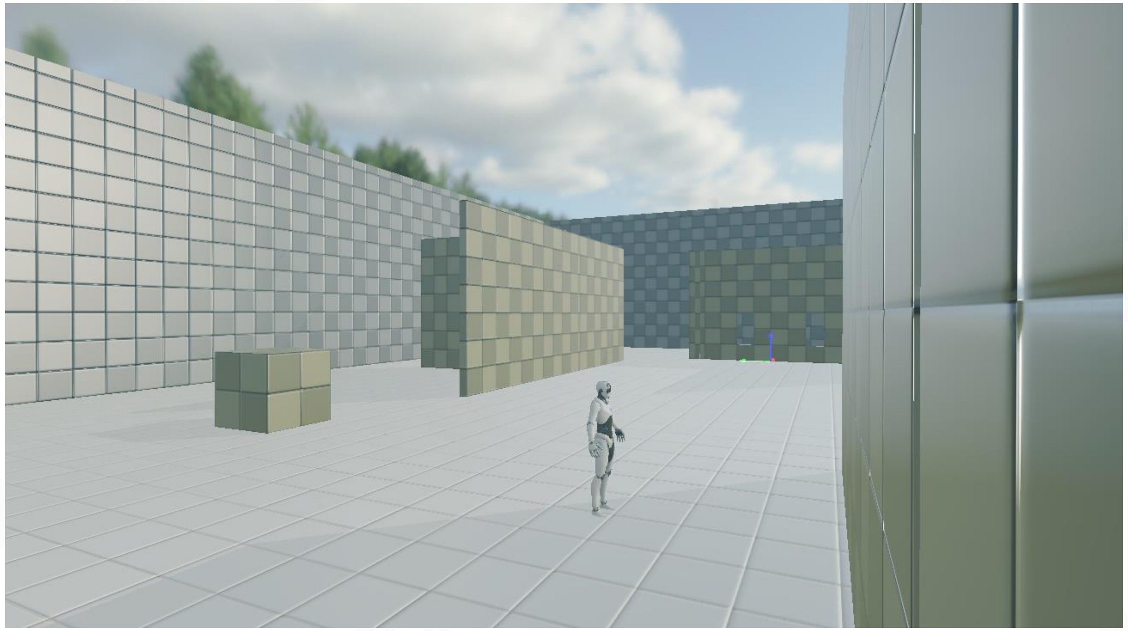
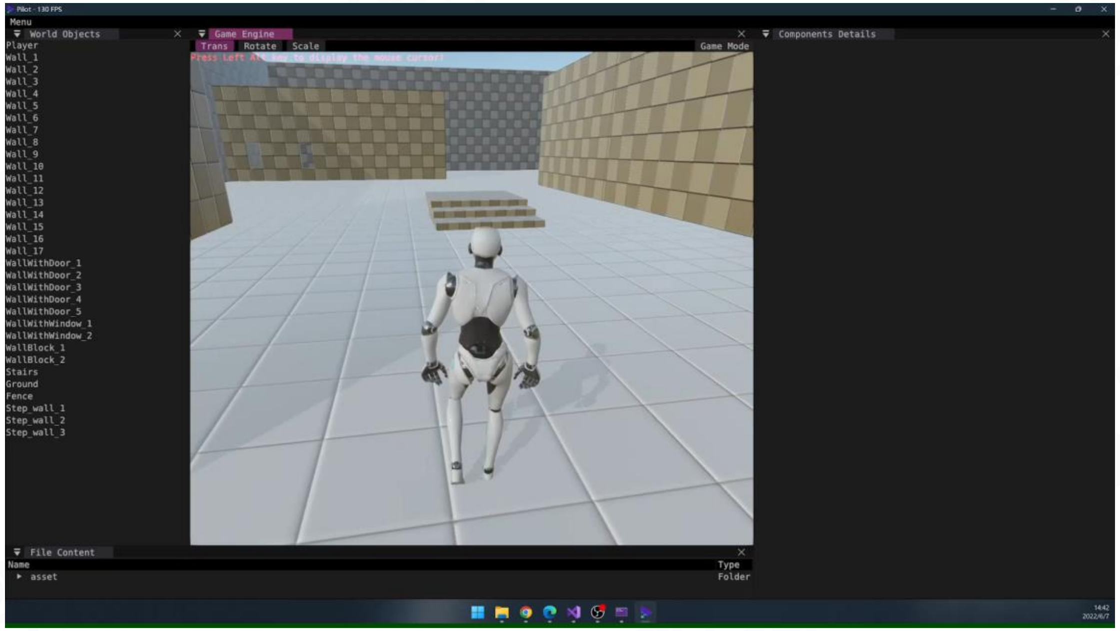
Homework Submission Extended

- Will extend Homework #2 and Homework #3 deadline to Aug 31th



Homework #2 : Rendering



Press Left Alt key to display the mouse cursor!

Name	Type
▸ asset	Folder

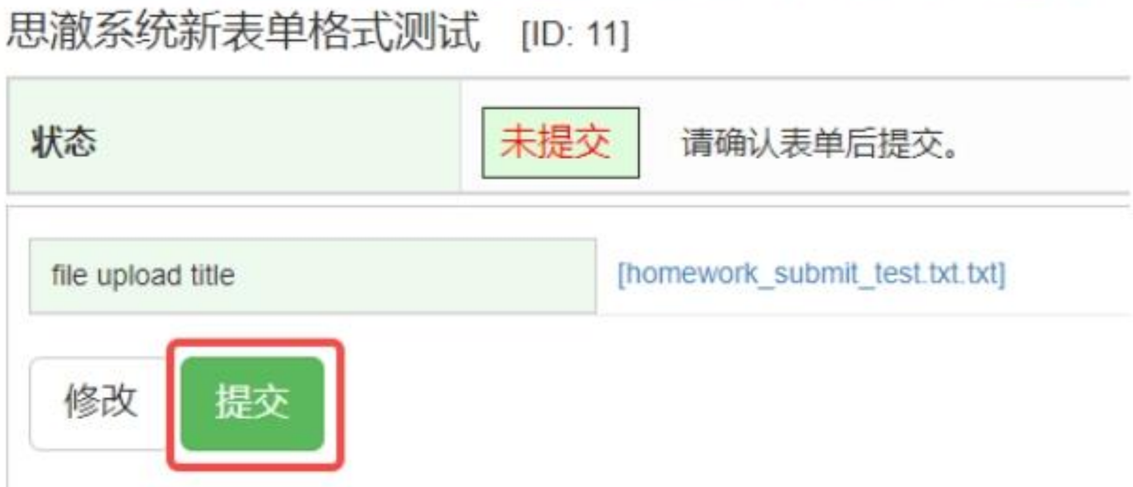
Homework #3 : Animation and Physics

Voice from Community

- Some submissions are reported lost
- We’ve tested the submission system after received the reports
- We noticed one critical step may easily be omitted
- We have highlighted the critical step and updated our submission guide
- Please refer: https://cdn.boomingtech.com/games104_static/upload/GAMES104_SmartChair_Submission_Guide.pdf , Page 9

点击“提交”（想要修改提交的文件可以点击“修改”）

注意：此步骤非常关键，否则作业不会被上传到系统，依然会显示“未提交”。



思澈系统新表单格式测试 [ID: 11]

状态	未提交 请确认表单后提交。
file upload title	[homework_submit_test.txt.txt]

Q&A about Piccolo Engine

- Q1: Is script system in Piccolo Engine’s roadmap? Which script language will Piccolo Engine support?
- Q2: Why did Piccolo Engine use CMake as meta build system instead of XMake?
- Q3: Why some source code will be recompiled even no code is modified?

Lecture 15

Gameplay

Gameplay Complexity and Building Blocks

WANG XI

GAMES 104

2022

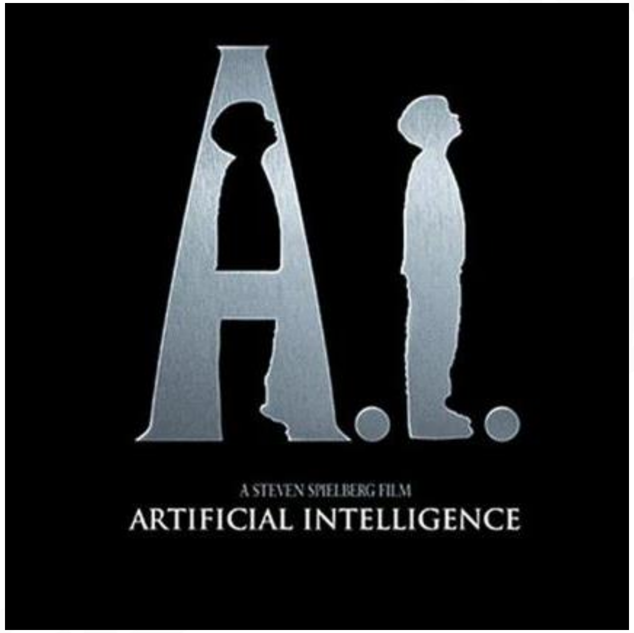
Outline of Gameplay System

01.

