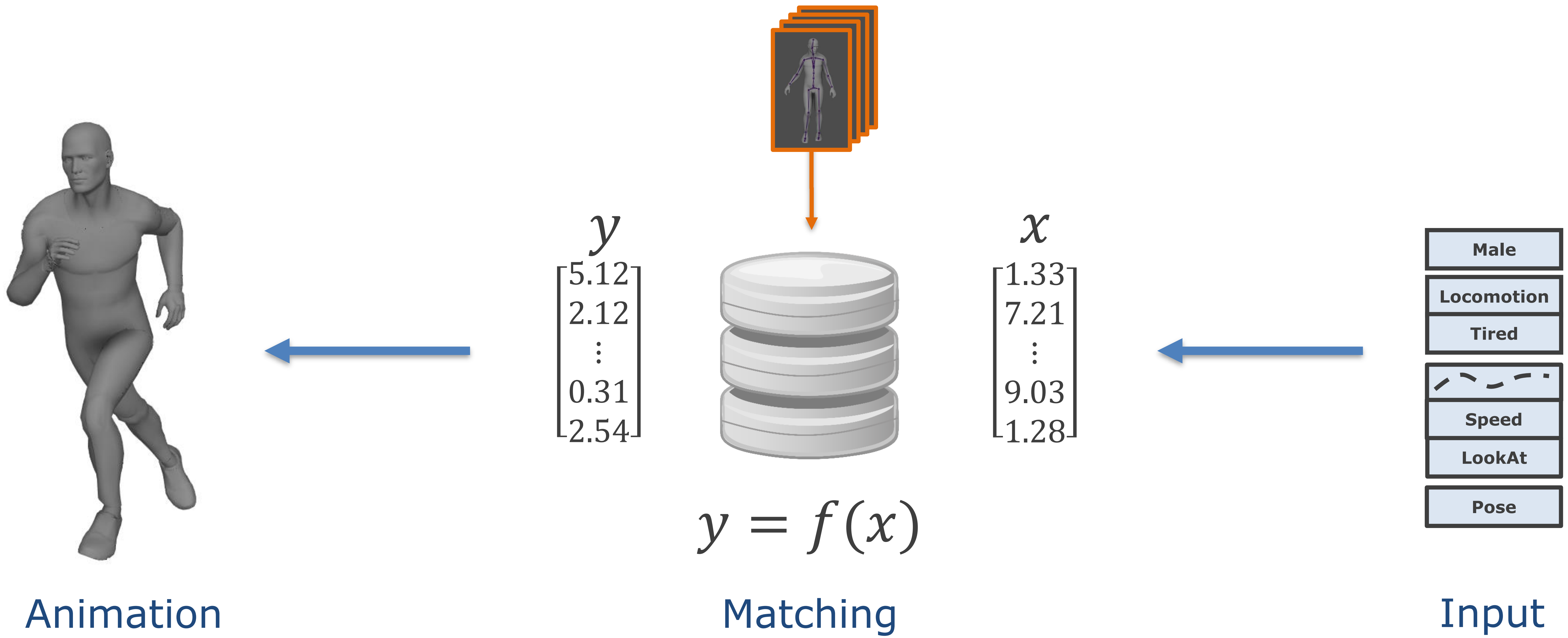
Locomotion

Tired

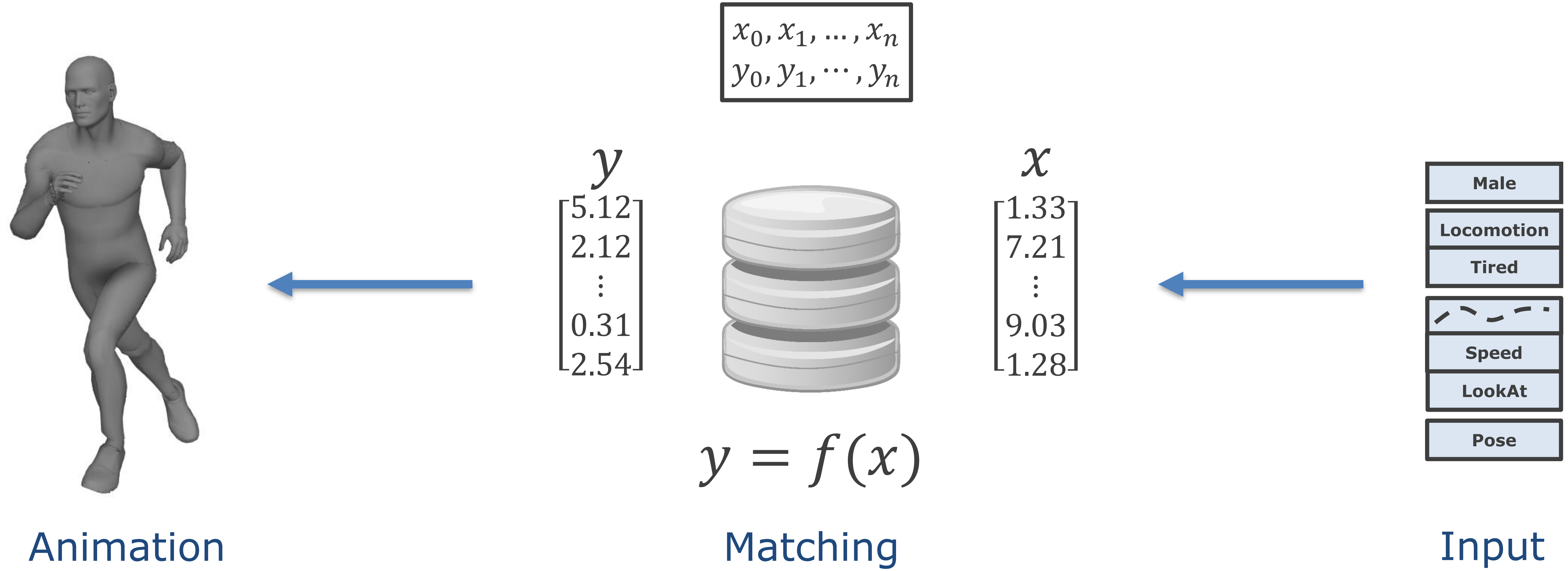
Speed

LookAt

Pose



y
$\begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix}$
x
$\begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$
Male
Locomotion
Tired
Speed
LookAt
Pose
$y = f(x)$
Animation
Matching
Input


$x_0, x_1, \dots, x_n$
$y_0, y_1, \cdots, y_n$
y
$\begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix}$
x
$\begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$
$y = f(x)$
Male
Locomotion
Tired
Speed
LookAt
Pose
Animation
Matching
Input

$$\boxed{\begin{matrix} x_0, x_1, \dots, x_n \\ y_0, y_1, \cdots, y_n \end{matrix}}$$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \qquad y = f(x) \qquad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

Matching

Input

$$\boxed{\begin{array}{c} x_0, x_1, \dots, x_n \\ y_0, y_1, \cdots, y_n \end{array}}$$

$$\downarrow$$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \qquad y = f(x) \qquad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

Supervised Learning

$$x_0, x_1, \dots, x_n$$
$$y_0, y_1, \cdots, y_n$$

$$y \quad \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \qquad y = f(x) \qquad x \quad \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$$

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

Matching

Input

$$x_0, x_1, \ldots, x_n$$
$$y_0, y_1, \cdots, y_n$$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \qquad y = f(x) \qquad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

Male
Locomotion
Tired
Speed
LookAt
Pose

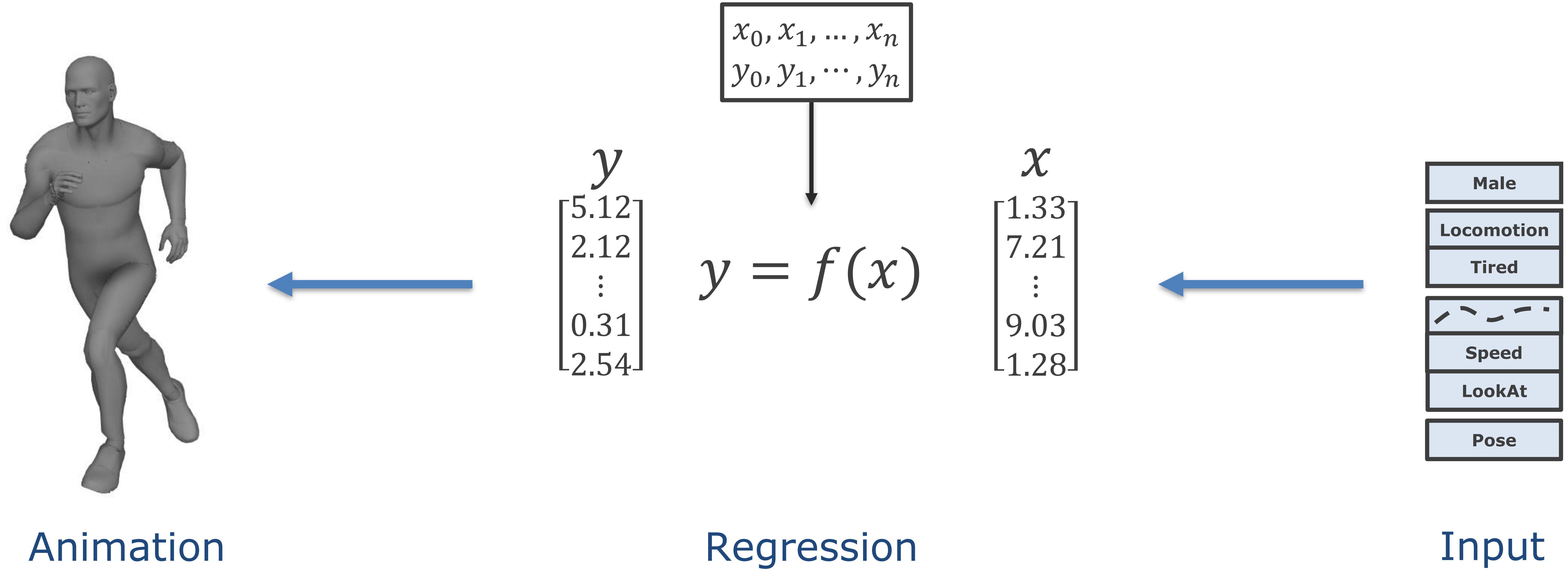
Animation

Regression

Input

Motion Matching is a special case where

$$f = NearestNeighbourRegression$$



$x_0, x_1, \ldots, x_n$
$y_0, y_1, \cdots, y_n$
y
$\begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix}$
$y = f(x)$
x
$\begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$
Male
Locomotion
Tired
Speed
LookAt
Pose
Animation
Regression
Input

$$\boxed{\begin{array}{l} x_0, x_1, \ldots, x_n \\ y_0, y_1, \cdots, y_n \end{array}} \qquad \|x_i - x\|^2$$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \quad y = f(x) \quad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

- Male
- Locomotion
- Tired
- Speed
- LookAt
- Pose

Animation

Nearest Neighbour Regression

Input

$3.1, 0.5, \ldots, 2.3$

$$\boxed{\begin{array}{l} x_0, x_1, \ldots, x_n \\ y_0, y_1, \cdots, y_n \end{array}}$$

$\|x_i - x\|^2$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \quad y = f(x) \quad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

Nearest Neighbour Regression

Input

$3.1, 0.5, \ldots, 2.3$

$$\begin{array}{c} x_0, x_1, \ldots, x_n \\ y_0, y_1, \cdots, y_n \end{array}$$

$\|x_i - x\|^2$

$$y = \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \qquad y = f(x) \qquad x = \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$$

Male

Locomotion

Tired

Speed

LookAt

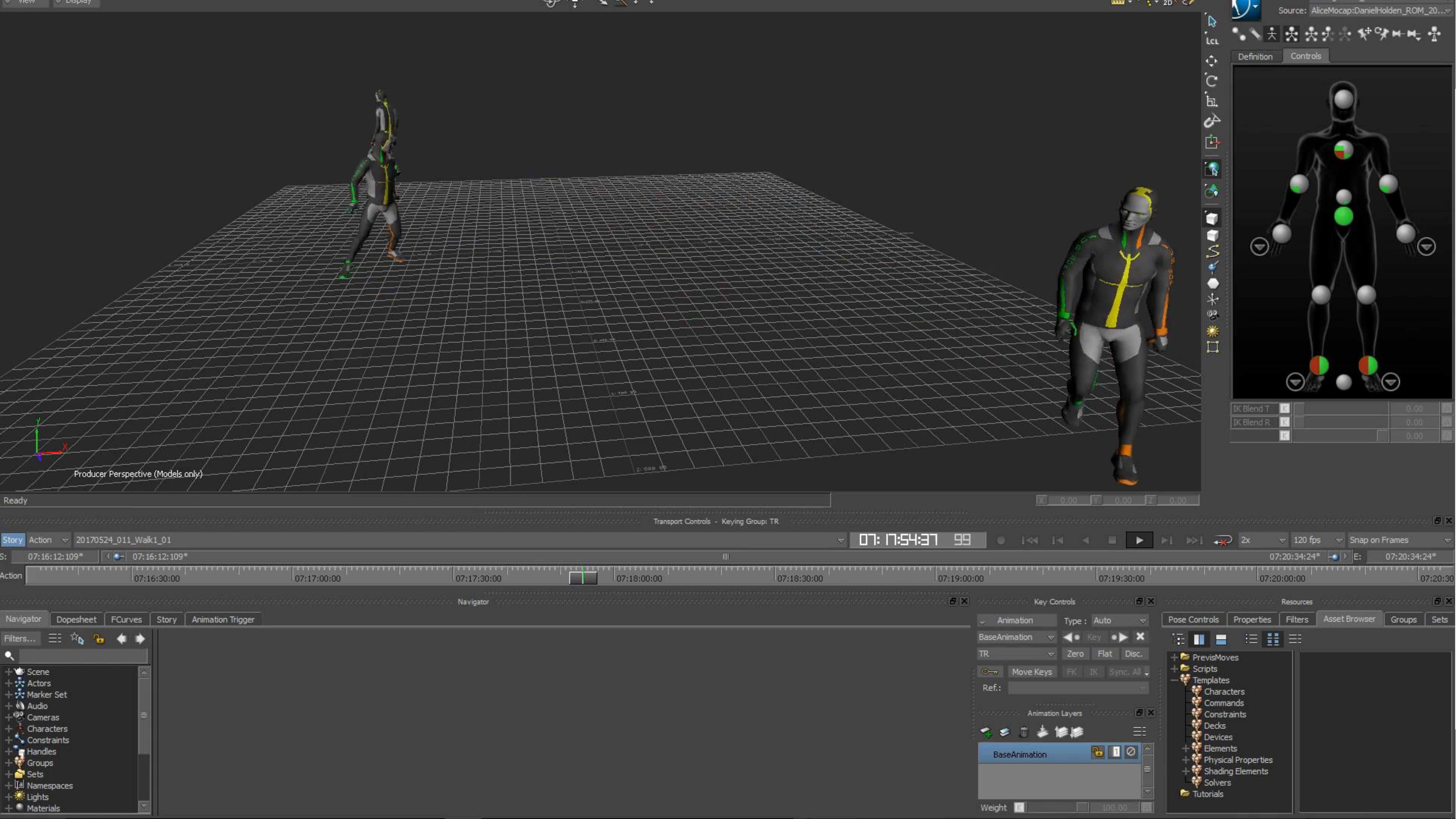
Pose

Animation

Nearest Neighbour Regression

Input

A Simple Example


Source: AliceMocap:DanielHolden_ROM_20...
Definition
Controls
Producer Perspective (Models only)
Ready
IK Blend T
IK Blend R
Transport Controls - Keying Group: TR
Story
Action
20170524_011_Walk1_01
07:17:54:37 99
2x
120 fps
Snap on Frames
07:16:12:109*
07:20:34:24*
07:16:30:00
07:17:00:00
07:17:30:00
07:18:00:00
07:18:30:00
07:19:00:00
07:19:30:00
07:20:00:00
Navigator
Dopesheet
FCurves
Animation Trigger
Filters...
Scene
Actors
Marker Set
Audio
Cameras
Characters
Constraints
Handles
Groups
Sets
Namespaces
Lights
Materials
Key Controls
Animation
Type : Auto
BaseAnimation
Key
TR
Zero
Flat
Disc.
Move Keys
FK
IK
Sync. All
Ref.:
Animation Layers
Weight
Resources
Pose Controls
Properties
Filters
Asset Browser
PrevisMoves
Scripts
Templates
Commands
Decks
Devices
Elements
Physical Properties
Shading Elements
Solvers
Tutorials

Input x

- **Joint Positions** in the previous frame.
- **Joint Velocities** in the previous frame.
- **Target Position** of the root in 1 second.
- **Target Velocity** of the root in 1 second.
- **Target Direction** of the root in 1 second.

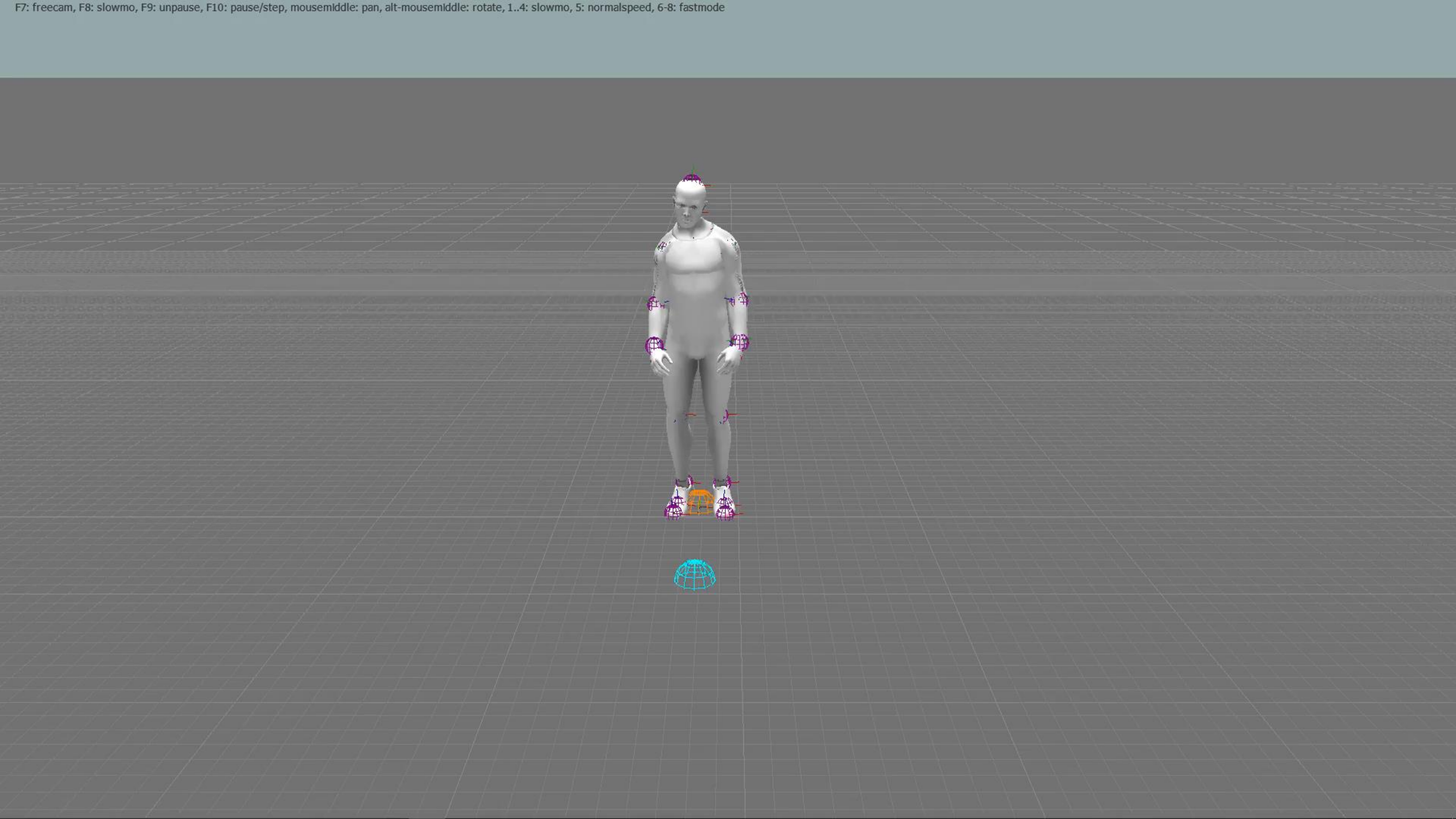
Output y

- **Joint Positions** for the next 1 second.
- **Joint Rotations** for the next 1 second.

