

# Game Control



Ken-Yi Lee

Game Programming, Fall 2020 @ National Taiwan University

# Game Programming

- Rendering
- Looping and control
- Math
- Behaviour and navigation (AI)
- Physics
- Animation and effects
- Networking

# Game Programming

- Rendering
- Looping and **control**
- Math
- Behaviour and navigation (AI)
- Physics
- Animation and effects
- Networking



[https://en.wikipedia.org/wiki/Game\\_controller](https://en.wikipedia.org/wiki/Game_controller)







[https://en.wikipedia.org/wiki/Game\\_controller](https://en.wikipedia.org/wiki/Game_controller)



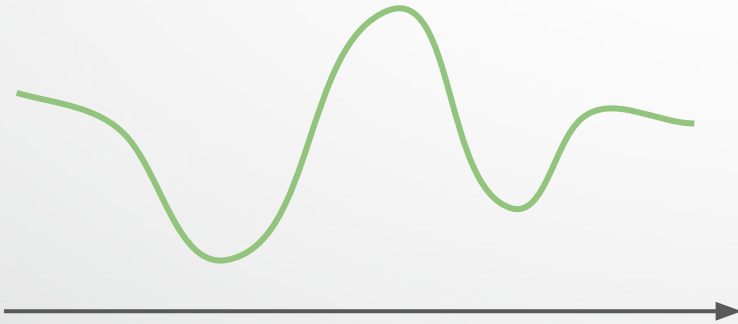
[https://en.wikipedia.org/wiki/Game\\_controller](https://en.wikipedia.org/wiki/Game_controller)



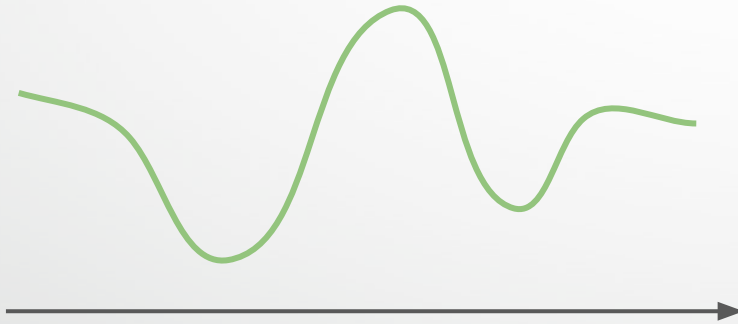


[https://en.wikipedia.org/wiki/Game\\_controller](https://en.wikipedia.org/wiki/Game_controller)