Gameplay Complexity and Building Blocks

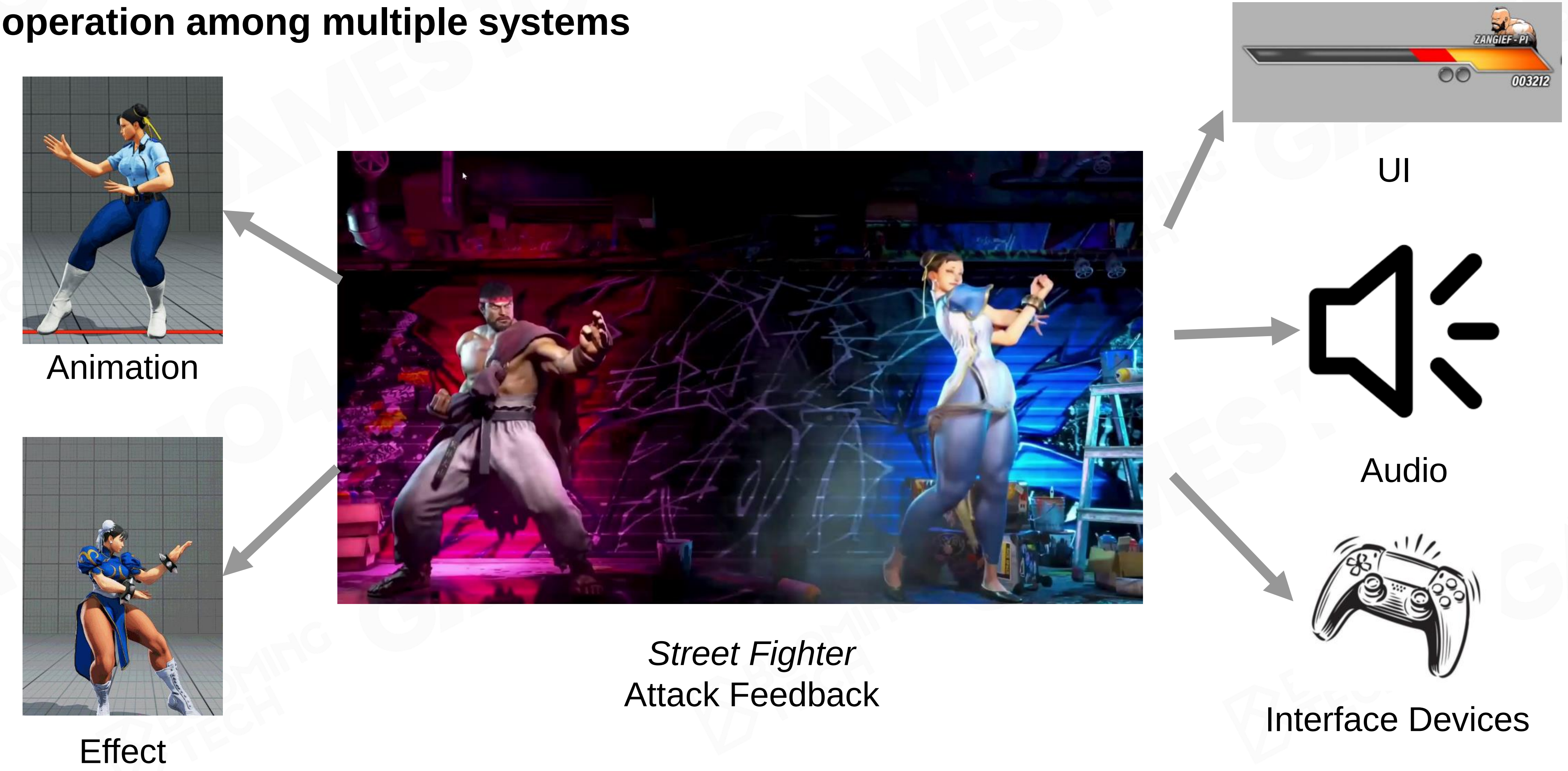
- Overview
- Event Mechanism
- Script System
- Visual Script
- Character, Control and Camera

02.



Challenges in GamePlay(1/3)

Cooperation among multiple systems



Animation

Effect

Street Fighter
Attack Feedback

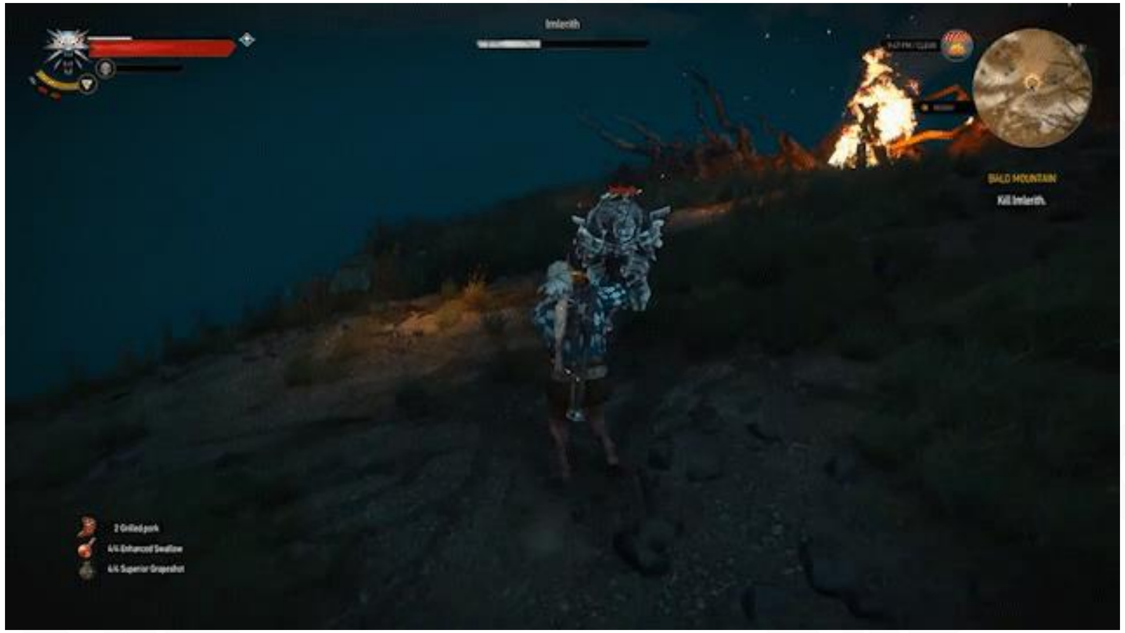
UI

Audio

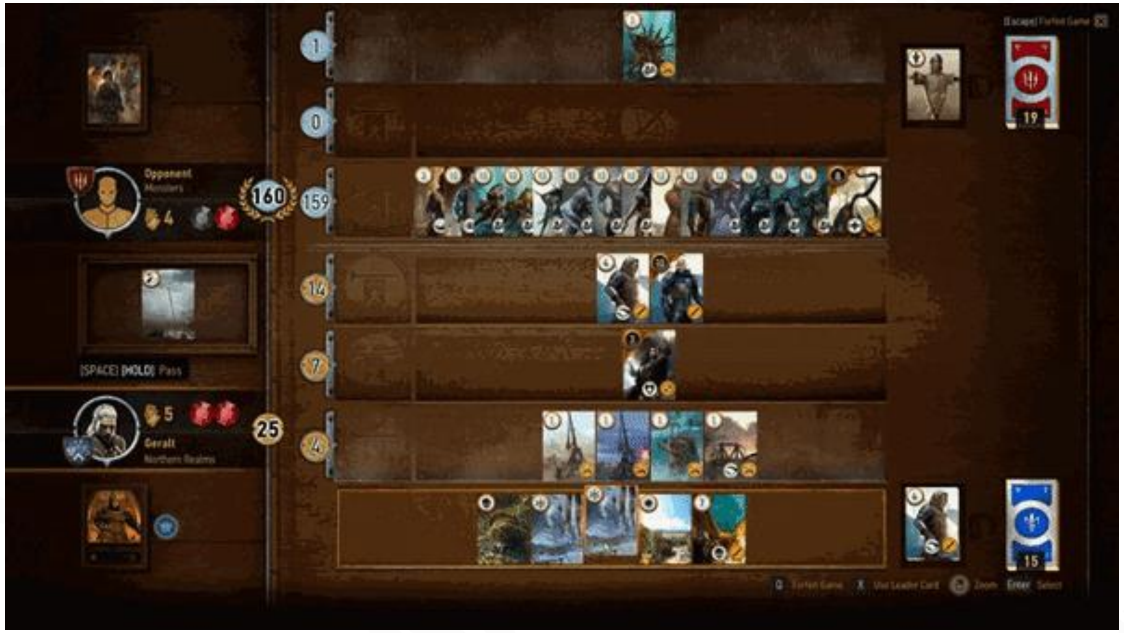
Interface Devices

Challenges in GamePlay (2/3)

Diversity of game play in the same game



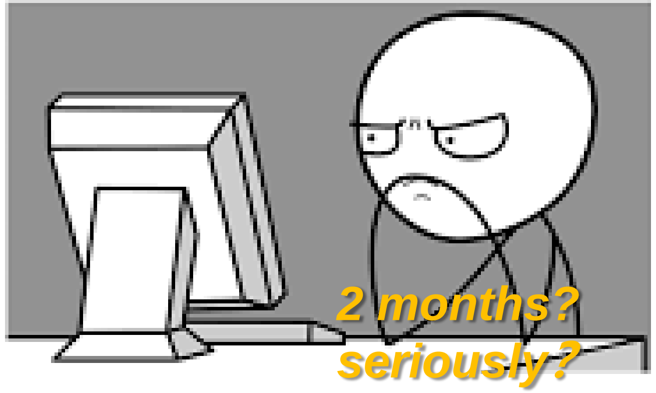
The Witcher 3: Wild Hunt
Combat Gameplay



The Witcher 3: Wild Hunt
Card-playing Mechanic

Challenges in GamePlay (3/3)

Rapid iteration



Fortnite: Save the World
TPS,tower defense, survival

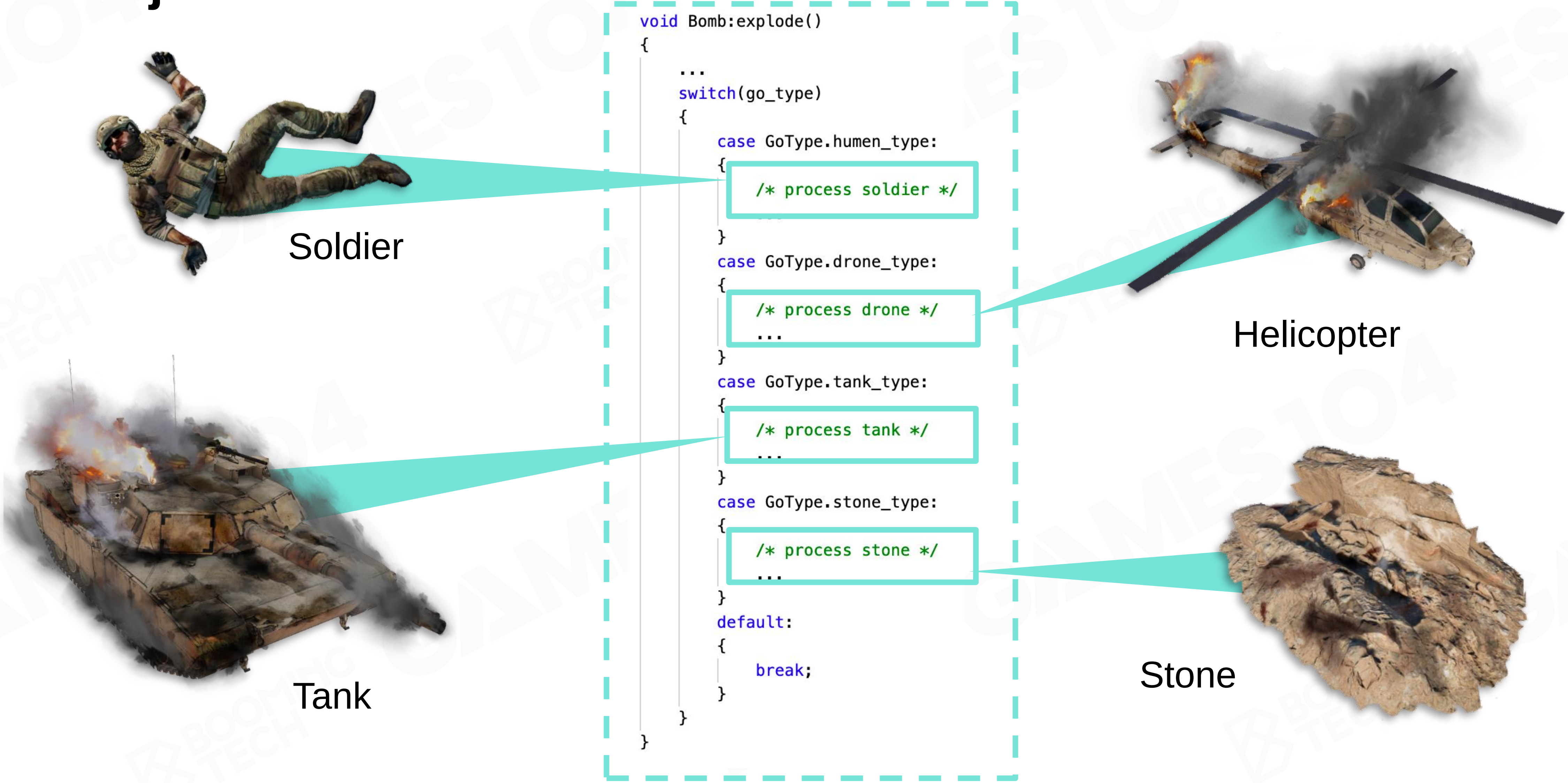
Fortnite: Battle Royale
battle royale

Epic acknowledged that within the Fortnite fundamentals, they could also do a battle royale mode, and rapidly developed their own version atop Fortnite in about **two months**.