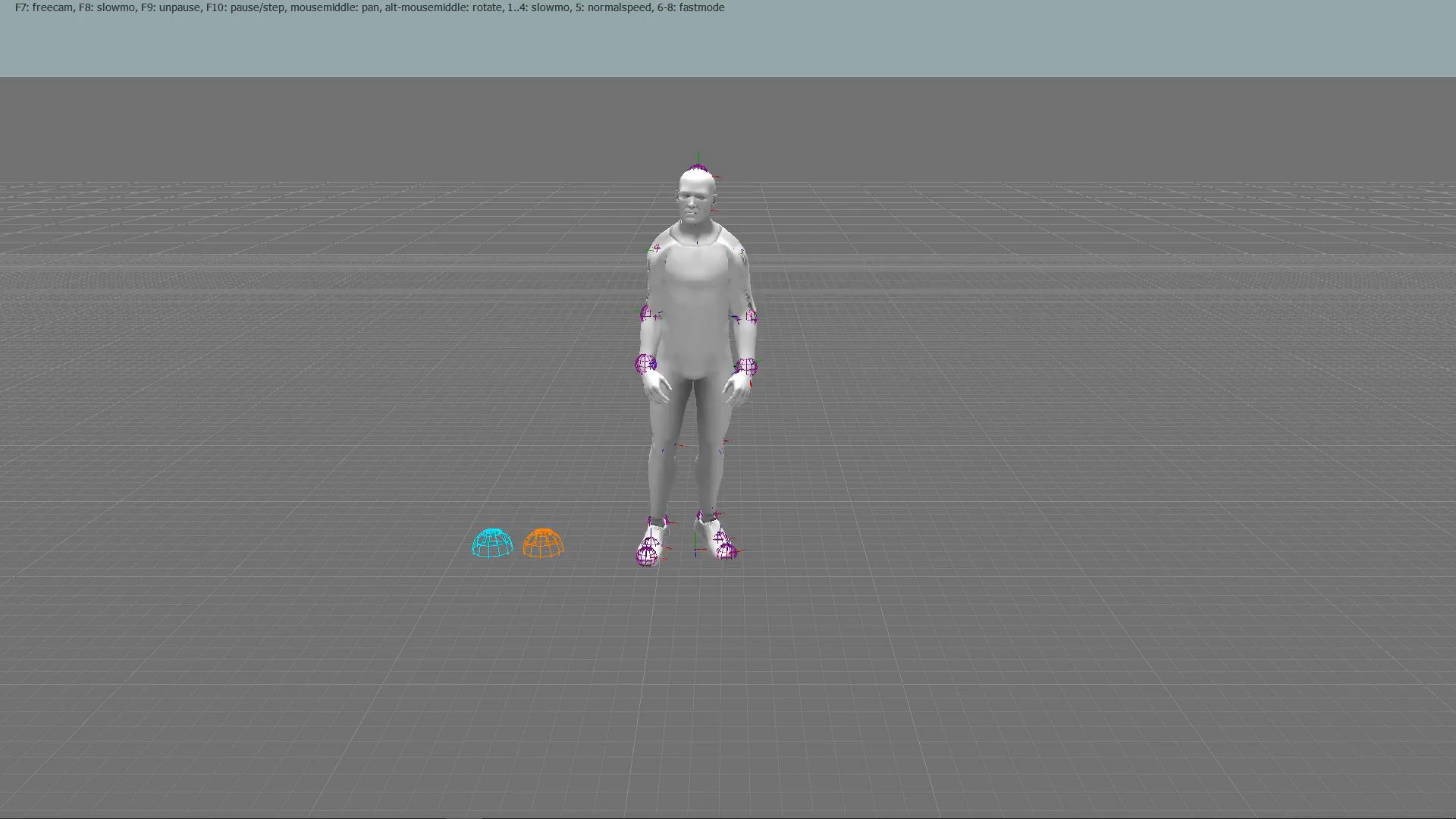
Function f

- **Call** every 1 second or…
- **Call** if the user input changes.

Nearest Neighbour Regression


F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode

Add Blending…


F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode

Memory	Runtime
~200 mb	~1 ms

$$x_0, x_1, \ldots, x_n$$
$$y_0, y_1, \cdots, y_n$$

$$y = \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \qquad y = f(x) \qquad x = \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$$

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation

Regression

Input

$x_0, x_1, \ldots, x_n$
$y_0, y_1, \cdots, y_n$

$$y = \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \qquad y = f(x) \qquad x = \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$$

Male
Locomotion
Tired
Speed
LookAt
Pose

Animation

Regression

Input

1. Supervised learning

1. Supervised learning

1.1. Generalized Linear Models
- 1.1.1. Ordinary Least Squares
  - 1.1.1.1. Ordinary Least Squares Complexity
- 1.1.2. Ridge Regression
  - 1.1.2.1. Ridge Complexity
  - 1.1.2.2. Setting the regularization parameter: generalized Cross-Validation
- 1.1.3. Lasso
  - 1.1.3.1. Setting regularization parameter
    - 1.1.3.1.1. Using cross-validation
    - 1.1.3.1.2. Information-criteria based model selection
    - 1.1.3.1.3. Comparison with the regularization parameter of SVM
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic Net
- 1.1.6. Multi-task Elastic Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
  - 1.1.8.1. Mathematical formulation
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
  - 1.1.10.1. Bayesian Ridge Regression
  - 1.1.10.2. Automatic Relevance Determination - ARD
- 1.1.11. Logistic regression
- 1.1.12. Stochastic Gradient Descent - SGD
- 1.1.13. Perceptron
- 1.1.14. Passive Aggressive Algorithms
- 1.1.15. Robustness regression: outliers and modeling errors
  - 1.1.15.1. Different scenario and useful concepts
  - 1.1.15.2. RANSAC: RANdom SAmple Consensus
    - 1.1.15.2.1. Details of the algorithm
  - 1.1.15.3. Theil-Sen estimator: generalized-median-based estimator
    - 1.1.15.3.1. Theoretical considerations
  - 1.1.15.4. Huber Regression
  - 1.1.15.5. Notes
- 1.1.16. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis
- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regression

1.4. Support Vector Machines
- 1.4.1. Classification
  - 1.4.1.1. Multi-class classification
  - 1.4.1.2. Scores and probabilities
  - 1.4.1.3. Unbalanced problems
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
  - 1.4.6.1. Custom Kernels
    - 1.4.6.1.1. Using Python functions as kernels
    - 1.4.6.1.2. Using the Gram matrix
    - 1.4.6.1.3. Parameters of the RBF Kernel
- 1.4.7. Mathematical formulation
  - 1.4.7.1. SVC
  - 1.4.7.2. NuSVC
  - 1.4.7.3. SVR
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent
- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse data
- 1.5.4. Complexity
- 1.5.5. Tips on Practical Use
- 1.5.6. Mathematical formulation
  - 1.5.6.1. SGD
- 1.5.7. Implementation details

1.6. Nearest Neighbors
- 1.6.1. Unsupervised Nearest Neighbors
  - 1.6.1.1. Finding the Nearest Neighbors
  - 1.6.1.2. KDTree and BallTree Classes
- 1.6.2. Nearest Neighbors Classification
- 1.6.3. Nearest Neighbors Regression
- 1.6.4. Nearest Neighbor Algorithms
  - 1.6.4.1. Brute Force
  - 1.6.4.2. K-D Tree
  - 1.6.4.3. Ball Tree
  - 1.6.4.4. Choice of Nearest Neighbors Algorithm
  - 1.6.4.5. Effect of `leaf_size`
- 1.6.5. Nearest Centroid Classifier
  - 1.6.5.1. Nearest Shrunken Centroid

1.7. Gaussian Processes
- 1.7.1. Gaussian Process Regression (GPR)
- 1.7.2. GPR examples
  - 1.7.2.1. GPR with noise-level estimation
  - 1.7.2.2. Comparison of GPR and Kernel Ridge Regression
  - 1.7.2.3. GPR on Mauna Loa CO2 data
- 1.7.3. Gaussian Process Classification (GPC)
- 1.7.4. GPC examples
  - 1.7.4.1. Probabilistic predictions with GPC
  - 1.7.4.2. Illustration of GPC on the XOR dataset
  - 1.7.4.3. Gaussian process classification (GPC) on iris dataset
- 1.7.5. Kernels for Gaussian Processes
  - 1.7.5.1. Gaussian Process Kernel API
  - 1.7.5.2. Basic kernels
  - 1.7.5.3. Kernel operators
  - 1.7.5.4. Radial-basis function (RBF) kernel
  - 1.7.5.5. Matérn kernel
  - 1.7.5.6. Rational quadratic kernel
  - 1.7.5.7. Exp-Sine-Squared kernel
  - 1.7.5.8. Dot-Product kernel
  - 1.7.5.9. References

1.8. Cross decomposition

1.9. Naive Bayes
- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Bernoulli Naive Bayes
- 1.9.4. Out-of-core naive Bayes model fitting

1.10. Decision Trees
- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
  - 1.10.7.1. Classification criteria
  - 1.10.7.2. Regression criteria

1.11. Ensemble methods
- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
  - 1.11.2.1. Random Forests
  - 1.11.2.2. Extremely Randomized Trees
  - 1.11.2.3. Parameters
  - 1.11.2.4. Parallelization
  - 1.11.2.5. Feature importance evaluation
  - 1.11.2.6. Totally Random Trees Embedding
- 1.11.3. AdaBoost
  - 1.11.3.1. Usage
- 1.11.4. Gradient Tree Boosting
  - 1.11.4.1. Classification
  - 1.11.4.2. Regression
  - 1.11.4.3. Fitting additional weak-learners
  - 1.11.4.4. Controlling the tree size
  - 1.11.4.5. Mathematical formulation
    - 1.11.4.5.1. Loss Functions
  - 1.11.4.6. Regularization
    - 1.11.4.6.1. Shrinkage
    - 1.11.4.6.2. Subsampling
  - 1.11.4.7. Interpretation
    - 1.11.4.7.1. Feature importance
    - 1.11.4.7.2. Partial dependence
- 1.11.5. Voting Classifier
  - 1.11.5.1. Majority Class Labels (Majority/Hard Voting)
    - 1.11.5.1.1. Usage
  - 1.11.5.2. Weighted Average Probabilities (Soft Voting)
  - 1.11.5.3. Using the VotingClassifier with GridSearch
    - 1.11.5.3.1. Usage

1.12. Multiclass and multilabel algorithms
- 1.12.1. Multilabel classification format
- 1.12.2. One-Vs-The-Rest
  - 1.12.2.1. Multiclass learning
  - 1.12.2.2. Multilabel learning
- 1.12.3. One-Vs-One
  - 1.12.3.1. Multiclass learning
- 1.12.4. Error-Correcting Output-Codes
  - 1.12.4.1. Multiclass learning
- 1.12.5. Multioutput regression
- 1.12.6. Multioutput classification
- 1.12.7. Classifier Chain

1.13. Feature selection
- 1.13.1. Removing features with low variance
- 1.13.2. Univariate feature selection
- 1.13.3. Recursive feature elimination
- 1.13.4. Feature selection using SelectFromModel
  - 1.13.4.1. L1-based feature selection
  - 1.13.4.2. Tree-based feature selection
- 1.13.5. Feature selection as part of a pipeline

1.14. Semi-Supervised
- 1.14.1. Label Propagation

1.15. Isotonic regression

1.16. Probability calibration

1.17. Neural network models (supervised)
- 1.17.1. Multi-layer Perceptron
- 1.17.2. Classification
- 1.17.3. Regression
- 1.17.4. Regularization
- 1.17.5. Algorithms
- 1.17.6. Complexity
- 1.17.7. Mathematical formulation
- 1.17.8. Tips on Practical Use
- 1.17.9. More control with warm_start

1. Supervised learning

1.1. Generalized Linear Models

- 1.1.1. Ordinary Least Squares
  - 1.1.1.1. Ordinary Least Squares Complexity
- 1.1.2. Ridge Regression
  - 1.1.2.1. Ridge Complexity
  - 1.1.2.2. Setting the regularization parameter: generalized Cross-Validation
- 1.1.3. Lasso
  - 1.1.3.1. Setting regularization parameter
    - 1.1.3.1.1. Using cross-validation
    - 1.1.3.1.2. Information-criteria based model selection
    - 1.1.3.1.3. Comparison with the regularization parameter of SVM
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic Net
- 1.1.6. Multi-task Elastic Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
  - 1.1.8.1. Mathematical formulation
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
  - 1.1.10.1. Bayesian Ridge Regression
  - 1.1.10.2. Automatic Relevance Determination - ARD
- 1.1.11. Logistic regression
- 1.1.12. Stochastic Gradient Descent - SGD
- 1.1.13. Perceptron
- 1.1.14. Passive Aggressive Algorithms
- 1.1.15. Robustness regression: outliers and modeling errors
  - 1.1.15.1. Different scenario and useful concepts
  - 1.1.15.2. RANSAC: RANdom SAmple Consensus
    - 1.1.15.2.1. Details of the algorithm
  - 1.1.15.3. Theil-Sen estimator: generalized-median-based estimator
    - 1.1.15.3.1. Theoretical considerations
  - 1.1.15.4. Huber Regression
  - 1.1.15.5. Notes
- 1.1.16. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis

- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regression

1.4. Support Vector Machines

- 1.4.1. Classification
  - 1.4.1.1. Multi-class classification
  - 1.4.1.2. Scores and probabilities
  - 1.4.1.3. Unbalanced problems
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
  - 1.4.6.1. Custom Kernels
    - 1.4.6.1.1. Using Python functions as kernels
    - 1.4.6.1.2. Using the Gram matrix
    - 1.4.6.1.3. Parameters of the RBF Kernel
- 1.4.7. Mathematical formulation
  - 1.4.7.1. SVC
  - 1.4.7.2. NuSVC
  - 1.4.7.3. SVR
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent

- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse data
- 1.5.4. Complexity
- 1.5.5. Tips on Practical Use
- 1.5.6. Mathematical formulation
  - 1.5.6.1. SGD
- 1.5.7. Implementation details

1.6. Nearest Neighbors

- 1.6.1. Unsupervised Nearest Neighbors
  - 1.6.1.1. Finding the Nearest Neighbors
  - 1.6.1.2. KDTree and BallTree Classes
- 1.6.2. Nearest Neighbors Classification
- 1.6.3. Nearest Neighbors Regression
- 1.6.4. Nearest Neighbor Algorithms
  - 1.6.4.1. Brute Force
  - 1.6.4.2. K-D Tree
  - 1.6.4.3. Ball Tree
  - 1.6.4.4. Choice of Nearest Neighbors Algorithm
  - 1.6.4.5. Effect of `leaf_size`
- 1.6.5. Nearest Centroid Classifier
  - 1.6.5.1. Nearest Shrunken Centroid

1.7. Gaussian Processes

- 1.7.1. Gaussian Process Regression (GPR)
- 1.7.2. GPR examples
  - 1.7.2.1. GPR with noise-level estimation
  - 1.7.2.2. Comparison of GPR and Kernel Ridge Regression
  - 1.7.2.3. GPR on Mauna Loa CO2 data
- 1.7.3. Gaussian Process Classification (GPC)
- 1.7.4. GPC examples
  - 1.7.4.1. Probabilistic predictions with GPC
  - 1.7.4.2. Illustration of GPC on the XOR dataset
  - 1.7.4.3. Gaussian process classification (GPC) on iris dataset
- 1.7.5. Kernels for Gaussian Processes
  - 1.7.5.1. Gaussian Process Kernel API
  - 1.7.5.2. Basic kernels
  - 1.7.5.3. Kernel operators
  - 1.7.5.4. Radial-basis function (RBF) kernel
  - 1.7.5.5. Matérn kernel
  - 1.7.5.6. Rational quadratic kernel
  - 1.7.5.7. Exp-Sine-Squared kernel
  - 1.7.5.8. Dot-Product kernel
  - 1.7.5.9. References

1.8. Cross decomposition

1.9. Naive Bayes

- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Bernoulli Naive Bayes
- 1.9.4. Out-of-core naive Bayes model fitting

1.10. Decision Trees

- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
  - 1.10.7.1. Classification criteria
  - 1.10.7.2. Regression criteria

1.11. Ensemble methods

- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
  - 1.11.2.1. Random Forests
  - 1.11.2.2. Extremely Randomized Trees
  - 1.11.2.3. Parameters
  - 1.11.2.4. Parallelization
  - 1.11.2.5. Feature importance evaluation
  - 1.11.2.6. Totally Random Trees Embedding
- 1.11.3. AdaBoost
  - 1.11.3.1. Usage
- 1.11.4. Gradient Tree Boosting
  - 1.11.4.1. Classification
  - 1.11.4.2. Regression
  - 1.11.4.3. Fitting additional weak-learners
  - 1.11.4.4. Controlling the tree size
  - 1.11.4.5. Mathematical formulation
    - 1.11.4.5.1. Loss Functions
  - 1.11.4.6. Regularization
    - 1.11.4.6.1. Shrinkage
    - 1.11.4.6.2. Subsampling
  - 1.11.4.7. Interpretation
    - 1.11.4.7.1. Feature importance
    - 1.11.4.7.2. Partial dependence
- 1.11.5. Voting Classifier
  - 1.11.5.1. Majority Class Labels (Majority/Hard Voting)
    - 1.11.5.1.1. Usage
  - 1.11.5.2. Weighted Average Probabilities (Soft Voting)
  - 1.11.5.3. Using the VotingClassifier with GridSearch
    - 1.11.5.3.1. Usage

1.12. Multiclass and multilabel algorithms

- 1.12.1. Multilabel classification format
- 1.12.2. One-Vs-The-Rest
  - 1.12.2.1. Multiclass learning
  - 1.12.2.2. Multilabel learning
- 1.12.3. One-Vs-One
  - 1.12.3.1. Multiclass learning
- 1.12.4. Error-Correcting Output-Codes
  - 1.12.4.1. Multiclass learning
- 1.12.5. Multioutput regression
- 1.12.6. Multioutput classification
- 1.12.7. Classifier Chain

1.13. Feature selection

- 1.13.1. Removing features with low variance
- 1.13.2. Univariate feature selection
- 1.13.3. Recursive feature elimination
- 1.13.4. Feature selection using SelectFromModel
  - 1.13.4.1. L1-based feature selection
  - 1.13.4.2. Tree-based feature selection
- 1.13.5. Feature selection as part of a pipeline

1.14. Semi-Supervised

- 1.14.1. Label Propagation

1.15. Isotonic regression

1.16. Probability calibration

1.17. Neural network models (supervised)

- 1.17.1. Multi-layer Perceptron
- 1.17.2. Classification
- 1.17.3. Regression
- 1.17.4. Regularization
- 1.17.5. Algorithms
- 1.17.6. Complexity
- 1.17.7. Mathematical formulation
- 1.17.8. Tips on Practical Use
- 1.17.9. More control with warm_start

1. Supervised learning

1.1. Generalized Linear Models

- 1.1.1. Ordinary Least Squares
  - 1.1.1.1. Ordinary Least Squares Complexity
- 1.1.2. Ridge Regression
  - 1.1.2.1. Ridge Complexity
  - 1.1.2.2. Setting the regularization parameter: generalized Cross-Validation
- 1.1.3. Lasso
  - 1.1.3.1. Setting regularization parameter
    - 1.1.3.1.1. Using cross-validation
    - 1.1.3.1.2. Information-criteria based model selection
    - 1.1.3.1.3. Comparison with the regularization parameter of SVM
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic Net
- 1.1.6. Multi-task Elastic Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
  - 1.1.8.1. Mathematical formulation
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
  - 1.1.10.1. Bayesian Ridge Regression
  - 1.1.10.2. Automatic Relevance Determination - ARD
- 1.1.11. Logistic regression
- 1.1.12. Stochastic Gradient Descent - SGD
- 1.1.13. Perceptron
- 1.1.14. Passive Aggressive Algorithms
- 1.1.15. Robustness regression: outliers and modeling errors
  - 1.1.15.1. Different scenario and useful concepts
  - 1.1.15.2. RANSAC: RANdom SAmple Consensus
    - 1.1.15.2.1. Details of the algorithm
  - 1.1.15.3. Theil-Sen estimator: generalized-median-based estimator
    - 1.1.15.3.1. Theoretical considerations
  - 1.1.15.4. Huber Regression
  - 1.1.15.5. Notes
- 1.1.16. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis

- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regression

1.4. Support Vector Machines

- 1.4.1. Classification
  - 1.4.1.1. Multi-class classification
  - 1.4.1.2. Scores and probabilities
  - 1.4.1.3. Unbalanced problems
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
  - 1.4.6.1. Custom Kernels
    - 1.4.6.1.1. Using Python functions as kernels
    - 1.4.6.1.2. Using the Gram matrix
    - 1.4.6.1.3. Parameters of the RBF Kernel
- 1.4.7. Mathematical formulation
  - 1.4.7.1. SVC
  - 1.4.7.2. NuSVC
  - 1.4.7.3. SVR
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent

- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse data
- 1.5.4. Complexity
- 1.5.5. Tips on Practical Use
- 1.5.6. Mathematical formulation
  - 1.5.6.1. SGD
- 1.5.7. Implementation details

1.6. Nearest Neighbors

- 1.6.1. Unsupervised Nearest Neighbors
  - 1.6.1.1. Finding the Nearest Neighbors
  - 1.6.1.2. KDTree and BallTree Classes
- 1.6.2. Nearest Neighbors Classification
- 1.6.3. Nearest Neighbors Regression
- 1.6.4. Nearest Neighbor Algorithms
  - 1.6.4.1. Brute Force
  - 1.6.4.2. K-D Tree
  - 1.6.4.3. Ball Tree
  - 1.6.4.4. Choice of Nearest Neighbors Algorithm
  - 1.6.4.5. Effect of `leaf_size`
- 1.6.5. Nearest Centroid Classifier
  - 1.6.5.1. Nearest Shrunken Centroid

1.7. Gaussian Processes

- 1.7.1. Gaussian Process Regression (GPR)
- 1.7.2. GPR examples
  - 1.7.2.1. GPR with noise-level estimation
  - 1.7.2.2. Comparison of GPR and Kernel Ridge Regression
  - 1.7.2.3. GPR on Mauna Loa CO2 data
- 1.7.3. Gaussian Process Classification (GPC)
- 1.7.4. GPC examples
  - 1.7.4.1. Probabilistic predictions with GPC
  - 1.7.4.2. Illustration of GPC on the XOR dataset
  - 1.7.4.3. Gaussian process classification (GPC) on iris dataset
- 1.7.5. Kernels for Gaussian Processes
  - 1.7.5.1. Gaussian Process Kernel API
  - 1.7.5.2. Basic kernels
  - 1.7.5.3. Kernel operators
  - 1.7.5.4. Radial-basis function (RBF) kernel
  - 1.7.5.5. Matérn kernel
  - 1.7.5.6. Rational quadratic kernel
  - 1.7.5.7. Exp-Sine-Squared kernel
  - 1.7.5.8. Dot-Product kernel
  - 1.7.5.9. References