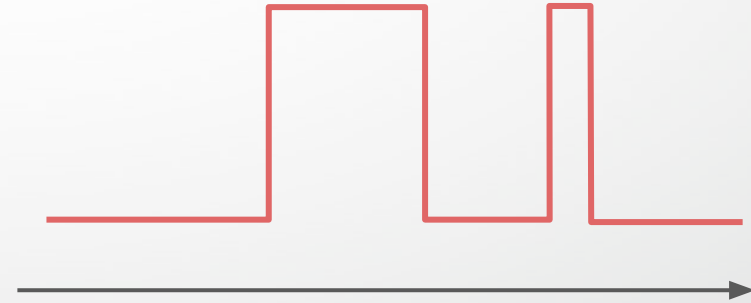




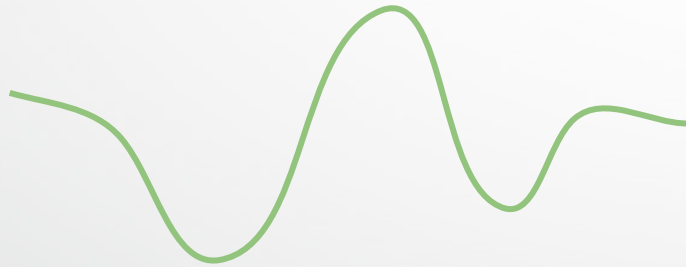
Analog



Analog



Digital

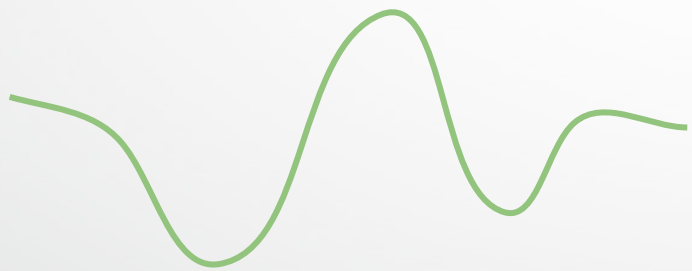


Analog<sup>2</sup>

+



Digital<sup>6</sup>

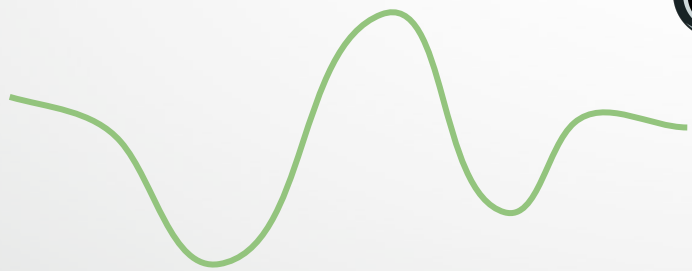


Float<sup>2</sup>

+



Bool<sup>6</sup>

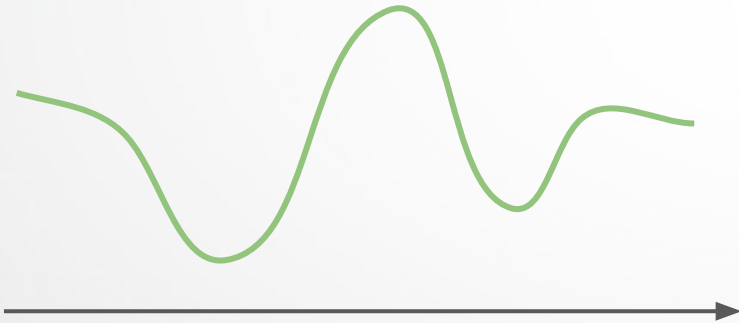


Float<sup>2</sup>

**X**



Bool

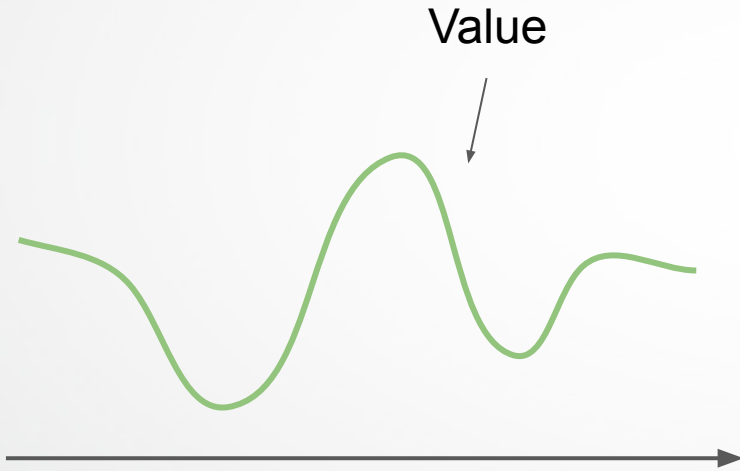


Float

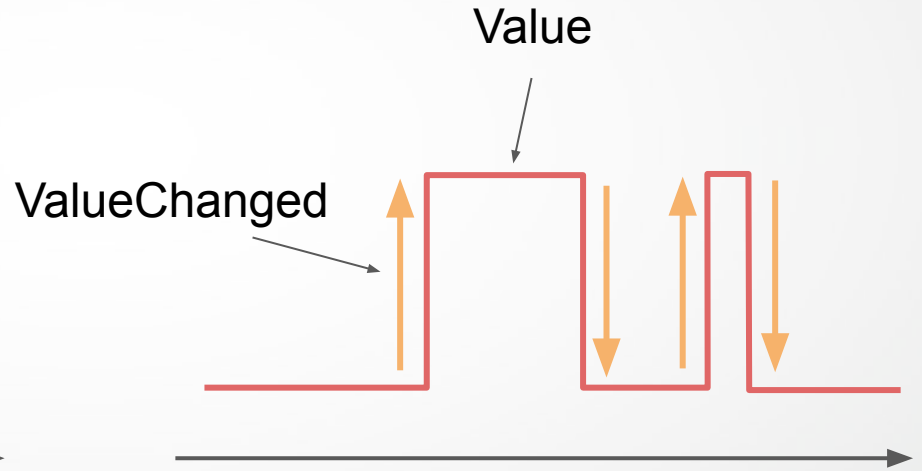


Bool

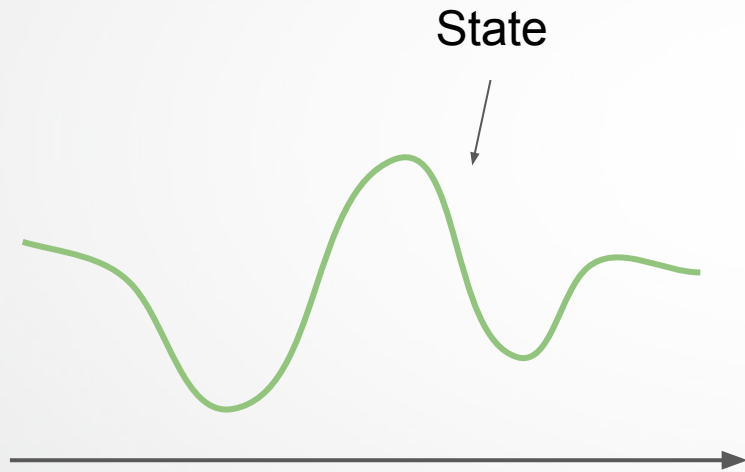




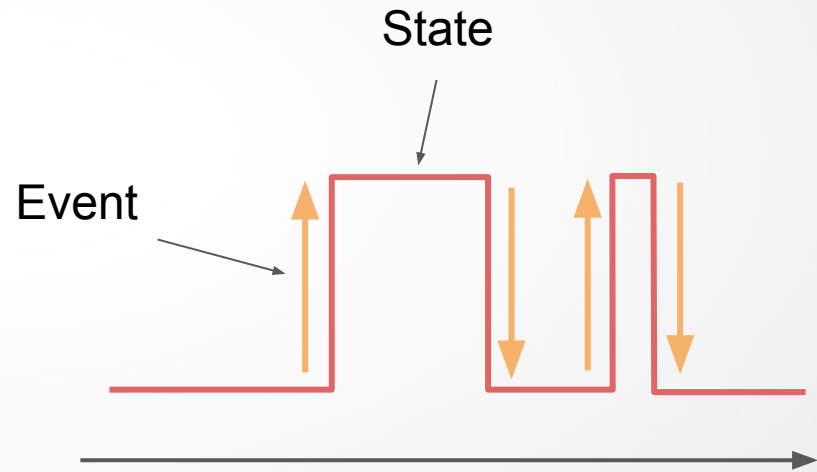
Float



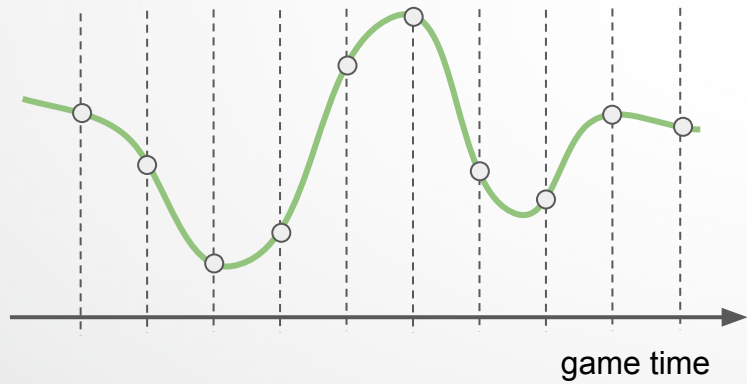
Bool



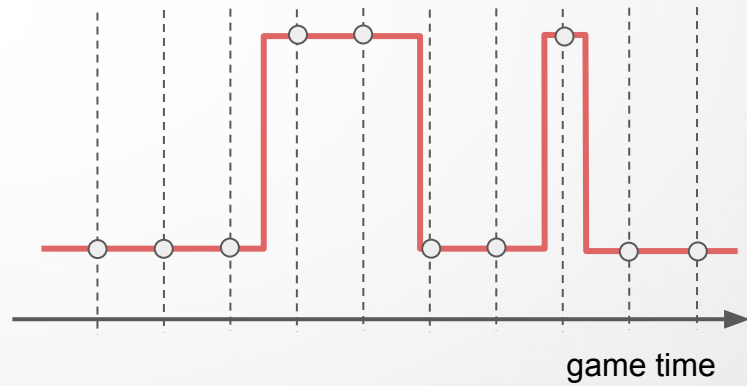
Float



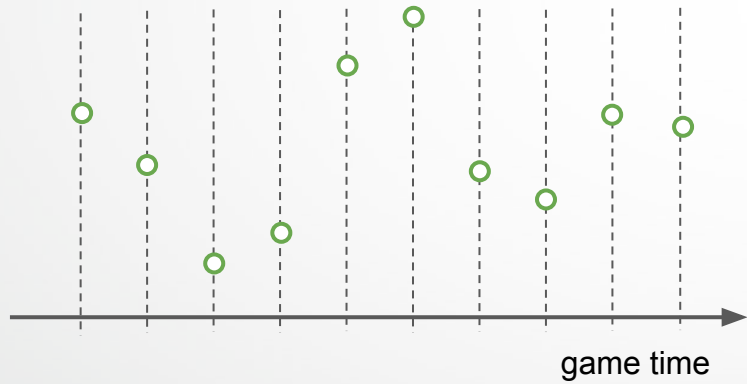
Bool



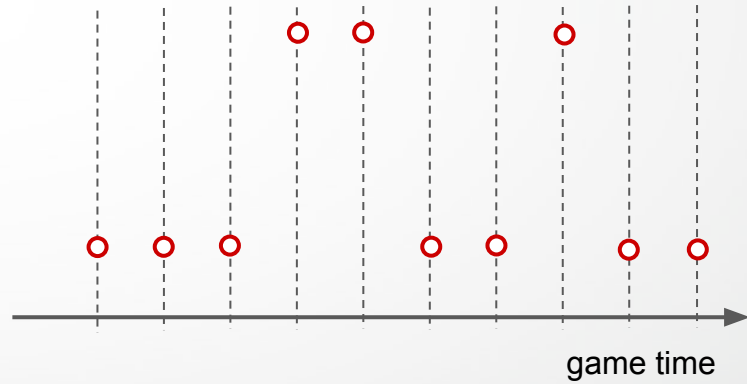
Float



Bool

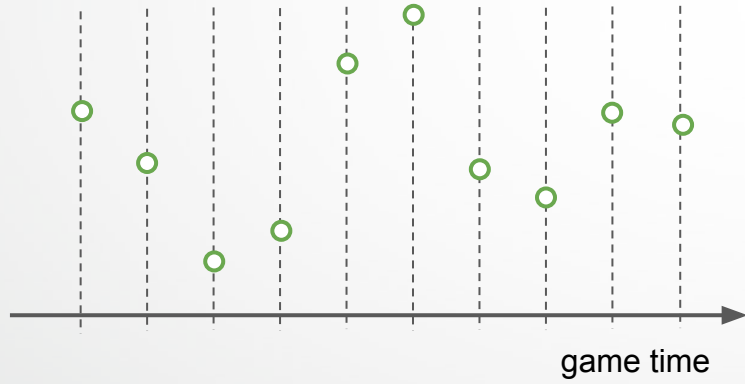


Float



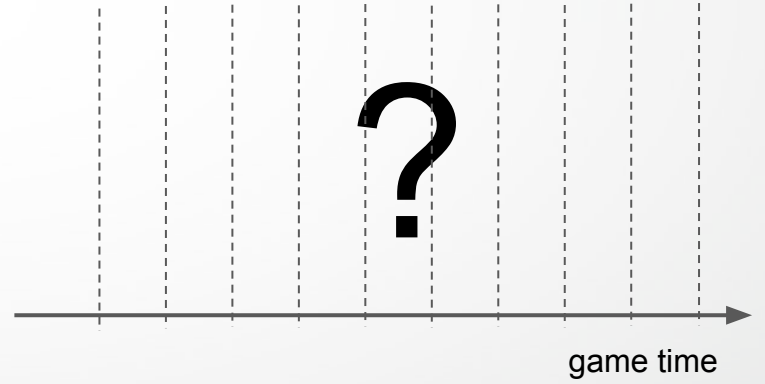
Bool

State



Float

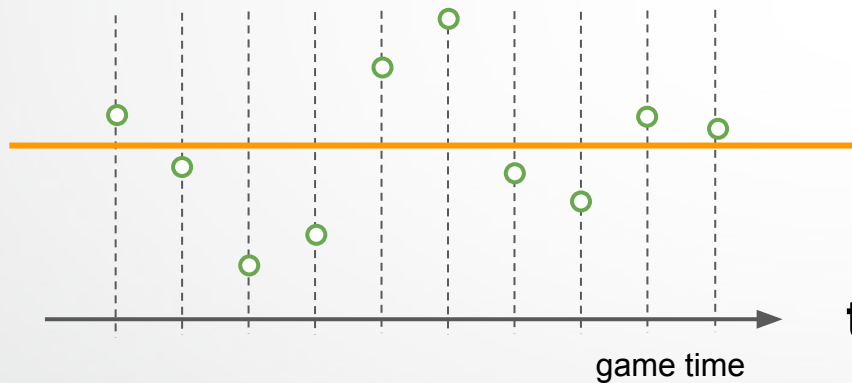
to



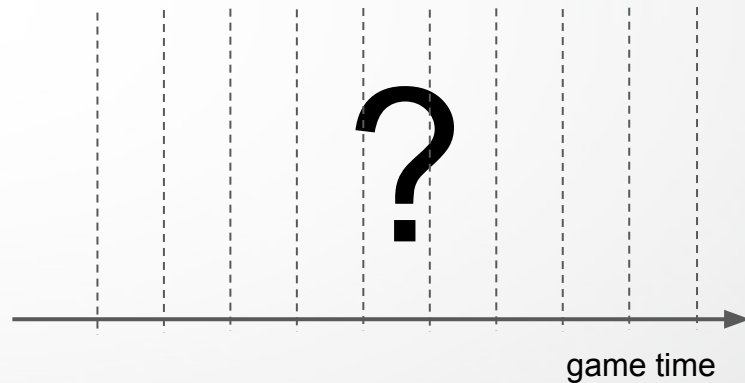
Bool

Value thresholding

State



to

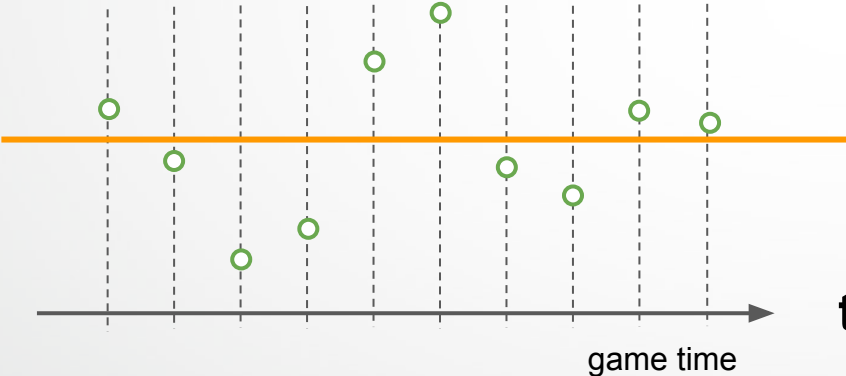


Float

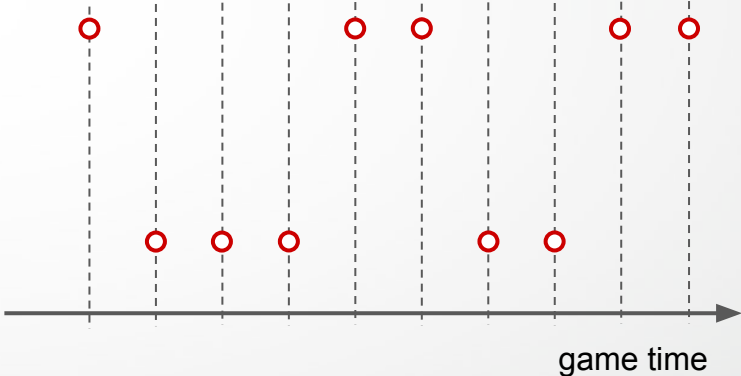
Bool

Value thresholding

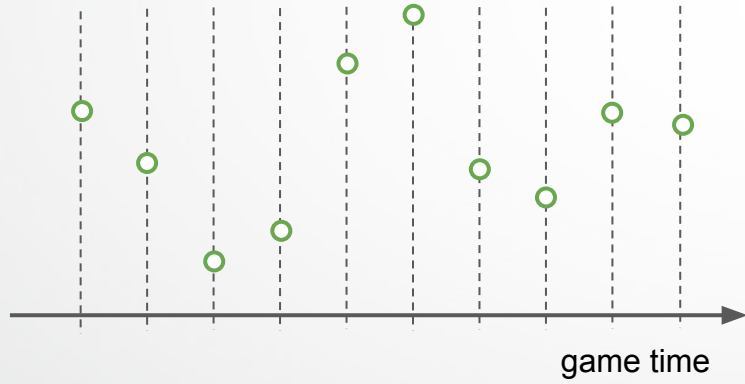
State  $\rightarrow$  State



to

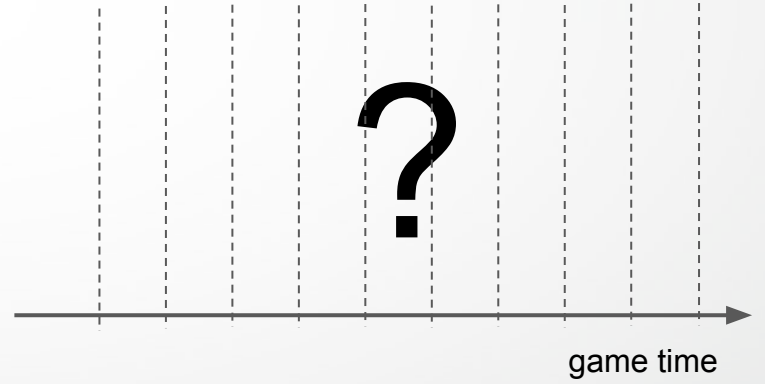


State



Float

to

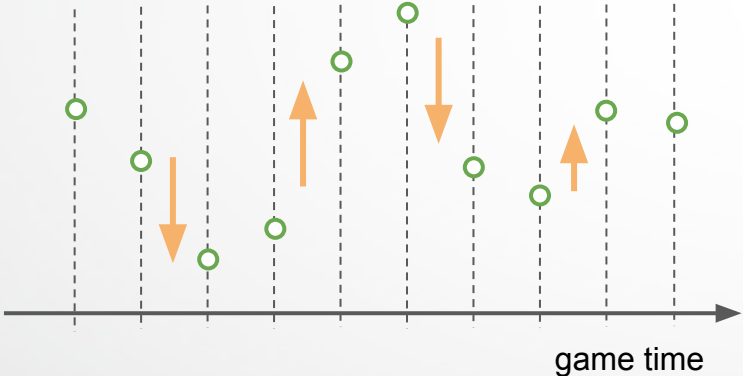


Bool



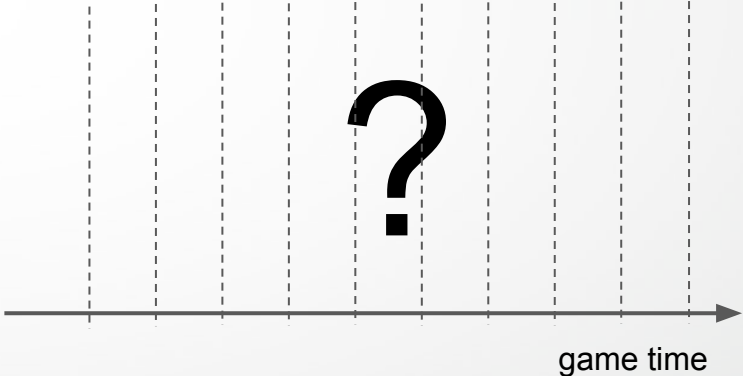
Gradient thresholding

State → Event



Float

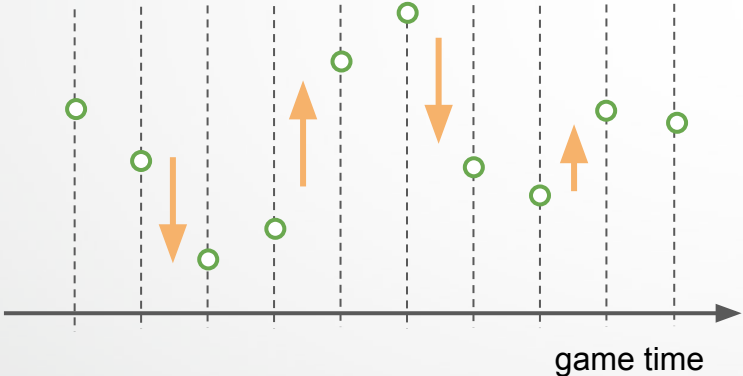
to



Bool

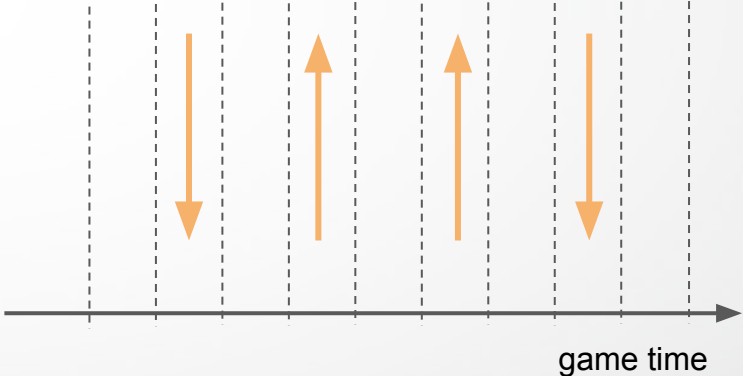
# Gradient thresholding

State → Event



Float

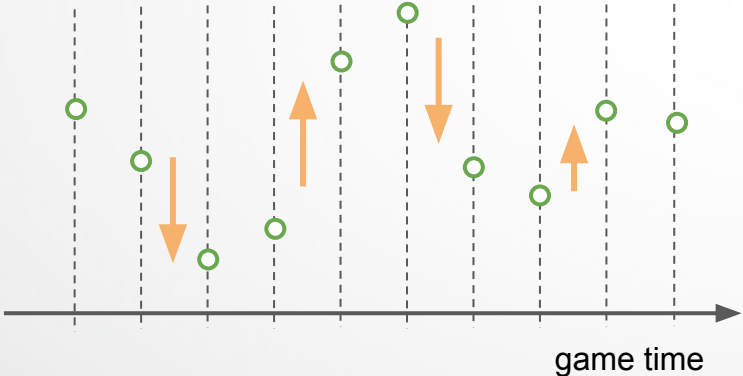
to



Bool

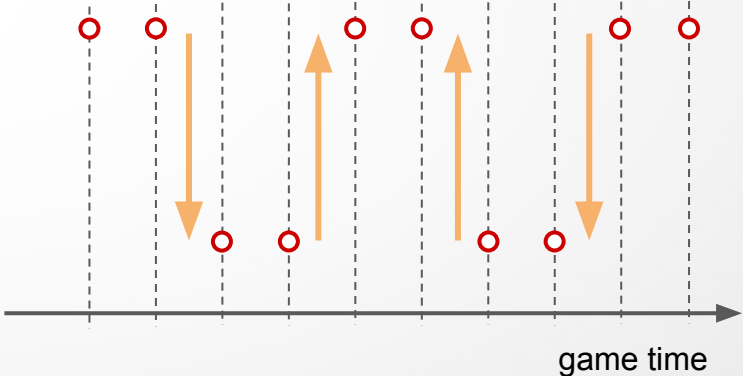
# Gradient thresholding

State → Event → State



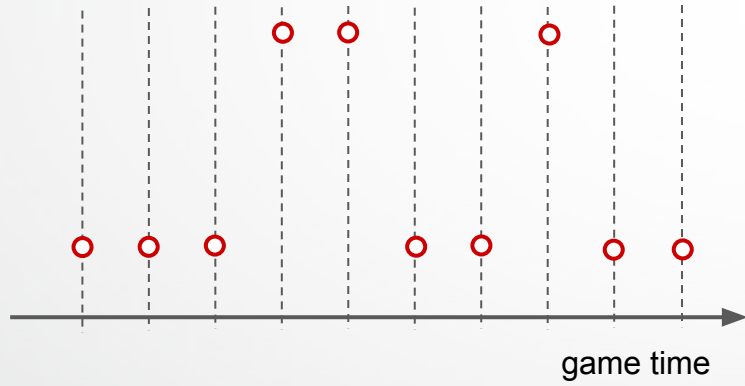
Float

to



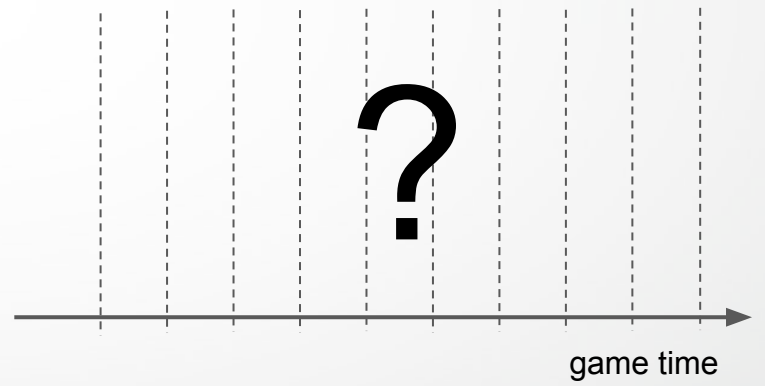
Bool

State



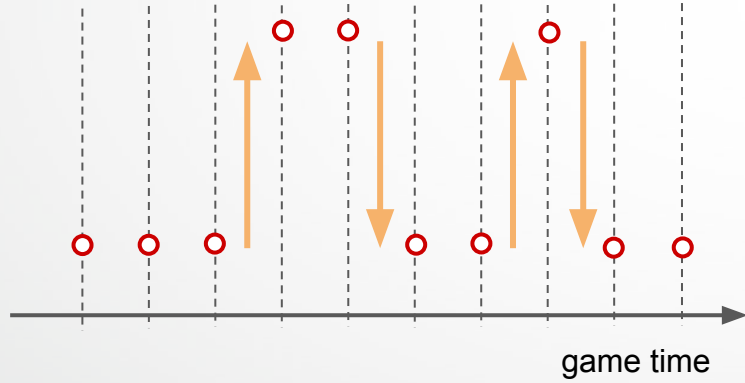
Bool

to



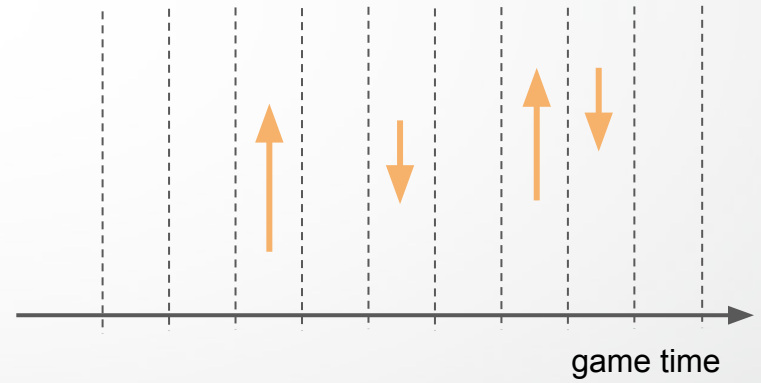
Float

State → Event



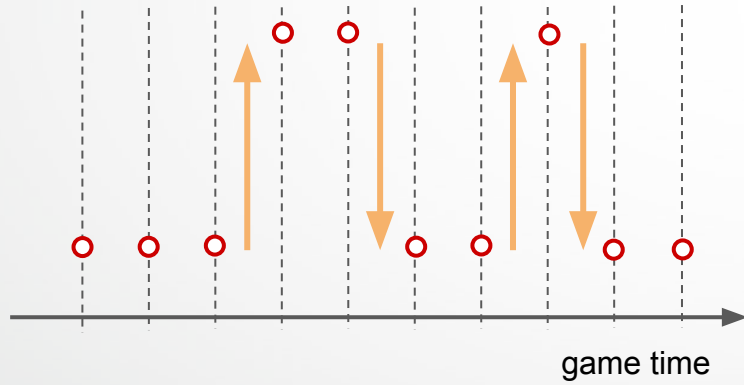
Bool

to



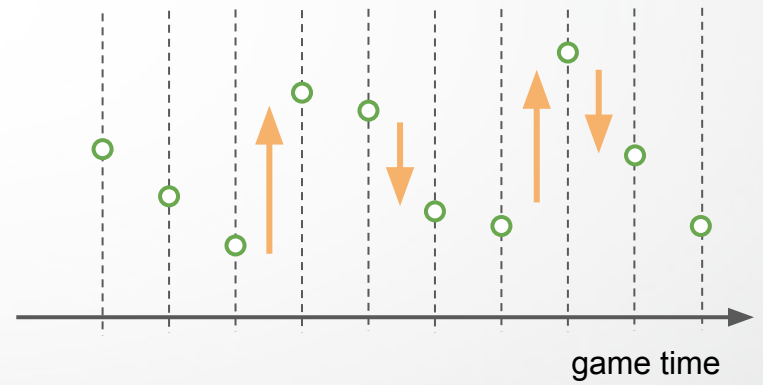
Float

State → Event → State



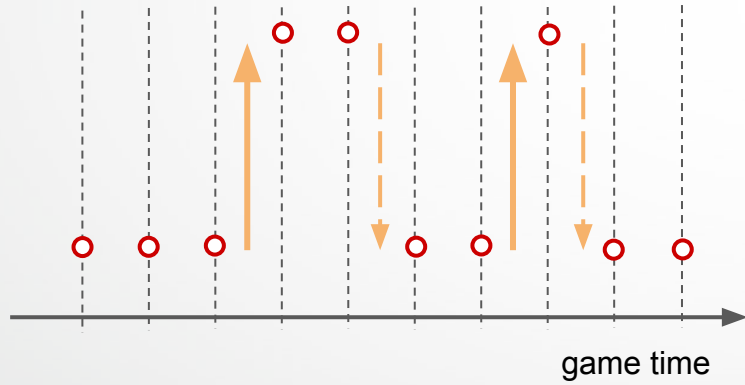
Bool

to



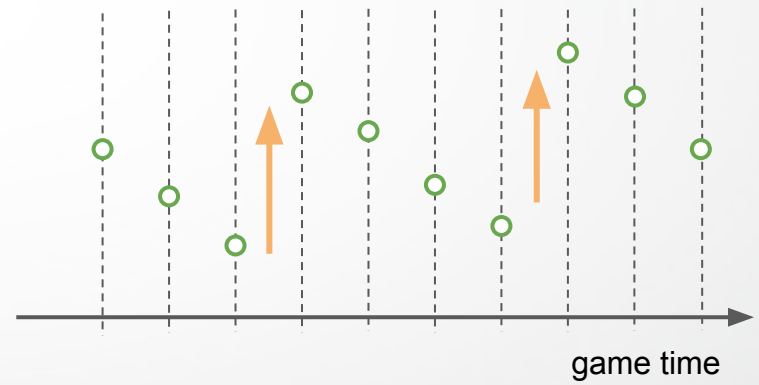
Float

State → Event → State



Bool

to



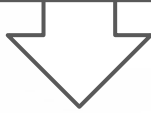
Float

Raw inputs

X-Axis,  
Button A,  
Button B



Raw inputs



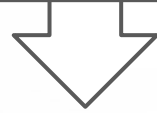
Semantic / Interpreted inputs

X-Axis,  
Button A,  
Button B

MoveHorizontal  
Attack  
Jump

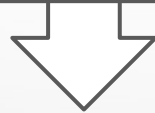
Raw inputs

X-Axis,  
Button A,  
Button B



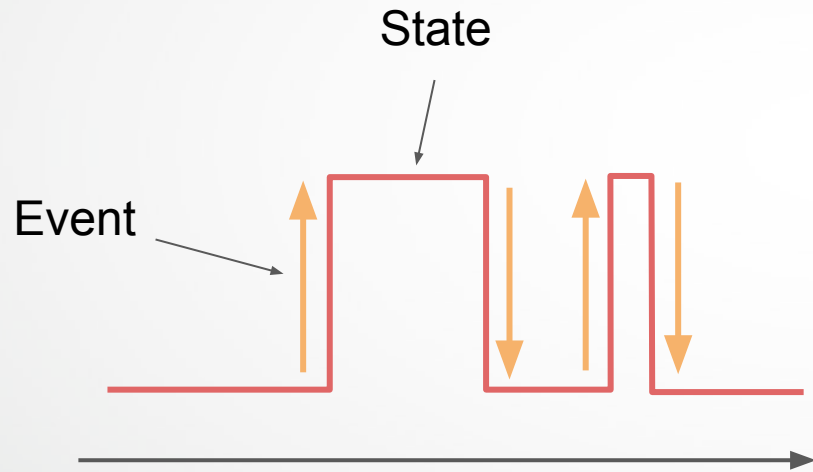
Semantic / Interpreted inputs

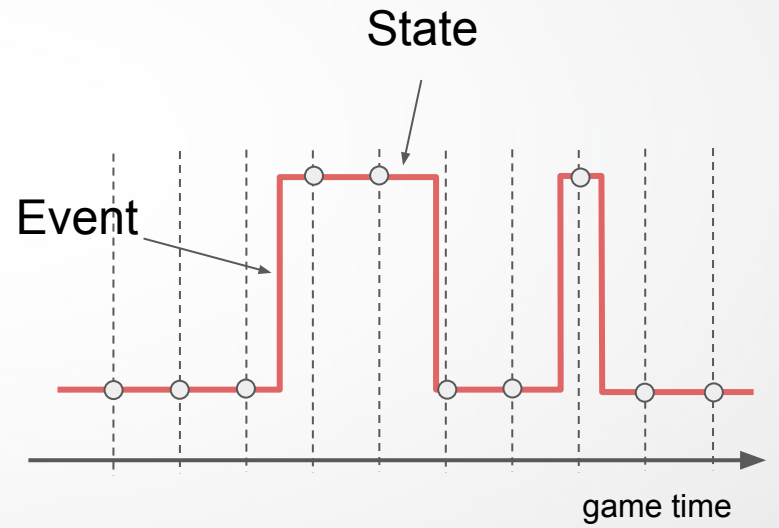
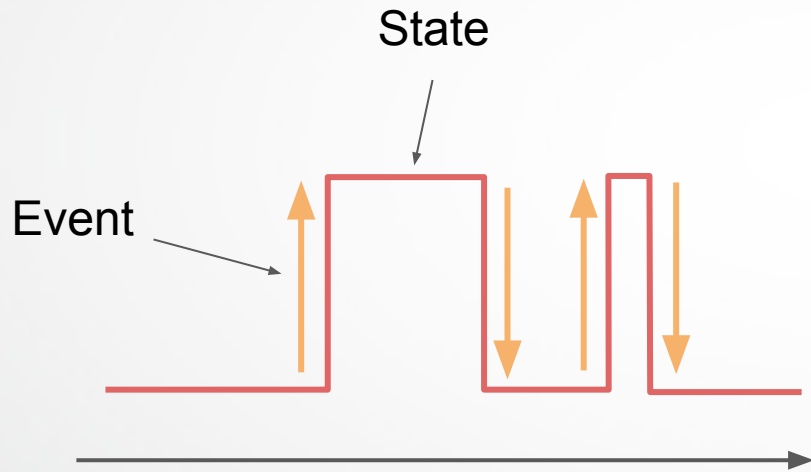
MoveHorizontal  
Attack  
Jump