GamePlay

Event Mechanism

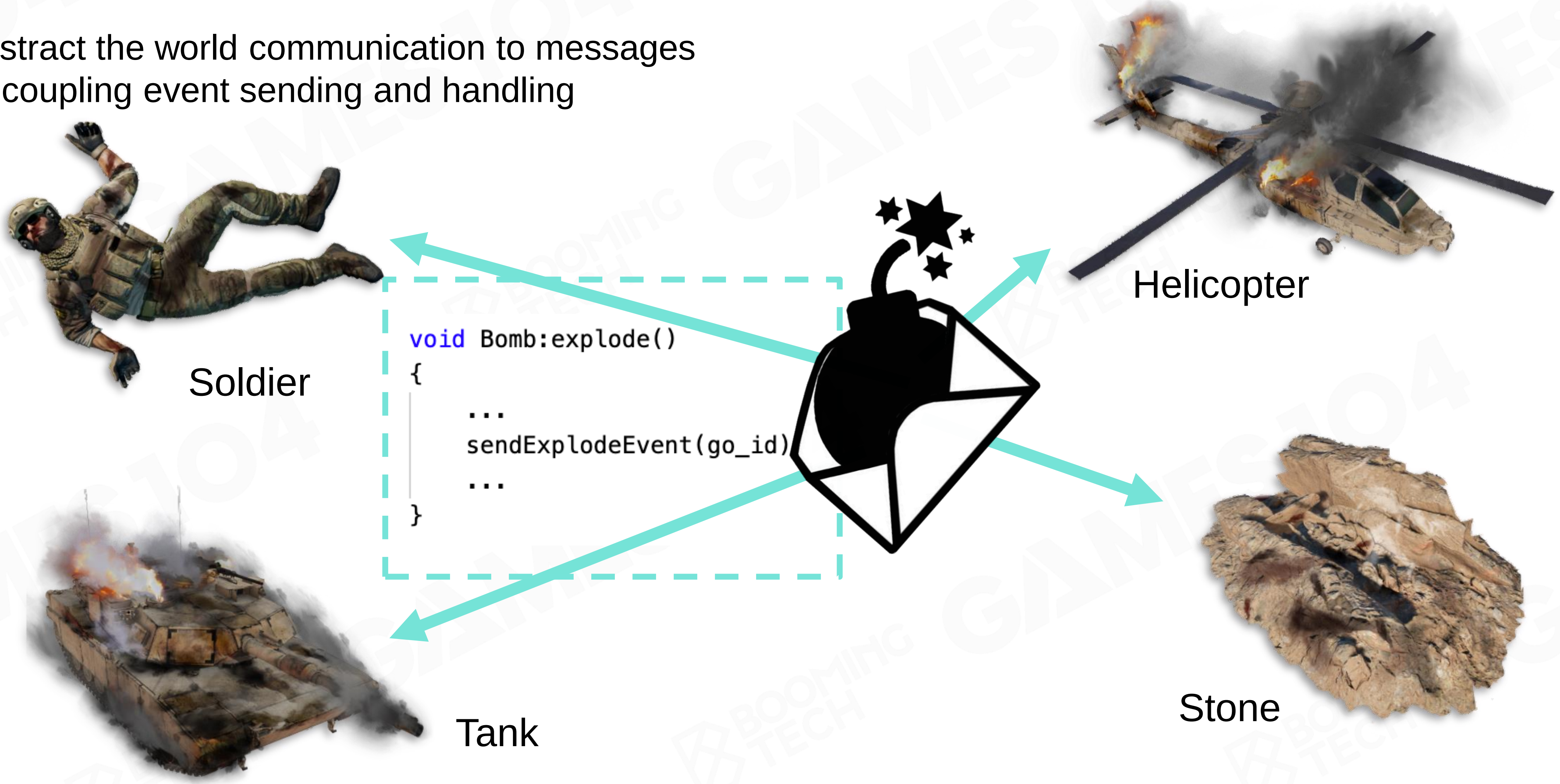
Let Objects Talk



```
void Bomb:explode()
{
    ...
    switch(go_type)
    {
        case GoType.humen_type:
        {
            /* process soldier */
            ...
        }
        case GoType.drone_type:
        {
            /* process drone */
            ...
        }
        case GoType.tank_type:
        {
            /* process tank */
            ...
        }
        case GoType.stone_type:
        {
            /* process stone */
            ...
        }
        default:
        {
            break;
        }
    }
}
```

Event/Message Mechanism

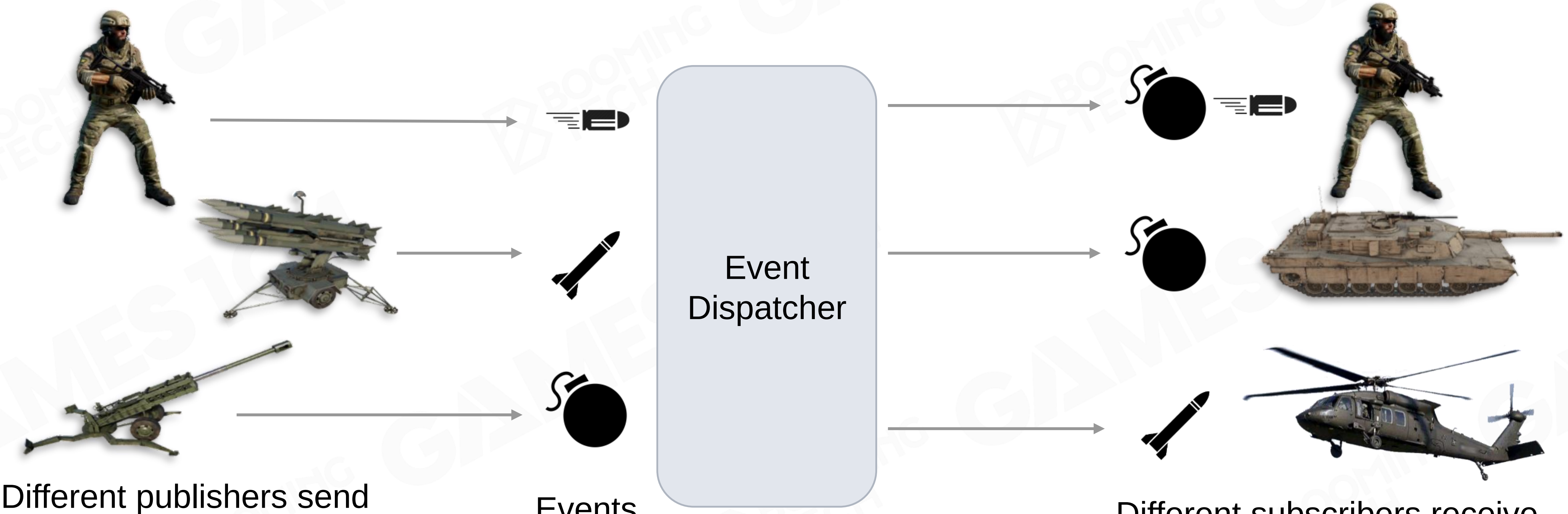
- Abstract the world communication to messages
- Decoupling event sending and handling



```
void Bomb:explode()
{
    ...
    sendExplodeEvent(go_id)
    ...
}
```

Publish-subscribe Pattern

- Publisher categorizes published messages (events) into classes
- Subscriber receive messages (events) that are of interest without knowledge of which publishers



Different publishers send different type of events

Different subscribers receive events of thier interest

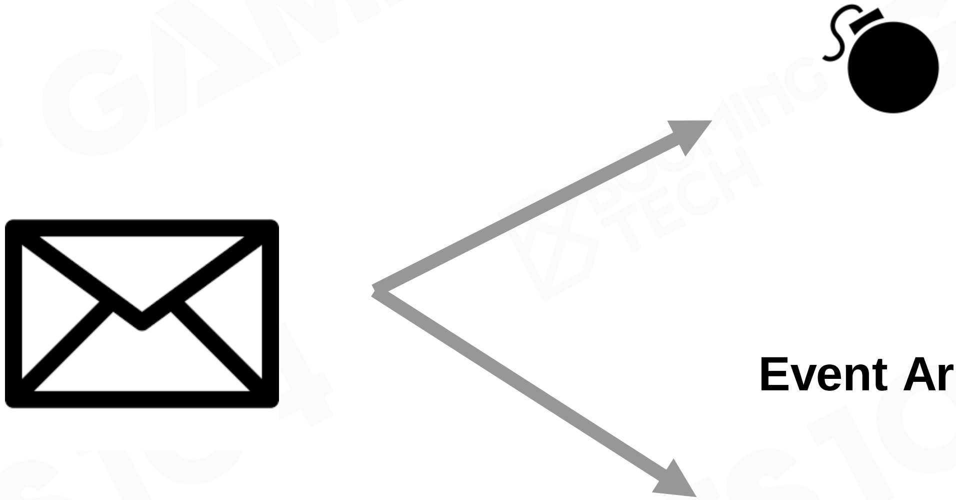
3 Key Components of Publish-subscribe Pattern

Event Definition

Callback Registration

Event Despatching

Event Definition



Event Type

"EVENT_TYPE_BOMB_EXPLOSION"

Event Argument

Key-value Table of Event Arguments

Key	Type	Value
"radius"	float	3.5
"damage"	int	40

Event Definition

Type and Arguments

```
class BombExplosionEvent : public Event
{
    Point m_center;
    float m_damage;
    float m_radius;
};
```

```
class BulletHitEvent : public Event
{
    float   m_final_speed;
    float   m_damage;
};
```

```
class MissileHitEvent : public Event
{
    float   m_damage;
};
```

```
... ...
```