1.8. Cross decomposition

1.9. Naive Bayes

- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Bernoulli Naive Bayes
- 1.9.4. Out-of-core naive Bayes model fitting

1.10. Decision Trees

- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
  - 1.10.7.1. Classification criteria
  - 1.10.7.2. Regression criteria

1.11. Ensemble methods

- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
  - 1.11.2.1. Random Forests
  - 1.11.2.2. Extremely Randomized Trees
  - 1.11.2.3. Parameters
  - 1.11.2.4. Parallelization
  - 1.11.2.5. Feature importance evaluation
  - 1.11.2.6. Totally Random Trees Embedding
- 1.11.3. AdaBoost
  - 1.11.3.1. Usage
- 1.11.4. Gradient Tree Boosting
  - 1.11.4.1. Classification
  - 1.11.4.2. Regression
  - 1.11.4.3. Fitting additional weak-learners
  - 1.11.4.4. Controlling the tree size
  - 1.11.4.5. Mathematical formulation
    - 1.11.4.5.1. Loss Functions
  - 1.11.4.6. Regularization
    - 1.11.4.6.1. Shrinkage
    - 1.11.4.6.2. Subsampling
  - 1.11.4.7. Interpretation
    - 1.11.4.7.1. Feature importance
    - 1.11.4.7.2. Partial dependence
- 1.11.5. Voting Classifier
  - 1.11.5.1. Majority Class Labels (Majority/Hard Voting)
    - 1.11.5.1.1. Usage
  - 1.11.5.2. Weighted Average Probabilities (Soft Voting)
  - 1.11.5.3. Using the VotingClassifier with GridSearch
    - 1.11.5.3.1. Usage

1.12. Multiclass and multilabel algorithms

- 1.12.1. Multilabel classification format
- 1.12.2. One-Vs-The-Rest
  - 1.12.2.1. Multiclass learning
  - 1.12.2.2. Multilabel learning
- 1.12.3. One-Vs-One
  - 1.12.3.1. Multiclass learning
- 1.12.4. Error-Correcting Output-Codes
  - 1.12.4.1. Multiclass learning
- 1.12.5. Multioutput regression
- 1.12.6. Multioutput classification
- 1.12.7. Classifier Chain

1.13. Feature selection

- 1.13.1. Removing features with low variance
- 1.13.2. Univariate feature selection
- 1.13.3. Recursive feature elimination
- 1.13.4. Feature selection using SelectFromModel
  - 1.13.4.1. L1-based feature selection
  - 1.13.4.2. Tree-based feature selection
- 1.13.5. Feature selection as part of a pipeline

1.14. Semi-Supervised

- 1.14.1. Label Propagation

1.15. Isotonic regression

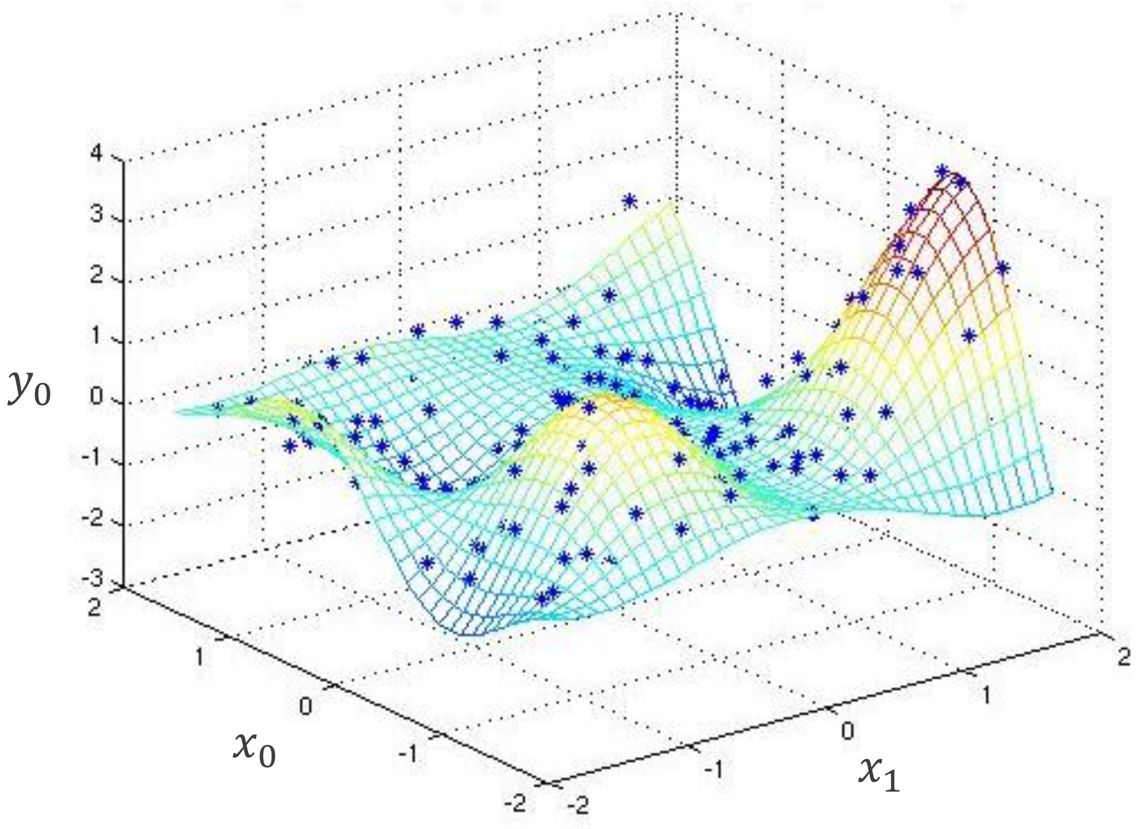
1.16. Probability calibration

1.17. Neural network models (supervised)

- 1.17.1. Multi-layer Perceptron
- 1.17.2. Classification
- 1.17.3. Regression
- 1.17.4. Regularization
- 1.17.5. Algorithms
- 1.17.6. Complexity
- 1.17.7. Mathematical formulation
- 1.17.8. Tips on Practical Use
- 1.17.9. More control with warm_start

Gaussian Processes

- Smooth interpolation of high dimensional data



Animation

$x_0, x_1, \ldots, x_n$
$y_0, y_1, \cdots, y_n$

$$y = \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \qquad y = f(x) \qquad x = \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}$$

Regression

Male

Locomotion

Tired

Speed

LookAt

Pose

Input

$$\boxed{\begin{matrix} x_0, x_1, \ldots, x_n \\ y_0, y_1, \cdots, y_n \end{matrix}} \xrightarrow{Training} GP$$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \quad y = f(x) \quad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

Male

Locomotion

Tired

Speed

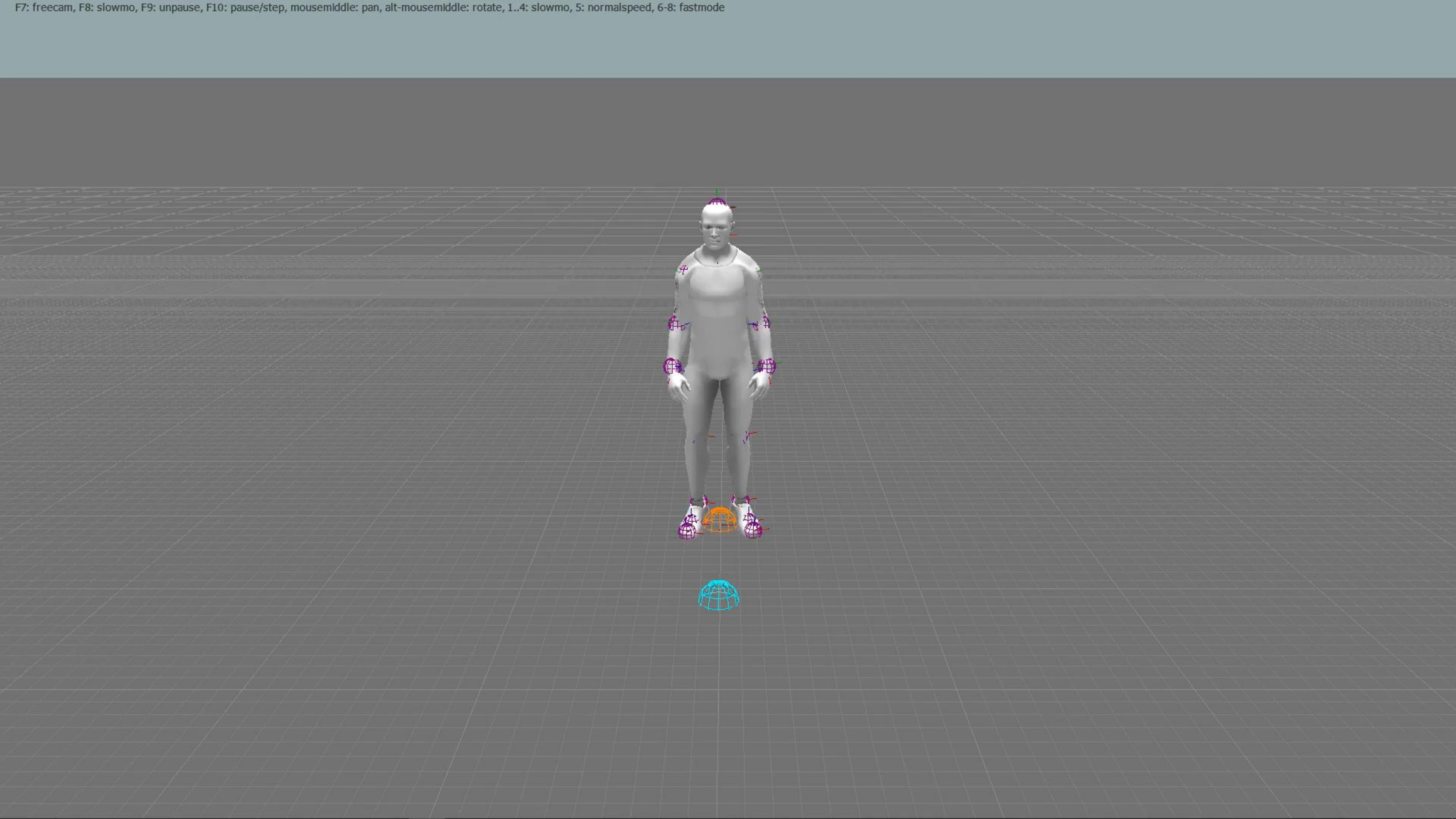
LookAt

Pose

Animation

Gaussian Process

Input


F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode

Gaussian Processes

Gaussian Processes

- Scales poorly with the amount of training data.

Gaussian Processes

- Scales poorly with the amount of training data.
- We could only use ~1000 samples for training.

Gaussian Processes

- Scales poorly with the amount of training data.
- We could only use ~1000 samples for training.
- Maybe we just didn't use enough data…

1. Supervised learning

1.1. Generalized Linear Models

- 1.1.1. Ordinary Least Squares
  - 1.1.1.1. Ordinary Least Squares Complexity
- 1.1.2. Ridge Regression
  - 1.1.2.1. Ridge Complexity
  - 1.1.2.2. Setting the regularization parameter: generalized Cross-Validation
- 1.1.3. Lasso
  - 1.1.3.1. Setting regularization parameter
    - 1.1.3.1.1. Using cross-validation
    - 1.1.3.1.2. Information-criteria based model selection
    - 1.1.3.1.3. Comparison with the regularization parameter of SVM
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic Net
- 1.1.6. Multi-task Elastic Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
  - 1.1.8.1. Mathematical formulation
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
  - 1.1.10.1. Bayesian Ridge Regression
  - 1.1.10.2. Automatic Relevance Determination - ARD
- 1.1.11. Logistic regression
- 1.1.12. Stochastic Gradient Descent - SGD
- 1.1.13. Perceptron
- 1.1.14. Passive Aggressive Algorithms
- 1.1.15. Robustness regression: outliers and modeling errors
  - 1.1.15.1. Different scenario and useful concepts
  - 1.1.15.2. RANSAC: RANdom SAmple Consensus
    - 1.1.15.2.1. Details of the algorithm
  - 1.1.15.3. Theil-Sen estimator: generalized-median-based estimator
    - 1.1.15.3.1. Theoretical considerations
  - 1.1.15.4. Huber Regression
  - 1.1.15.5. Notes
- 1.1.16. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis

- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regression

1.4. Support Vector Machines

- 1.4.1. Classification
  - 1.4.1.1. Multi-class classification
  - 1.4.1.2. Scores and probabilities
  - 1.4.1.3. Unbalanced problems
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
  - 1.4.6.1. Custom Kernels
    - 1.4.6.1.1. Using Python functions as kernels
    - 1.4.6.1.2. Using the Gram matrix
    - 1.4.6.1.3. Parameters of the RBF Kernel
- 1.4.7. Mathematical formulation
  - 1.4.7.1. SVC
  - 1.4.7.2. NuSVC
  - 1.4.7.3. SVR
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent

- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse data
- 1.5.4. Complexity
- 1.5.5. Tips on Practical Use
- 1.5.6. Mathematical formulation
  - 1.5.6.1. SGD
- 1.5.7. Implementation details

1.6. Nearest Neighbors

- 1.6.1. Unsupervised Nearest Neighbors
  - 1.6.1.1. Finding the Nearest Neighbors
  - 1.6.1.2. KDTree and BallTree Classes
- 1.6.2. Nearest Neighbors Classification
- 1.6.3. Nearest Neighbors Regression
- 1.6.4. Nearest Neighbor Algorithms
  - 1.6.4.1. Brute Force
  - 1.6.4.2. K-D Tree
  - 1.6.4.3. Ball Tree
  - 1.6.4.4. Choice of Nearest Neighbors Algorithm
  - 1.6.4.5. Effect of `leaf_size`
- 1.6.5. Nearest Centroid Classifier
  - 1.6.5.1. Nearest Shrunken Centroid

1.7. Gaussian Processes

- 1.7.1. Gaussian Process Regression (GPR)
- 1.7.2. GPR examples
  - 1.7.2.1. GPR with noise-level estimation
  - 1.7.2.2. Comparison of GPR and Kernel Ridge Regression
  - 1.7.2.3. GPR on Mauna Loa CO2 data
- 1.7.3. Gaussian Process Classification (GPC)
- 1.7.4. GPC examples
  - 1.7.4.1. Probabilistic predictions with GPC
  - 1.7.4.2. Illustration of GPC on the XOR dataset
  - 1.7.4.3. Gaussian process classification (GPC) on iris dataset
- 1.7.5. Kernels for Gaussian Processes
  - 1.7.5.1. Gaussian Process Kernel API
  - 1.7.5.2. Basic kernels
  - 1.7.5.3. Kernel operators
  - 1.7.5.4. Radial-basis function (RBF) kernel
  - 1.7.5.5. Matérn kernel
  - 1.7.5.6. Rational quadratic kernel
  - 1.7.5.7. Exp-Sine-Squared kernel
  - 1.7.5.8. Dot-Product kernel
  - 1.7.5.9. References

1.8. Cross decomposition

1.9. Naive Bayes

- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Bernoulli Naive Bayes
- 1.9.4. Out-of-core naive Bayes model fitting

1.10. Decision Trees

- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
  - 1.10.7.1. Classification criteria
  - 1.10.7.2. Regression criteria

1.11. Ensemble methods

- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
  - 1.11.2.1. Random Forests
  - 1.11.2.2. Extremely Randomized Trees
  - 1.11.2.3. Parameters
  - 1.11.2.4. Parallelization
  - 1.11.2.5. Feature importance evaluation
  - 1.11.2.6. Totally Random Trees Embedding
- 1.11.3. AdaBoost
  - 1.11.3.1. Usage
- 1.11.4. Gradient Tree Boosting
  - 1.11.4.1. Classification
  - 1.11.4.2. Regression
  - 1.11.4.3. Fitting additional weak-learners
  - 1.11.4.4. Controlling the tree size
  - 1.11.4.5. Mathematical formulation
    - 1.11.4.5.1. Loss Functions
  - 1.11.4.6. Regularization
    - 1.11.4.6.1. Shrinkage
    - 1.11.4.6.2. Subsampling
  - 1.11.4.7. Interpretation
    - 1.11.4.7.1. Feature importance
    - 1.11.4.7.2. Partial dependence
- 1.11.5. Voting Classifier
  - 1.11.5.1. Majority Class Labels (Majority/Hard Voting)
    - 1.11.5.1.1. Usage
  - 1.11.5.2. Weighted Average Probabilities (Soft Voting)
  - 1.11.5.3. Using the VotingClassifier with GridSearch
    - 1.11.5.3.1. Usage

1.12. Multiclass and multilabel algorithms

- 1.12.1. Multilabel classification format
- 1.12.2. One-Vs-The-Rest
  - 1.12.2.1. Multiclass learning
  - 1.12.2.2. Multilabel learning
- 1.12.3. One-Vs-One
  - 1.12.3.1. Multiclass learning
- 1.12.4. Error-Correcting Output-Codes
  - 1.12.4.1. Multiclass learning
- 1.12.5. Multioutput regression
- 1.12.6. Multioutput classification
- 1.12.7. Classifier Chain

1.13. Feature selection

- 1.13.1. Removing features with low variance
- 1.13.2. Univariate feature selection
- 1.13.3. Recursive feature elimination
- 1.13.4. Feature selection using SelectFromModel
  - 1.13.4.1. L1-based feature selection
  - 1.13.4.2. Tree-based feature selection
- 1.13.5. Feature selection as part of a pipeline

1.14. Semi-Supervised

- 1.14.1. Label Propagation

1.15. Isotonic regression

1.16. Probability calibration

1.17. Neural network models (supervised)

- 1.17.1. Multi-layer Perceptron
- 1.17.2. Classification
- 1.17.3. Regression
- 1.17.4. Regularization
- 1.17.5. Algorithms
- 1.17.6. Complexity
- 1.17.7. Mathematical formulation
- 1.17.8. Tips on Practical Use
- 1.17.9. More control with warm_start

1. Supervised learning

1.1. Generalized Linear Models
- 1.1.1. Ordinary Least Squares
  - 1.1.1.1. Ordinary Least Squares Complexity
- 1.1.2. Ridge Regression
  - 1.1.2.1. Ridge Complexity
  - 1.1.2.2. Setting the regularization parameter: generalized Cross-Validation
- 1.1.3. Lasso
  - 1.1.3.1. Setting regularization parameter
    - 1.1.3.1.1. Using cross-validation
    - 1.1.3.1.2. Information-criteria based model selection
    - 1.1.3.1.3. Comparison with the regularization parameter of SVM
- 1.1.4. Multi-task Lasso
- 1.1.5. Elastic Net
- 1.1.6. Multi-task Elastic Net
- 1.1.7. Least Angle Regression
- 1.1.8. LARS Lasso
  - 1.1.8.1. Mathematical formulation
- 1.1.9. Orthogonal Matching Pursuit (OMP)
- 1.1.10. Bayesian Regression
  - 1.1.10.1. Bayesian Ridge Regression
  - 1.1.10.2. Automatic Relevance Determination - ARD
- 1.1.11. Logistic regression
- 1.1.12. Stochastic Gradient Descent - SGD
- 1.1.13. Perceptron
- 1.1.14. Passive Aggressive Algorithms
- 1.1.15. Robustness regression: outliers and modeling errors
  - 1.1.15.1. Different scenario and useful concepts
  - 1.1.15.2. RANSAC: RANdom SAmple Consensus
    - 1.1.15.2.1. Details of the algorithm
  - 1.1.15.3. Theil-Sen estimator: generalized-median-based estimator
    - 1.1.15.3.1. Theoretical considerations
  - 1.1.15.4. Huber Regression
  - 1.1.15.5. Notes
- 1.1.16. Polynomial regression: extending linear models with basis functions

1.2. Linear and Quadratic Discriminant Analysis
- 1.2.1. Dimensionality reduction using Linear Discriminant Analysis
- 1.2.2. Mathematical formulation of the LDA and QDA classifiers
- 1.2.3. Mathematical formulation of LDA dimensionality reduction
- 1.2.4. Shrinkage
- 1.2.5. Estimation algorithms

1.3. Kernel ridge regression

1.4. Support Vector Machines
- 1.4.1. Classification
  - 1.4.1.1. Multi-class classification
  - 1.4.1.2. Scores and probabilities
  - 1.4.1.3. Unbalanced problems
- 1.4.2. Regression
- 1.4.3. Density estimation, novelty detection
- 1.4.4. Complexity
- 1.4.5. Tips on Practical Use
- 1.4.6. Kernel functions
  - 1.4.6.1. Custom Kernels
    - 1.4.6.1.1. Using Python functions as kernels
    - 1.4.6.1.2. Using the Gram matrix
    - 1.4.6.1.3. Parameters of the RBF Kernel
- 1.4.7. Mathematical formulation
  - 1.4.7.1. SVC
  - 1.4.7.2. NuSVC
  - 1.4.7.3. SVR
- 1.4.8. Implementation details

1.5. Stochastic Gradient Descent
- 1.5.1. Classification
- 1.5.2. Regression
- 1.5.3. Stochastic Gradient Descent for sparse data
- 1.5.4. Complexity
- 1.5.5. Tips on Practical Use
- 1.5.6. Mathematical formulation
  - 1.5.6.1. SGD
- 1.5.7. Implementation details

1.6. Nearest Neighbors
- 1.6.1. Unsupervised Nearest Neighbors
  - 1.6.1.1. Finding the Nearest Neighbors
  - 1.6.1.2. KDTree and BallTree Classes
- 1.6.2. Nearest Neighbors Classification
- 1.6.3. Nearest Neighbors Regression
- 1.6.4. Nearest Neighbor Algorithms
  - 1.6.4.1. Brute Force
  - 1.6.4.2. K-D Tree
  - 1.6.4.3. Ball Tree
  - 1.6.4.4. Choice of Nearest Neighbors Algorithm
  - 1.6.4.5. Effect of `leaf_size`
- 1.6.5. Nearest Centroid Classifier
  - 1.6.5.1. Nearest Shrunken Centroid