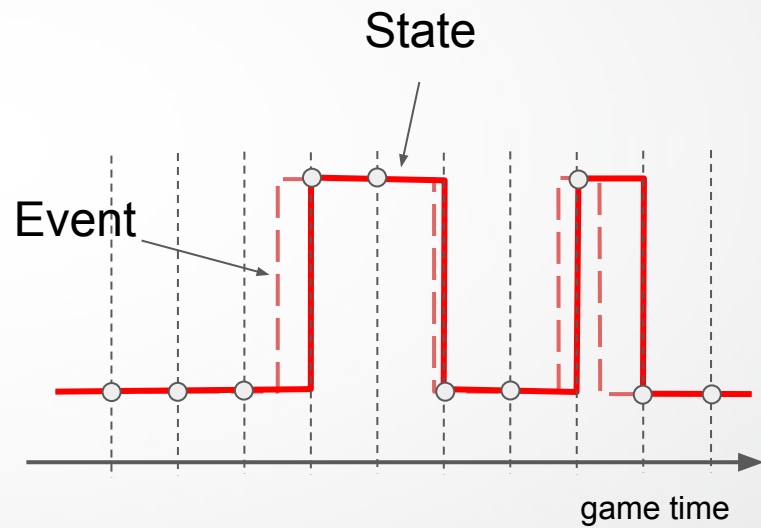
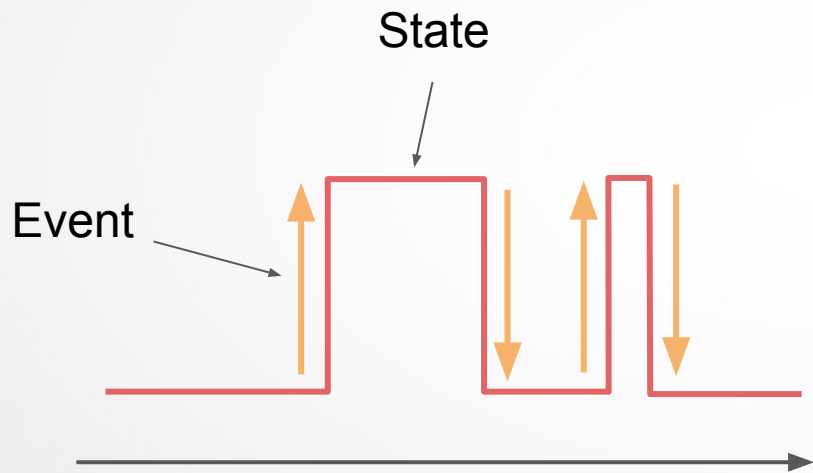


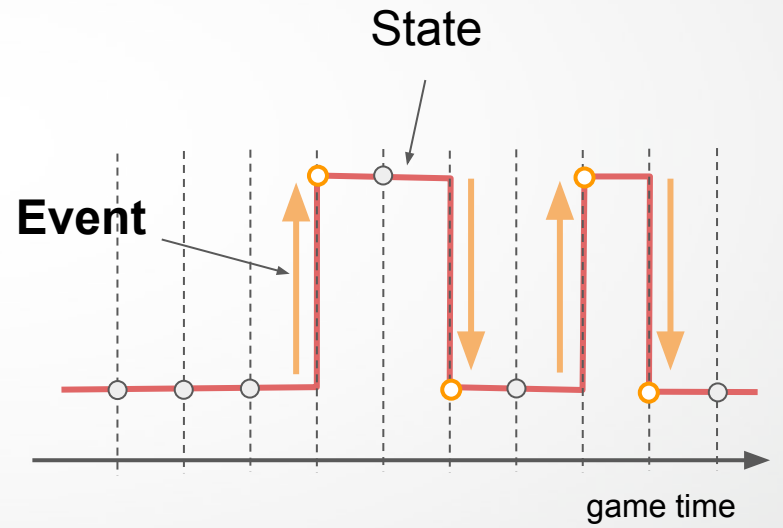
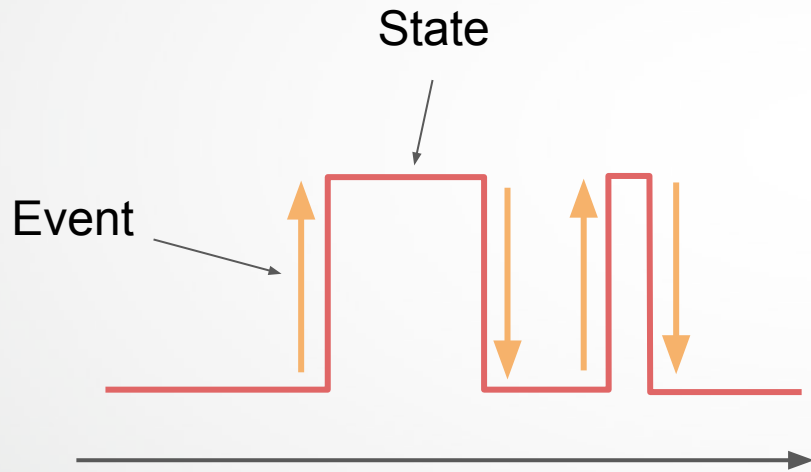
Game code

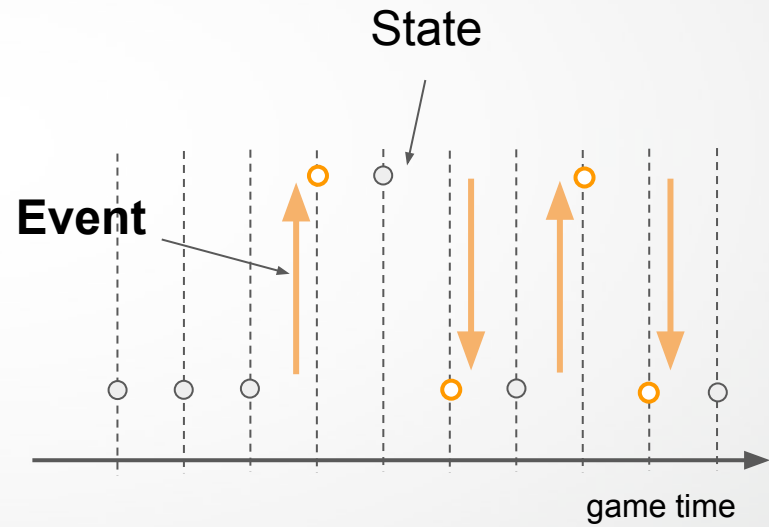
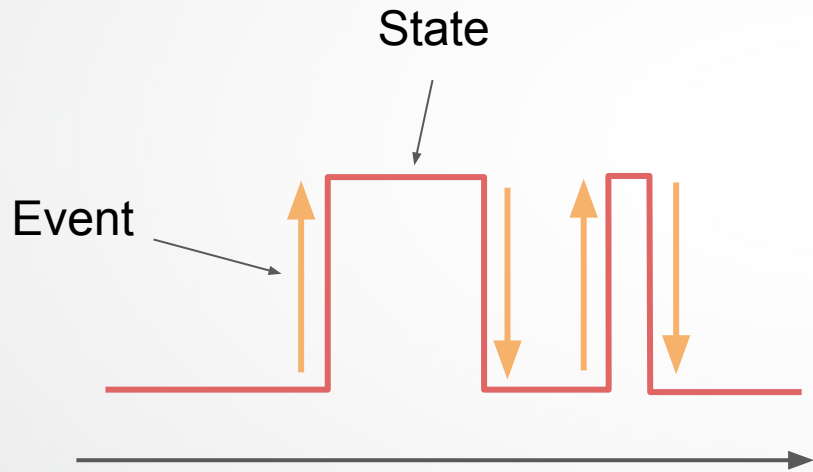
GetState("MoveHorizontal"),  
OnEvent("Attack")  
OnEvent("Jump")

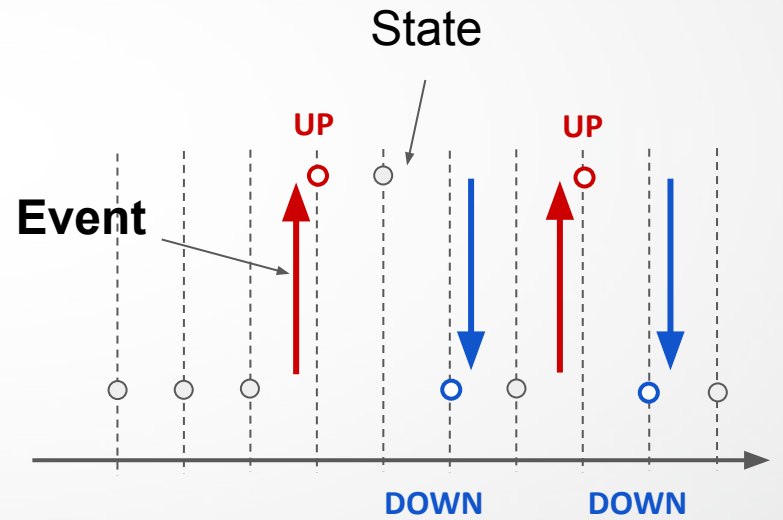
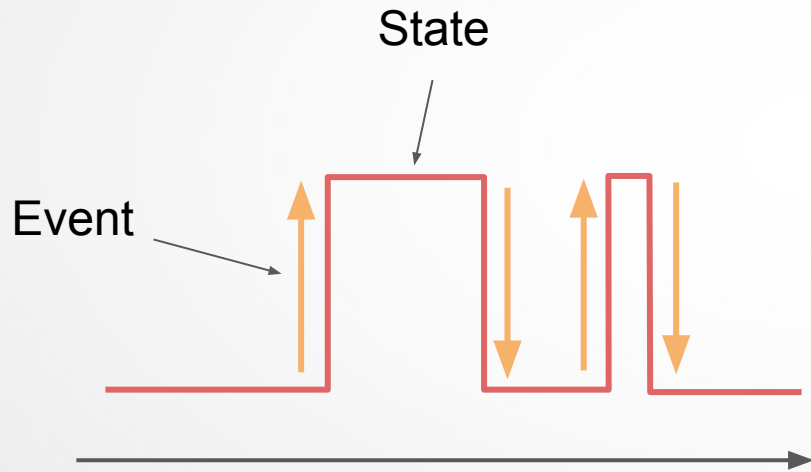




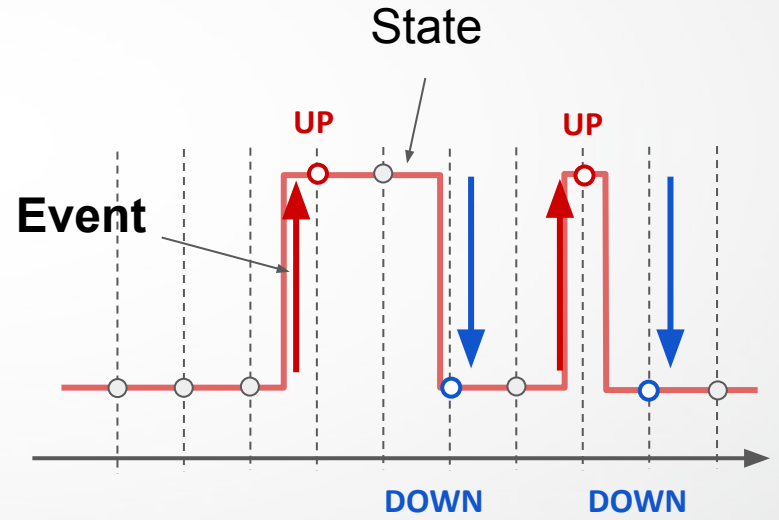
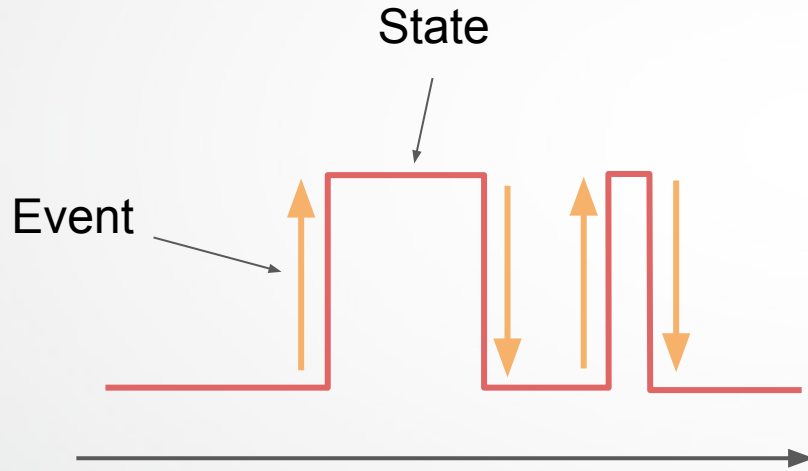














# UnityEngine.Input

<a href="#">GetAxis</a>	Returns the value of the virtual axis identified by axisName.
<a href="#">GetAxisRaw</a>	Returns the value of the virtual axis identified by axisName with no smoothing filtering applied.
<a href="#">GetButton</a>	Returns true while the virtual button identified by buttonName is held down.
<a href="#">GetButtonDown</a>	Returns true during the frame the user pressed down the virtual button identified by buttonName.
<a href="#">GetButtonUp</a>	Returns true the first frame the user releases the virtual button identified by buttonName.
<a href="#">GetJoystickNames</a>	Retrieves a list of input device names corresponding to the index of an Axis configured within Input Manager.
<a href="#">GetKey</a>	Returns true while the user holds down the key identified by name.
<a href="#">GetKeyDown</a>	Returns true during the frame the user starts pressing down the key identified by name.
<a href="#">GetKeyUp</a>	Returns true during the frame the user releases the key identified by name.
<a href="#">GetMouseButton</a>	Returns whether the given mouse button is held down.
<a href="#">GetMouseButtonDown</a>	Returns true during the frame the user pressed the given mouse button.
<a href="#">GetMouseButtonUp</a>	Returns true during the frame the user releases the given mouse button.



# UnityEngine.Input

<a href="#">GetAxis</a> <b>State</b>	Returns the value of the virtual axis identified by axisName.
<a href="#">GetAxisRaw</a>	Returns the value of the virtual axis identified by axisName with no smoothing filtering applied.
<a href="#">GetButton</a>	Returns true while the virtual button identified by buttonName is held down.
<a href="#">GetButtonDown</a>	Returns true during the frame the user pressed down the virtual button identified by buttonName.
<a href="#">GetButtonUp</a> <b>Event</b>	Returns true the first frame the user releases the virtual button identified by buttonName.
<a href="#">GetJoystickNames</a>	Retrieves a list of input device names corresponding to the index of an Axis configured within Input Manager.
<a href="#">GetKey</a>	Returns true while the user holds down the key identified by name.
<a href="#">GetKeyDown</a>	Returns true during the frame the user starts pressing down the key identified by name.
<a href="#">GetKeyUp</a>	Returns true during the frame the user releases the key identified by name.
<a href="#">GetMouseButton</a>	Returns whether the given mouse button is held down.
<a href="#">GetMouseButtonDown</a>	Returns true during the frame the user pressed the given mouse button.
<a href="#">GetMouseButtonUp</a>	Returns true during the frame the user releases the given mouse button.



# UnityEngine.Input

<u>GetAxis</u> <b>State</b>	Returns the value of the virtual axis identified by axisName.
<u>GetAxisRaw</u>	Returns the value of the virtual axis identified by axisName with no smoothing filtering applied.
<u>GetButton</u>	Returns true while the virtual button identified by buttonName is held down.
<u>GetButtonDown</u>	Returns true during the frame the user pressed down the virtual button identified by buttonName.
<u>GetButtonUp</u> <b>Event</b>	Returns true the first frame the user releases the virtual button identified by buttonName.
<u>GetJoystickNames</u>	Retrieves a list of input device names corresponding to the index of an Axis configured within Input Manager.
<u>GetKey</u>	Returns true while the user holds down the key identified by name.
<u>GetKeyDown</u>	Returns true during the frame the user starts pressing down the key identified by name.
<u>GetKeyUp</u>	Returns true during the frame the user releases the key identified by name.
<u>GetMouseButton</u>	Returns whether the mouse button is currently pressed.
<u>GetMouseButtonDown</u>	Returns true during the frame the user pressed the given mouse button.
<u>GetMouseButtonUp</u>	Returns true during the frame the user releases the given mouse button.

Event triggered in FixedUpdate() or Update() ?



# UnityEngine.Input

<u>GetAxis</u> <b>State</b>	Returns the value of the virtual axis identified by axisName.
<u>GetAxisRaw</u>	Returns the value of the virtual axis identified by axisName with no smoothing filtering applied.
<u>GetButton</u>	Returns true while the virtual button identified by buttonName is held down.
<u>GetButtonDown</u>	Returns true during the frame the user pressed down the virtual button identified by buttonName.
<u>GetButtonUp</u> <b>Event</b>	Returns true the first frame the user releases the virtual button identified by buttonName.
<u>GetJoystickNames</u>	Retrieves a list of input device names corresponding to the index of an Axis configured within Input Manager.
<u>GetKey</u>	Returns true while the user holds down the key identified by name.
<u>GetKeyDown</u>	Returns true during the frame the user starts pressing down the key identified by name.
<u>GetKeyUp</u>	Returns true during the frame the user releases the key identified by name.
<u>GetMouseButton</u>	Returns whether
<u>GetMouseButtonDown</u>	Returns true during the frame the user pressed the given mouse button.
<u>GetMouseButtonUp</u>	Returns true during the frame the user releases the given mouse button.

Event triggered in Update()



# Input System Package (in Dev)

Project Settings

**Input System Package**

Update Mode: Process Events In Dynamic Update

Timeslice Events

Filter Noise On Current

Compensate For Screen Or

Default Deadzone Min: 0.125

Default Deadzone Max: 0.925

Default Button Press Point: 0.5

Default Tap Time: 0.2

Default Slow Tap Time: 0.5

Default Hold Time: 0.4

Tap Radius: 5

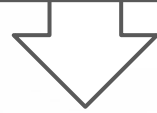
Multi Tap Delay Time: 0.75

Leave 'Supported Devices' empty if you want the input system to support all input devices it can recognize. If, however, you are only interested in a certain set of devices, adding them here will narrow the scope of what's presented in the editor and avoid picking up input from devices not relevant to the project.