Impossible for hardcode

Event Definition

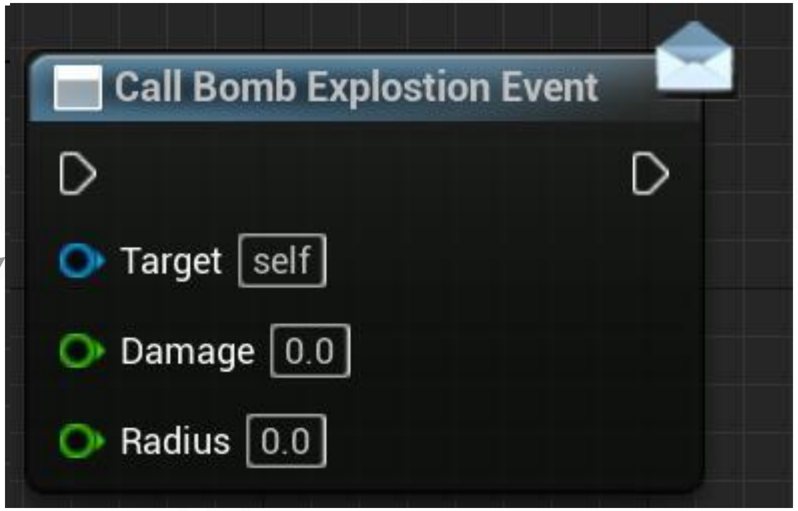
Type and Arguments

- Editable

```
class BombExplosionEvent
        : public Event
{
    float   m_damage;
    float   m_radius;
};
```

Hardcode

Reflection



Call Bomb Explostion Event

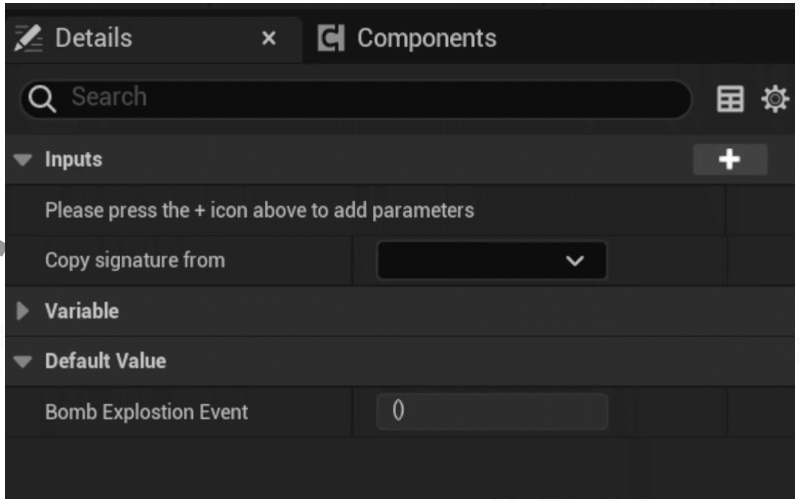
Target self

Damage 0.0

Radius 0.0

View in Editor

Code Generator



Inputs +

Please press the + icon above to add parameters

Copy signature from

Variable

Default Value

Bomb Explosion Event ()

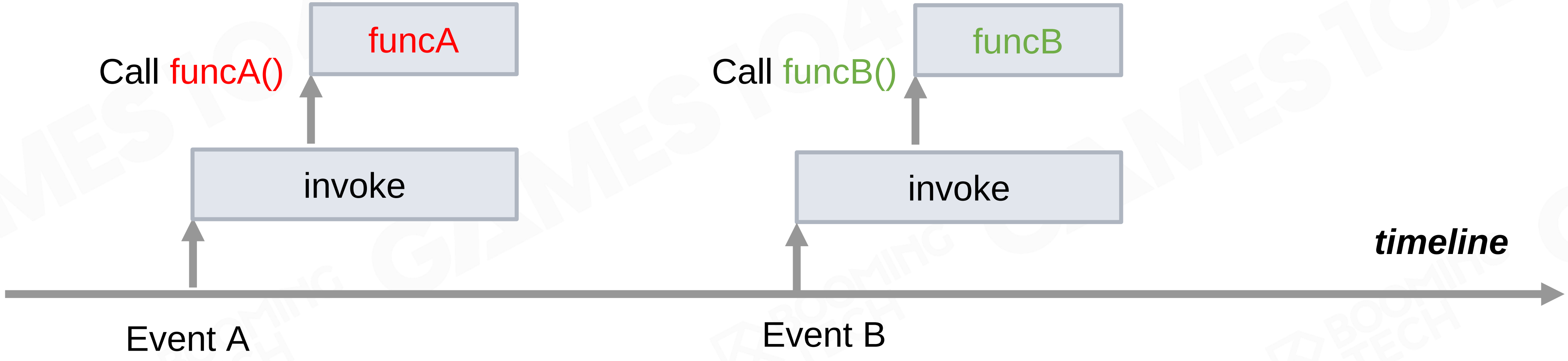
Editable

Callback Registration

Callback (function)

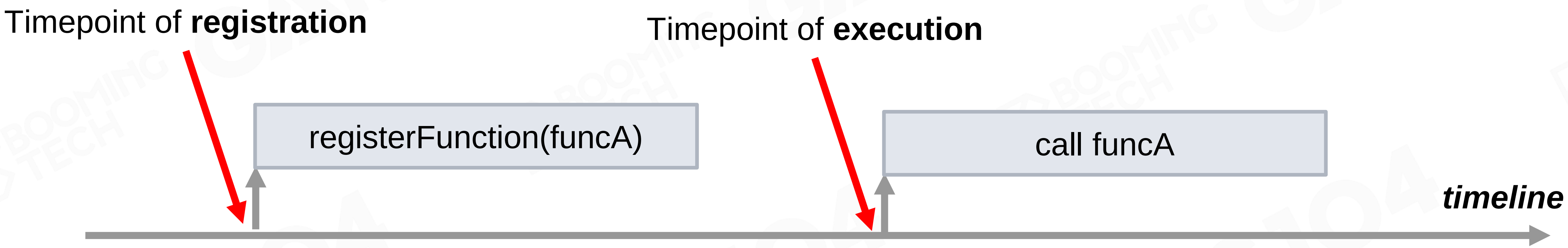
- Any reference to executable code that is passed as an argument to another piece of code

```
function invoke(call_back_function)
{
    // ...
    call_back_function()
}
```

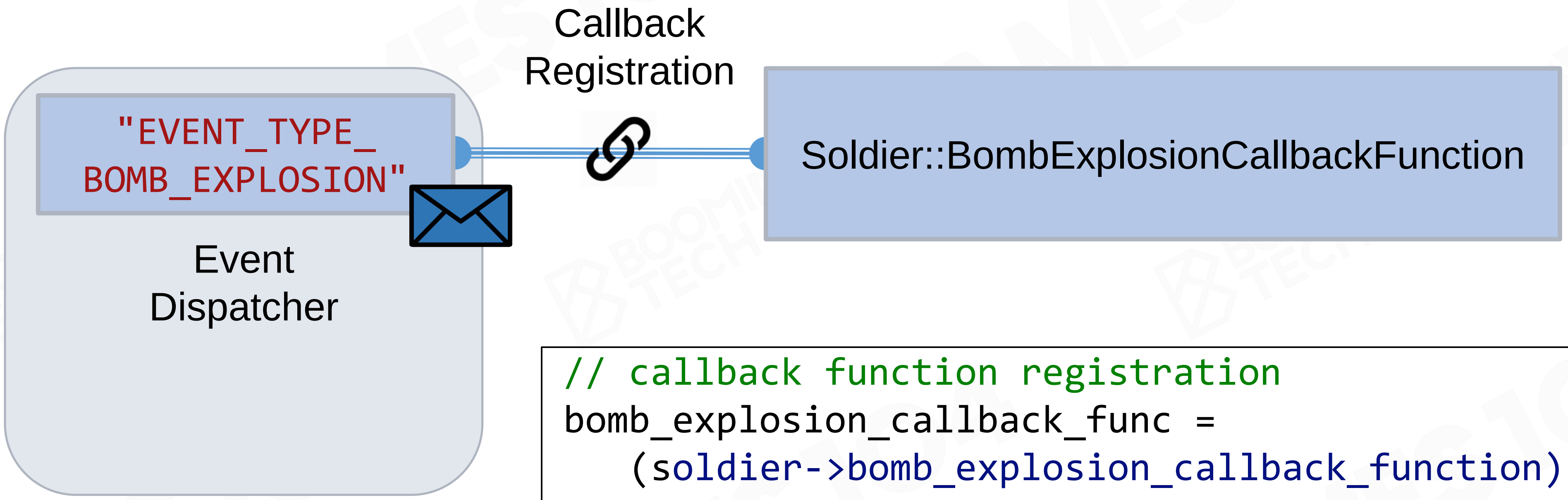


Object Lifespan and Callback Safety

Time points of registration and execution differs



Object Lifespan and Callback Safety

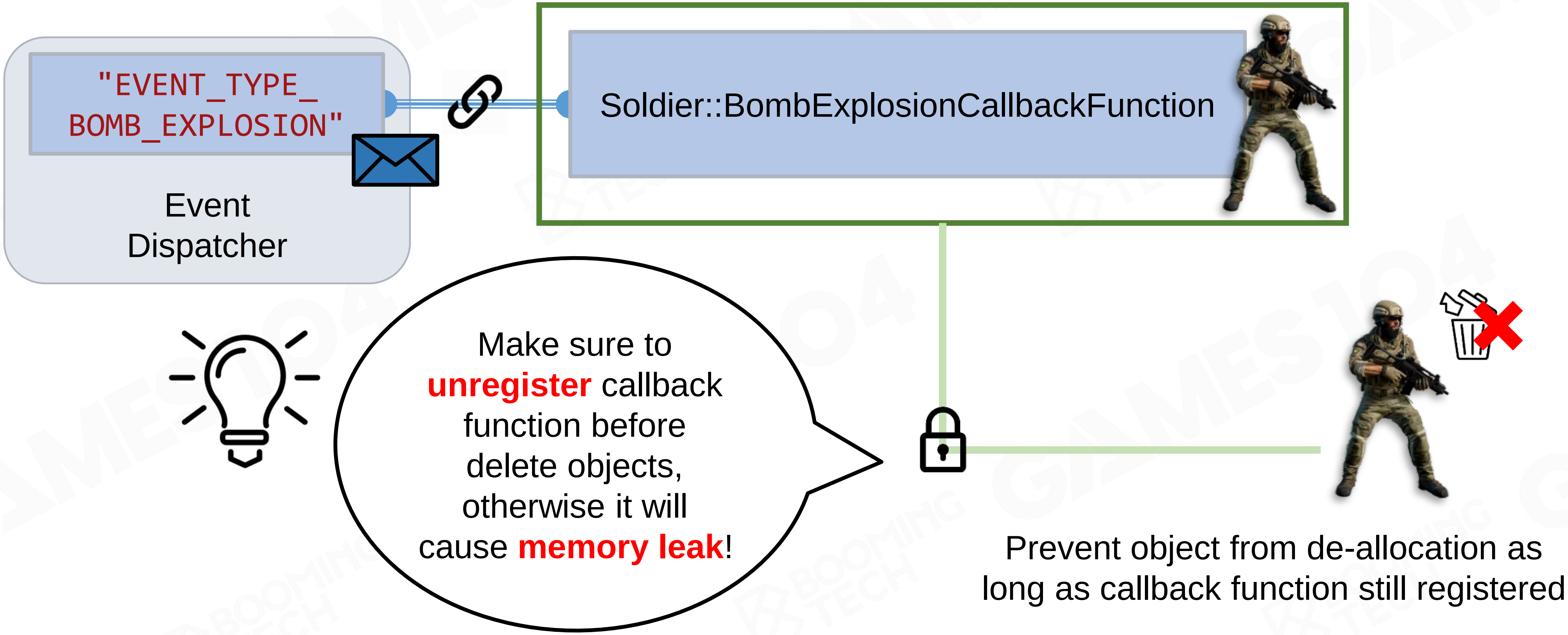


```
// callback function registration
bomb_explosion_callback_func =
   (soldier->bomb_explosion_callback_function)

//...
// when EVENT_TYPE_BOMB_EXPLOSION event comes
invoke(soldier, bomb_explosion_callback_func)
```

What if soldier already destroyed?