1.7. Gaussian Processes
- 1.7.1. Gaussian Process Regression (GPR)
- 1.7.2. GPR examples
  - 1.7.2.1. GPR with noise-level estimation
  - 1.7.2.2. Comparison of GPR and Kernel Ridge Regression
  - 1.7.2.3. GPR on Mauna Loa CO2 data
- 1.7.3. Gaussian Process Classification (GPC)
- 1.7.4. GPC examples
  - 1.7.4.1. Probabilistic predictions with GPC
  - 1.7.4.2. Illustration of GPC on the XOR dataset
  - 1.7.4.3. Gaussian process classification (GPC) on iris dataset
- 1.7.5. Kernels for Gaussian Processes
  - 1.7.5.1. Gaussian Process Kernel API
  - 1.7.5.2. Basic kernels
  - 1.7.5.3. Kernel operators
  - 1.7.5.4. Radial-basis function (RBF) kernel
  - 1.7.5.5. Matérn kernel
  - 1.7.5.6. Rational quadratic kernel
  - 1.7.5.7. Exp-Sine-Squared kernel
  - 1.7.5.8. Dot-Product kernel
  - 1.7.5.9. References

1.8. Cross decomposition

1.9. Naive Bayes
- 1.9.1. Gaussian Naive Bayes
- 1.9.2. Multinomial Naive Bayes
- 1.9.3. Bernoulli Naive Bayes
- 1.9.4. Out-of-core naive Bayes model fitting

1.10. Decision Trees
- 1.10.1. Classification
- 1.10.2. Regression
- 1.10.3. Multi-output problems
- 1.10.4. Complexity
- 1.10.5. Tips on practical use
- 1.10.6. Tree algorithms: ID3, C4.5, C5.0 and CART
- 1.10.7. Mathematical formulation
  - 1.10.7.1. Classification criteria
  - 1.10.7.2. Regression criteria

1.11. Ensemble methods
- 1.11.1. Bagging meta-estimator
- 1.11.2. Forests of randomized trees
  - 1.11.2.1. Random Forests
  - 1.11.2.2. Extremely Randomized Trees
  - 1.11.2.3. Parameters
  - 1.11.2.4. Parallelization
  - 1.11.2.5. Feature importance evaluation
  - 1.11.2.6. Totally Random Trees Embedding
- 1.11.3. AdaBoost
  - 1.11.3.1. Usage
- 1.11.4. Gradient Tree Boosting
  - 1.11.4.1. Classification
  - 1.11.4.2. Regression
  - 1.11.4.3. Fitting additional weak-learners
  - 1.11.4.4. Controlling the tree size
  - 1.11.4.5. Mathematical formulation
    - 1.11.4.5.1. Loss Functions
  - 1.11.4.6. Regularization
    - 1.11.4.6.1. Shrinkage
    - 1.11.4.6.2. Subsampling
  - 1.11.4.7. Interpretation
    - 1.11.4.7.1. Feature importance
    - 1.11.4.7.2. Partial dependence
- 1.11.5. Voting Classifier
  - 1.11.5.1. Majority Class Labels (Majority/Hard Voting)
    - 1.11.5.1.1. Usage
  - 1.11.5.2. Weighted Average Probabilities (Soft Voting)
  - 1.11.5.3. Using the VotingClassifier with GridSearch
    - 1.11.5.3.1. Usage

1.12. Multiclass and multilabel algorithms
- 1.12.1. Multilabel classification format
- 1.12.2. One-Vs-The-Rest
  - 1.12.2.1. Multiclass learning
  - 1.12.2.2. Multilabel learning
- 1.12.3. One-Vs-One
  - 1.12.3.1. Multiclass learning
- 1.12.4. Error-Correcting Output-Codes
  - 1.12.4.1. Multiclass learning
- 1.12.5. Multioutput regression
- 1.12.6. Multioutput classification
- 1.12.7. Classifier Chain

1.13. Feature selection
- 1.13.1. Removing features with low variance
- 1.13.2. Univariate feature selection
- 1.13.3. Recursive feature elimination
- 1.13.4. Feature selection using SelectFromModel
  - 1.13.4.1. L1-based feature selection
  - 1.13.4.2. Tree-based feature selection
- 1.13.5. Feature selection as part of a pipeline

1.14. Semi-Supervised
- 1.14.1. Label Propagation

1.15. Isotonic regression

1.16. Probability calibration

1.17. Neural network models (supervised)
- 1.17.1. Multi-layer Perceptron
- 1.17.2. Classification
- 1.17.3. Regression
- 1.17.4. Regularization
- 1.17.5. Algorithms
- 1.17.6. Complexity
- 1.17.7. Mathematical formulation
- 1.17.8. Tips on Practical Use
- 1.17.9. More control with warm_start

Neural Networks

- Capacity for virtually unlimited training data.

Neural Networks

- Capacity for virtually unlimited training data.

- Data can be discarded once network is trained.

Neural Networks

- Capacity for virtually unlimited training data.
- Data can be discarded once network is trained.
- Fast to evaluate and low memory usage.

Neural Networks

In Five Minutes

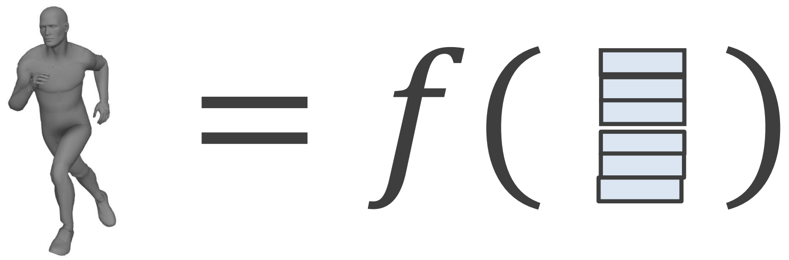
A Neural Network is just a function…

$$\boldsymbol{y} = f(\boldsymbol{x})$$

One example of a simple function…

$$\boldsymbol{y} = \sin(3\,\boldsymbol{x} + 2)$$

It takes some input and produces some output…



The inputs and outputs are represented as vectors

$$\overset{\boldsymbol{y}}{\begin{bmatrix} 2.3 \\ 9.2 \\ \vdots \\ 7.9 \end{bmatrix}} = f\left(\overset{\boldsymbol{x}}{\begin{bmatrix} 6.2 \\ 0.3 \\ \vdots \\ 8.5 \end{bmatrix}}\right)$$

A single “layer” is described by the following function…

$$\boldsymbol{y} = \sigma(\boldsymbol{W}\,\boldsymbol{x} + \boldsymbol{b})$$

The variables $\boldsymbol{W}$ and $\boldsymbol{b}$ are the network “weights”…

$$\boldsymbol{y} = \sigma(\boldsymbol{W}\,\boldsymbol{x} + \boldsymbol{b})$$

The input $\boldsymbol{x}$ and output $\boldsymbol{y}$ appear on either side…

$$\overset{\boldsymbol{y}}{\begin{bmatrix} 2.3 \\ 9.2 \\ \vdots \\ 7.9 \end{bmatrix}} = \sigma\left(\boldsymbol{W} \overset{\boldsymbol{x}}{\begin{bmatrix} 6.2 \\ 0.3 \\ \vdots \\ 8.5 \end{bmatrix}} + \boldsymbol{b}\right)$$

The first operation is for $\boldsymbol{x}$ to be multiplied by $\boldsymbol{W}$…

$$\boldsymbol{y} = \sigma(\underbrace{\begin{bmatrix} 1.5 & \cdots & 0.5 \\ \vdots & \ddots & \vdots \\ 0.2 & \cdots & 7.2 \end{bmatrix}}^{\boldsymbol{W}} \boldsymbol{x} + \boldsymbol{b})$$

The result is added to the vector $\boldsymbol{b}$, called the “bias”…

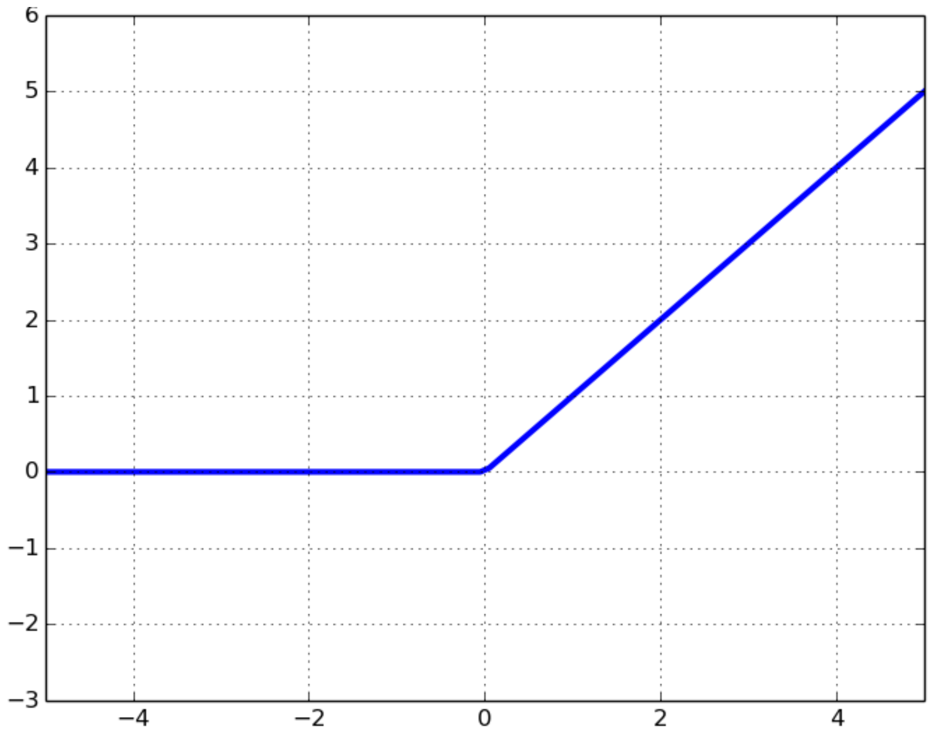
$$\begin{array}{cccc} & & \boldsymbol{W} & & \boldsymbol{b} \\ \boldsymbol{y} = \sigma\Bigg(& & \begin{bmatrix} 1.5 & \cdots & 0.5 \\ \vdots & \ddots & \vdots \\ 0.2 & \cdots & 7.2 \end{bmatrix} \boldsymbol{x} + & \begin{bmatrix} 4.5 \\ 1.3 \\ \vdots \\ 3.7 \end{bmatrix} \Bigg) \end{array}$$

Each value is then passed through σ, the “activation function”.

$$\boldsymbol{y} = \sigma\Big(\overset{\boldsymbol{W}}{\begin{bmatrix} 1.5 & \cdots & 0.5 \\ \vdots & \ddots & \vdots \\ 0.2 & \cdots & 7.2 \end{bmatrix}}\boldsymbol{x} + \overset{\boldsymbol{b}}{\begin{bmatrix} 4.5 \\ 1.3 \\ \vdots \\ 3.7 \end{bmatrix}}\Big)$$

This function produces a “bend” or “non-linearity” in the output.

$$\sigma(\boldsymbol{h}) = \max(\boldsymbol{h}, 0)$$



Looks Familiar…

$$\boldsymbol{y} = \sigma(\boldsymbol{W}\,\boldsymbol{x} + \boldsymbol{b})$$

$$\boldsymbol{y} = \sin(3\,\boldsymbol{x} + 2)$$

We can "stack" multiple layers by nesting the function inside itself…

$$\boldsymbol{y} = \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\sigma(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}_0) + \boldsymbol{b}_1) + \boldsymbol{b}_2$$

This produces the final equation for our Neural Network…

$$\underbrace{\boldsymbol{y} = \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\sigma(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}_0) + \boldsymbol{b}_1) + \boldsymbol{b}_2}$$

$$\begin{array}{ccc} y & & x \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} & y = f(x) & \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

We put the training data through the network and measure the error…

$$
\begin{array}{ccc}
y_0 & y_1 & y_2 \\
\begin{bmatrix} 9.01 \\ 0.98 \\ \vdots \\ 5.23 \\ 0.95 \end{bmatrix} &
\begin{bmatrix} 8.91 \\ 4.23 \\ \vdots \\ 7.11 \\ 3.61 \end{bmatrix} &
\begin{bmatrix} 6.63 \\ 6.23 \\ \vdots \\ 8.12 \\ 1.62 \end{bmatrix} \\
\updownarrow & \updownarrow & \updownarrow \\
\begin{bmatrix} 5.72 \\ 3.12 \\ \vdots \\ 6.92 \\ 2.85 \end{bmatrix} &
\begin{bmatrix} 4.62 \\ 1.93 \\ \vdots \\ 4.52 \\ 7.11 \end{bmatrix} &
\begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix}
\end{array}
\quad y = f(x) \quad
\begin{array}{ccc}
x_0 & x_1 & x_2 \\
\begin{bmatrix} 0.11 \\ 5.55 \\ \vdots \\ 2.43 \\ 1.15 \end{bmatrix} &
\begin{bmatrix} 1.66 \\ 3.11 \\ \vdots \\ 2.56 \\ 7.34 \end{bmatrix} &
\begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}
\end{array}
$$

Then use the calculated error to update the weights…

$$\boldsymbol{y} = \boldsymbol{W}_2\sigma(\boldsymbol{W}_1\sigma(\boldsymbol{W}_0\boldsymbol{x} + \boldsymbol{b}_0) + \boldsymbol{b}_1) + \boldsymbol{b}_2$$

$$\begin{bmatrix} 5.72 \\ 3.12 \\ \vdots \\ 6.92 \\ 2.85 \end{bmatrix} \begin{bmatrix} 4.62 \\ 1.93 \\ \vdots \\ 4.52 \\ 7.11 \end{bmatrix} \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix}$$

And repeat thousands of times on the GPU…

$$\begin{array}{ccc} y_0 & y_1 & y_2 \\ \begin{bmatrix} 9.01 \\ 0.98 \\ \vdots \\ 5.23 \\ 0.95 \end{bmatrix} & \begin{bmatrix} 8.91 \\ 4.23 \\ \vdots \\ 7.11 \\ 3.61 \end{bmatrix} & \begin{bmatrix} 6.63 \\ 6.23 \\ \vdots \\ 8.12 \\ 1.62 \end{bmatrix} \\ \updownarrow & \updownarrow & \updownarrow \\ \begin{bmatrix} 5.72 \\ 3.12 \\ \vdots \\ 6.92 \\ 2.85 \end{bmatrix} & \begin{bmatrix} 4.62 \\ 1.93 \\ \vdots \\ 4.52 \\ 7.11 \end{bmatrix} & \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \quad y = f(x) \quad \begin{array}{ccc} x_0 & x_1 & x_2 \\ \begin{bmatrix} 0.11 \\ 5.55 \\ \vdots \\ 2.43 \\ 1.15 \end{bmatrix} & \begin{bmatrix} 1.66 \\ 3.11 \\ \vdots \\ 2.56 \\ 7.34 \end{bmatrix} & \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

Animation

$$x_0, x_1, \dots, x_n$$
$$y_0, y_1, \cdots, y_n$$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \qquad y = f(x) \qquad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

Regression

Male

Locomotion

Tired

Speed

LookAt

Pose

Input

$$\boxed{\begin{matrix} x_0, x_1, \dots, x_n \\ y_0, y_1, \cdots, y_n \end{matrix}} \xrightarrow{Training} NN$$

$$NN \downarrow$$

$$\underset{y}{\begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix}} \quad y = f(x) \quad \underset{x}{\begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix}}$$

Male

Locomotion

Tired

Speed

LookAt

Pose

Animation ← Neural Network ← Input

$$\cancel{\begin{array}{l} x_0, x_1, \ldots, x_n \\ y_0, y_1, \ldots, y_n \end{array}}$$

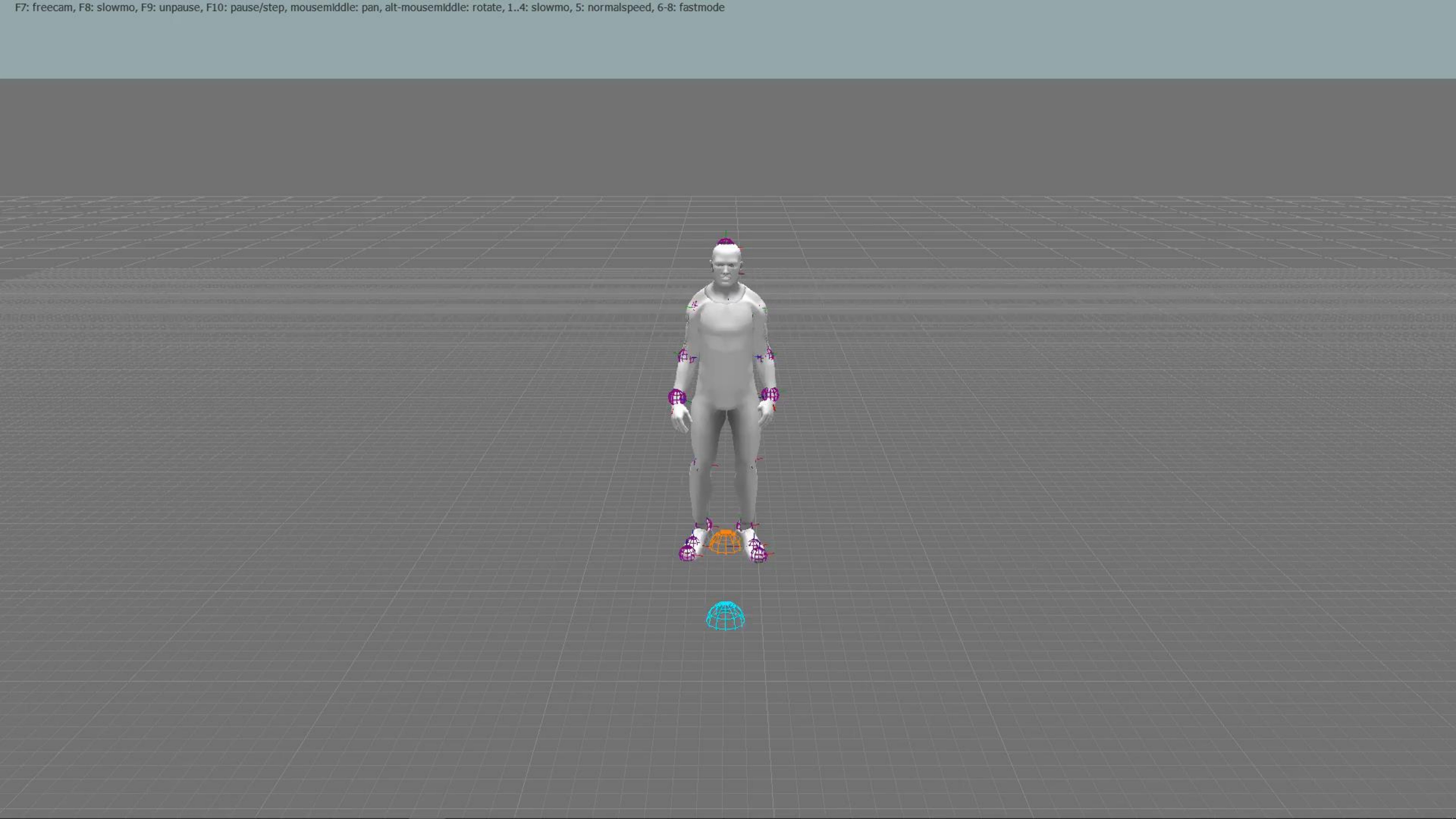
$$NN$$

$$\downarrow$$

$$\begin{array}{c} y \\ \begin{bmatrix} 5.12 \\ 2.12 \\ \vdots \\ 0.31 \\ 2.54 \end{bmatrix} \end{array} \quad y = f(x) \quad \begin{array}{c} x \\ \begin{bmatrix} 1.33 \\ 7.21 \\ \vdots \\ 9.03 \\ 1.28 \end{bmatrix} \end{array}$$