Supported Devices

Raw inputs

X-Axis,  
Button A,  
Button B



**Semantic / Interpreted inputs**

MoveHorizontal  
Attack  
Jump



Game code

GetState("MoveHorizontal"),  
OnEvent("Attack")  
OnEvent("Jump")



# Project Settings > InputManager (Legacy)

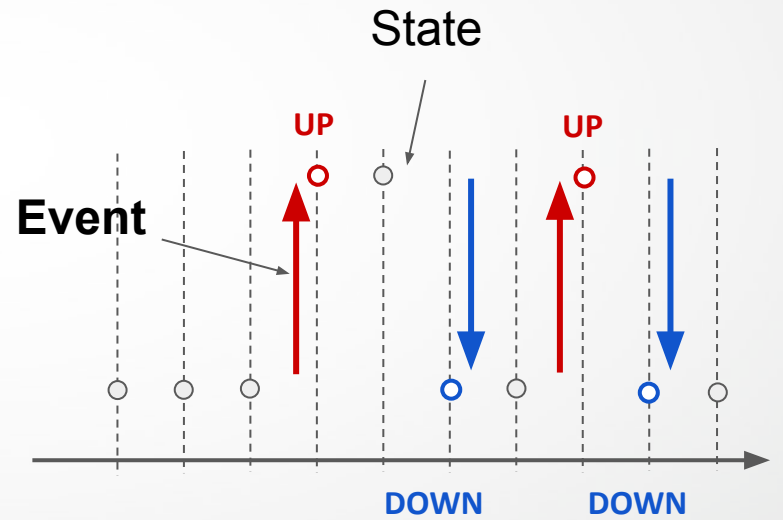
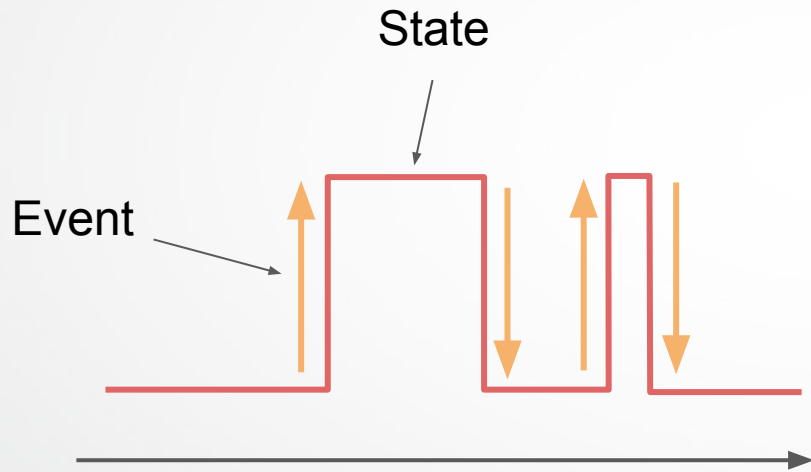
The screenshot shows the Unity Project Settings window for the Input Manager (Legacy). The left sidebar lists various settings categories, with 'Input Manager' selected. The main panel displays the configuration for the 'Horizontal' axis, which is highlighted with a red box. A warning message at the top suggests using the new Input System Package instead.

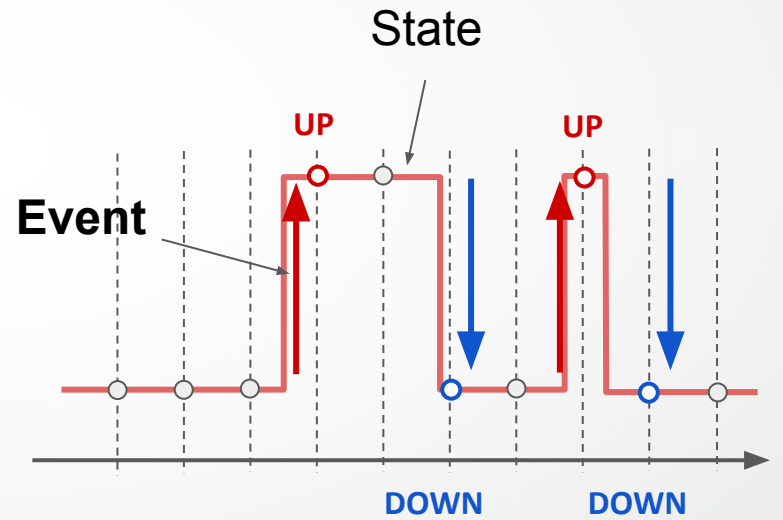
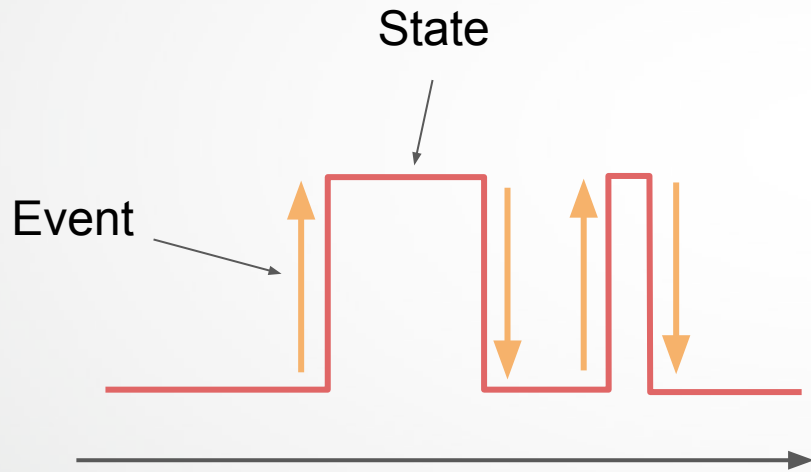
**Input Manager**

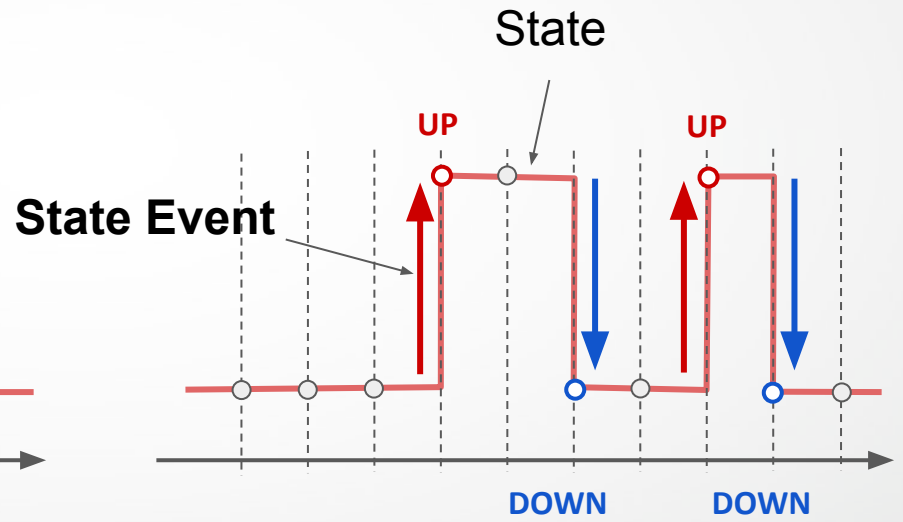
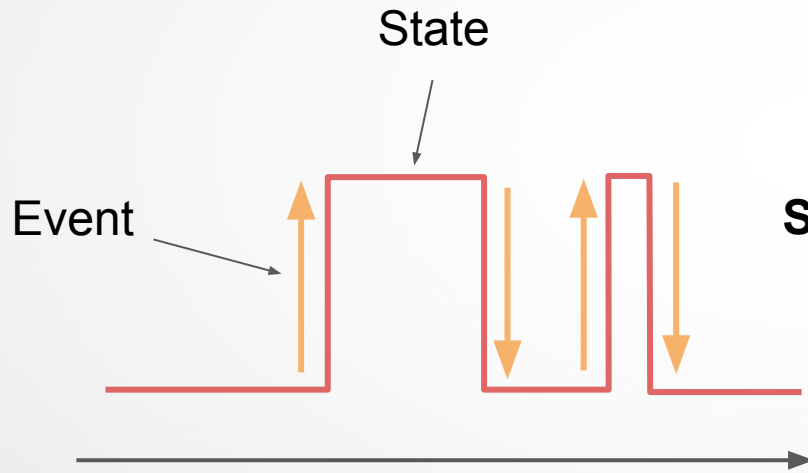
! This is where you can configure the controls to use with the UnityEngine.Input API. Consider using the new Input System Package instead.

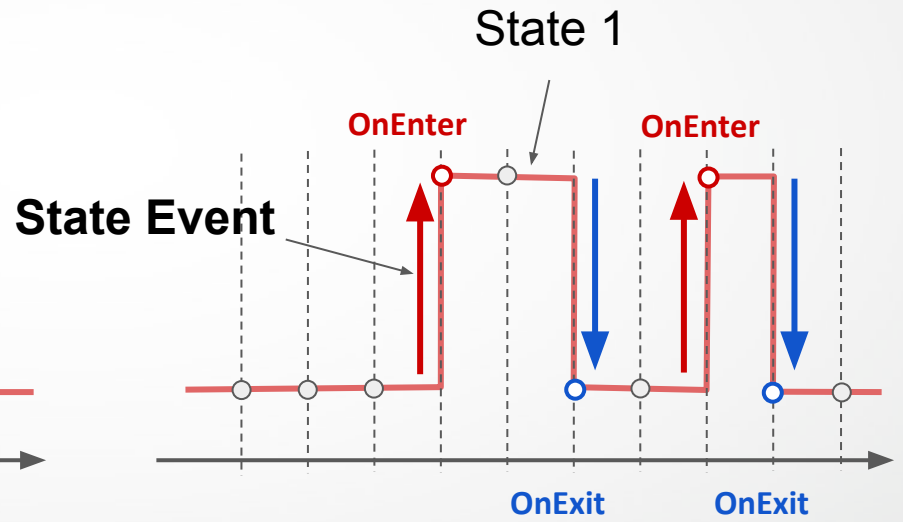
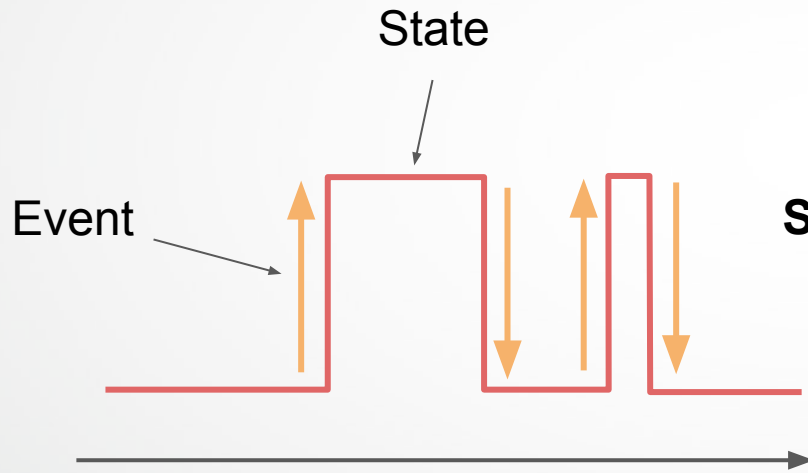
- ▼ Axes
  - Size: 23
  - ▼ Horizontal
    - Name: **Horizontal**
    - Descriptive Name:
    - Descriptive Negative Name:
    - Negative Button: left
    - Positive Button: right
    - Alt Negative Button: a
    - Alt Positive Button: d
    - Gravity: 1000
    - Dead: 0.001
    - Sensitivity: 1000
    - Snap:
    - Invert:
    - Type: Key or Mouse Button
    - Axis: X axis
    - Joy Num: Get Motion from all Joysticks
  - ▶ Vertical
    - ▶ Fire1
    - ▶ Fire2
    - ▶ Fire3
    - ▶ Jump

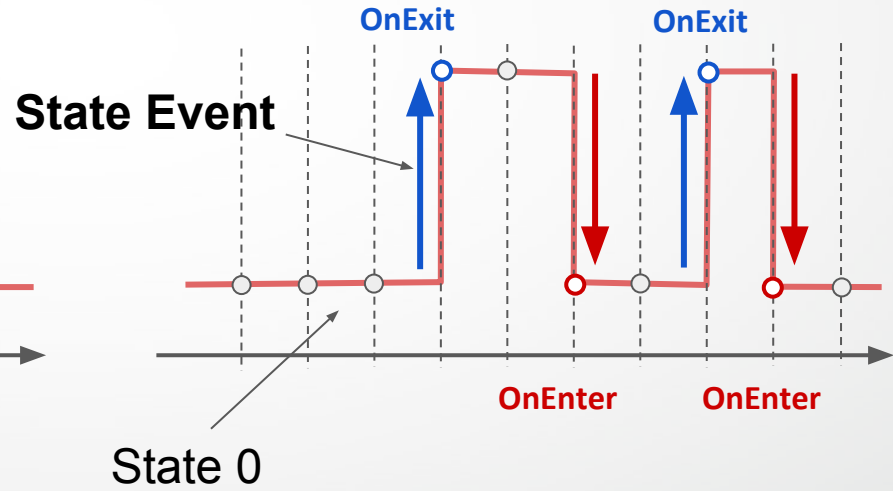
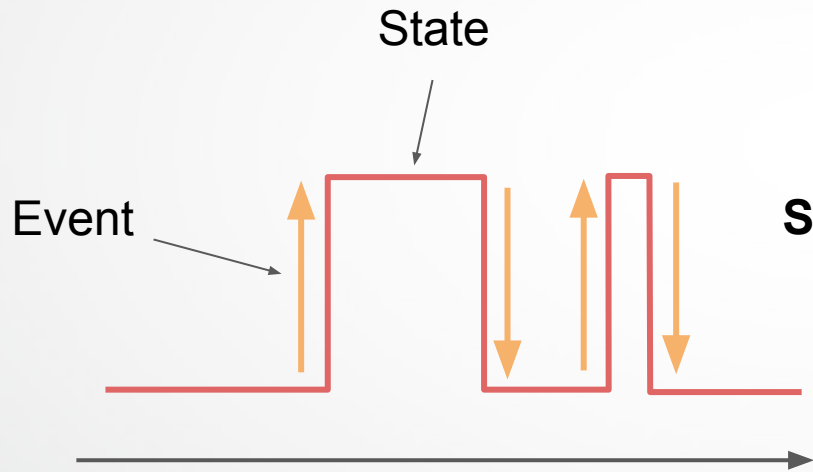






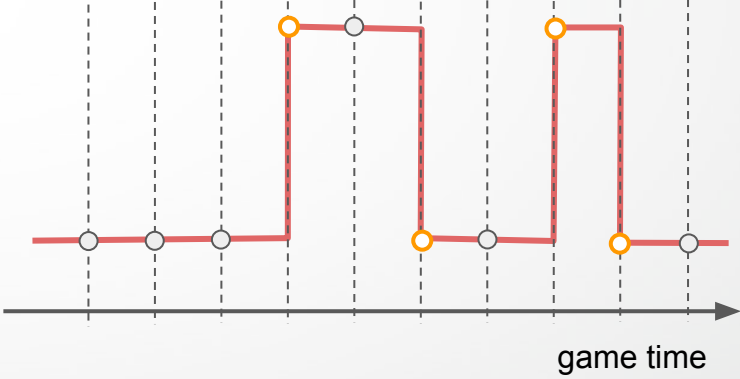


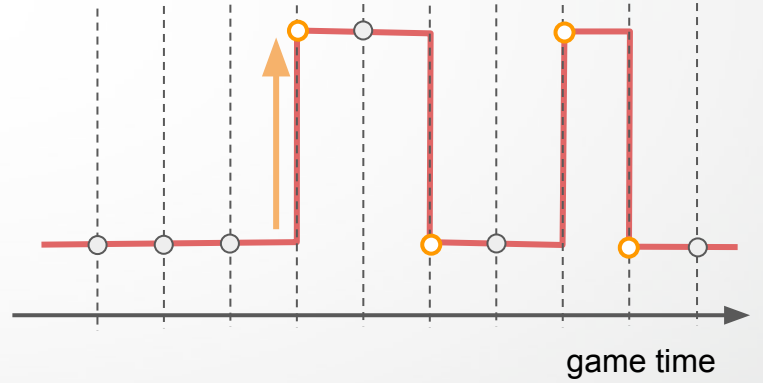
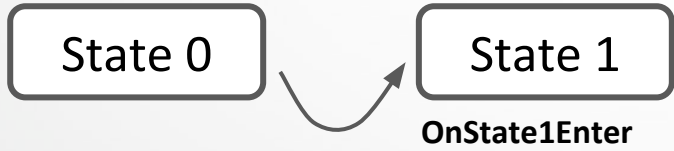


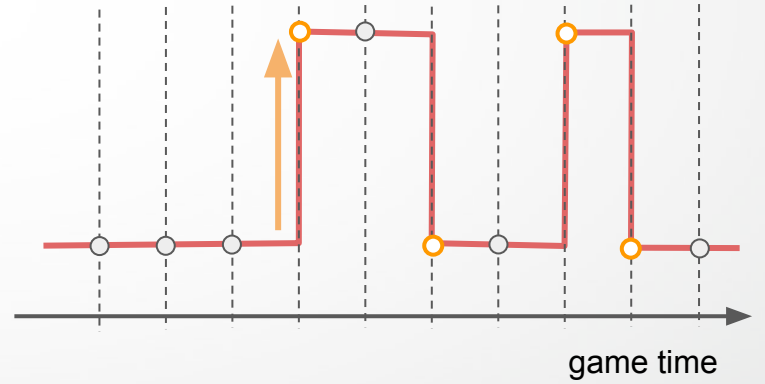
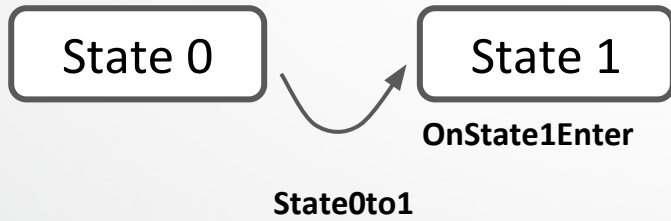


State 0

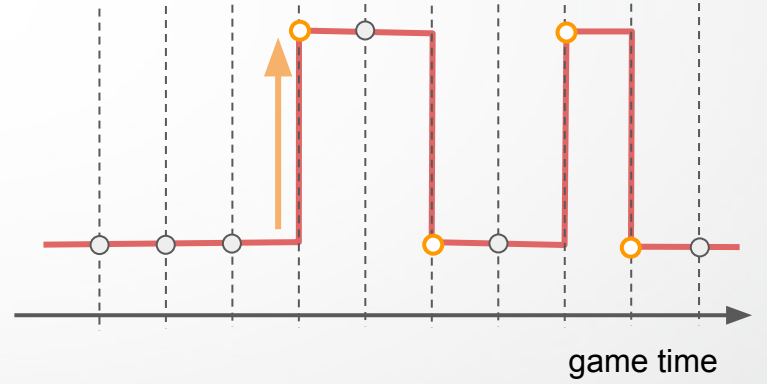
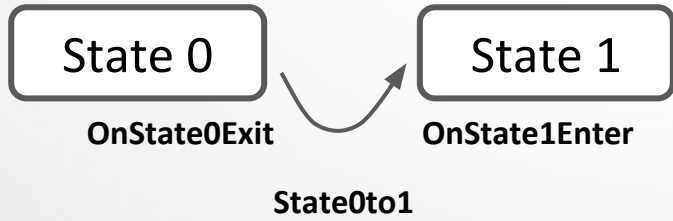
State 1



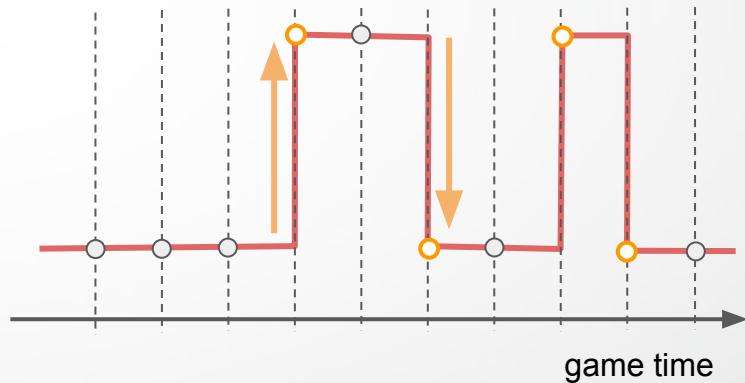
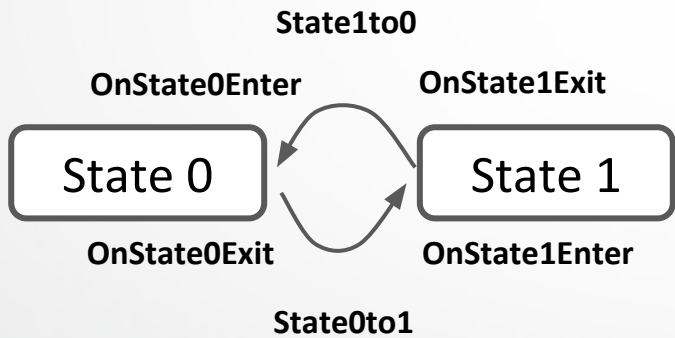












How many button states ?