Wild pointer! Crashed!

Object Strong Reference



"EVENT_TYPE_ BOMB_EXPLOSION"

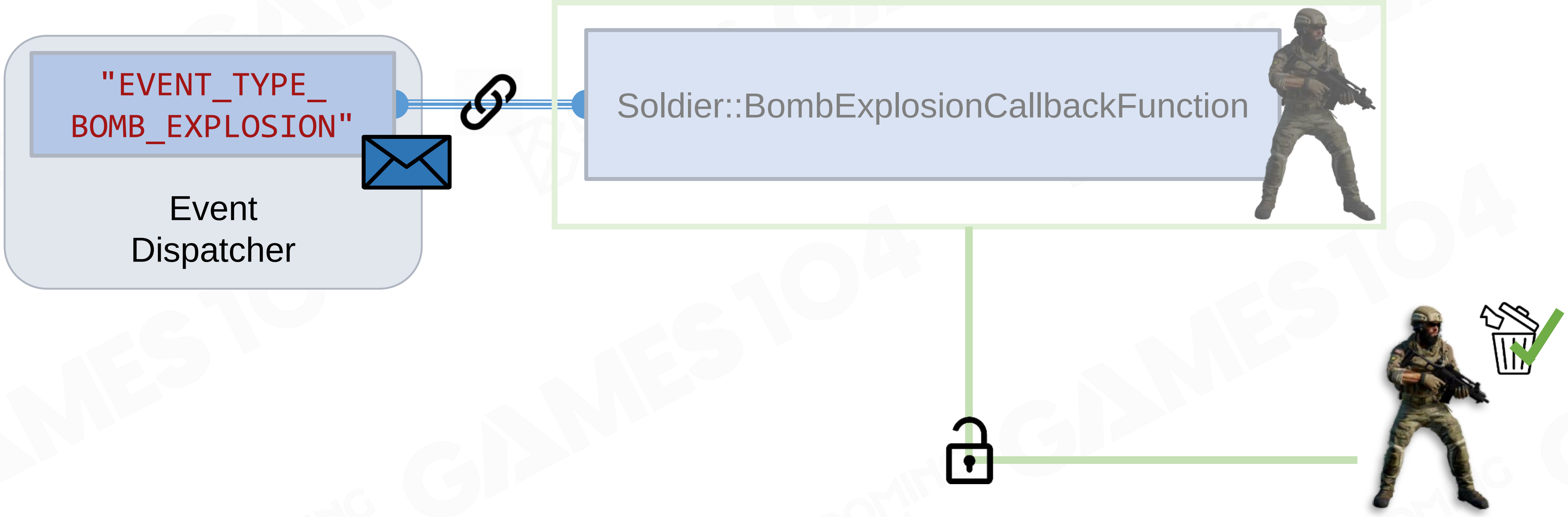
Event Dispatcher

Soldier::BombExplosionCallbackFunction

Make sure to **unregister** callback function before delete objects, otherwise it will cause **memory leak**!

Prevent object from de-allocation as long as callback function still registered

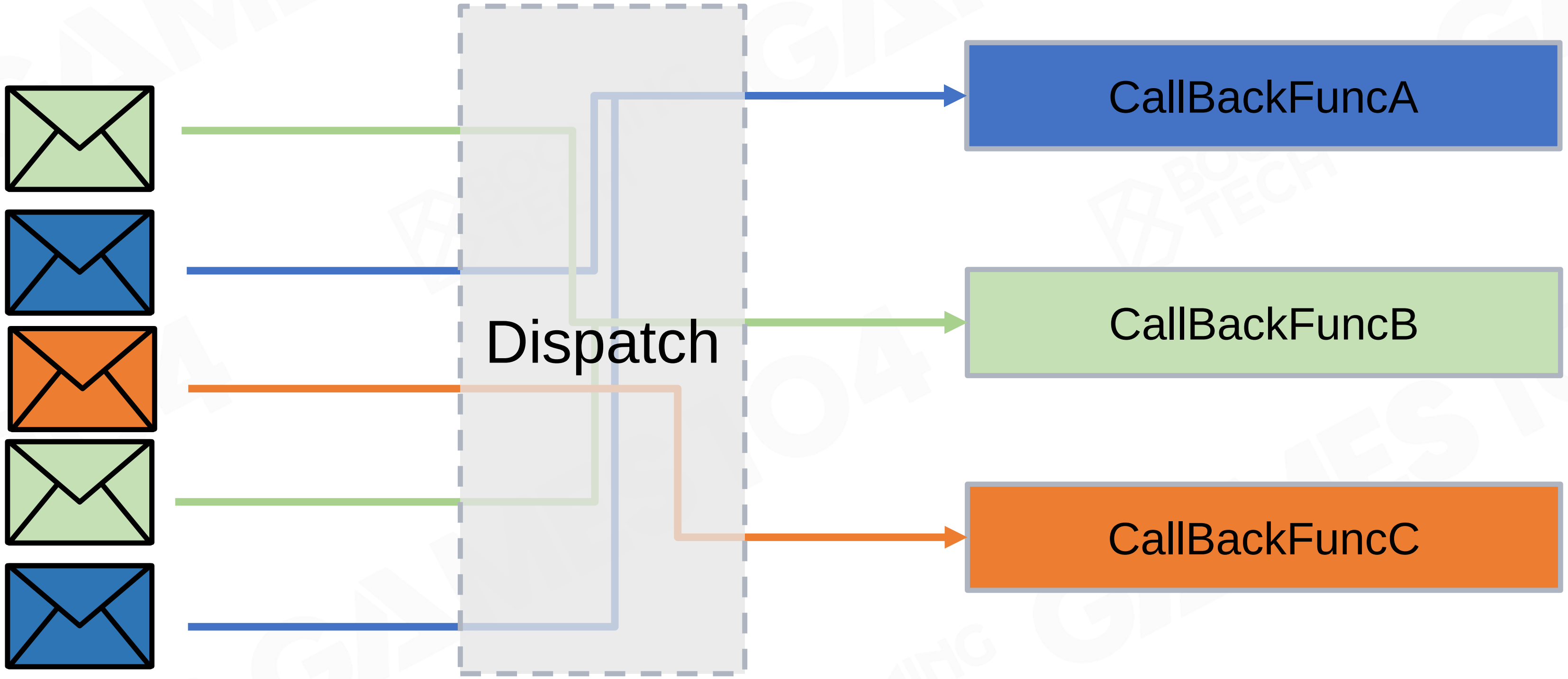
Object Weak Reference



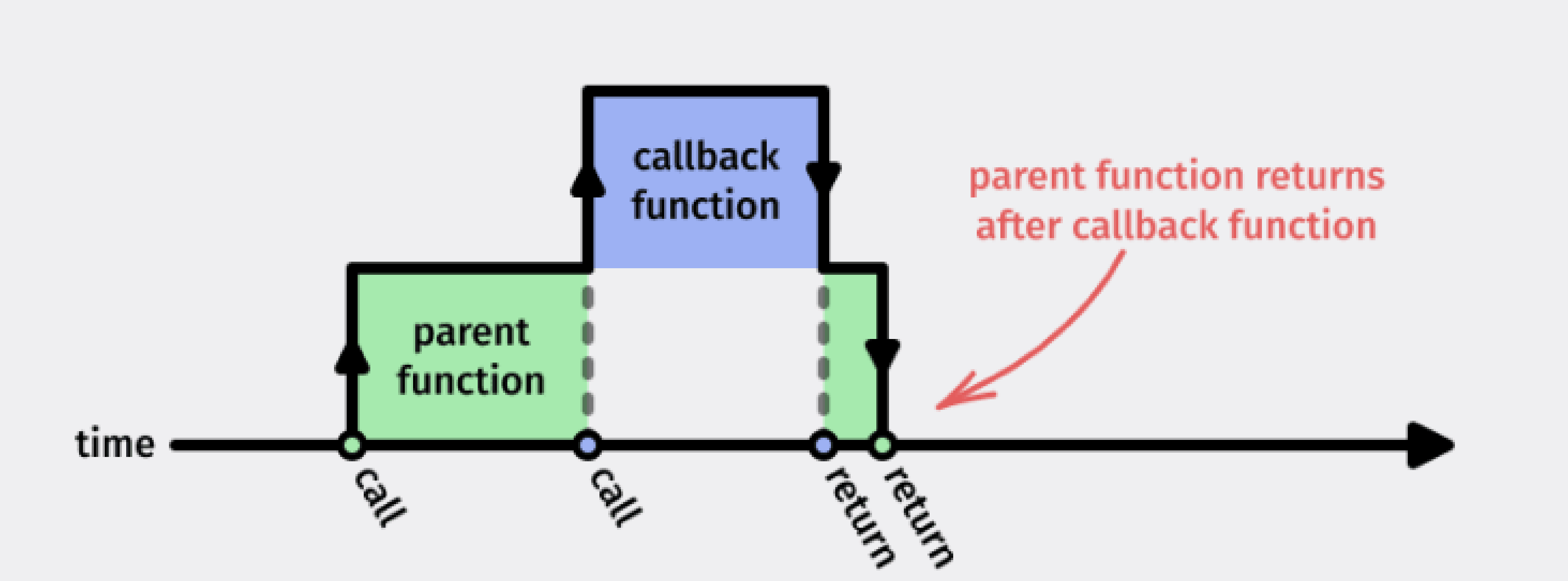
Object could be de-allocated, and will check callback function if valid