- Male
- Locomotion
- Tired
- Speed
- LookAt
- Pose

Animation ← Trained Neural Network ← Input


F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode

Machine Learning isn’t Magic

The results depend on…

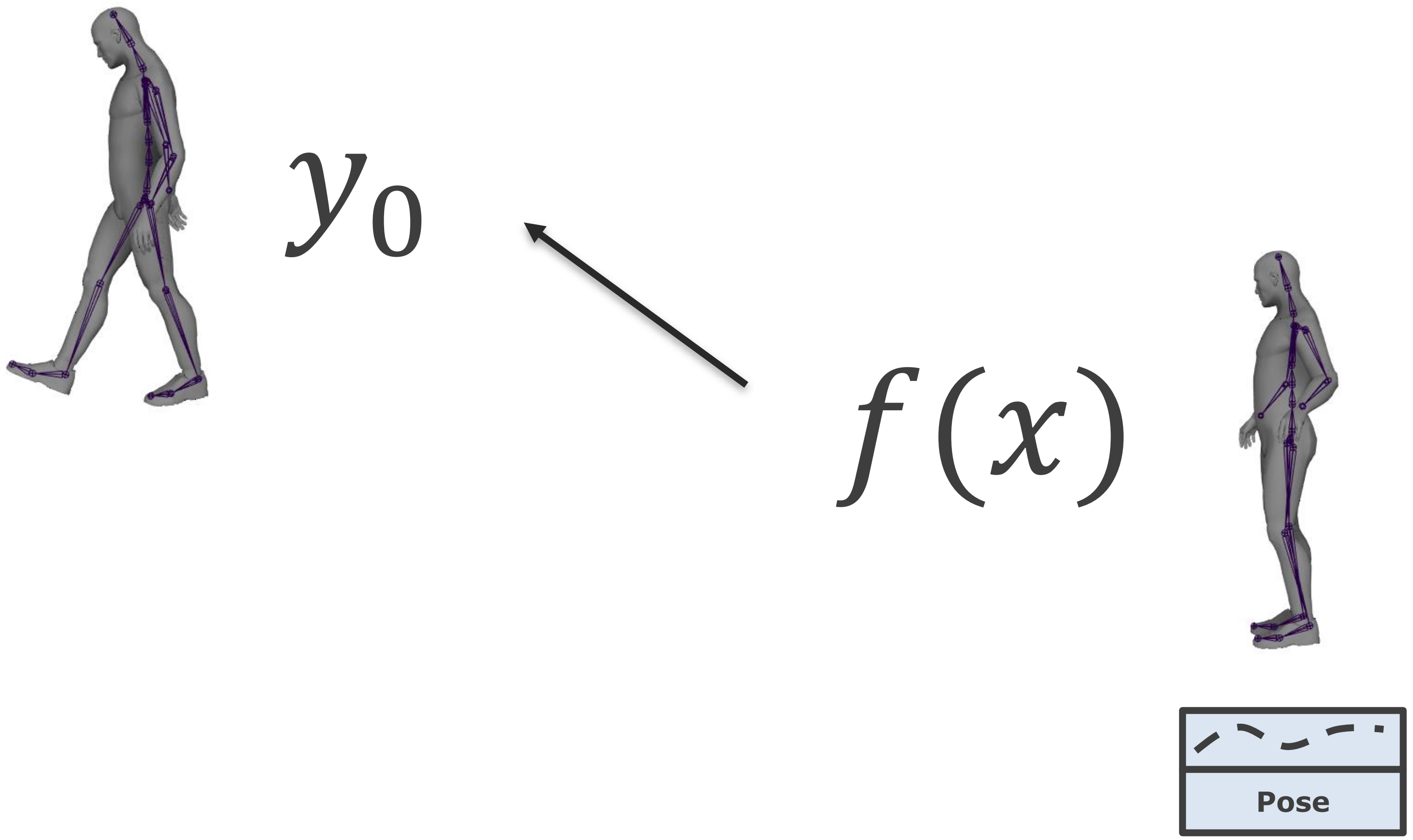
- The input representation x
- The output representation y
- How and when you use f

The function f isn’t well defined

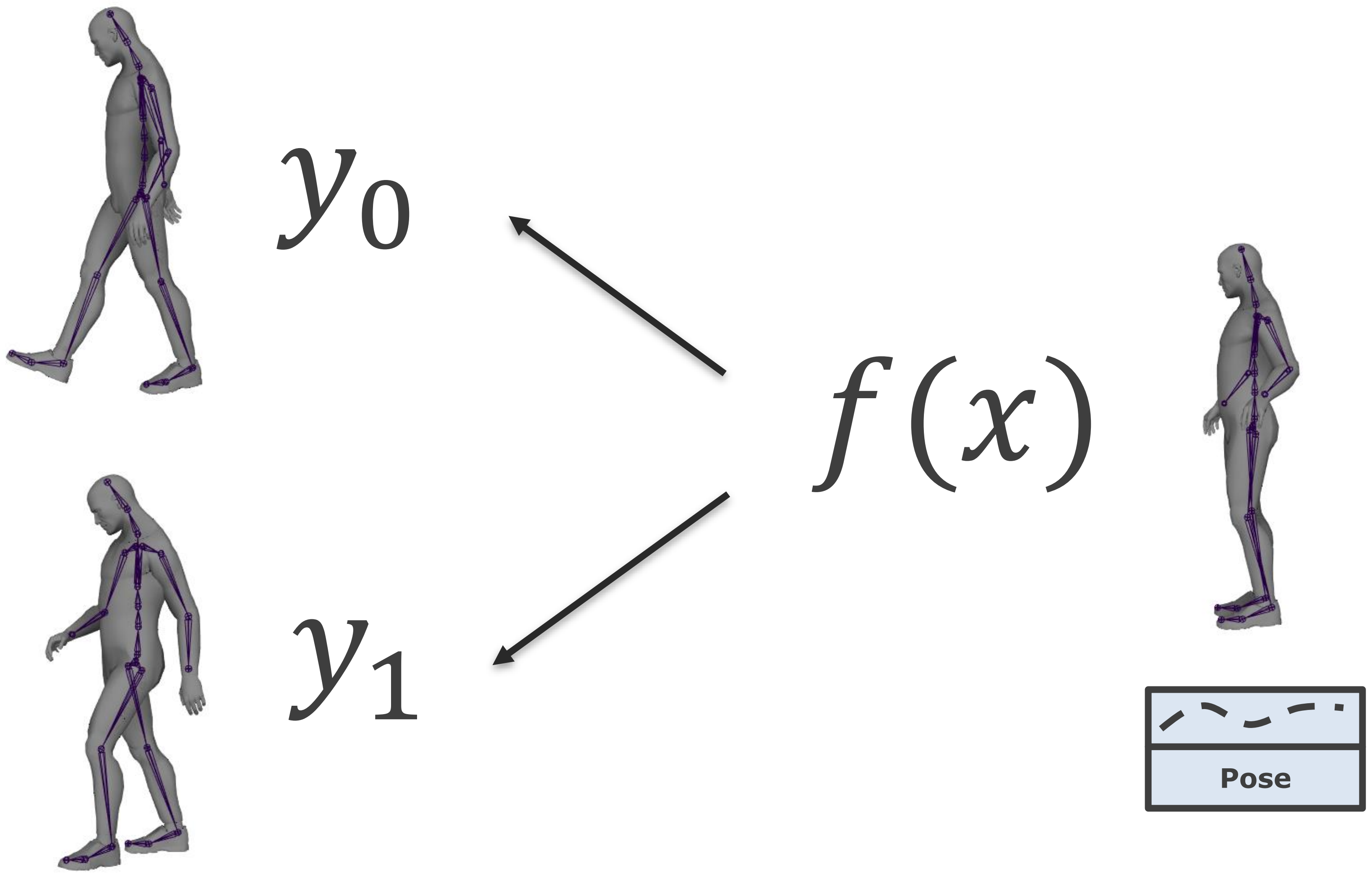
There are multiple y values for a single x.

$$f(x)$$

There are multiple y values for a single x.



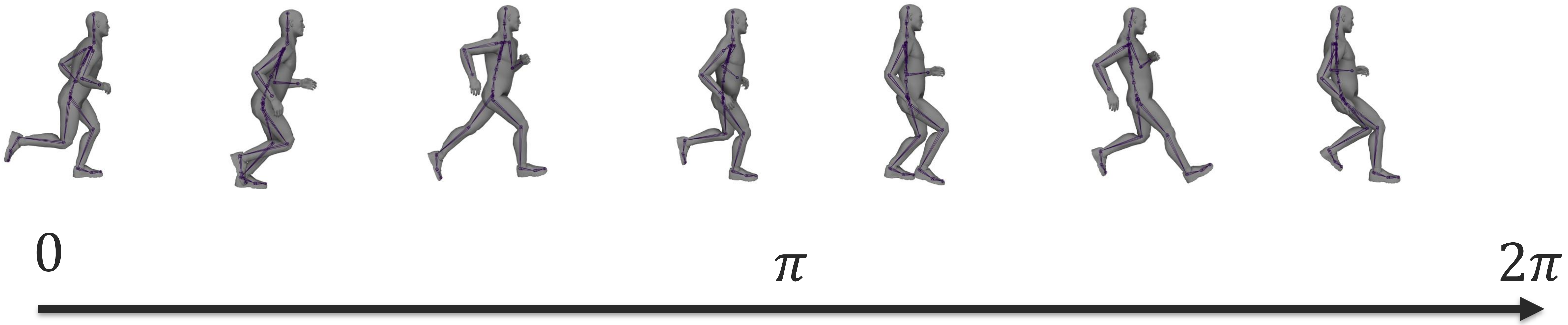
There are multiple y values for a single x.



Can we resolve the ambiguity?

The Phase

A variable representing the timing of the pose in the cycle.

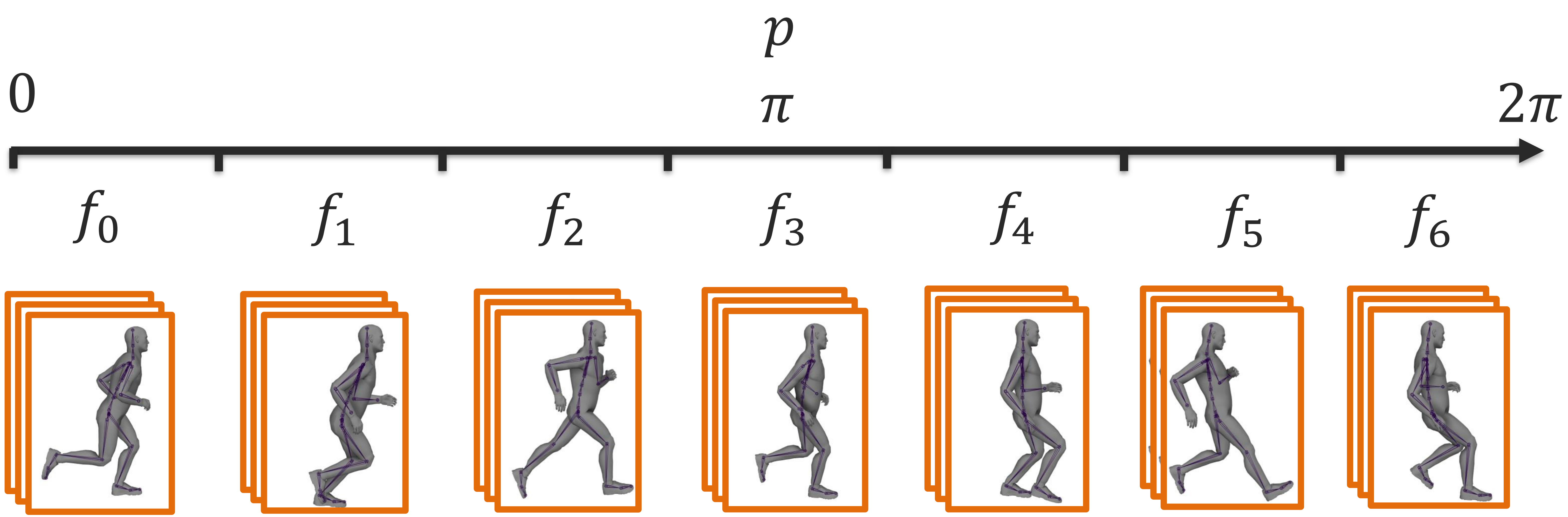


Use a separate f depending on the phase

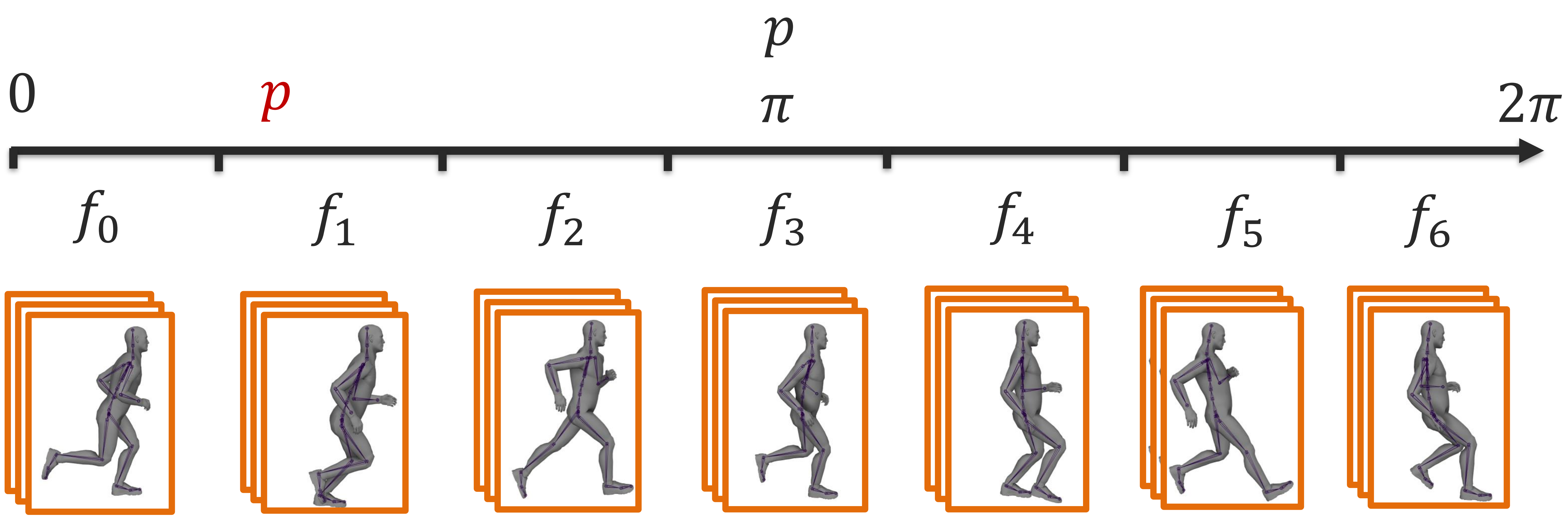
- Separate x and y into bins using the phase p.

- Separate x and y into bins using the phase p.
- At runtime select the bin for the current phase p.

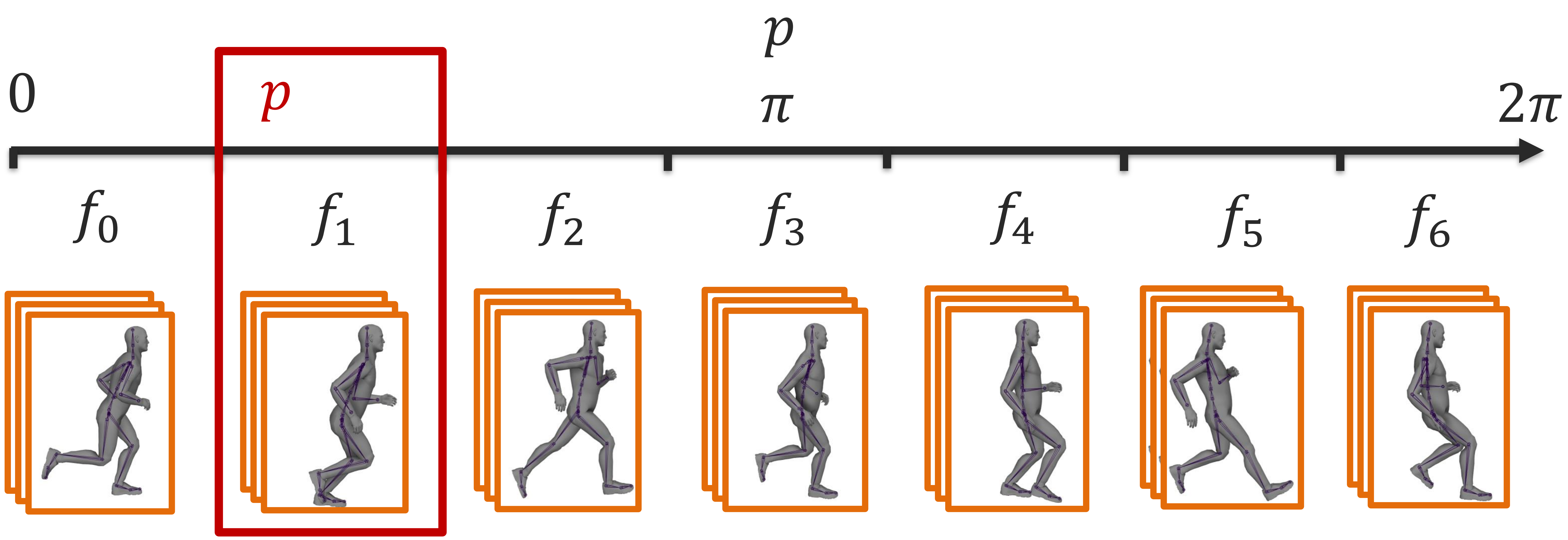
- Separate x and y into bins using the phase p.
- At runtime select the bin for the current phase p.
- Output the pose y in the selected bin using input x.