# How many button states ?



Normal

Pressed

# How many button states ?



Normal

`onMouseDown`

Pressed

`onMouseEnter`

`onMouseOver`

# How many button states ?



Normal

`onMouseDown`

`onMouseUp ?`

Pressed

`onMouseEnter`

`onMouseOver`

# How many button states ?



Normal

`onMouseDown`

`onMouseUp`

Pressed

`onMouseEnter`

Highlighted

`onMouseExit`

`onMouseDown`

`onMouseOver`



# How many button states ?



Normal

`onMouseDown`

`onMouseUp`

Pressed

`onMouseEnter`

Highlighted

`onMouseExit`

Disabled

`onMouseDown`

`onMouseOver`

# How many button states ?



Normal

`onMouseDown`

`onMouseUp`

Pressed

`onMouseEnter`

Highlighted

`onMouseExit`

Disabled

`onMouseDown`

`onMouseOver`

Clicked ?

# How many button states ?



Normal

`onMouseDown`

`onMouseUp`

`onMouseUpAsButton`

Pressed

`onMouseEnter`

Highlighted

`onMouseExit`

Disabled

`onMouseDown`

`onMouseOver`

Clicked ?

# How many button states ?



Normal

`onMouseDown`

`onMouseUp`

`onMouseUpAsButton`

Pressed

`onMouseEnter`

Highlighted

`onMouseExit`

Disabled

`onMouseDown`

`onMouseOver`

Clicked ?

# How many button states ?



Normal

`onMouseDown`

`onMouseUp`

`onMouseUpAsButton`

Pressed

`onMouseEnter`

Highlighted

`onMouseExit`

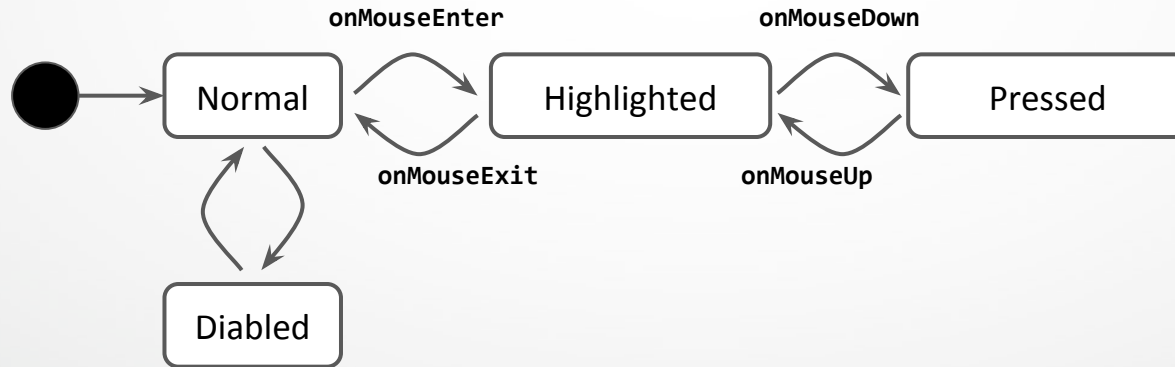
Disabled

`onMouseDown`

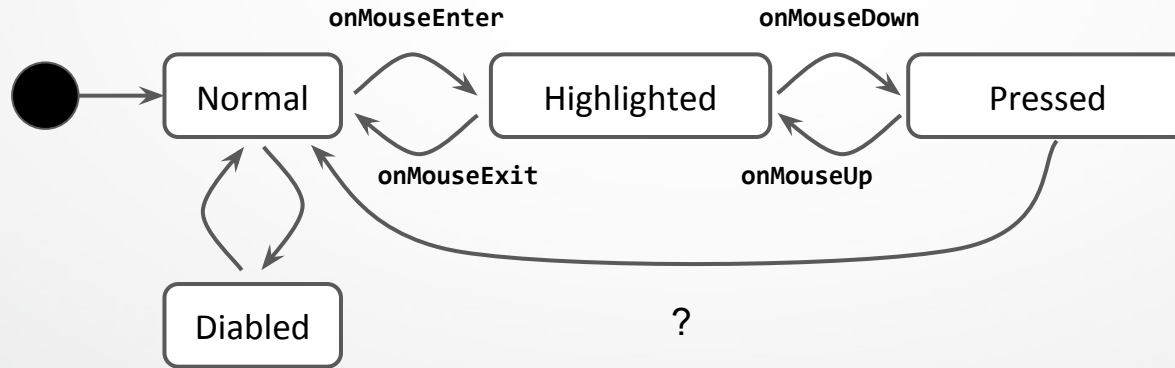
`onMouseOver`

Clicked ?

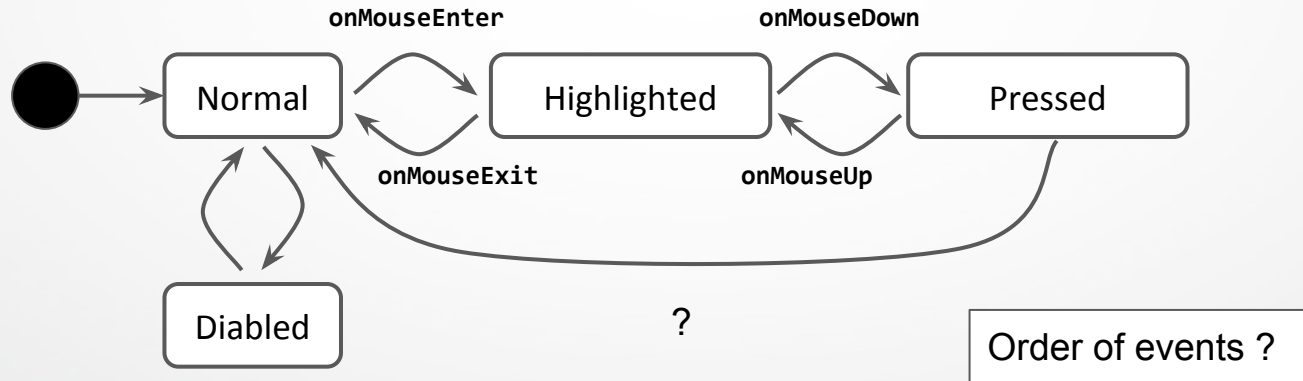
# Finite-state machine



# Finite-state machine



# Finite-state machine





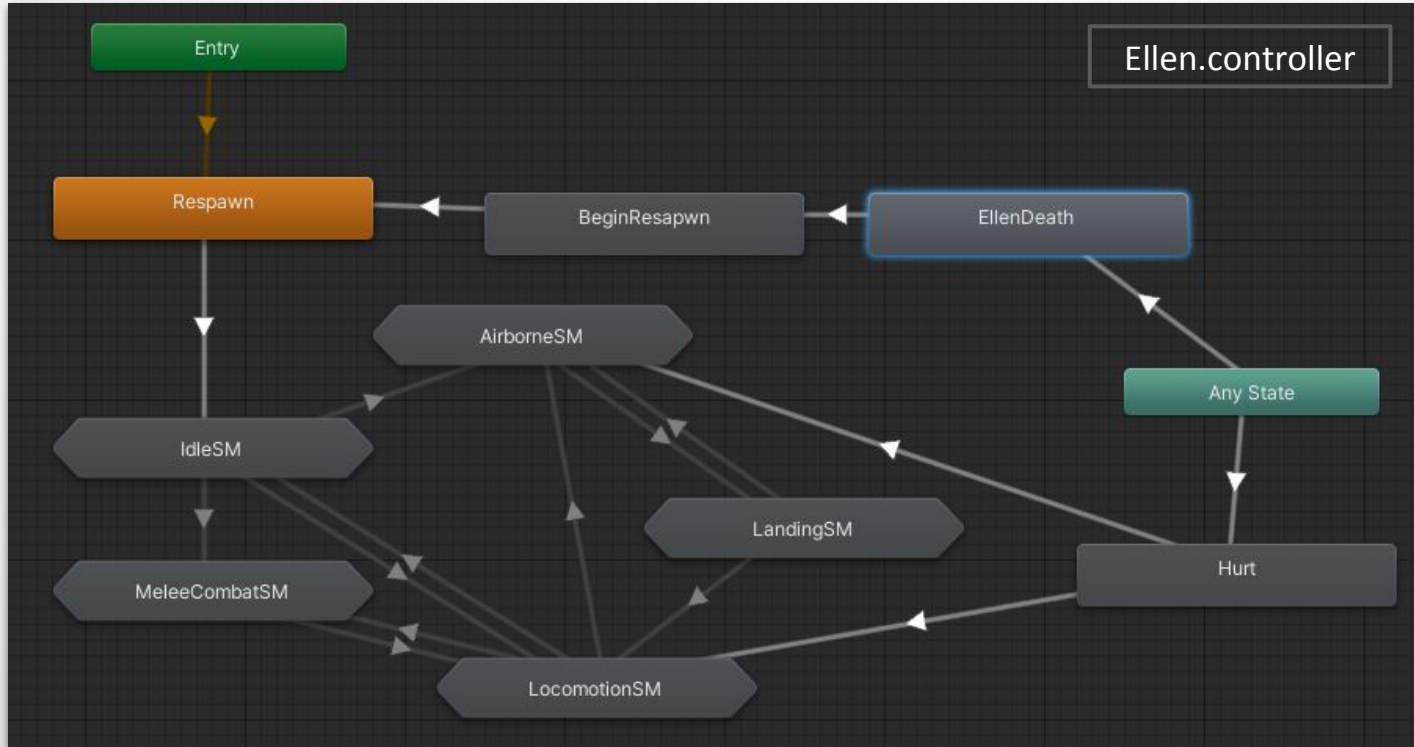


 unity Learn

# 3D Game Kit



# Animator Controller





# Animator Parameters

The screenshot displays the Unity Animator interface. On the left, a list of parameters is shown with their current values:

Parameter Name	Value
FootFall	0.0
VerticalSpeed	0.0
AirborneVerticalSpeed	0.0
ForwardSpeed	0.0
AngleDeltaRad	0.0
RandomIdle	0
TimeoutToldle	<input type="radio"/>
Grounded	<input type="checkbox"/>
InputDetected	<input type="checkbox"/>
MeleeAttack	<input type="radio"/>
Hurt	<input type="radio"/>
Death	<input type="radio"/>
Respawn	<input type="radio"/>

The main graph area shows a state machine with the following elements:

- Base Layer**: Contains an **Entry** state (green box).
- Transition**: A yellow arrow points from **Entry** to the **respawn** state.
- respawn** state (orange box): A transition arrow points to the **IdleSM** state.
- BeginResapwn** state (grey box): A transition arrow points to the **respawn** state.
- IdleSM** state (grey arrow): A transition arrow points to the **AirborneSM** state.
- AirborneSM** state (grey arrow): Multiple transition arrows point to other states, including **IdleSM**.



# Animator Parameters

The screenshot shows the Unity Animator Parameters window. The 'Parameters' tab is active, displaying a list of parameters under the 'Base Layer'. A green box highlights the following parameters:

Name	Value
FootFall	0.0
VerticalSpeed	0.0
AirborneVerticalSpeed	0.0
ForwardSpeed	0.0
AngleDeltaRad	0.0
RandomIdle	0

The state machine diagram shows the following states and transitions:

- Entry** (green box) transitions to **Respawn** (orange box).
- Respawn** transitions to **IdleSM** (grey box).
- Respawn** transitions to **AirborneSM** (grey box).
- BeginRespawn** (grey box) transitions to **Respawn**.
- IdleSM** transitions to **AirborneSM**.
- AirborneSM** transitions to **Respawn**.



# Animator Parameters

The screenshot shows the Unity Animator Parameters window. The left pane lists parameters for the 'Base Layer'. The right pane shows a state machine diagram with states like 'Entry', 'Respawn', 'AirborneSM', and 'IdleSM'.

Name	Value
FootFall	0.0
VerticalSpeed	0.0
AirborneVerticalSpeed	0.0
ForwardSpeed	0.0
AngleDeltaRad	0.0
RandomIdle	0
TimeoutToldle	<input type="radio"/>
Grounded	<input type="checkbox"/>
InputDetected	<input type="checkbox"/>
MeleeAttack	<input type="radio"/>
Hurt	<input type="radio"/>
Death	<input type="radio"/>
Respawn	<input type="radio"/>



# Animator Parameters

The screenshot shows the Unity Animator Parameters window. The left pane lists parameters for the 'Base Layer'. The right pane shows a state machine diagram with states like 'Entry', 'Respawn', 'IdleSM', and 'AirborneSM'.

Name	Value
FootFall	0.0
VerticalSpeed	0.0
AirborneVerticalSpeed	0.0
ForwardSpeed	0.0
AngleDeltaRad	0.0
RandomIdle	0
TimeoutToldle	<input type="checkbox"/>
Grounded	<input type="checkbox"/>
InputDetected	<input type="checkbox"/>
MeleeAttack	<input type="checkbox"/>
Hurt	<input type="checkbox"/>
Death	<input type="checkbox"/>
Respawn	<input type="checkbox"/>

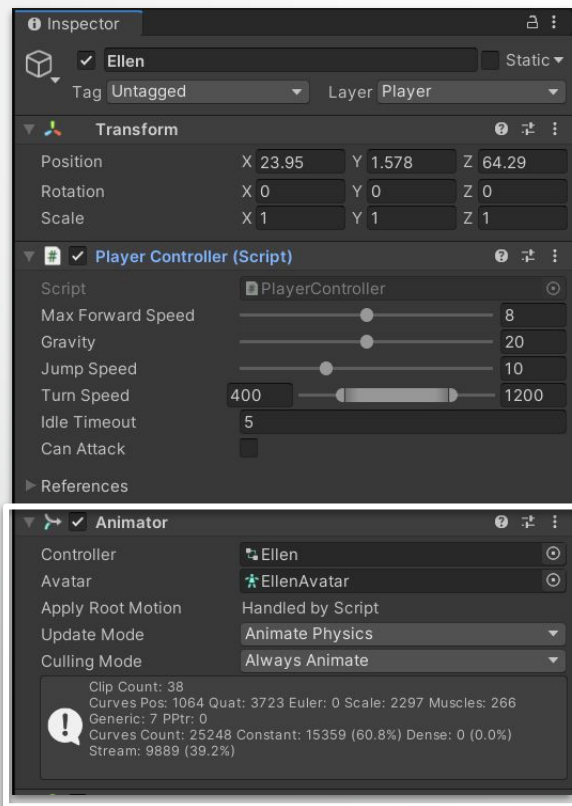


# UnityEngine.Animator

<a href="#"><u>SetBool</u></a>	Sets the value of the given boolean parameter.
<a href="#"><u>SetFloat</u></a>	Send float values to the Animator to affect transitions.
<a href="#"><u>SetInteger</u></a>	Sets the value of the given integer parameter.
<a href="#"><u>SetTrigger</u></a>	Sets the value of the given trigger parameter.



# Control the animator with your custom script

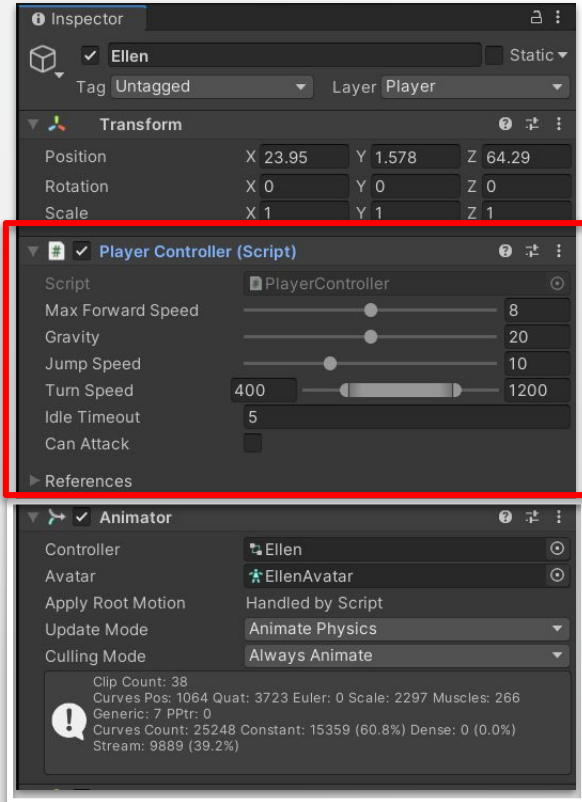


Animator





# Control the animator with your custom script

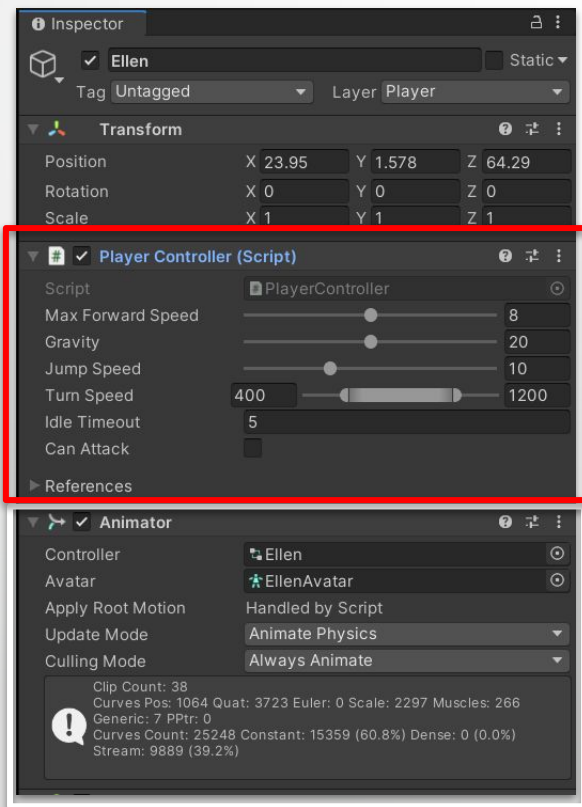


Your custom script

Animator



# Control the animator with your custom script



Your custom script

Animator

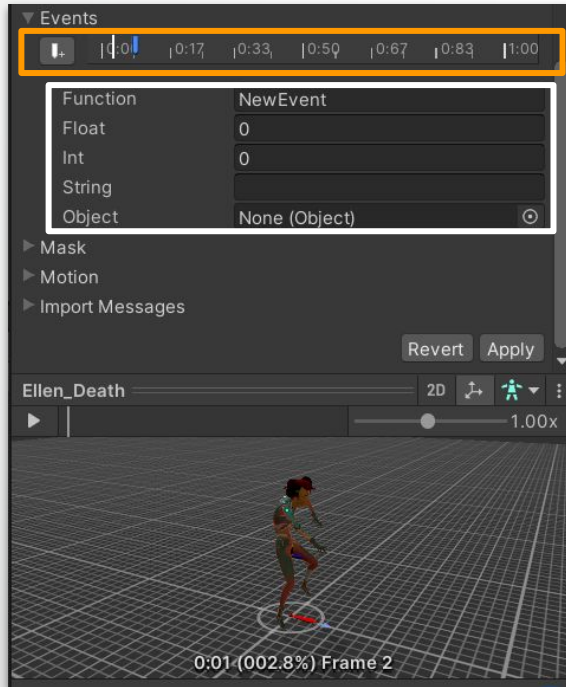
Update animator parameters





# Animation event

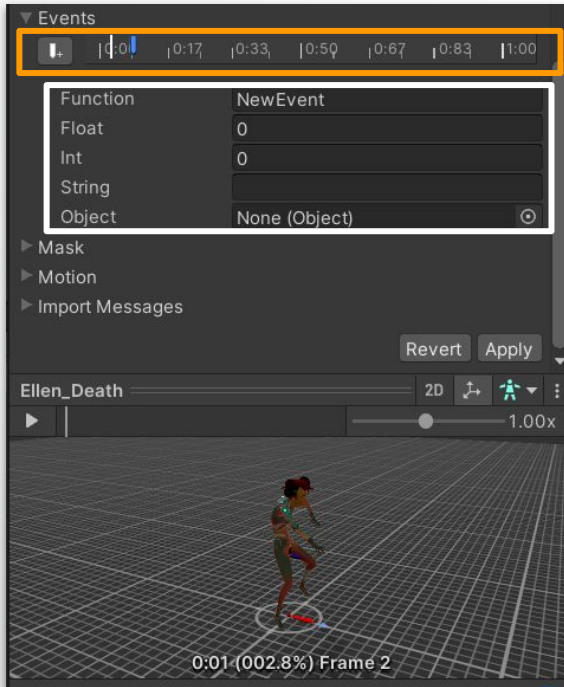
FBX Importer Settings (Inspector) (.fbx)