Event Dispatch

- Send event to appropriate destination



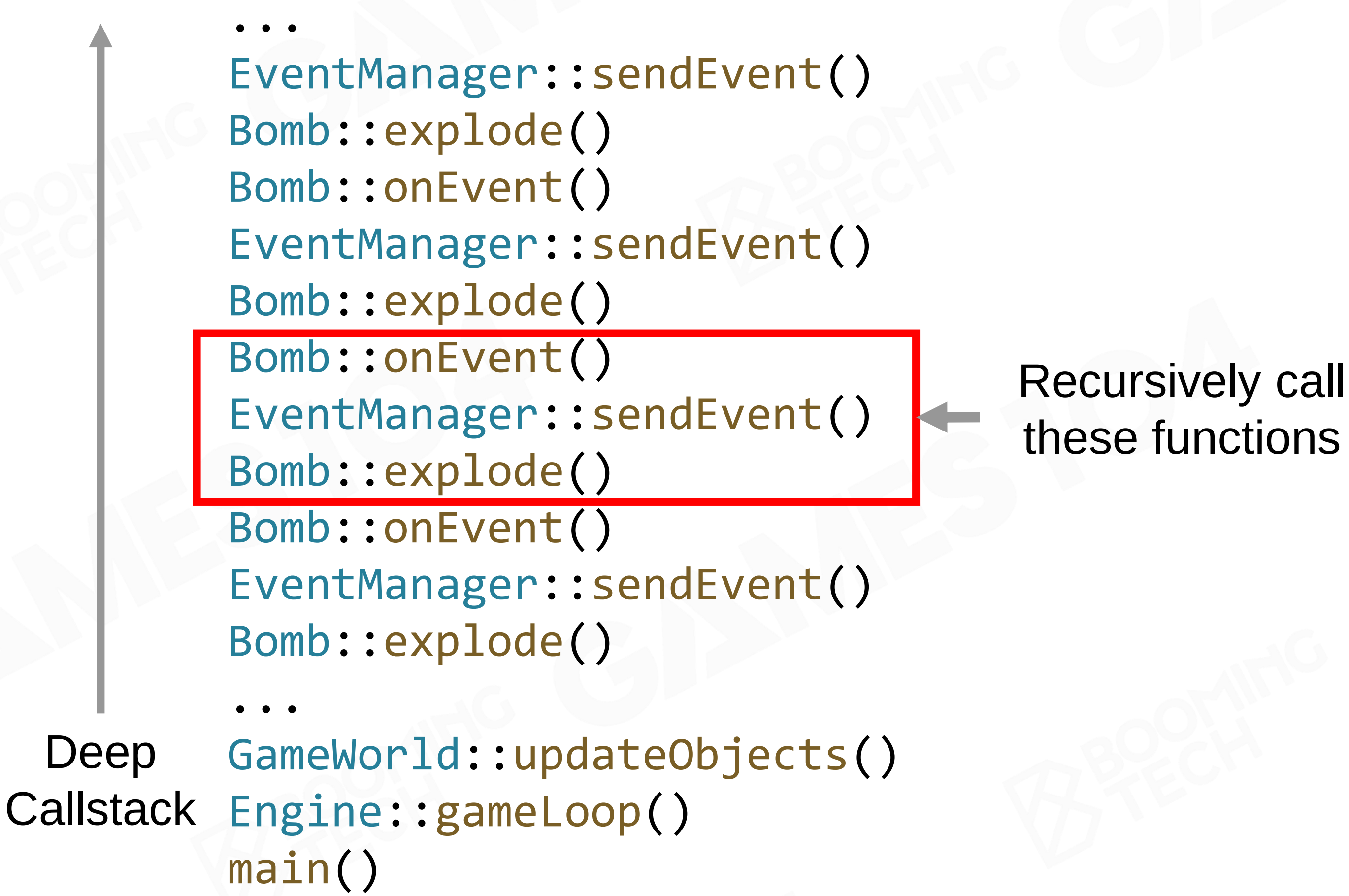
Event Dispatch : Immediate



parent function returns after callback function

Event Dispatch : Immediate

- Deep well of callbacks

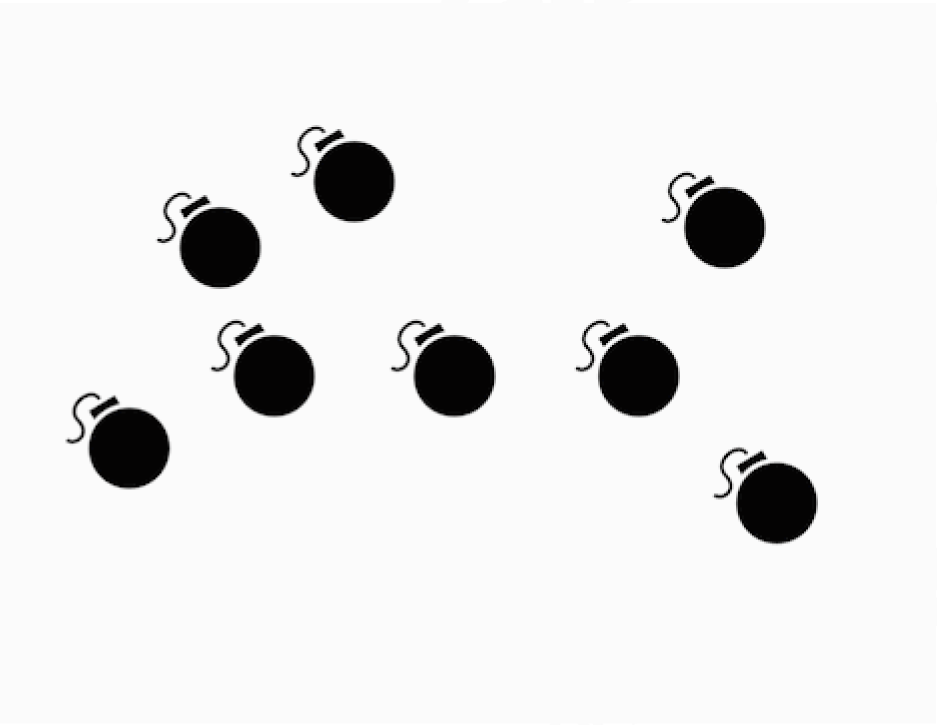


```
...
EventManager::sendEvent()
Bomb::explode()
Bomb::onEvent()
EventManager::sendEvent()
Bomb::explode()
Bomb::onEvent()
EventManager::sendEvent()
Bomb::explode()
Bomb::onEvent()
EventManager::sendEvent()
Bomb::explode()
...
GameWorld::updateObjects()
Engine::gameLoop()
main()
```

Deep Callstack

Recursively call these functions

When a bomb explodes near others......



Event Dispatch : Immediate

Problem

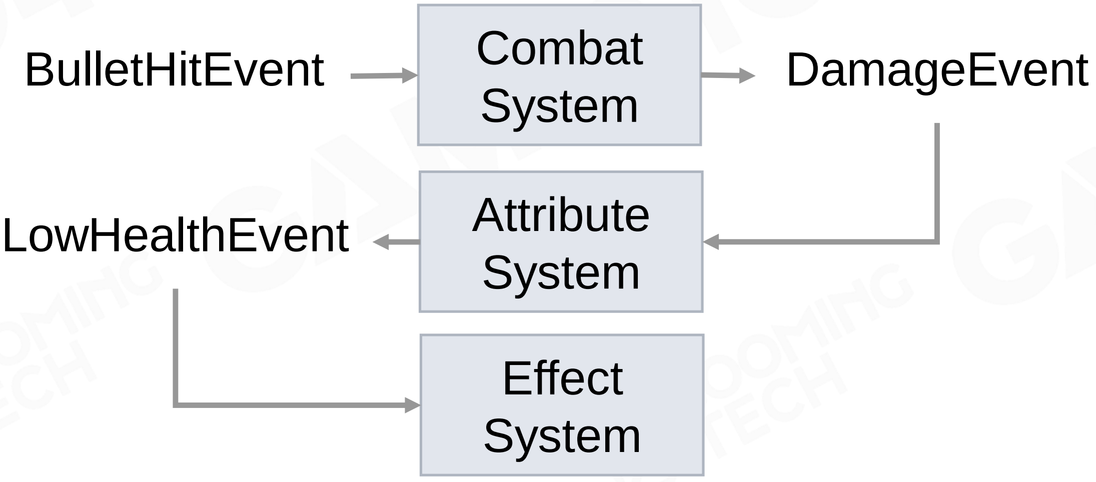
- Blocked by function

```
...
EventManager::sendEvent()
EffectSystem::addEffect()
EffectSystem::onEvent()
EventManager::sendEvent()
AttributeSystem::updateHealth()
AttributeSystem::onEvent()
EventManager::sendEvent()
CombatSystem::calculateDamage()
CombatSystem::onEvent()
GameWorld::updateSystem()
Engine::gameLoop()
main()
```

🚫 Blocked

The bleeding effect should be loaded
but cost plenty of time in this function call

Soldier begin
bleeding after hitted

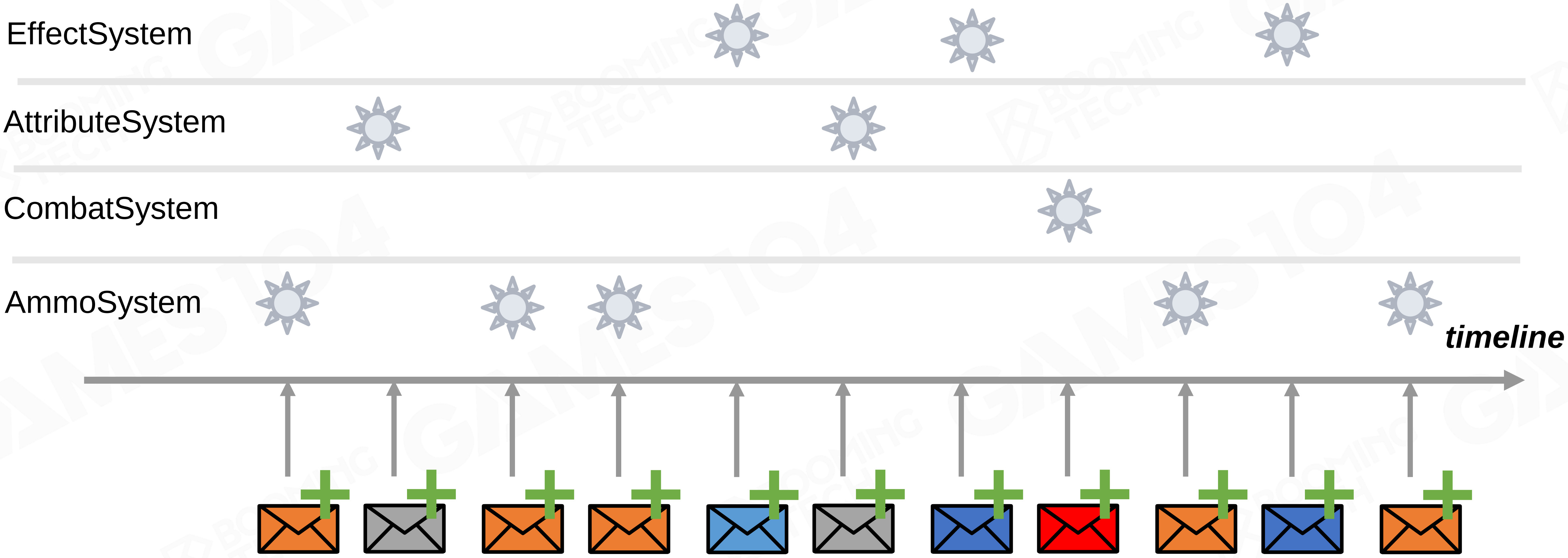


BulletHitEvent → Combat System → DamageEvent → Attribute System → LowHealthEvent → Effect System