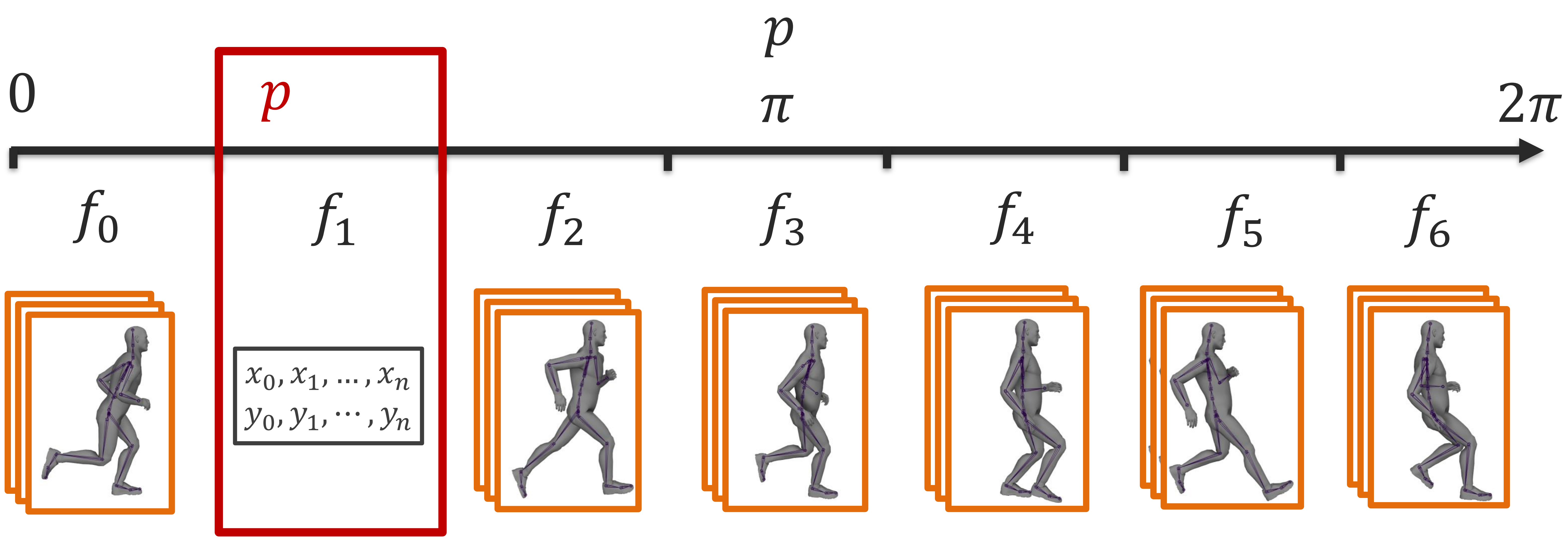


p
0
π
2π
f_0
f_1
f_2
f_3
f_4
f_5
f_6

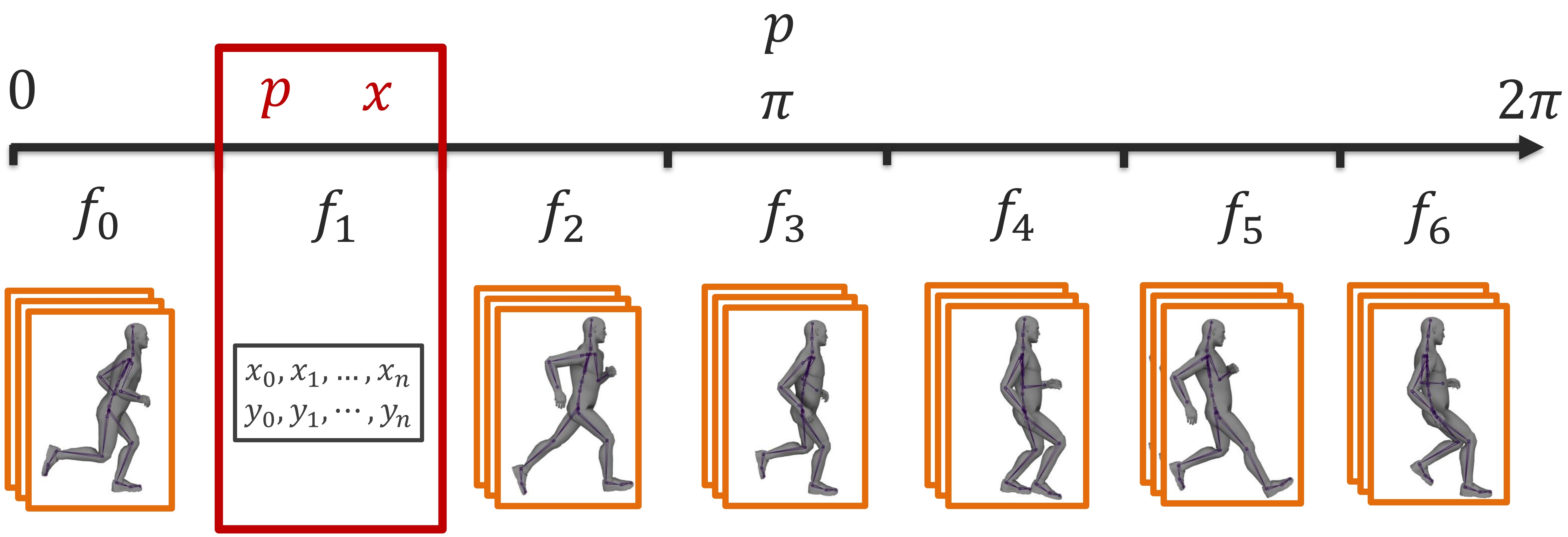


p
0
p
π
2π
f_0 f_1 f_2 f_3 f_4 f_5 f_6

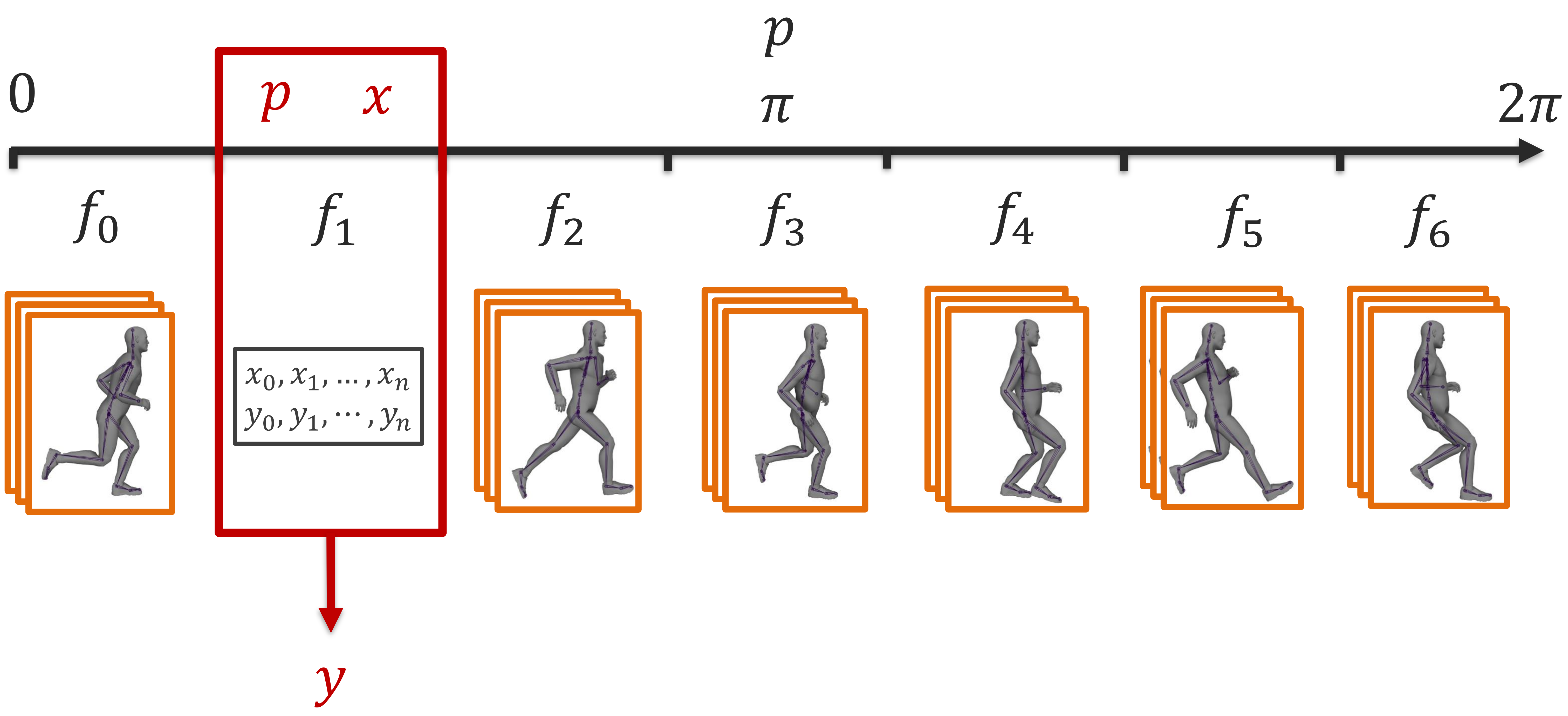

p
0
p
π
2π
f_0
f_1
f_2
f_3
f_4
f_5
f_6



p
0
p
π
2π
f_0
f_1
f_2
f_3
f_4
f_5
f_6
$x_0, x_1, \ldots, x_n$
$y_0, y_1, \cdots, y_n$



0
$p \quad x$
p
π
2π
f_0
f_1
f_2
f_3
f_4
f_5
f_6
$x_0, x_1, \ldots, x_n$
$y_0, y_1, \cdots, y_n$


p
0
p x
π
2π
f_0
f_1
f_2
f_3
f_4
f_5
f_6
$x_0, x_1, \ldots, x_n$
$y_0, y_1, \cdots, y_n$
y

Another Example

Input x

- **Joint Positions** in the previous frame.
- **Joint Velocities** in the previous frame.
- **Target Position** of the root in 1 second.
- **Target Velocity** of the root in 1 second.
- **Target Direction** of the root in 1 second.

Output y

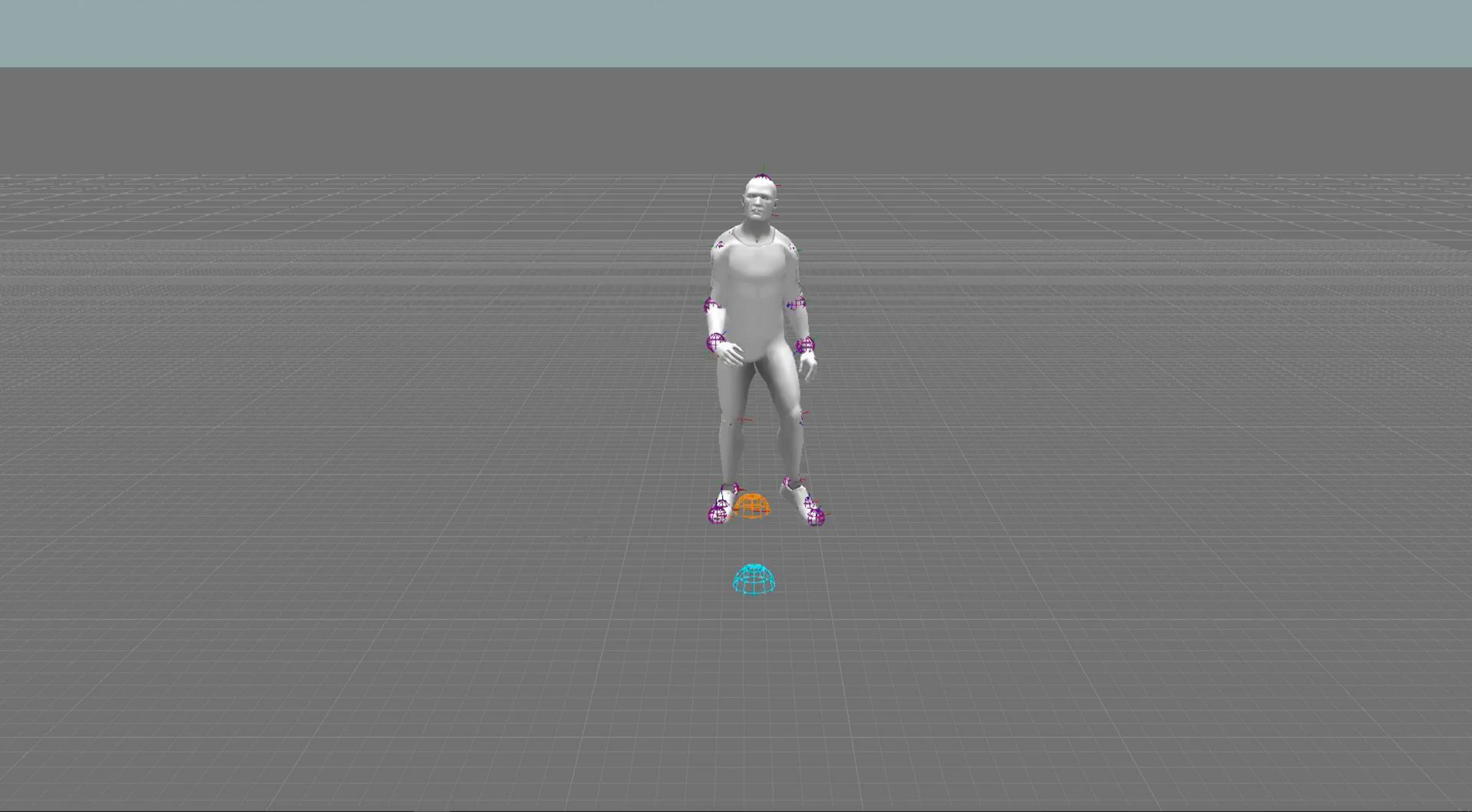
- **Joint Positions** for the current frame.
- **Joint Velocities** for the current frame.
- **Joint Rotations** for the current frame.

Function f

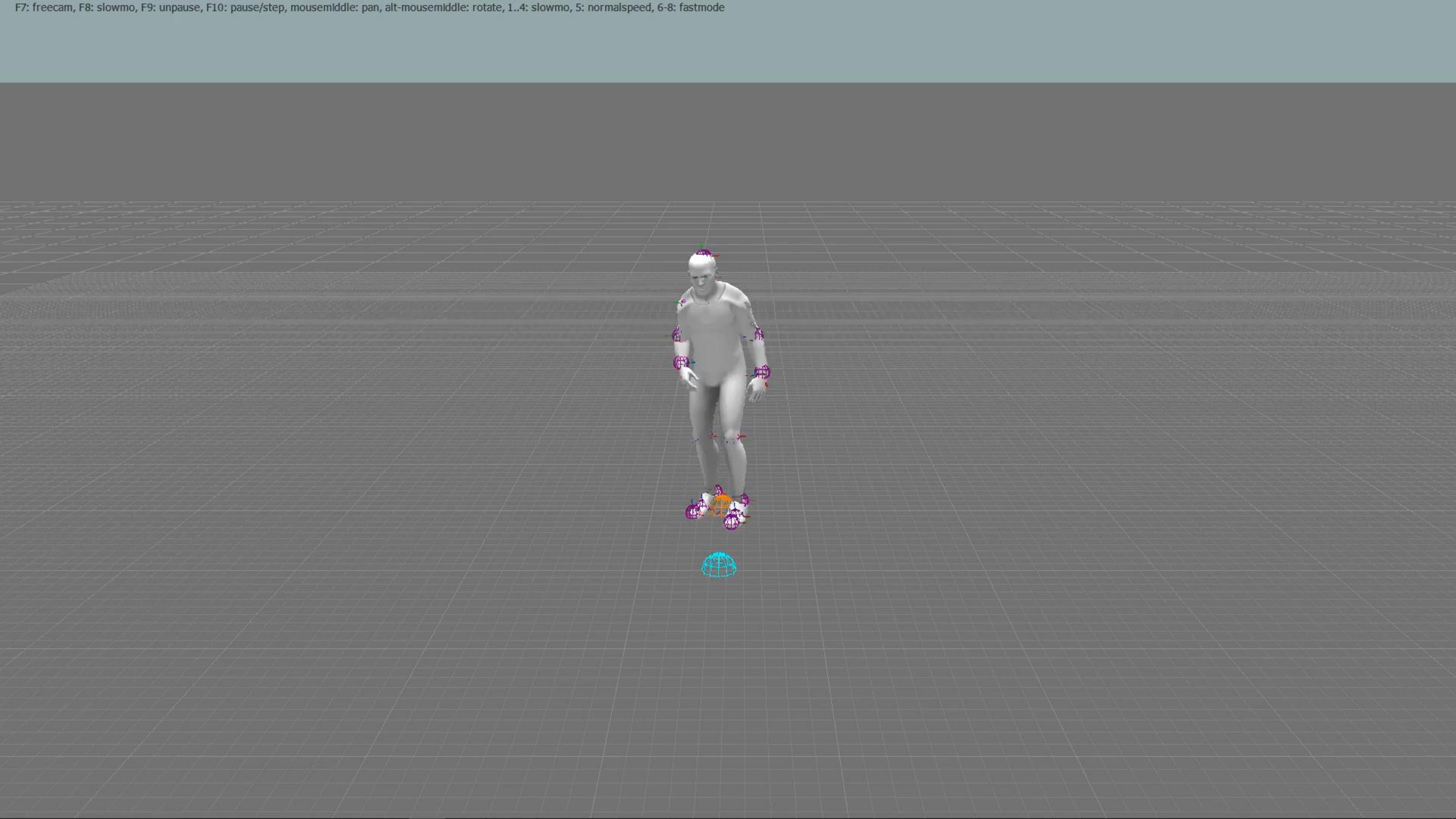
- **Select** an f each frame using the phase p.
- **Call** the chosen f.
- **Update** the phase value p.

Phase-Conditioned Nearest Neighbour

F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode



Phase-Conditioned Gaussian Process

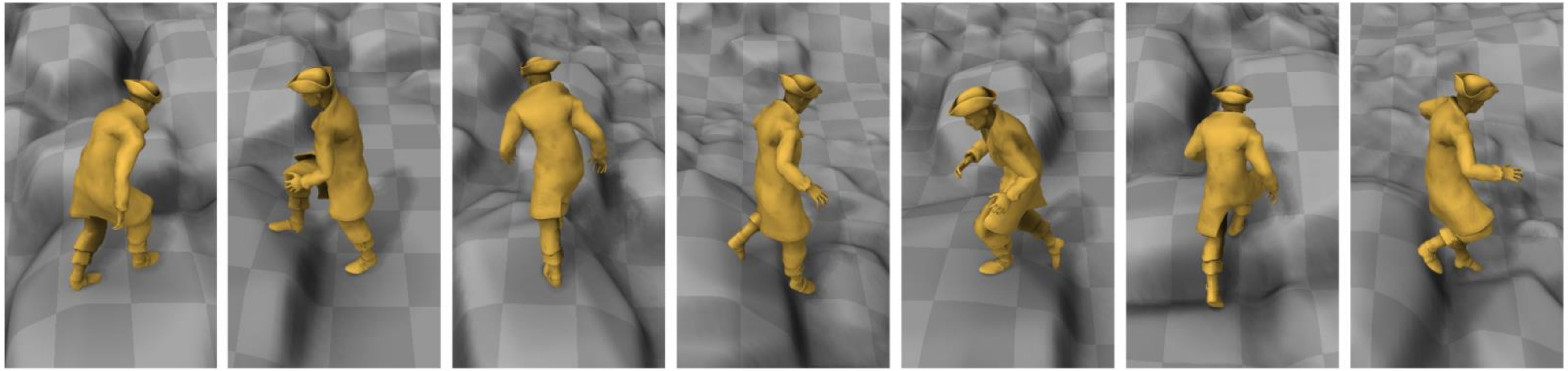

F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode

- What if the phase lies in-between two bins?

- What if the phase lies in-between two bins?
- Is it a waste to train multiple functions f ?

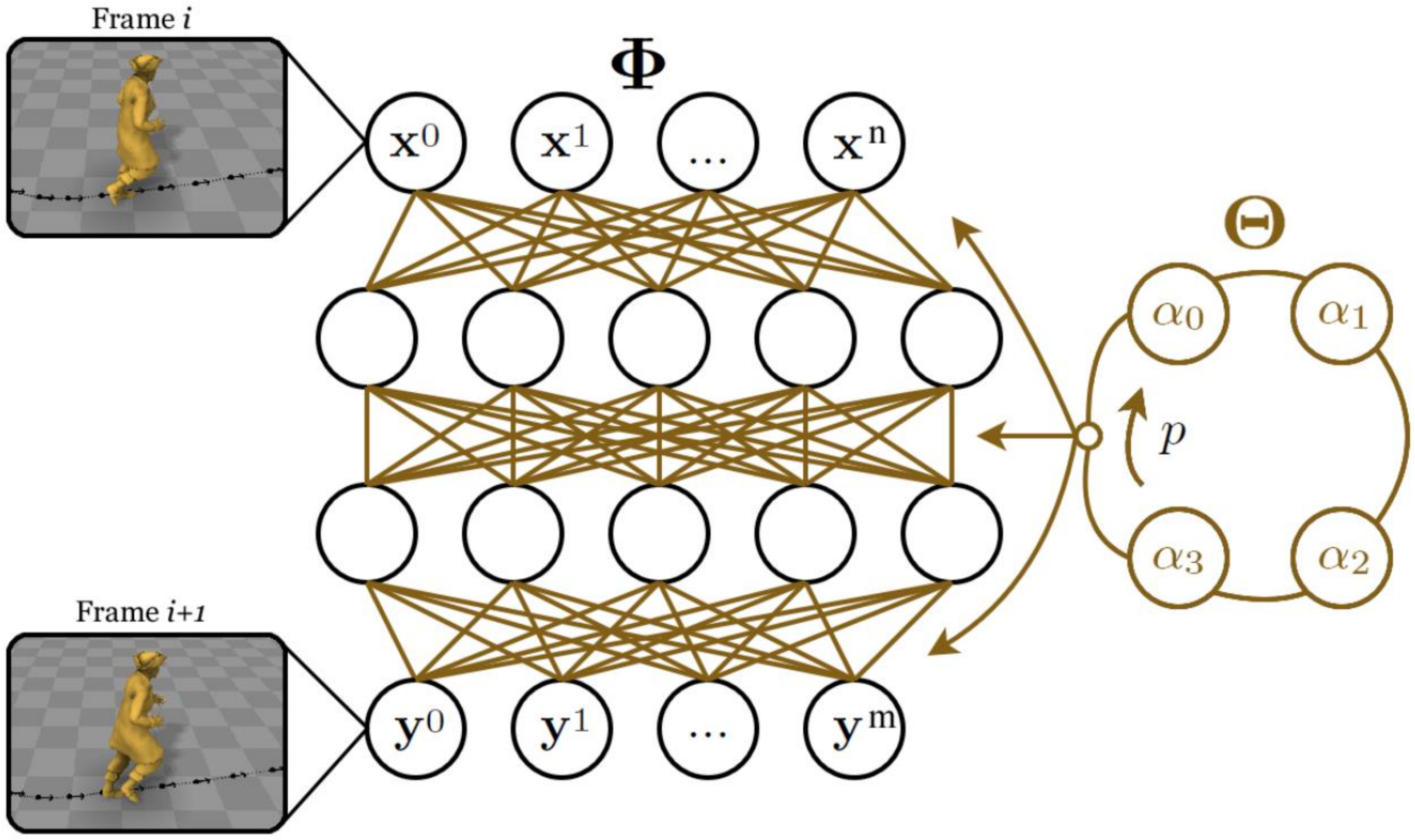
- What if the phase lies in-between two bins?
- Is it a waste to train multiple functions f ?
- How can we use Neural Networks to solve this?