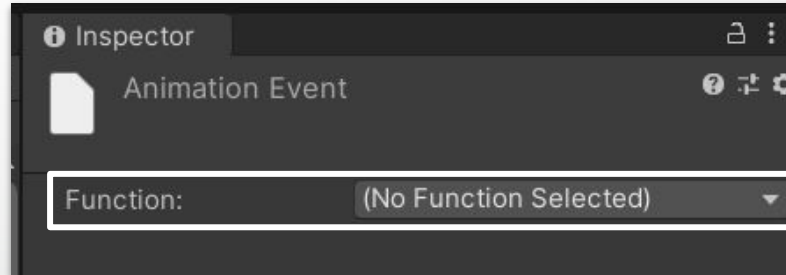
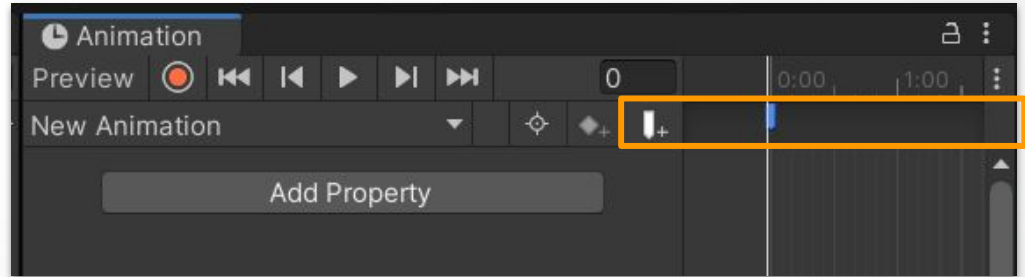


# Animation event

FBX Importer Settings (Inspector) (.fbx)

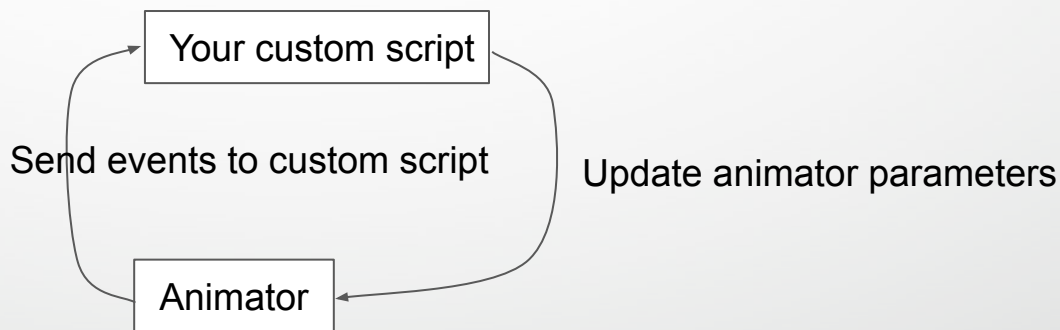
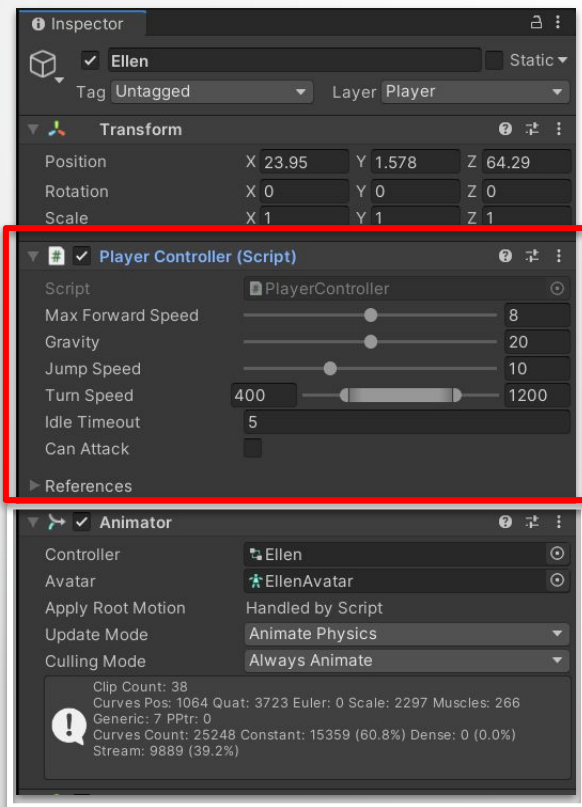


AnimationClip (Animation Window) (.anim)



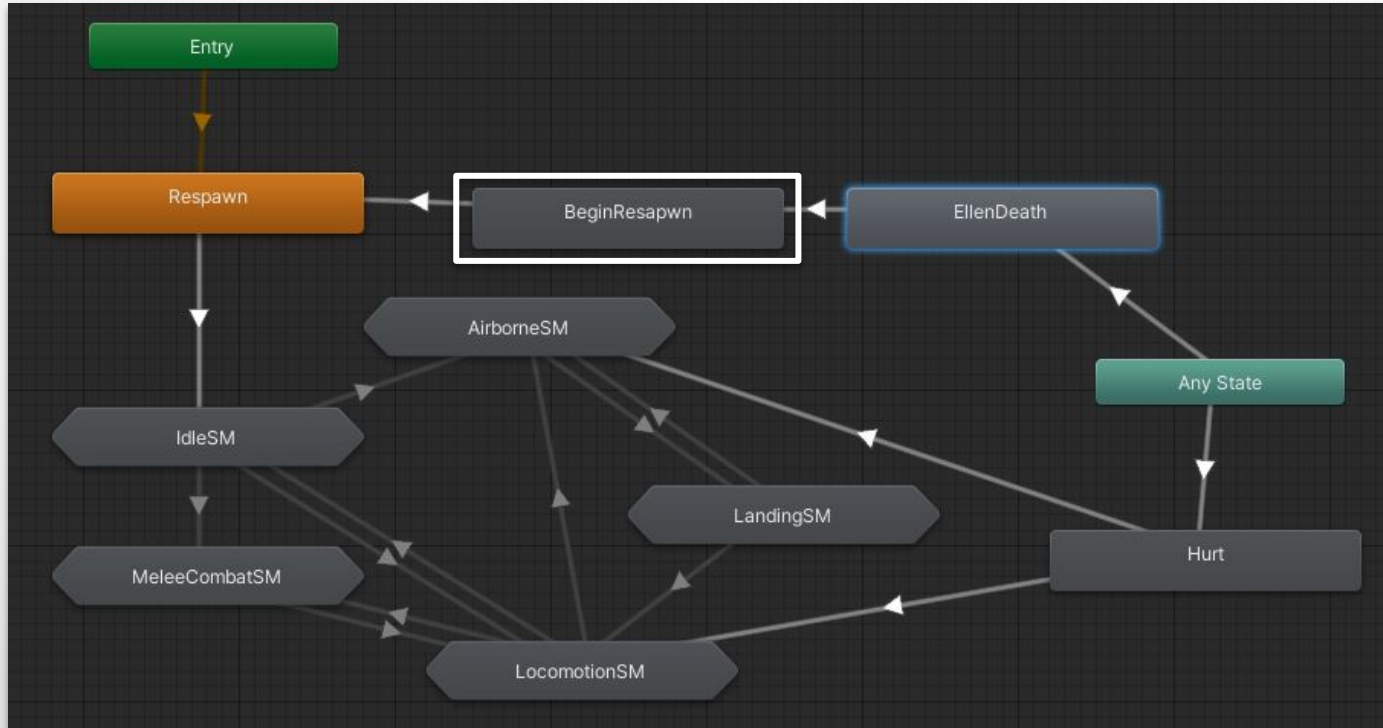


# Send animation events to custom script



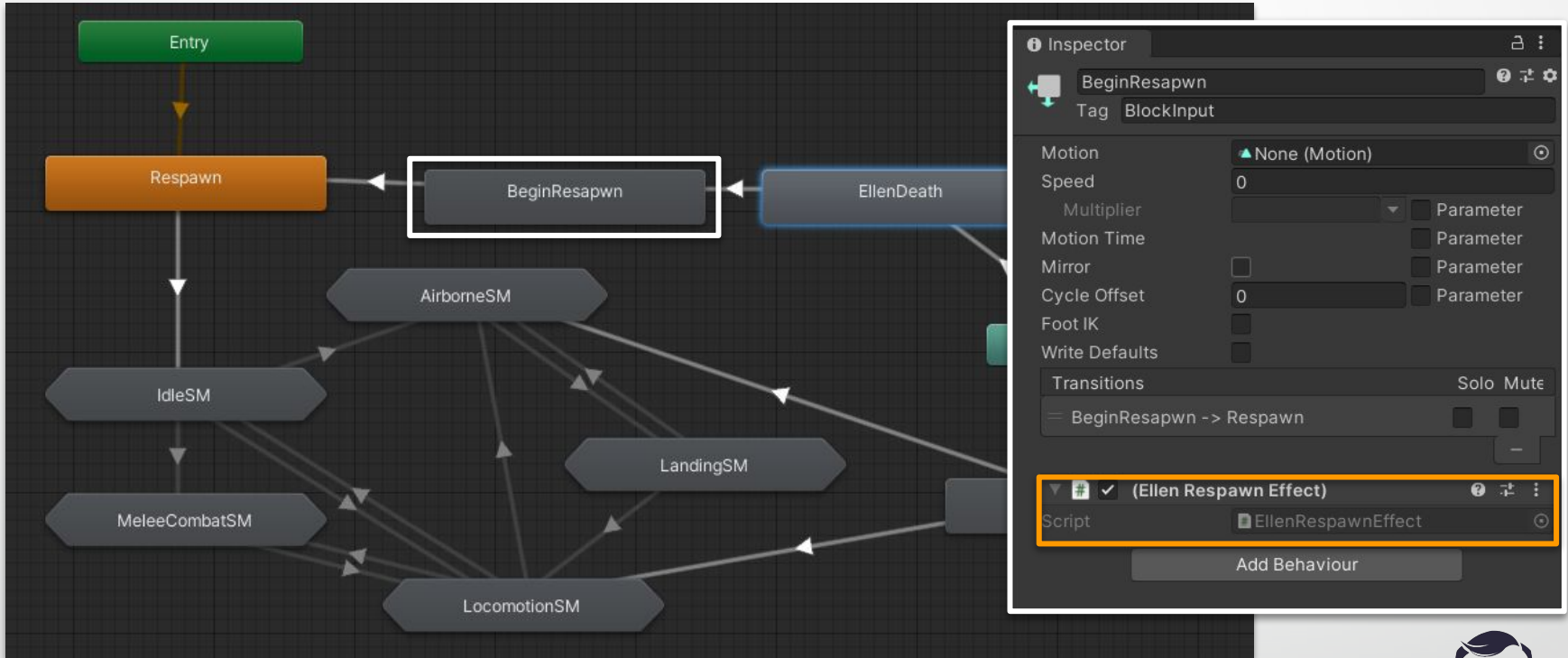


# StateMachineBehaviour





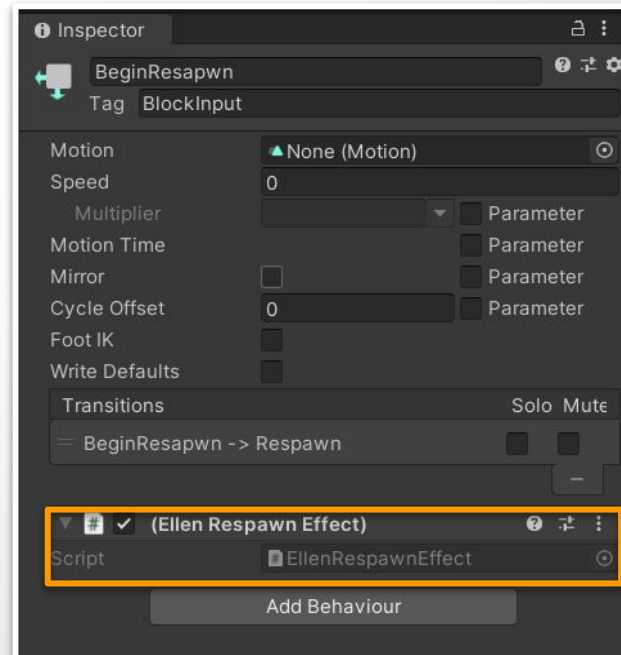
# StateMachineBehaviour





# StateMachineBehaviour

```
namespace Gamekit3D
{
    @ Unity 指令碼 | 0 個參考
    public class EllenRespawnEffect : StateMachineBehaviour
    {
        @ Unity Message | 3 個參考
        public override void OnStateEnter(
            Animator animator,
            AnimatorStateInfo stateInfo,
            int layerIndex)
        {
            animator.GetComponent<PlayerController>().Respawn();
        }
    }
}
```







# Animator Override Controller

Inspector

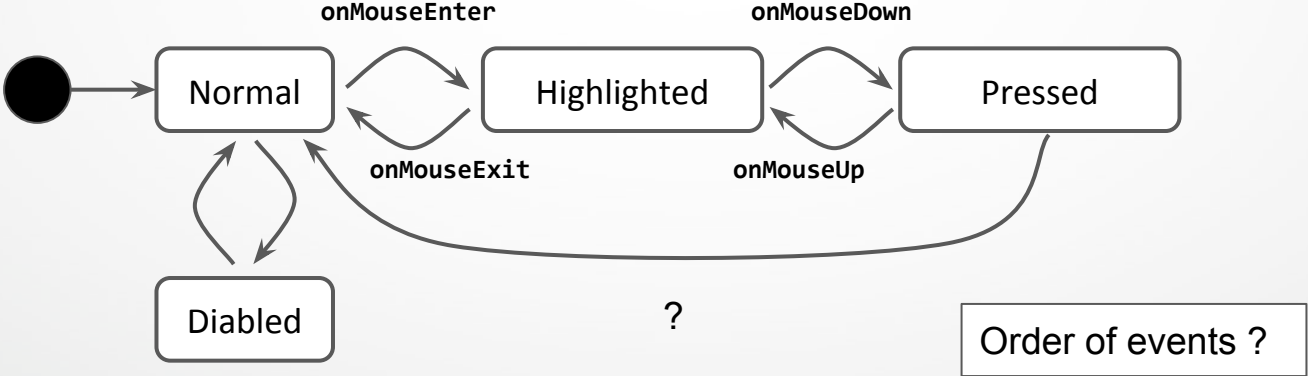
New Animator Override Controller

Open

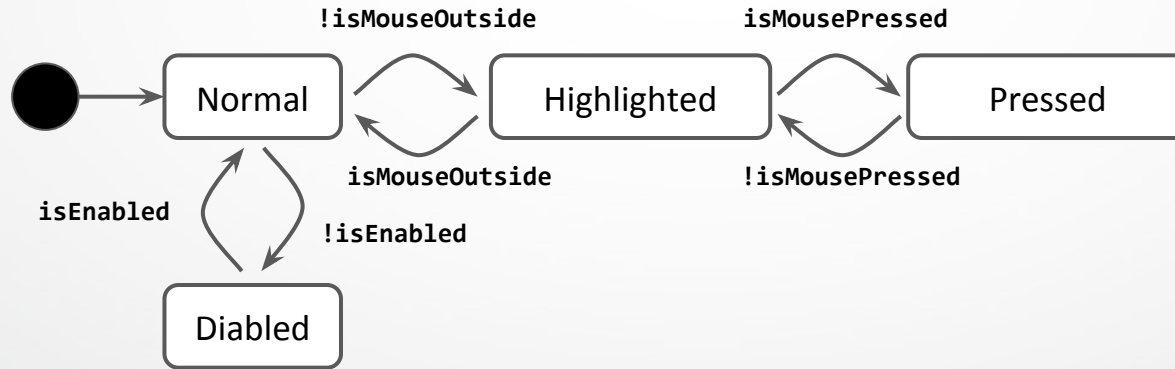
Controller: Ellen

Original	Override
Ellen_Combo1	▲ None (Animation Clip)
Ellen_Combo2	▲ None (Animation Clip)
Ellen_Combo3	▲ None (Animation Clip)
Ellen_Combo4	▲ None (Animation Clip)
Ellen_Death	▲ None (Animation Clip)
Ellen_HitBack	▲ None (Animation Clip)
Ellen_HitBackLeft	▲ None (Animation Clip)
Ellen_HitBackRight	▲ None (Animation Clip)
Ellen_HitFront	▲ None (Animation Clip)
Ellen_HitFrontLeft	▲ None (Animation Clip)
Ellen_HitFrontRight	▲ None (Animation Clip)

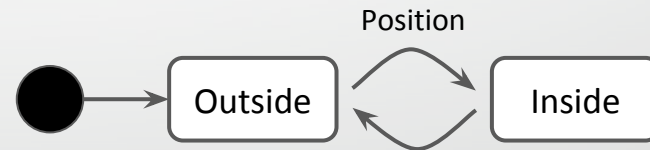
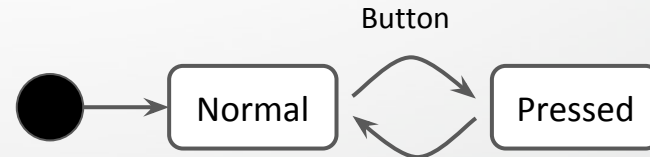
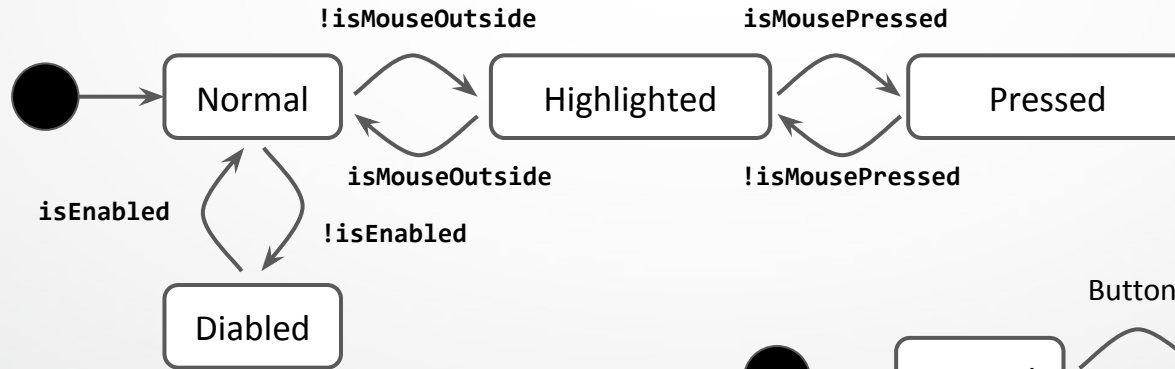
# Finite-state machine