Event Dispatch : Immediate

Problem

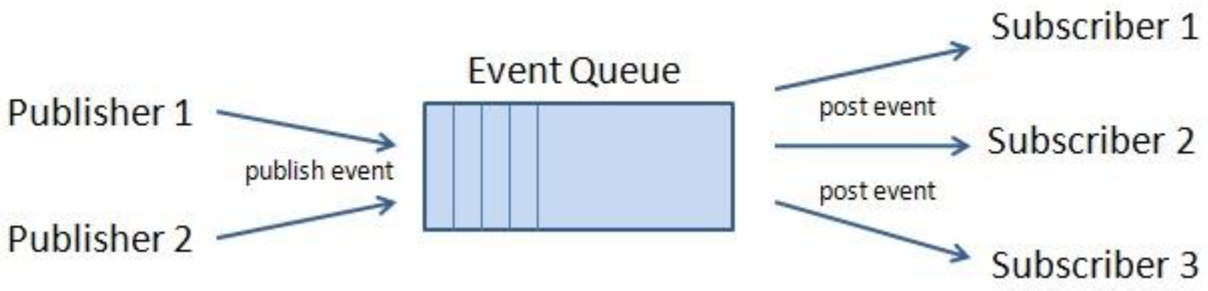
- Difficult for parallelization

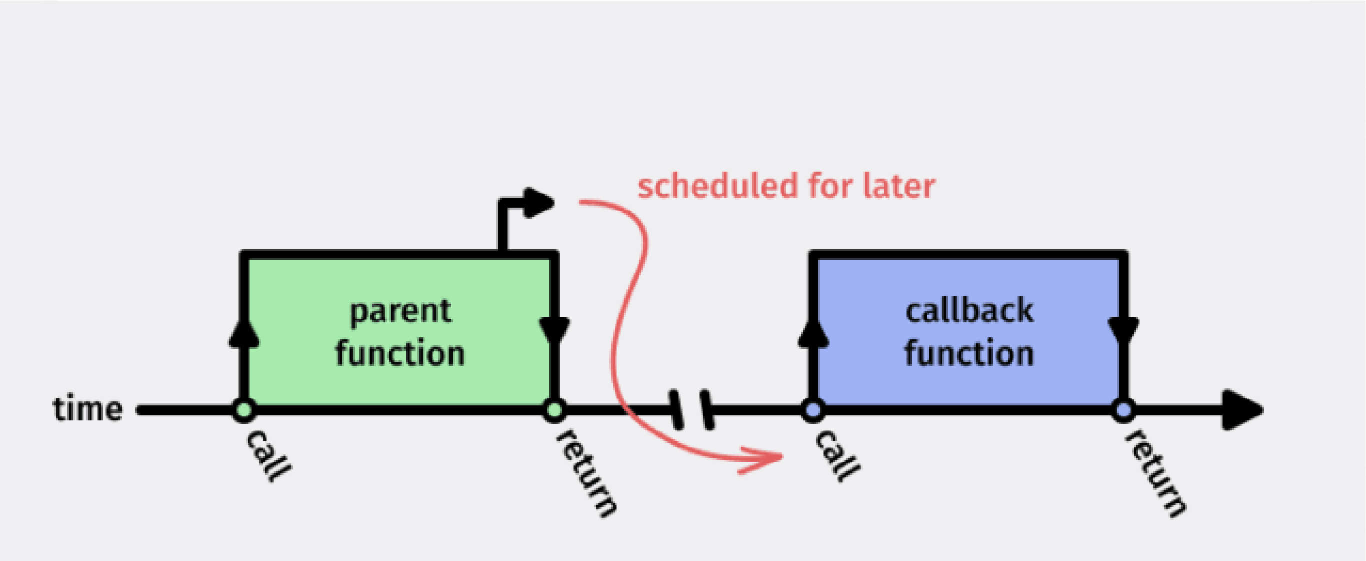


Event Queue

Basic Implementation

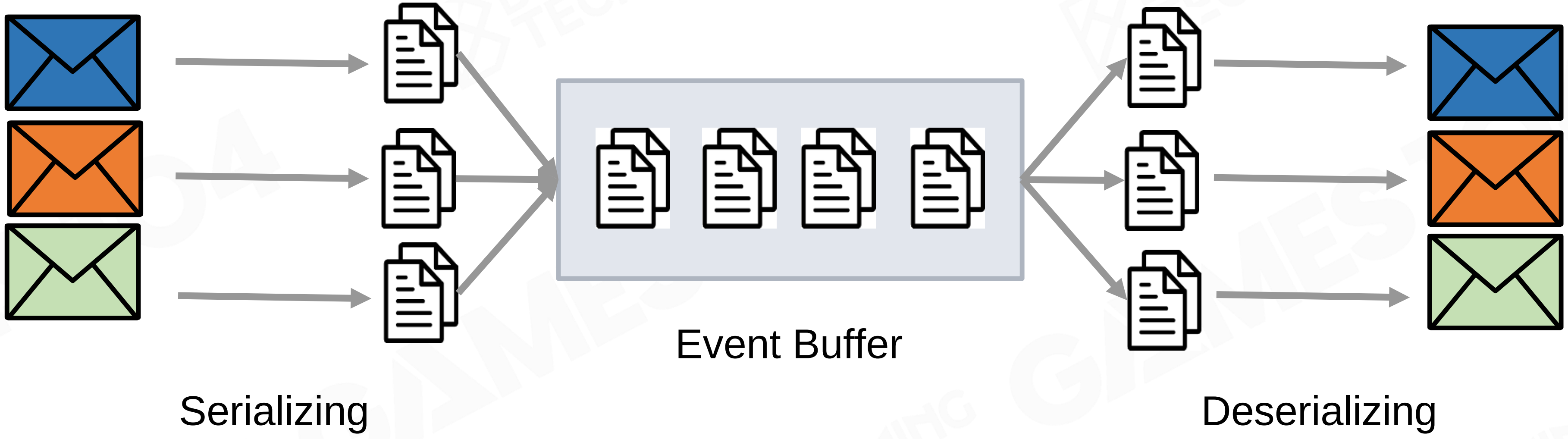
- Store events in queue for handling at an arbitrary future time