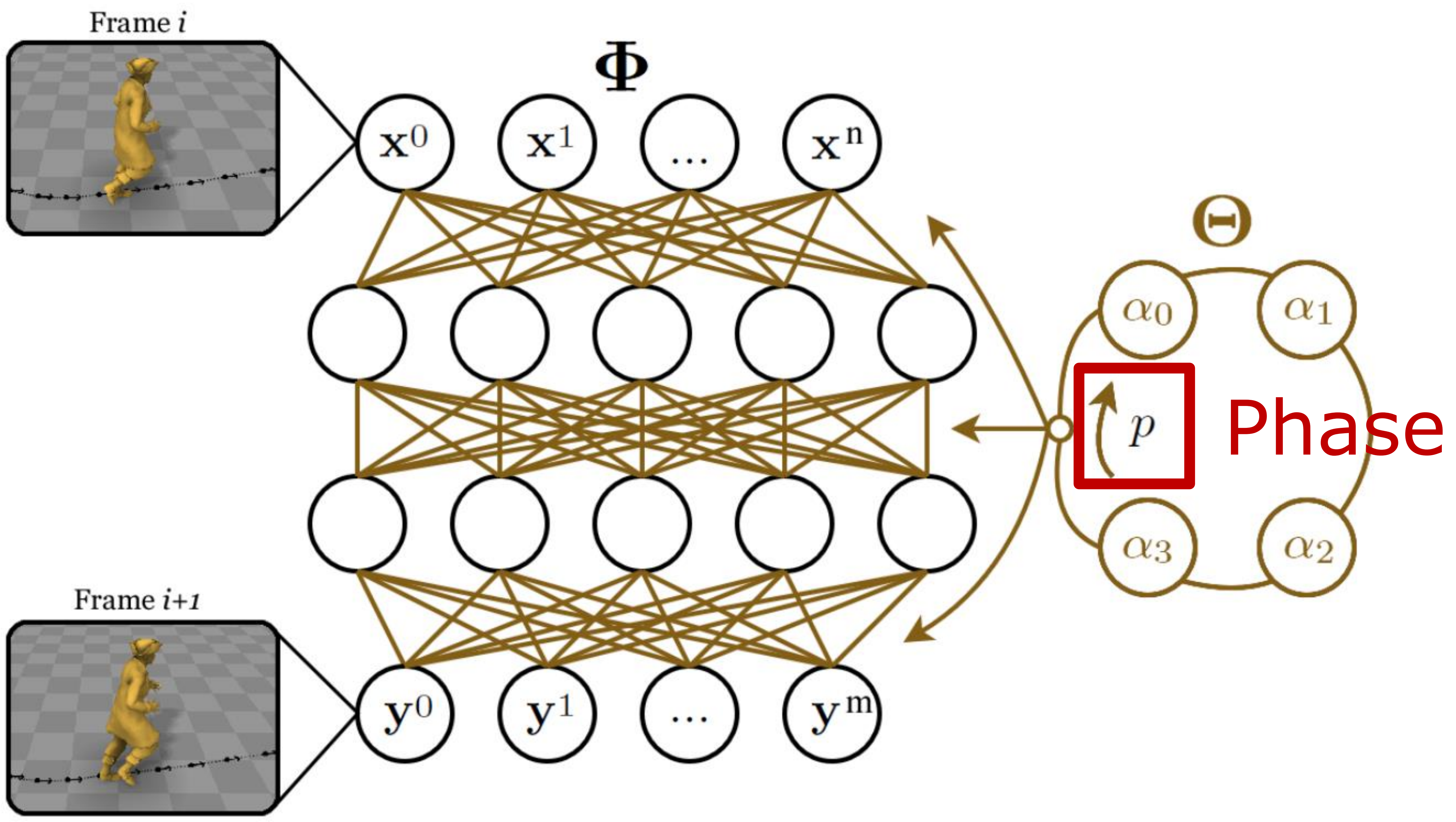
Phase-Functioned Neural Network



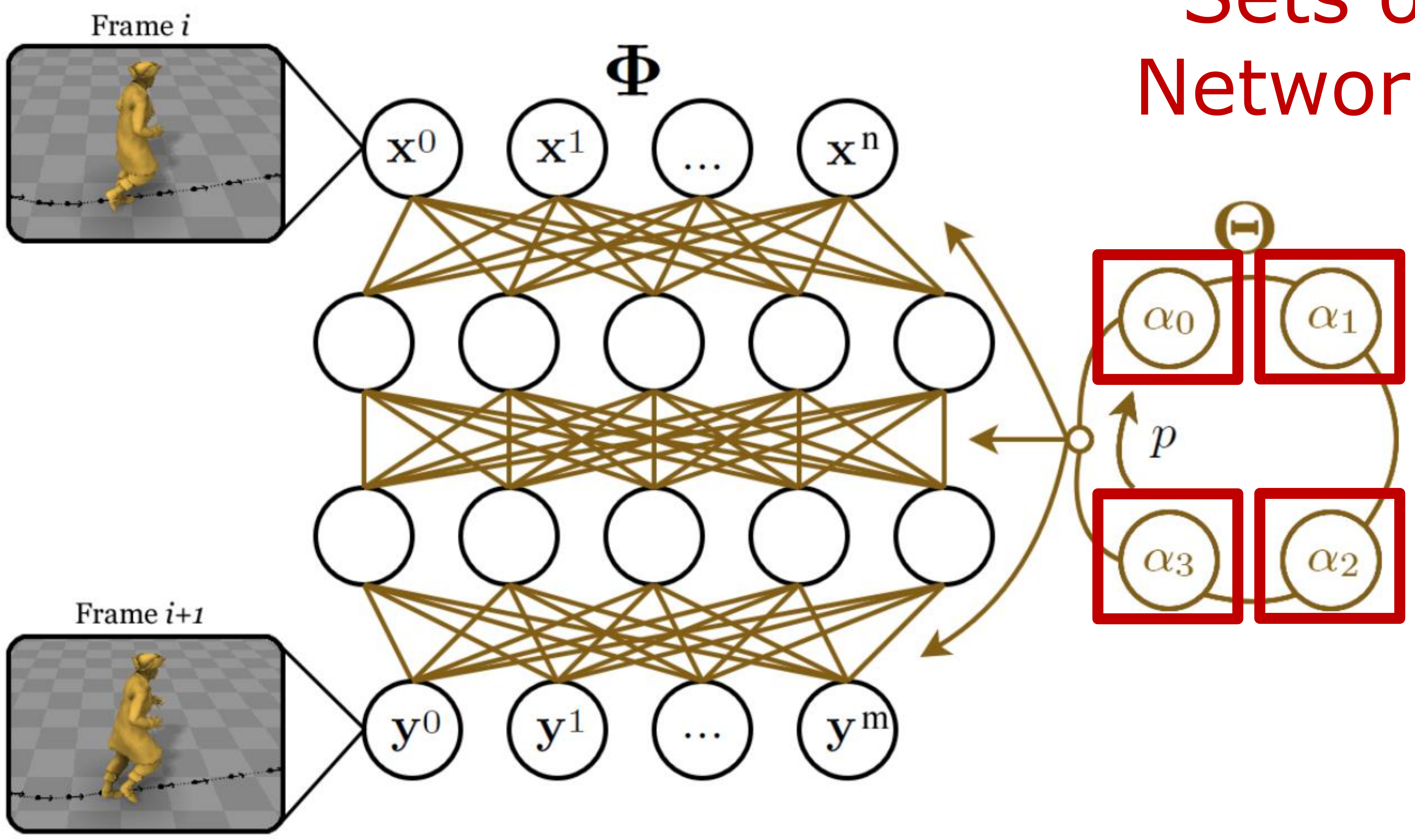
Phase-Functioned Neural Network

A Neural Network where the weights of the network are generated from the phase.

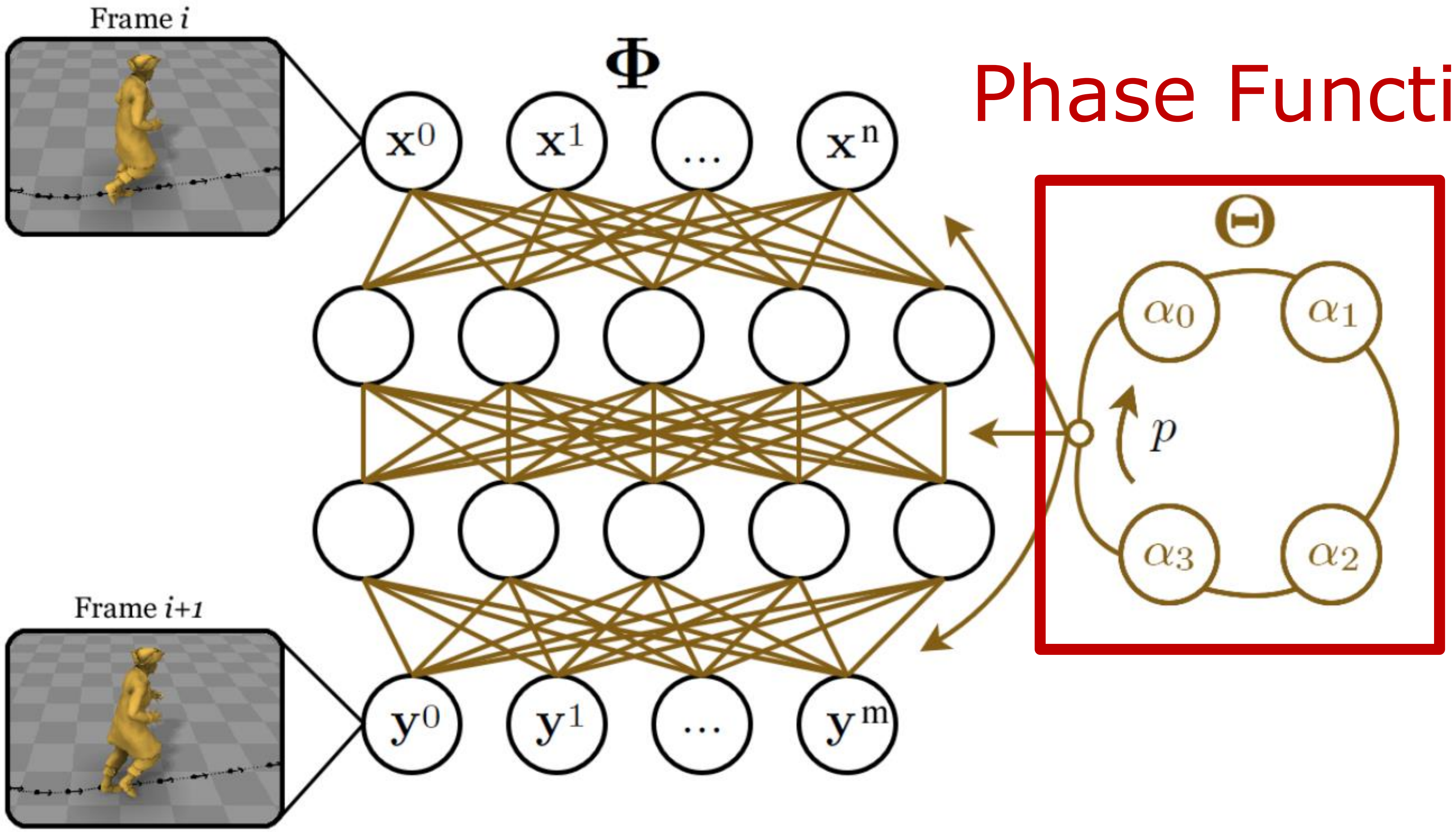

Frame i
Φ
x^0
x^1
...
x^n
Θ
α_0
α_1
p
α_3
α_2
Frame i+1
y^0
y^1
...
y^m

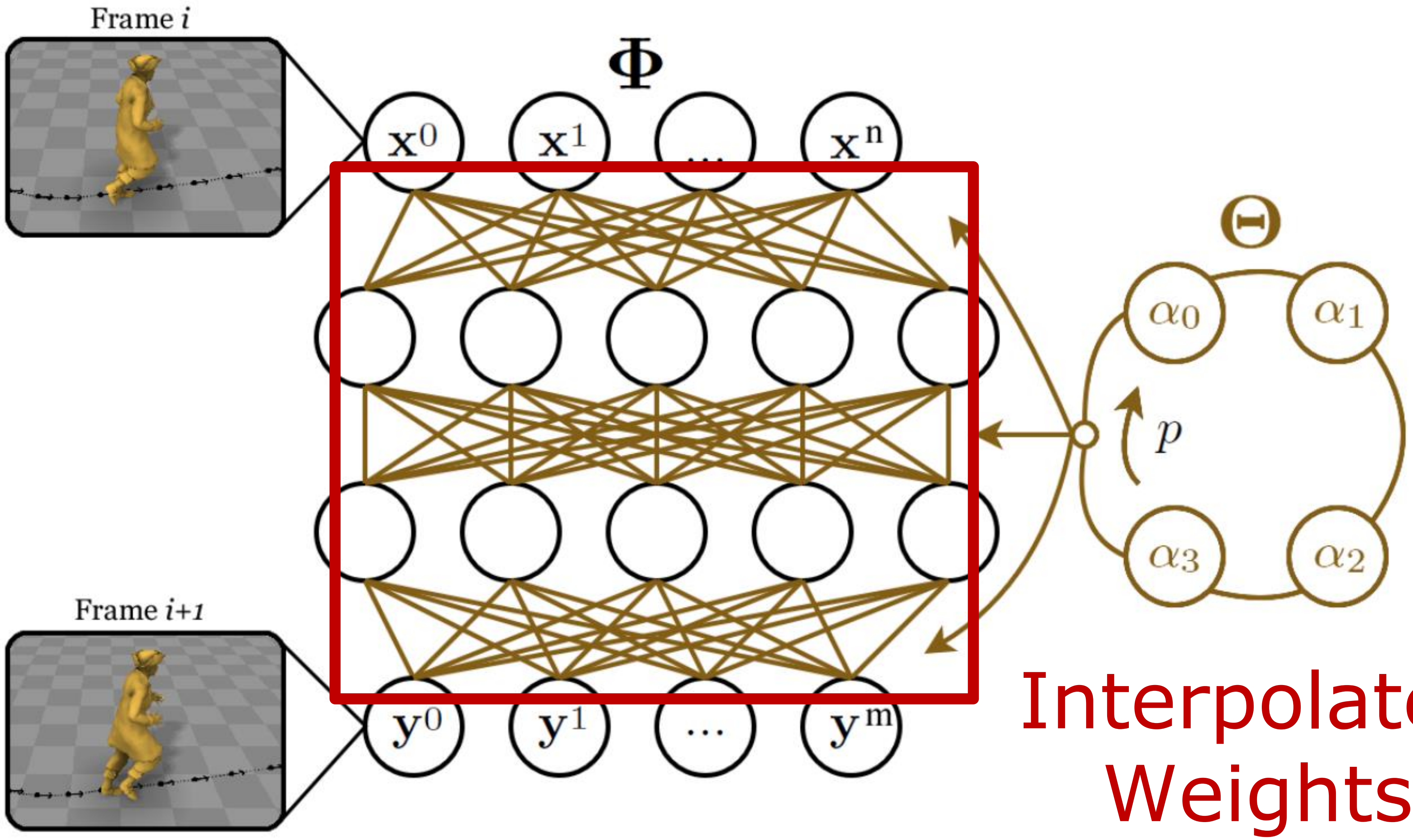

Frame i
Φ
$\mathbf{x}^0$
$\mathbf{x}^1$
...
$\mathbf{x}^n$
Θ
α_0
α_1
p
Phase
α_3
α_2
Frame i+1
$\mathbf{y}^0$
$\mathbf{y}^1$
...
$\mathbf{y}^m$