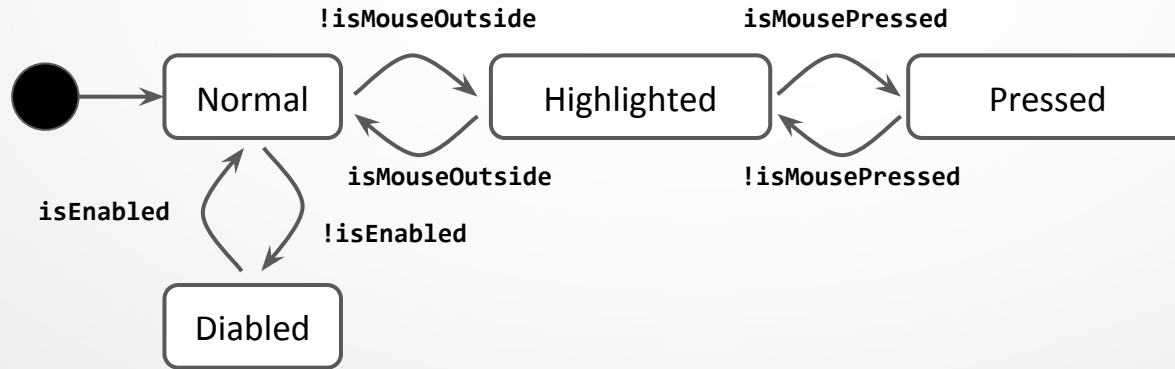
# State as triggered event



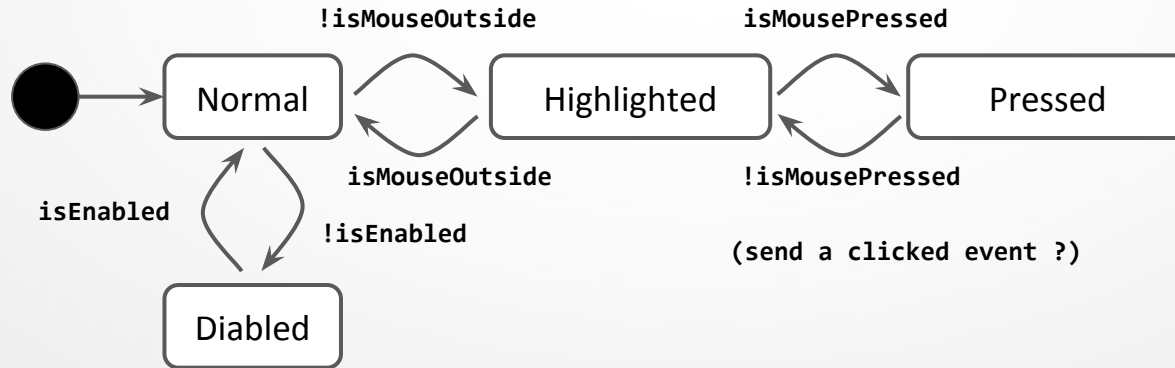
# State as triggered event



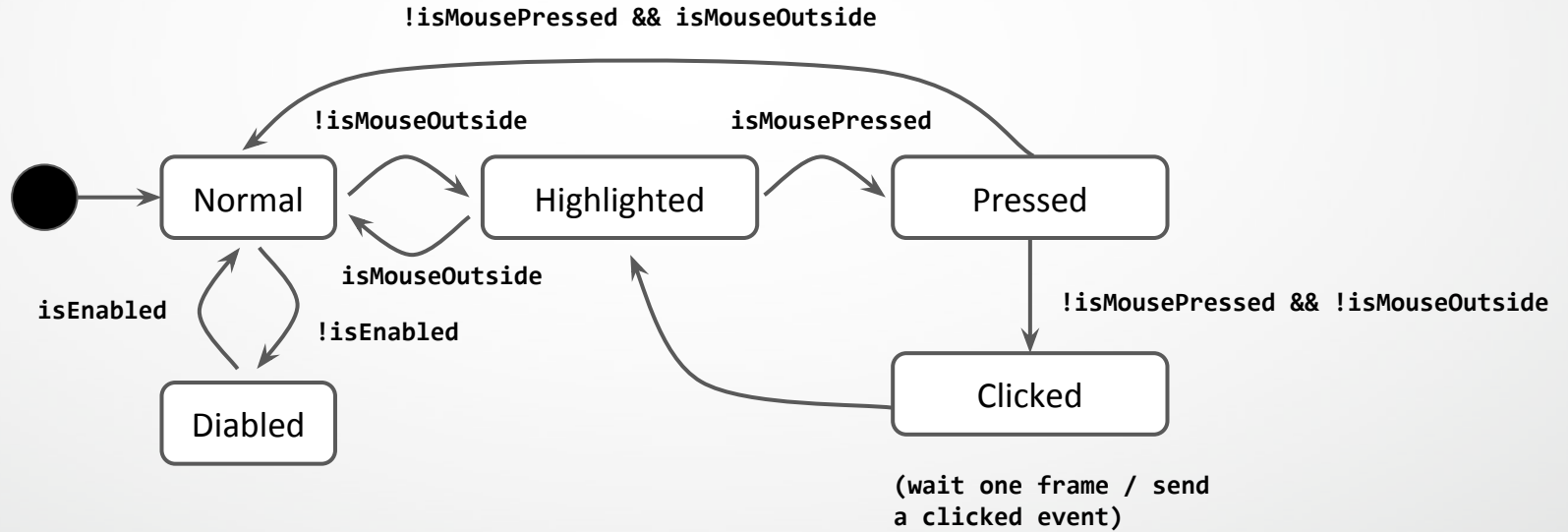
# Clicked ?



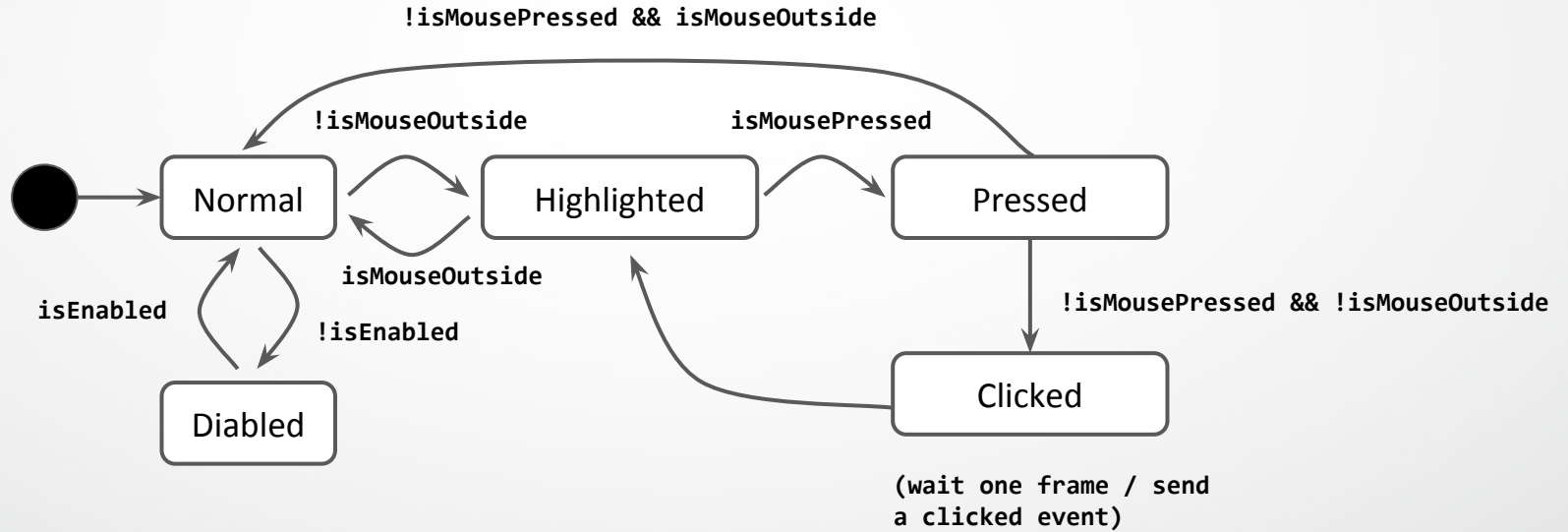
# Clicked ?



# Clicked ?

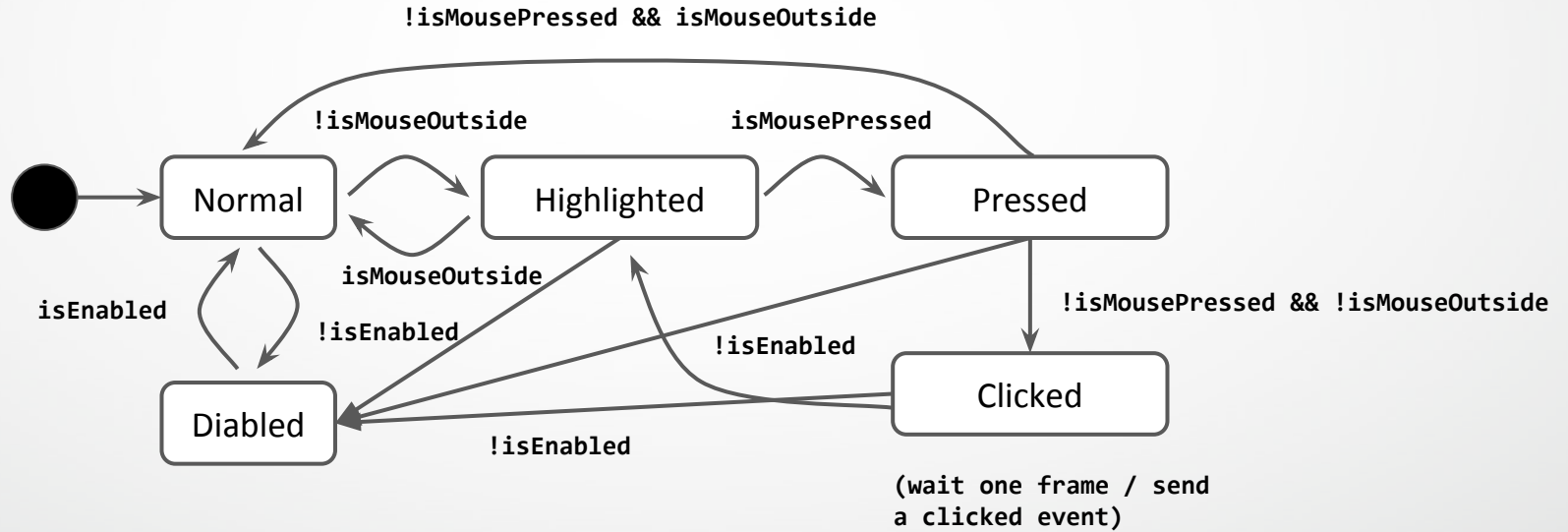


# isEnabled = false ?

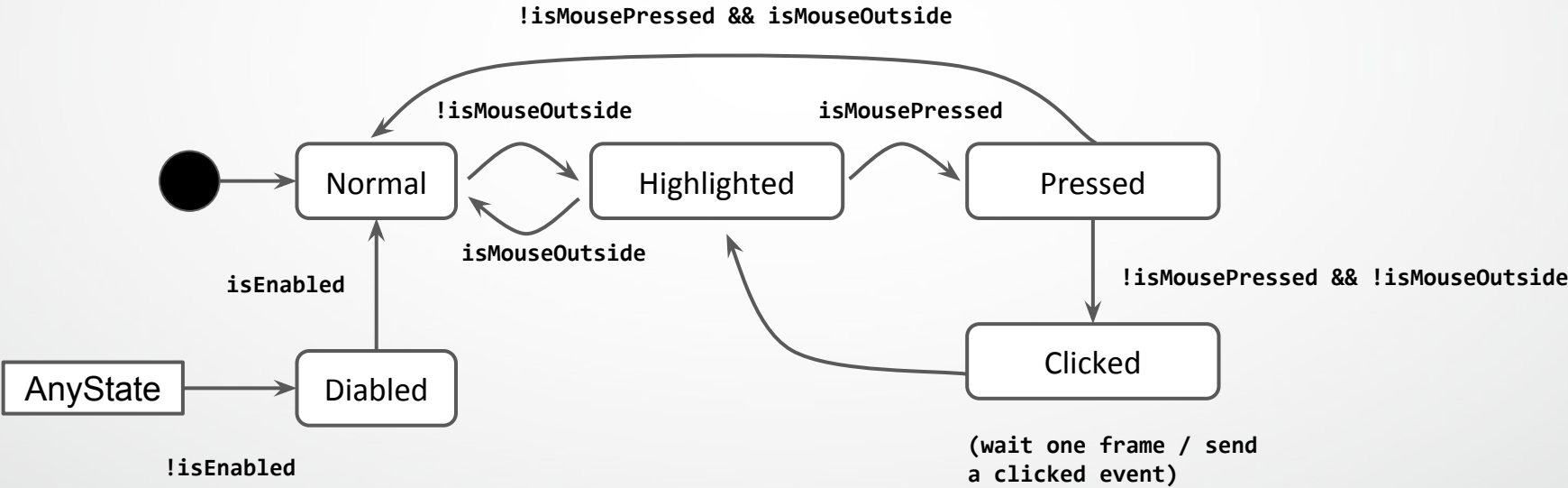




# isEnabled = false ?

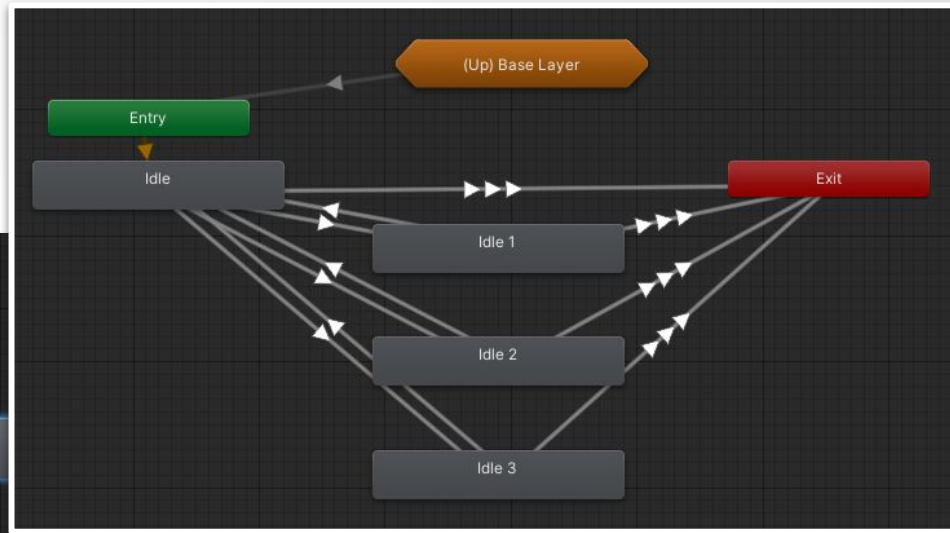
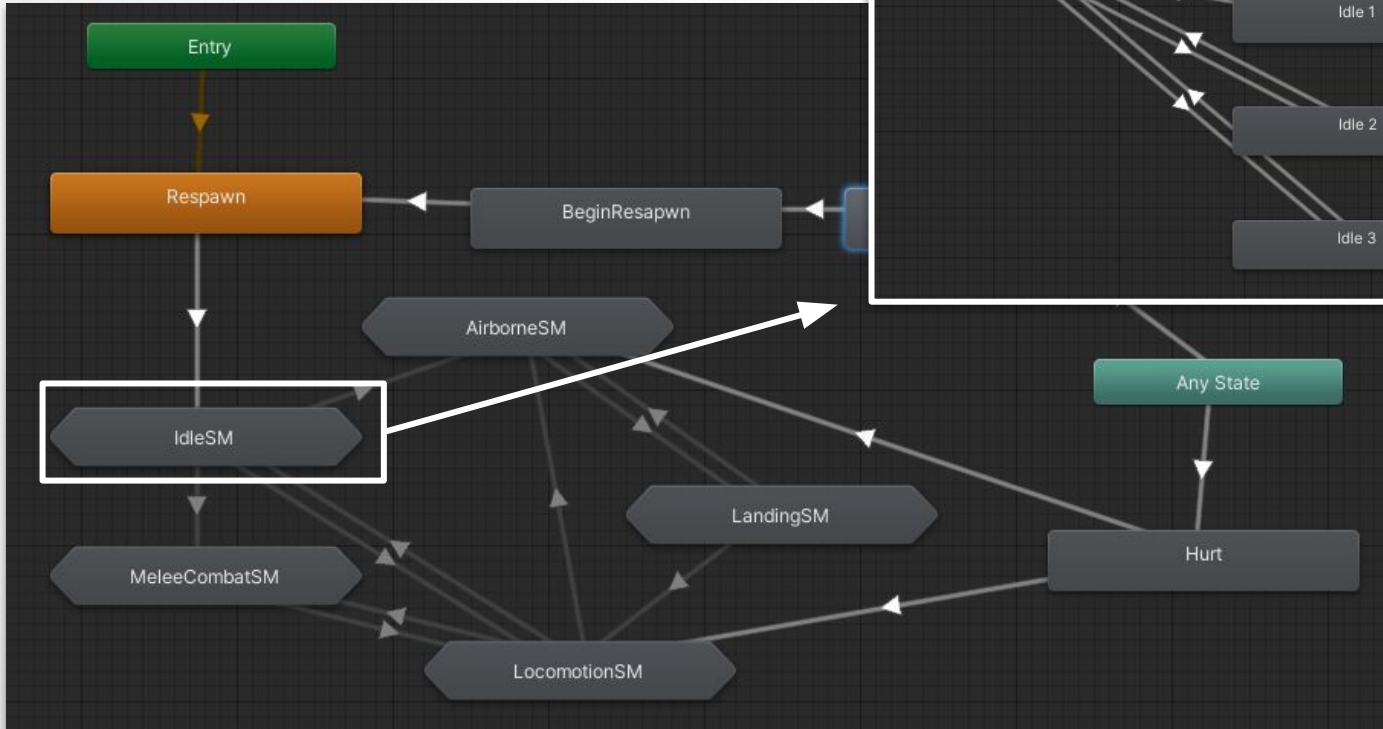


# IsEnabled = false ?

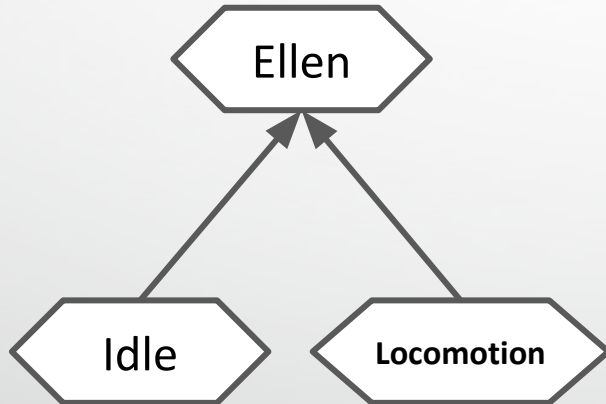




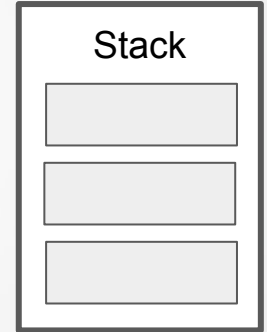
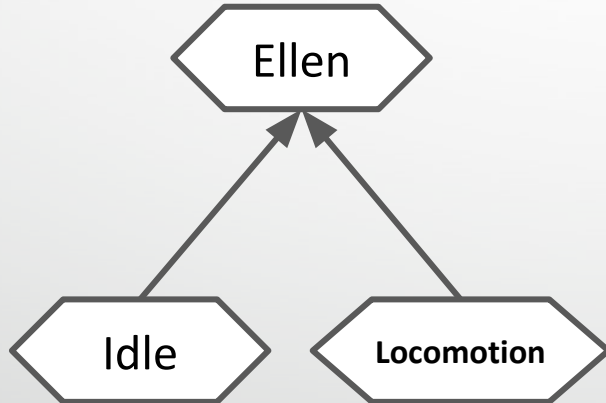
# Sub-state machines



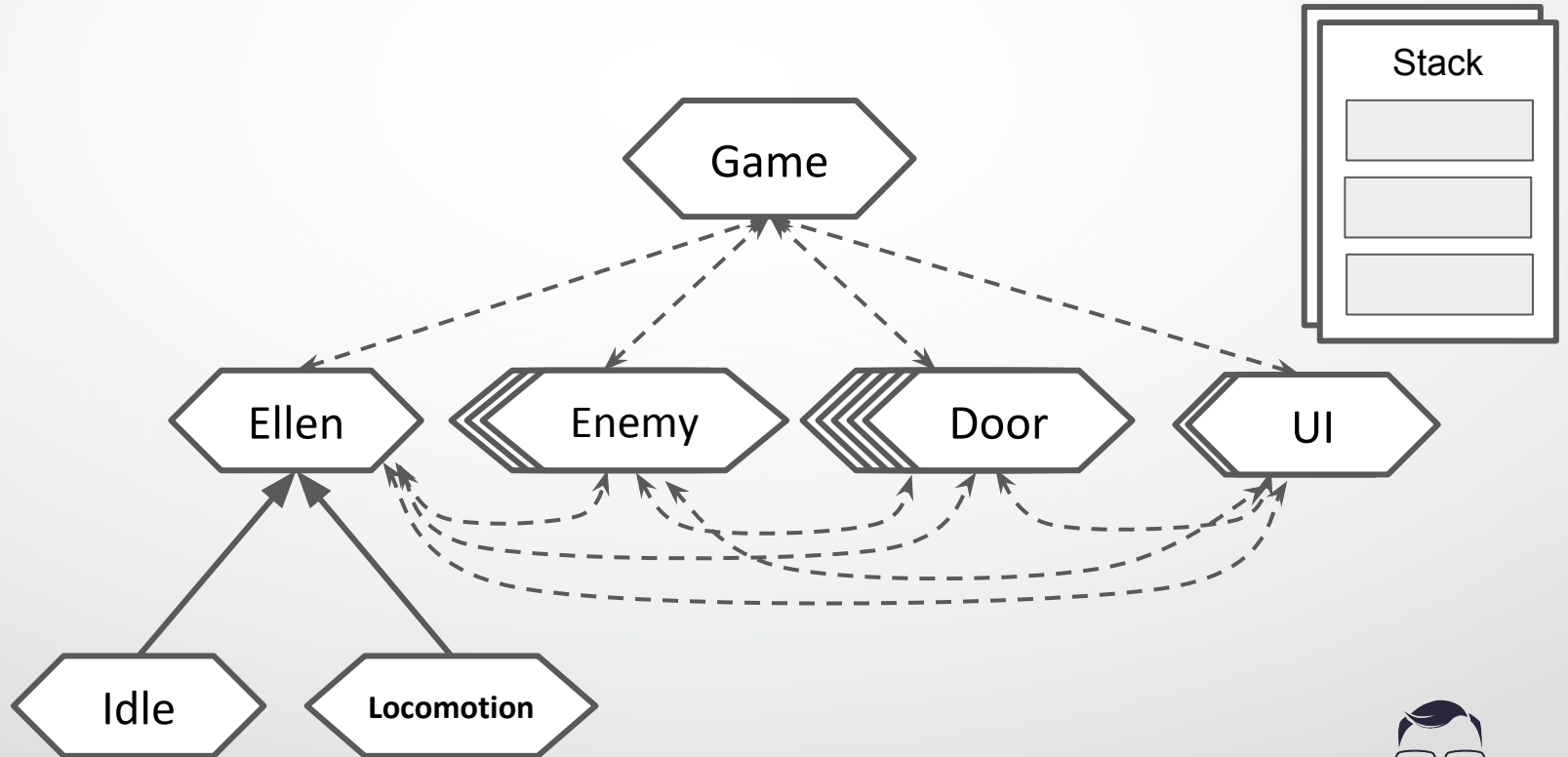
# Hierarchical state machine



# Pushdown automata



# Concurrent state machines



# Game state tree / store

Singleton pattern,  
Global variables, ...

---

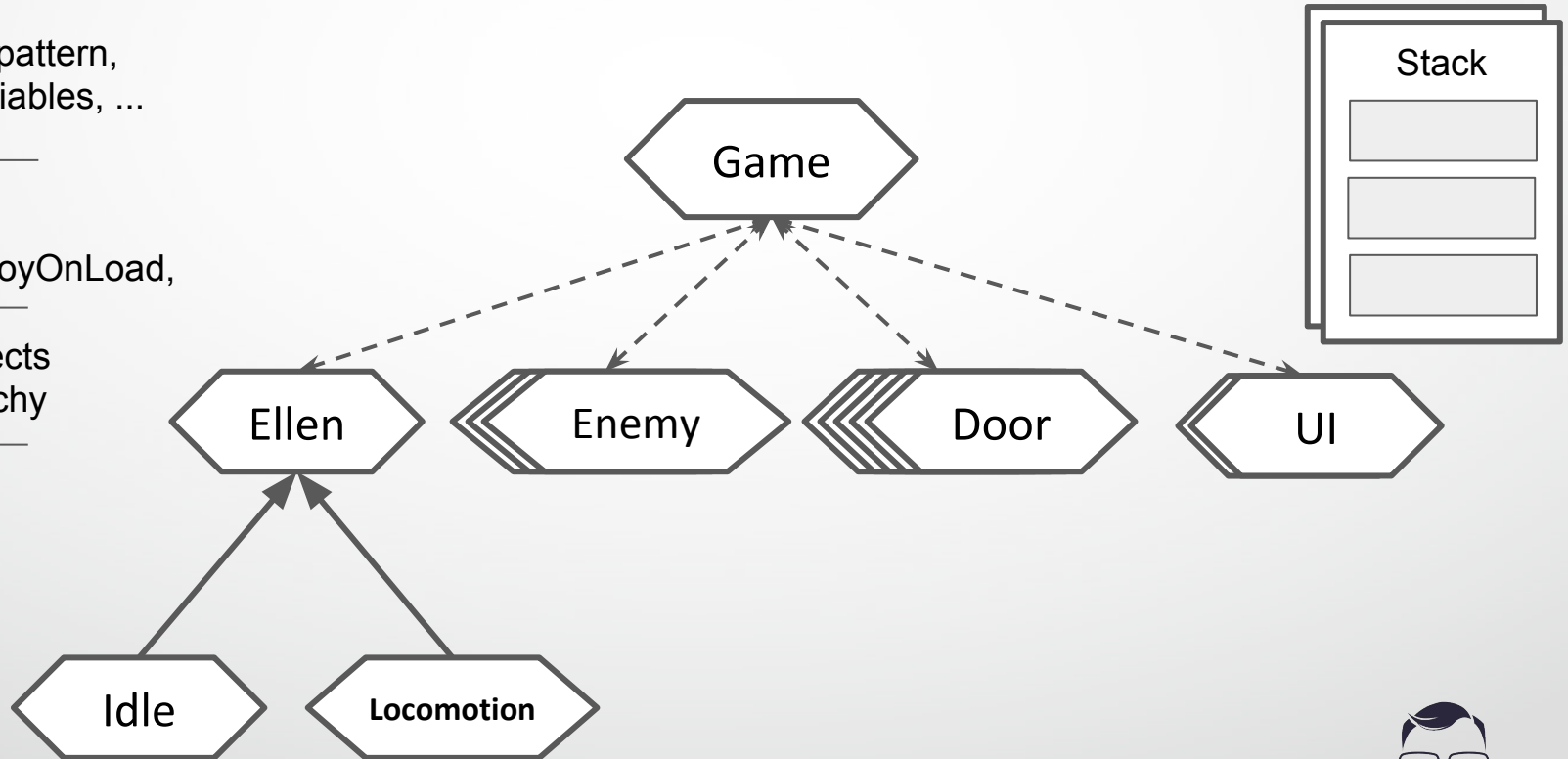
Scenes,  
DontDestroyOnLoad,

---

GameObjects  
in a hierarchy

---

Animators



# Save / Load?

Singleton pattern,  
Global variables, ...

---

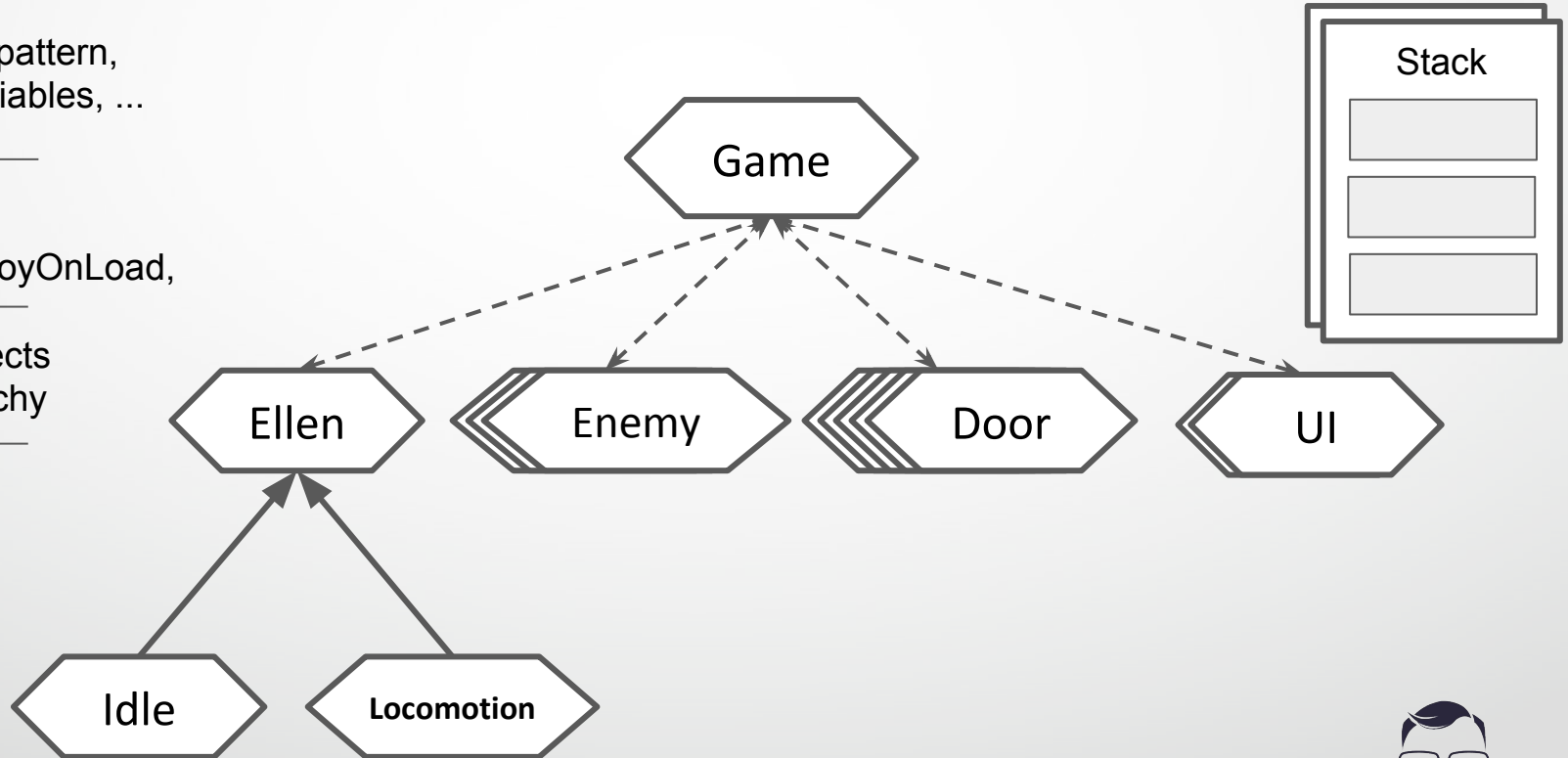
Scenes,  
DontDestroyOnLoad,

---

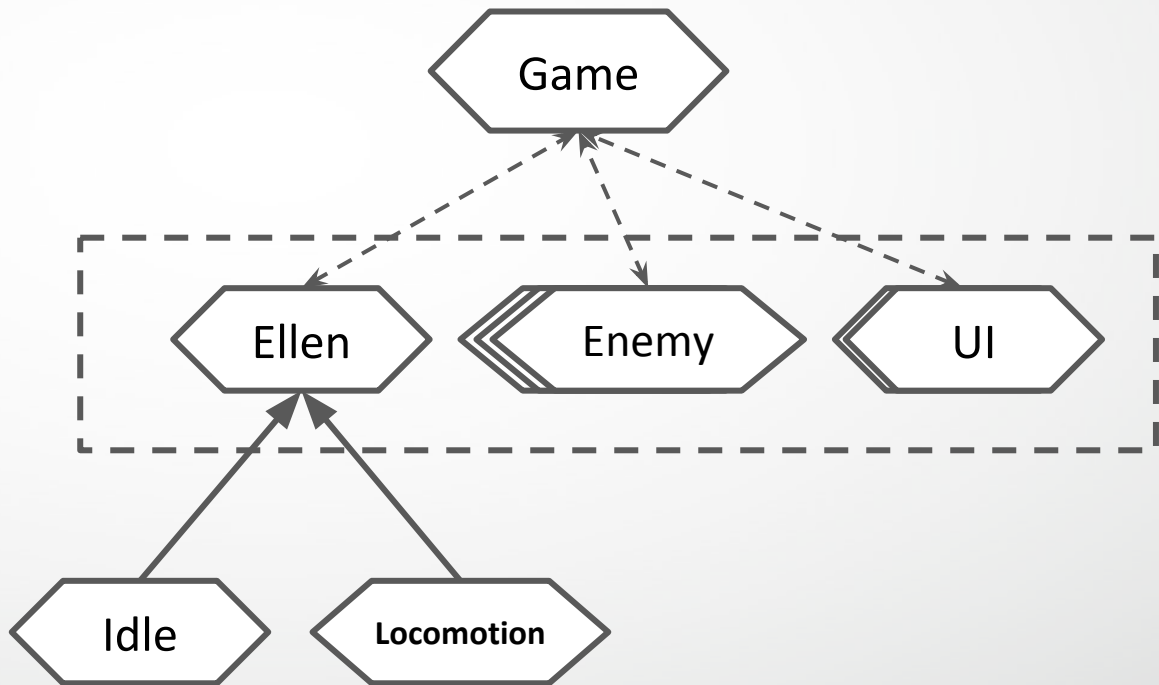
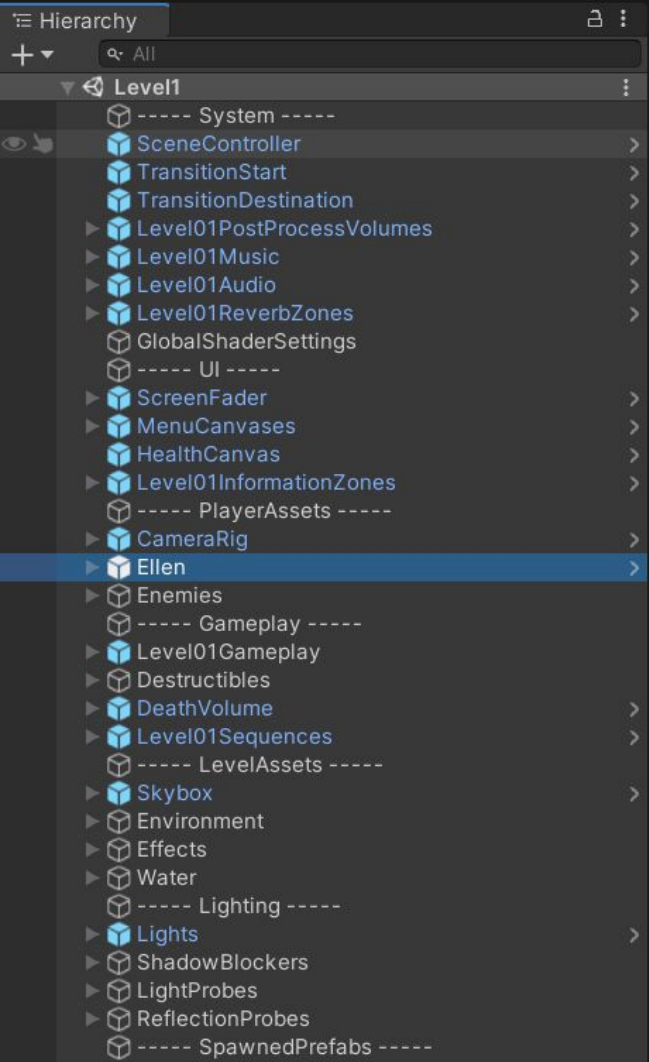
GameObjects  
in a hierarchy

---

Animators





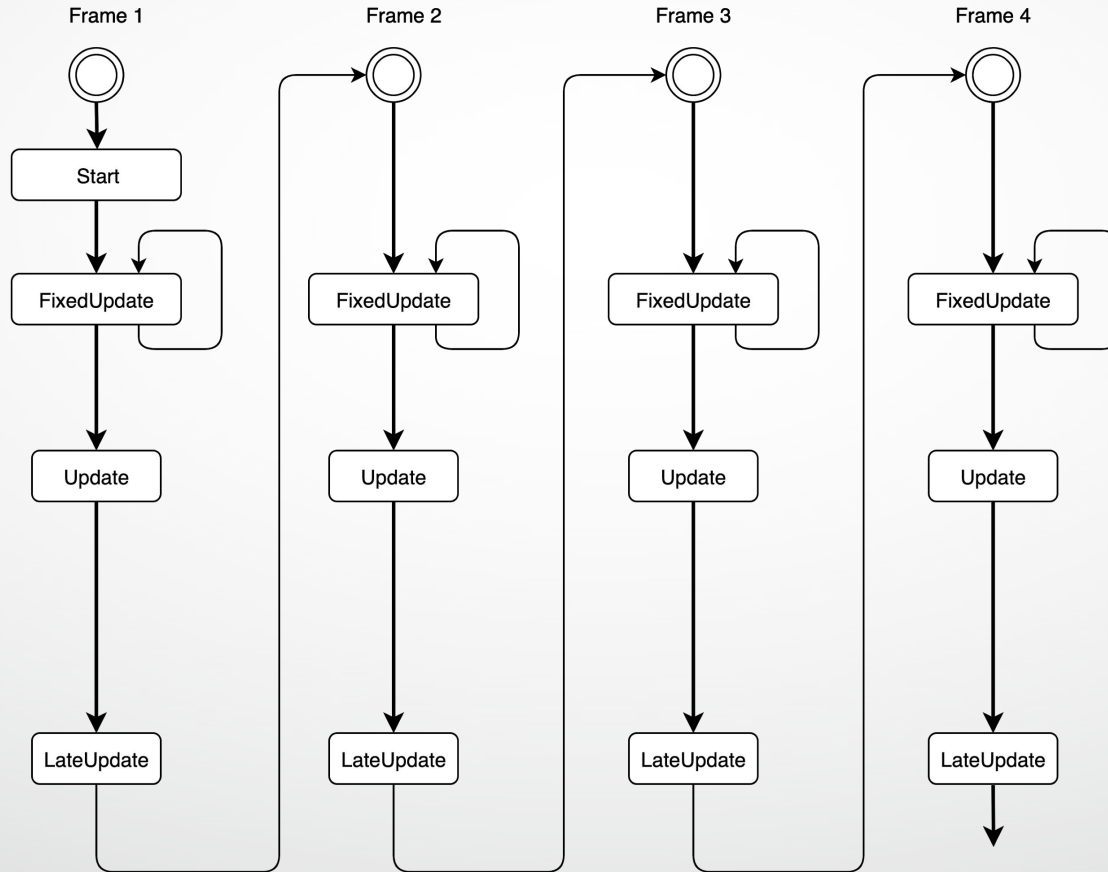


How to “Pause” a game ?





# MonoBehaviour.Update() / FixedUpdate()

- Hold the local state
  - enum State { Normal, Pressed, Highlighted, Disabled, Clicked };
    - with other data members
- Update the local state in Update() / FixedUpdate() or Coroutine with a hierarchical approach.
  - only read other objects' states if possible.
  - only receive / send "state events" if unavoidable
- Apply If-else chain, switch or state pattern





# Script Execution Order settings

**Script Execution Order**  

Add scripts to the custom order and drag them to reorder.

Scripts in the custom order can execute before or after the default time and are executed from top to bottom. All other scripts execute at the default time in the order they are loaded.

(Changing the order of a script may modify the meta data for more than one script.)

Default Time

UnityEngine.EventSystems.HoloLensInput	100	-
LoadBundle	200	-
UnityEngine.XR.WSA.SpatialMappingBase	250	-
LoadTextures	300	-
BuildiOSAppSlices	400	-

+ ▾

Revert Apply