Sets of Neural Network weights



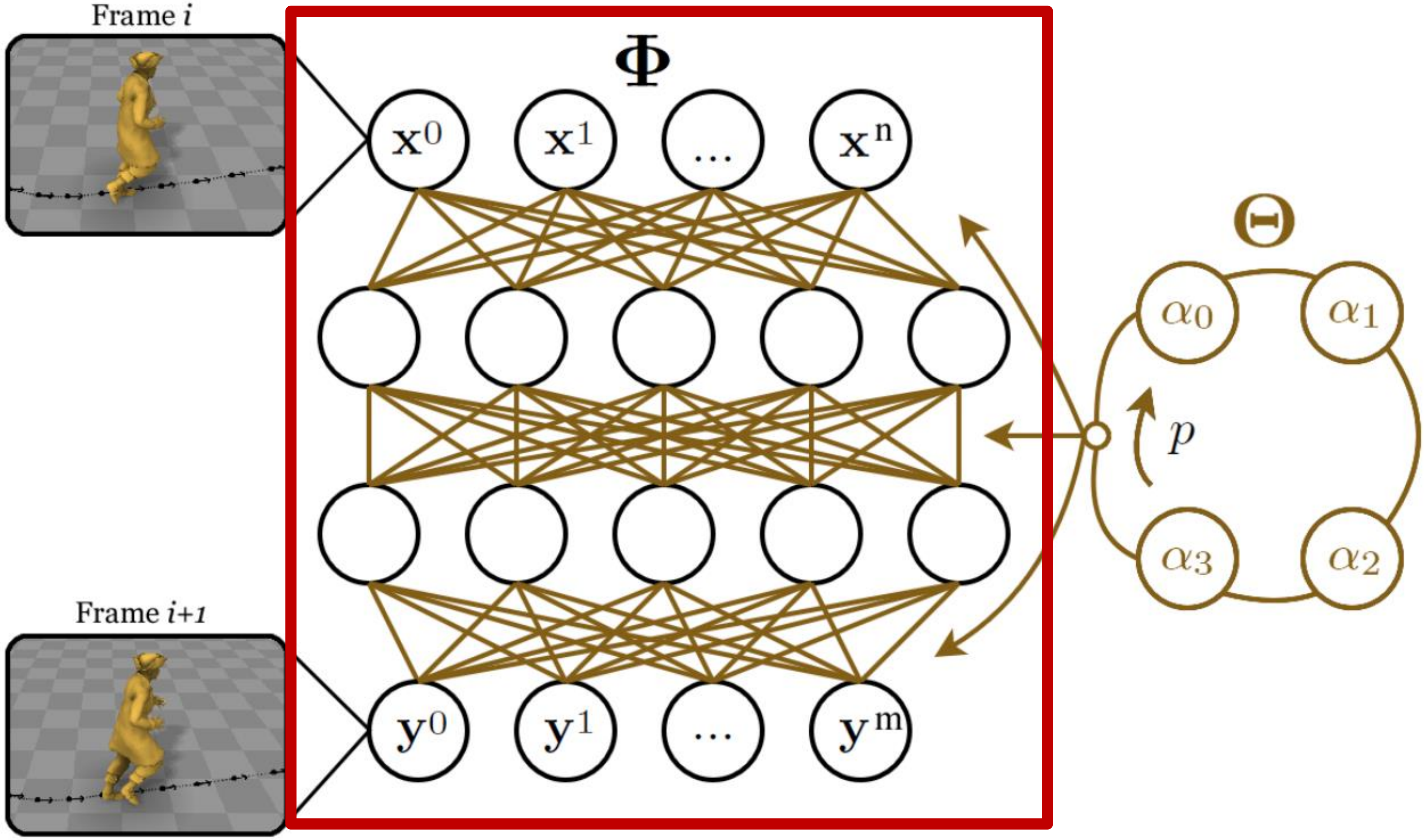
Phase Function

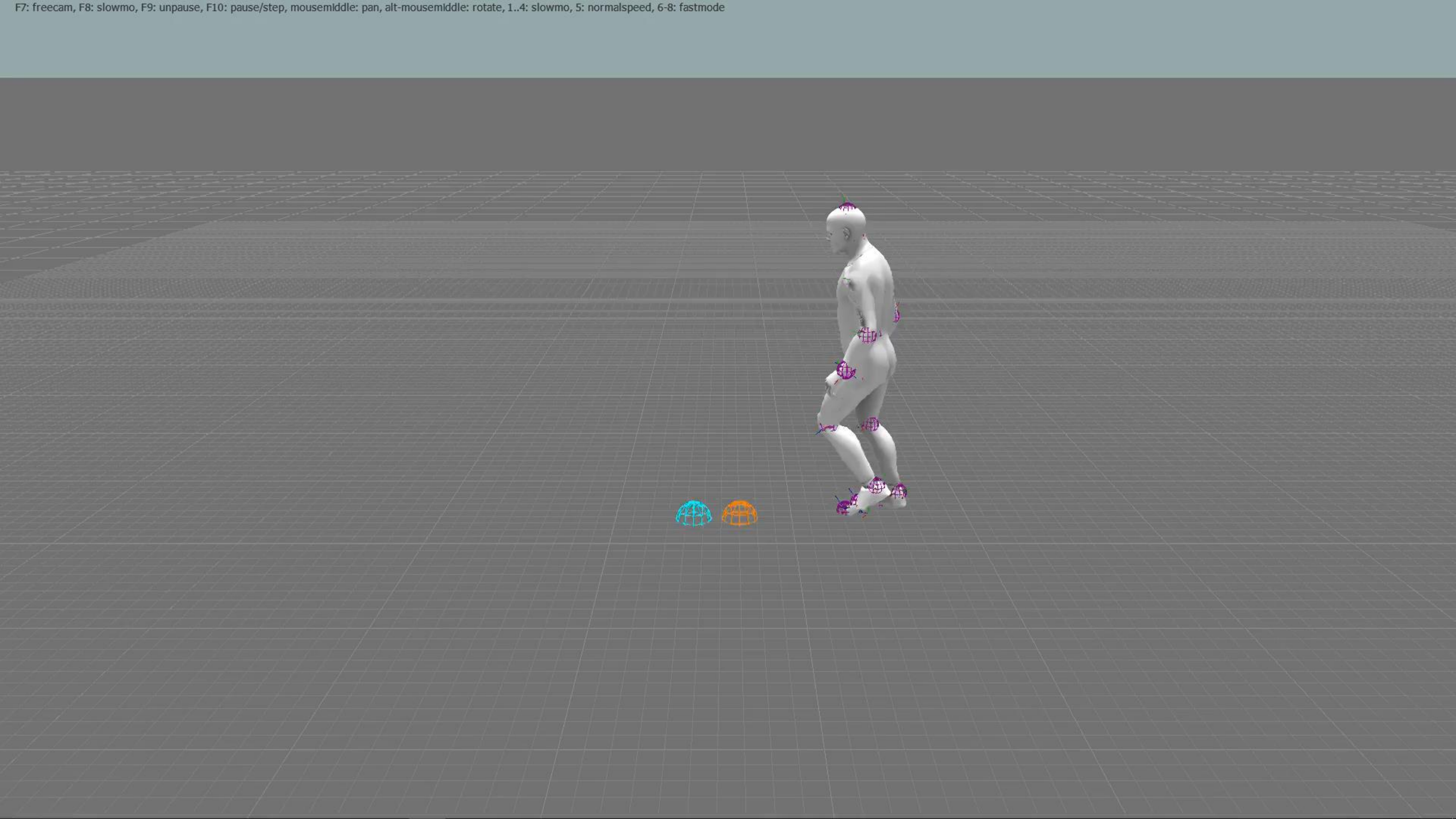




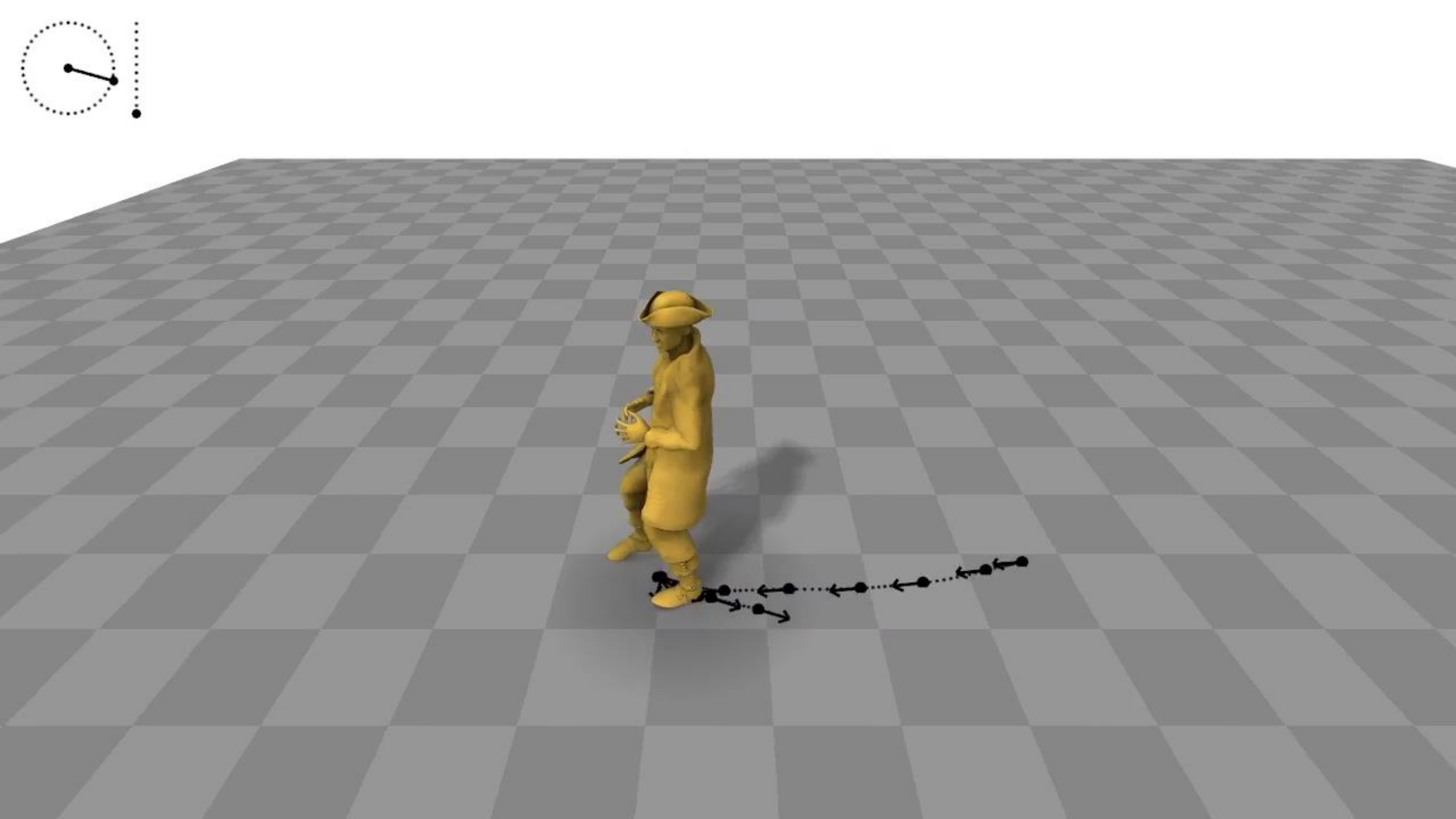
Interpolated Weights

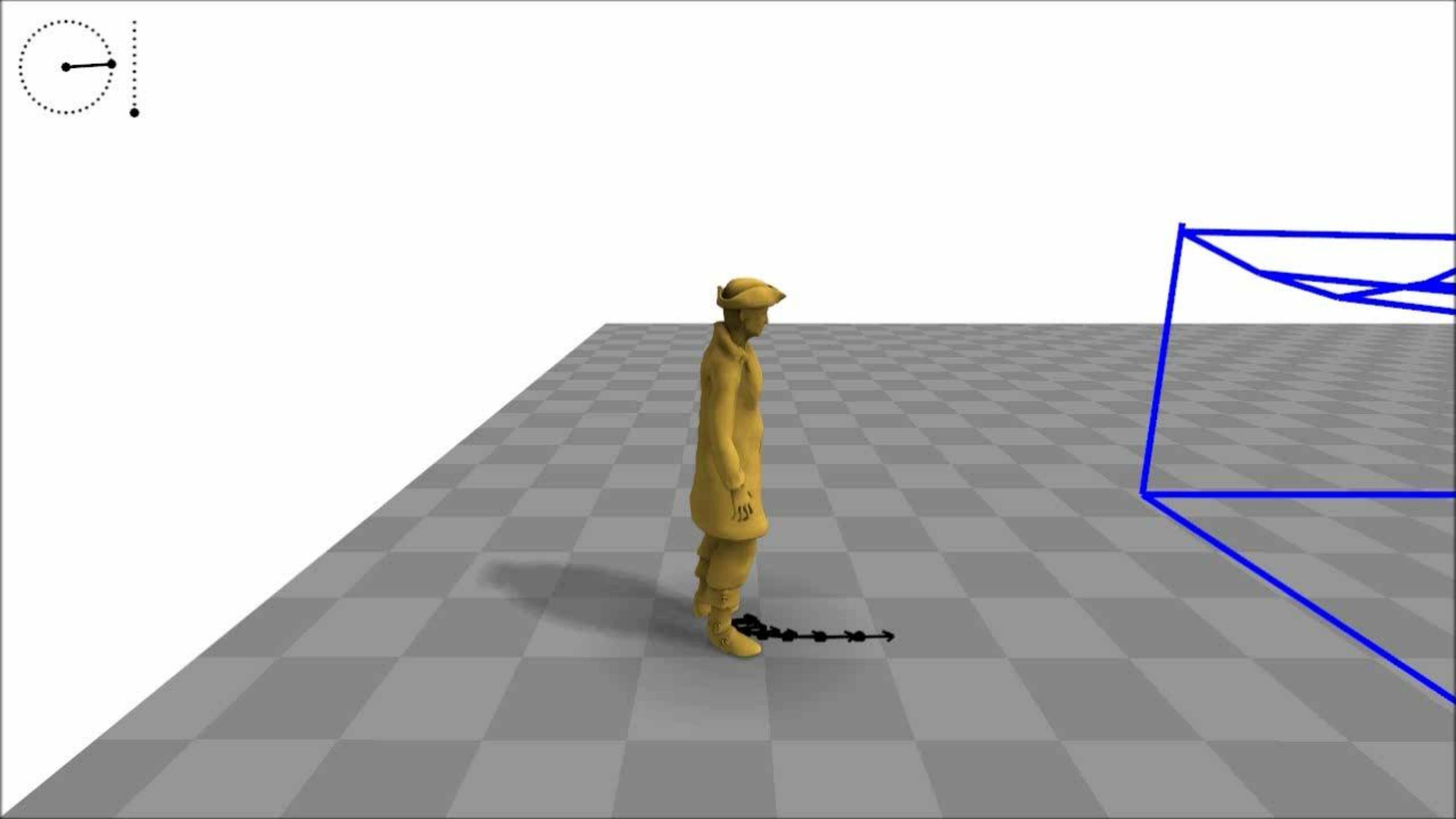
Neural Network

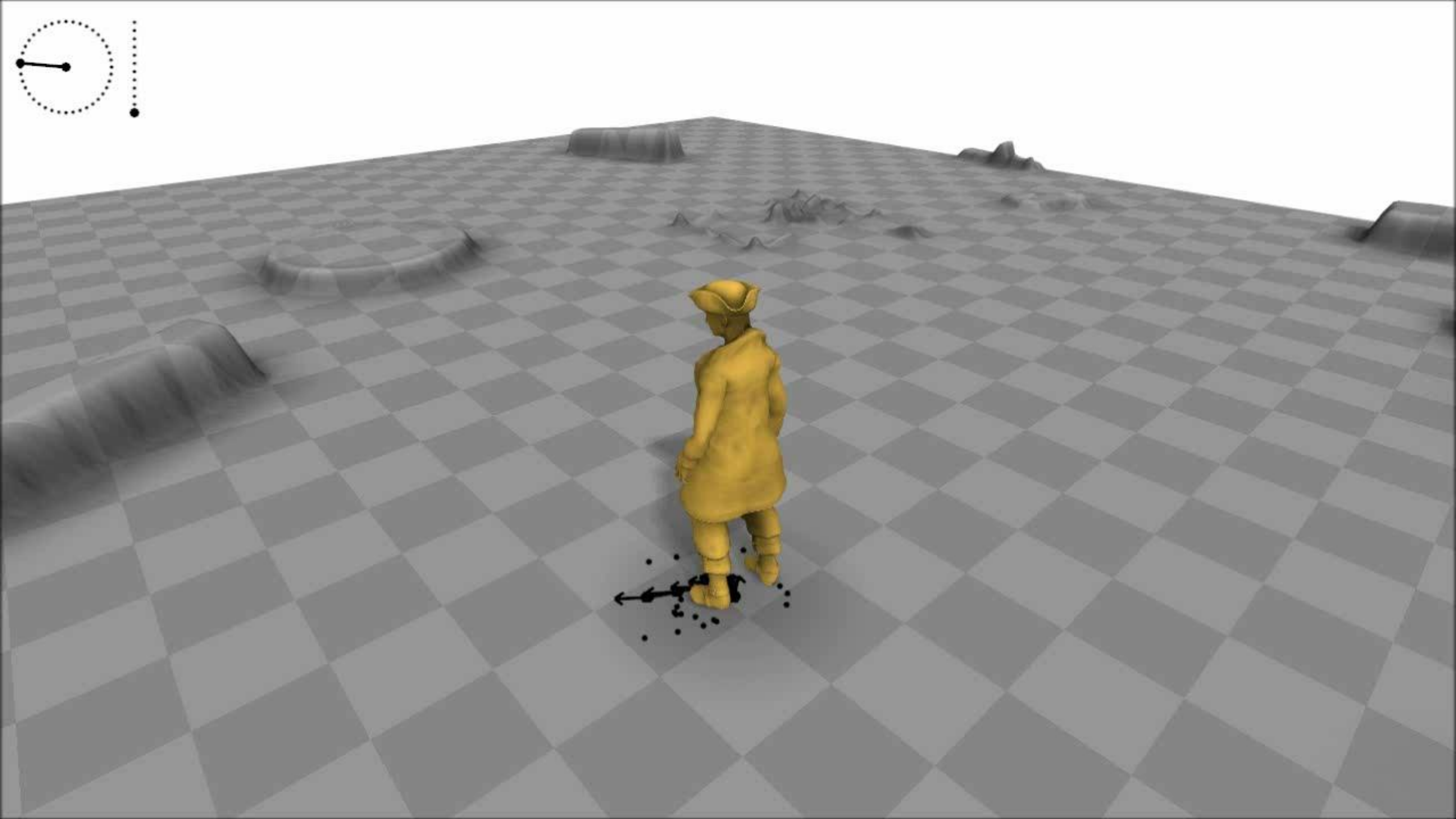



F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode

Continue tweaking x and y…







Memory		Runtime
~10 mb		~1 ms
	OR	
~100 mb		~0.5 ms

- Separate Data ✓
- Specify Desired Variables ✓
- Generalize Solution ✓

Conclusion

Sacrifices

Sacrifices

- We must give up precise control.

Sacrifices

- We must give up precise control.
- Requires learning a whole new skill set.

Sacrifices

- We must give up precise control.
- Requires learning a whole new skill set.
- Does not deal with many special cases.

Scalability

Scalability

- Animation quality is losing the battle against complexity.

Scalability

- Animation quality is losing the battle against complexity.
- We *can* use Machine Learning to balance this fight.

Scalability

- Animation quality is losing the battle against complexity.
- We *can* use Machine Learning to balance this fight.
- These ideas are one way of making progress.

The Future

- How can we remove the phase variable?

The Future

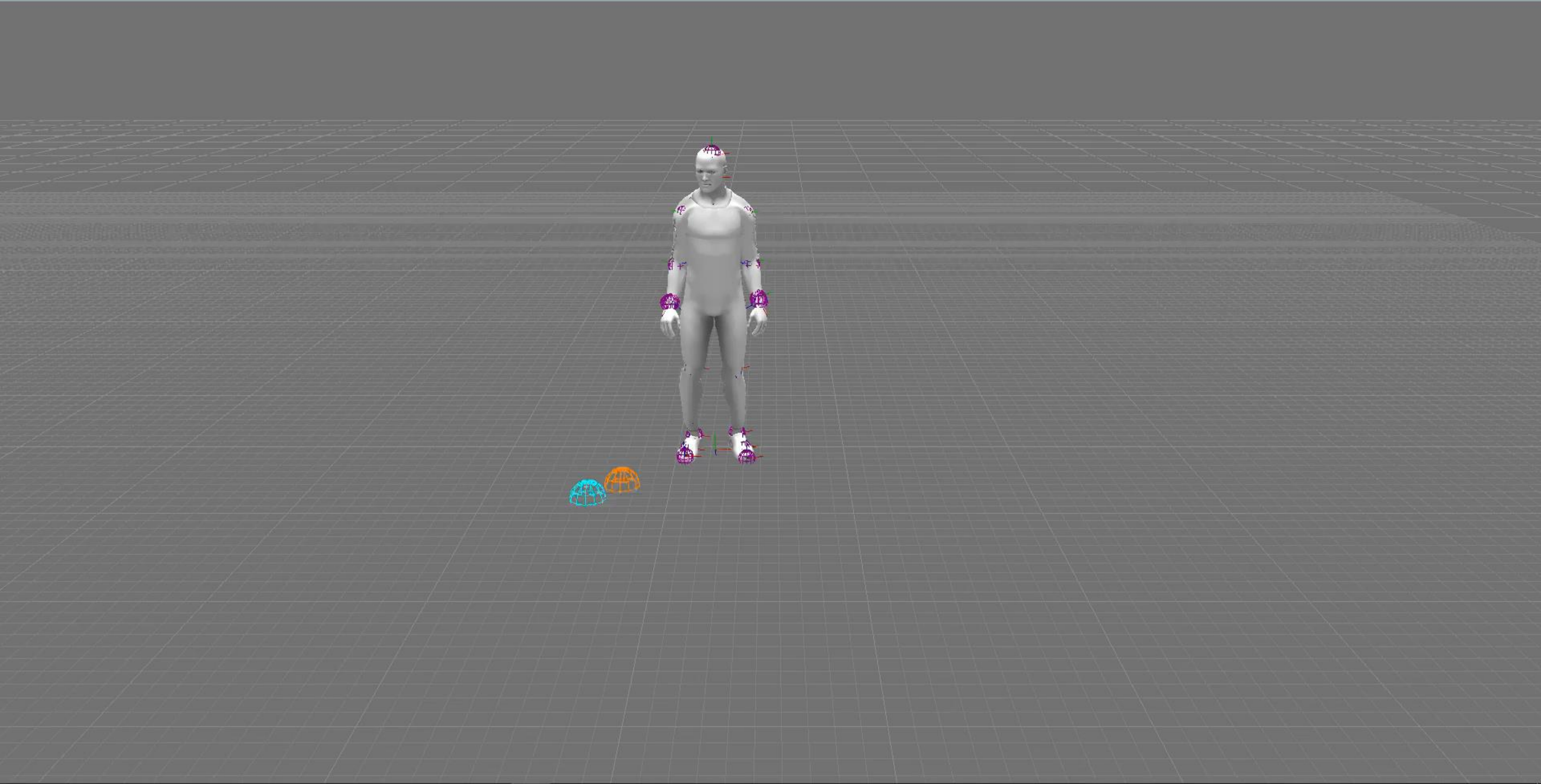
- How can we remove the phase variable?
- Can we scale to hundreds of different styles?

The Future

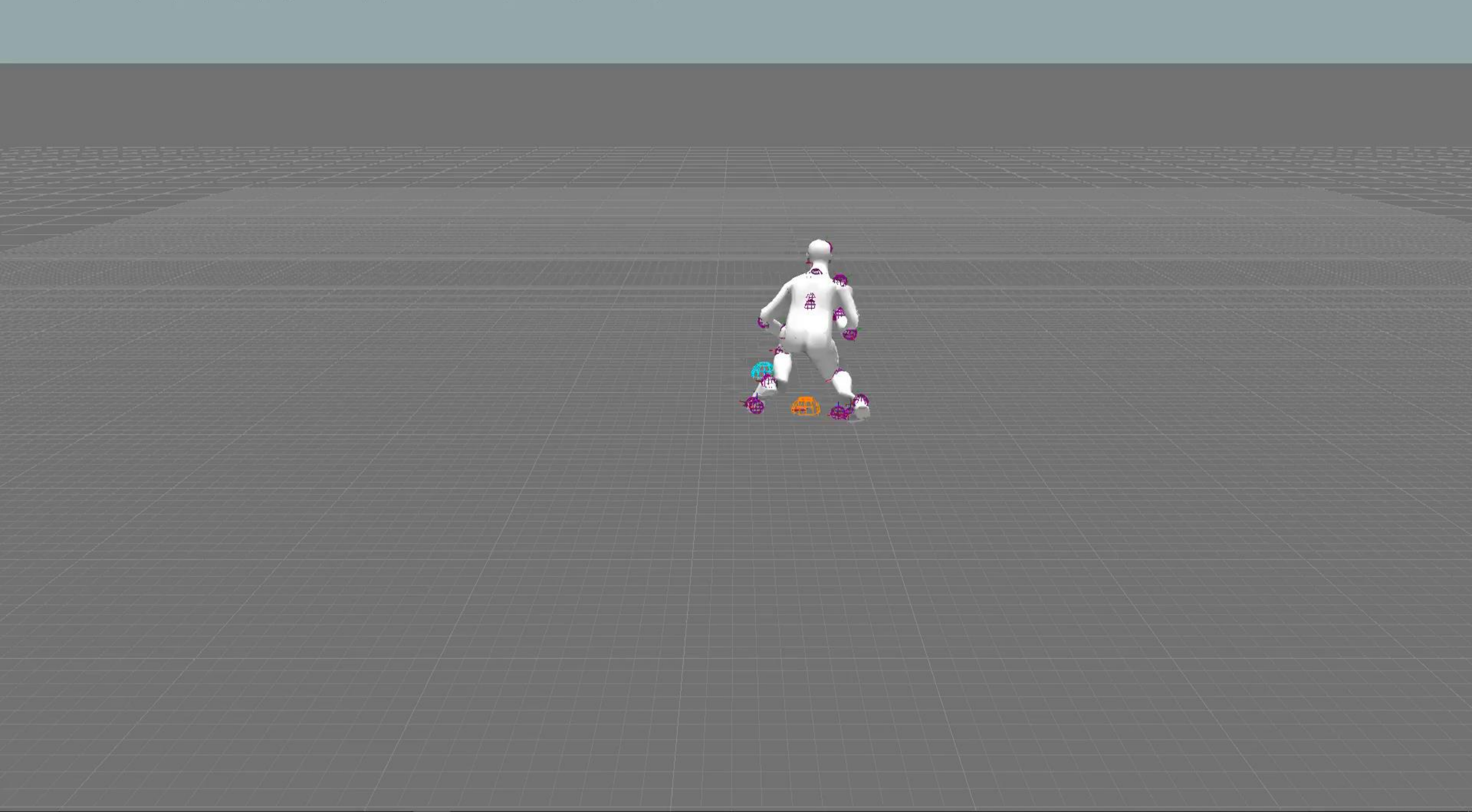
- How can we remove the phase variable?
- Can we scale to hundreds of different styles?
- How can we continue to improve the quality?

What Machine Learning is Really Like

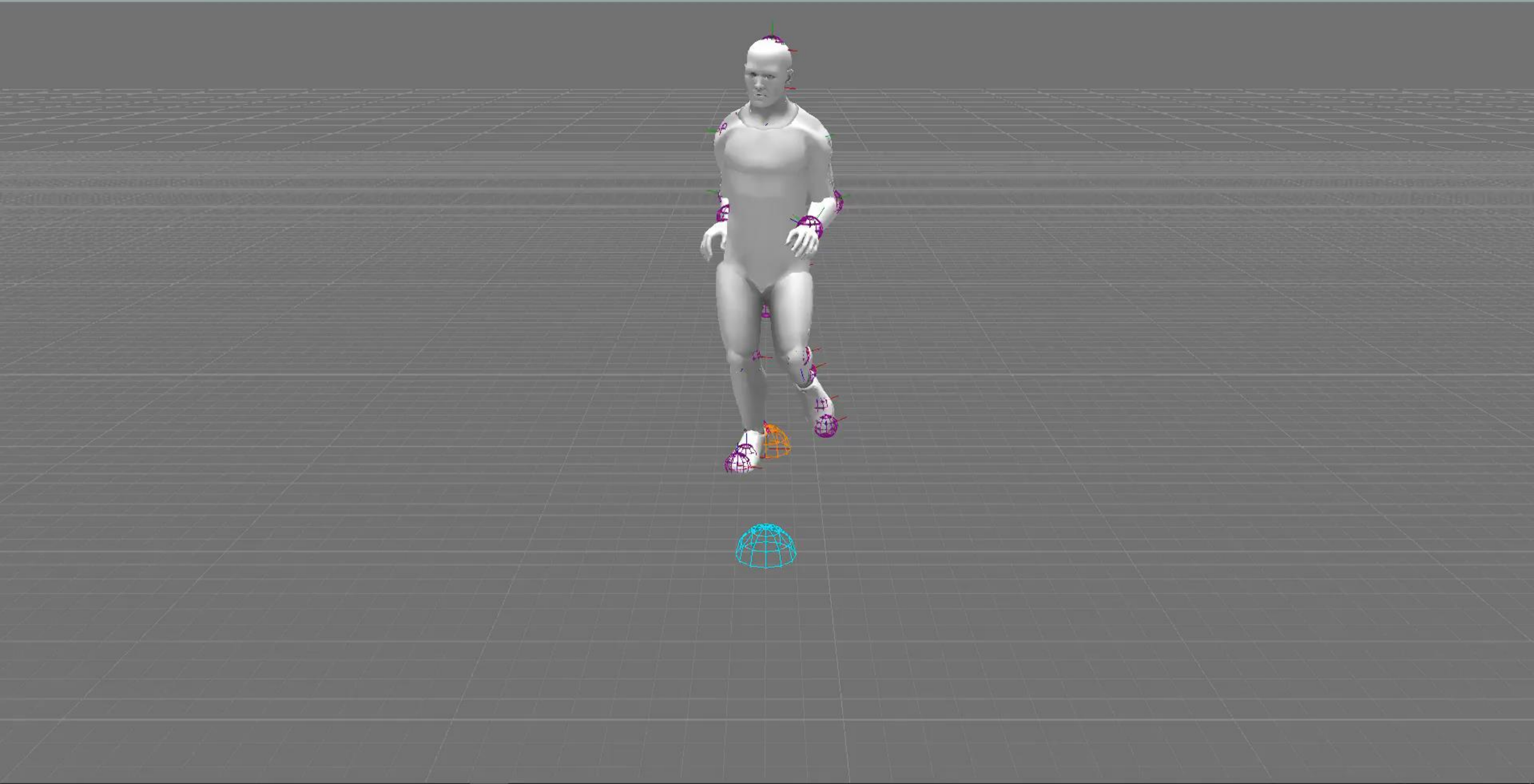
F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode



F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode



F7: freecam, F8: slowmo, F9: unpause, F10: pause/step, mousemiddle: pan, alt-mousemiddle: rotate, 1..4: slowmo, 5: normalspeed, 6-8: fastmode



Thanks!

Any Questions?