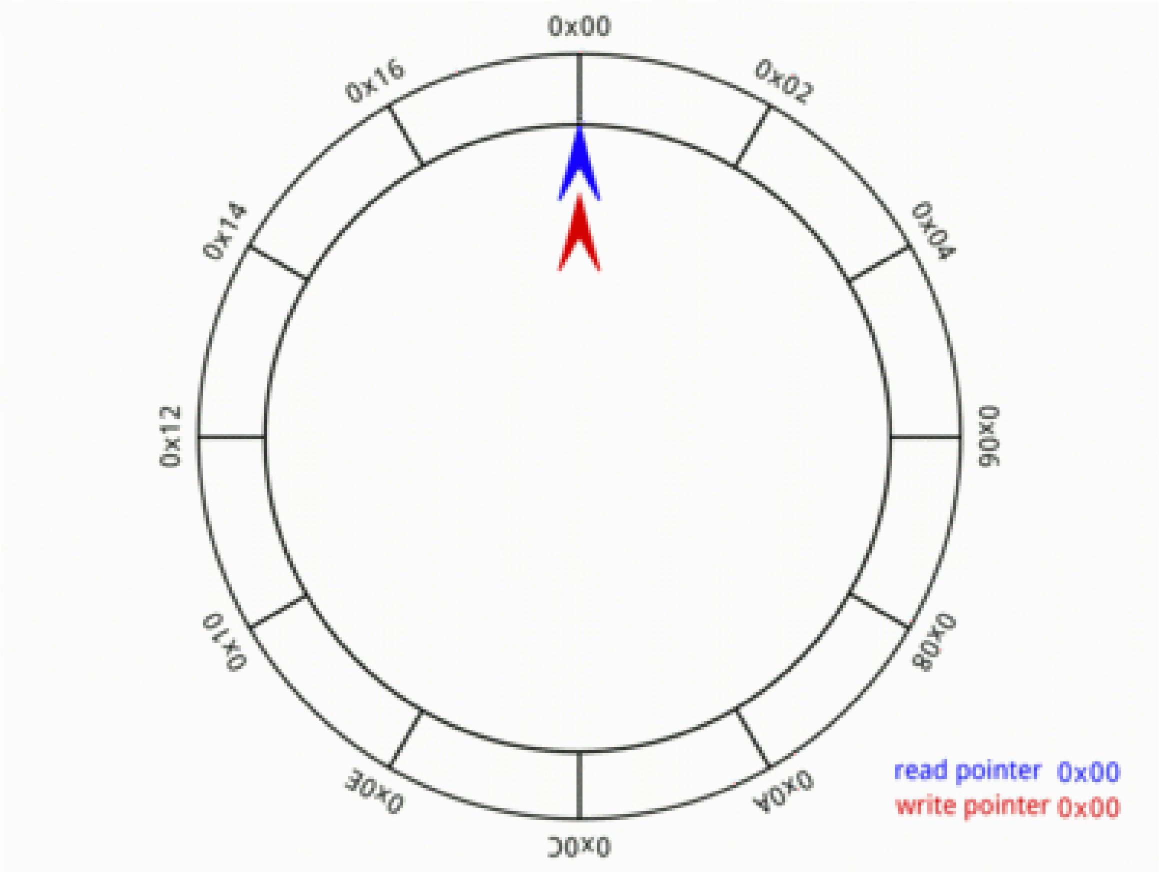
Event Serializing and Deserializing

- To store various types of events



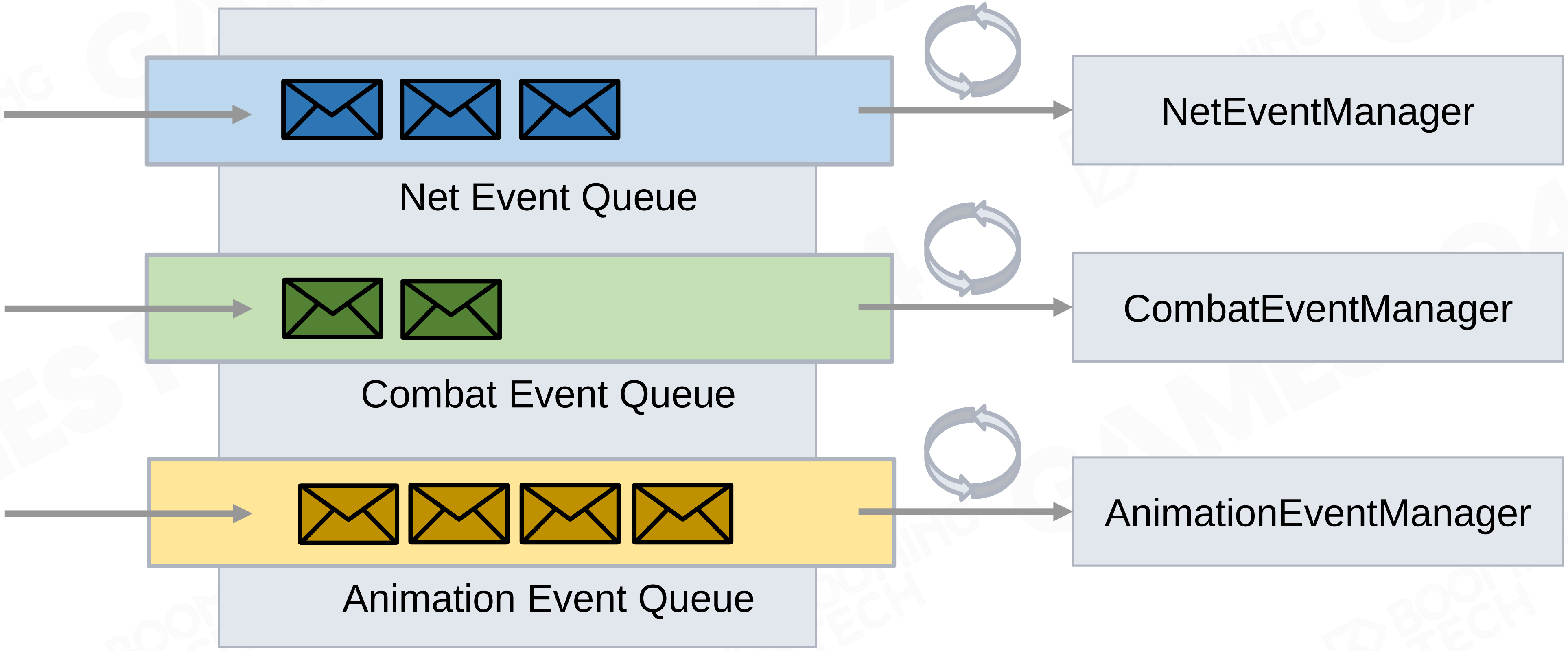
Event Queue

Ring buffer



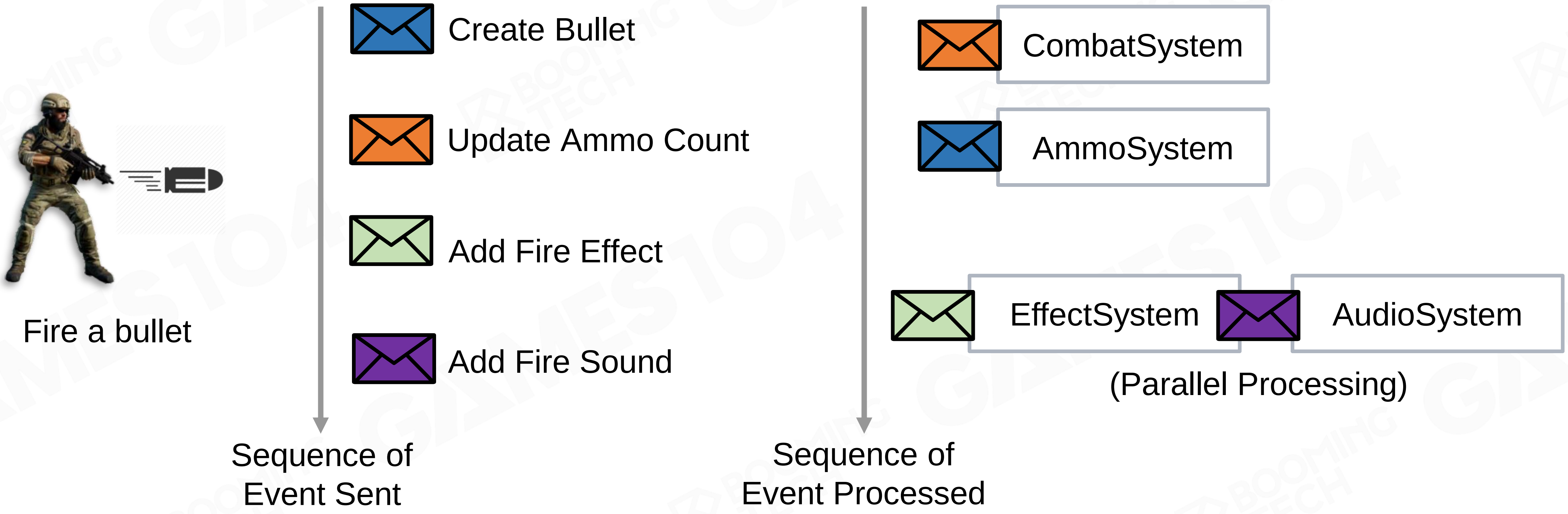
Event Queue

Batching



Problems of Event Queue (1/2)

- Timeline not determined by publisher



Fire a bullet

Create Bullet

Update Ammo Count

Add Fire Effect

Add Fire Sound

Sequence of Event Sent

CombatSystem

AmmoSystem

EffectSystem

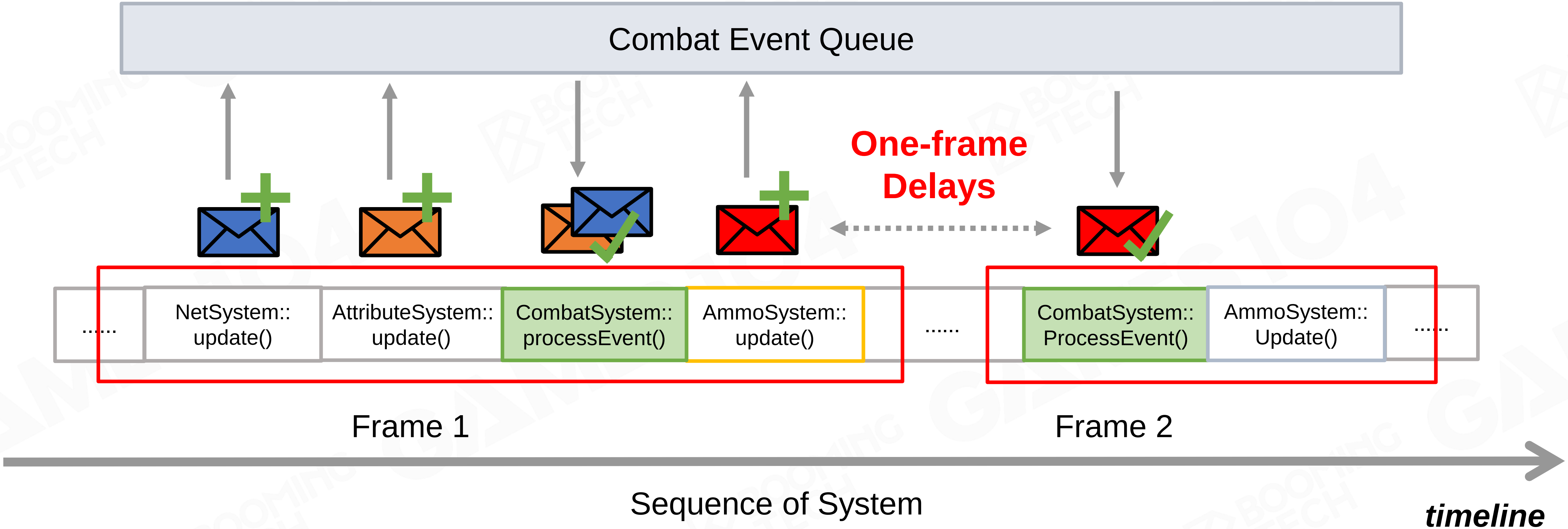
AudioSystem

(Parallel Processing)

Sequence of Event Processed

Problems of Event Queue (2/2)

- One-frame delays



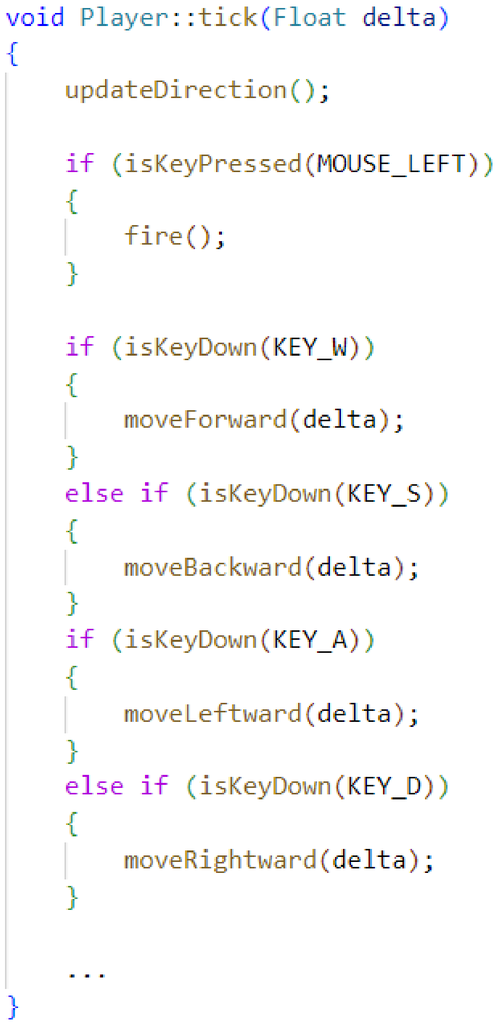
GamePlay

Game Logic

Early Stage Game Logic Programming

Compiled language(mostly C/C++)

- Compiled to machine code with high performance
- More easier to use than assembly language



```
void Player::tick(Float delta)
{
    updateDirection();

    if (isKeyPressed(MOUSE_LEFT))
    {
        fire();
    }

    if (isKeyDown(KEY_W))
    {
        moveForward(delta);
    }
    else if (isKeyDown(KEY_S))
    {
        moveBackward(delta);
    }
    if (isKeyDown(KEY_A))
    {
        moveLeftward(delta);
    }
    else if (isKeyDown(KEY_D))
    {
        moveRightward(delta);
    }

    ...
}
```

Problem of Compiled Languages

Game requirements get complex as hardware evolves

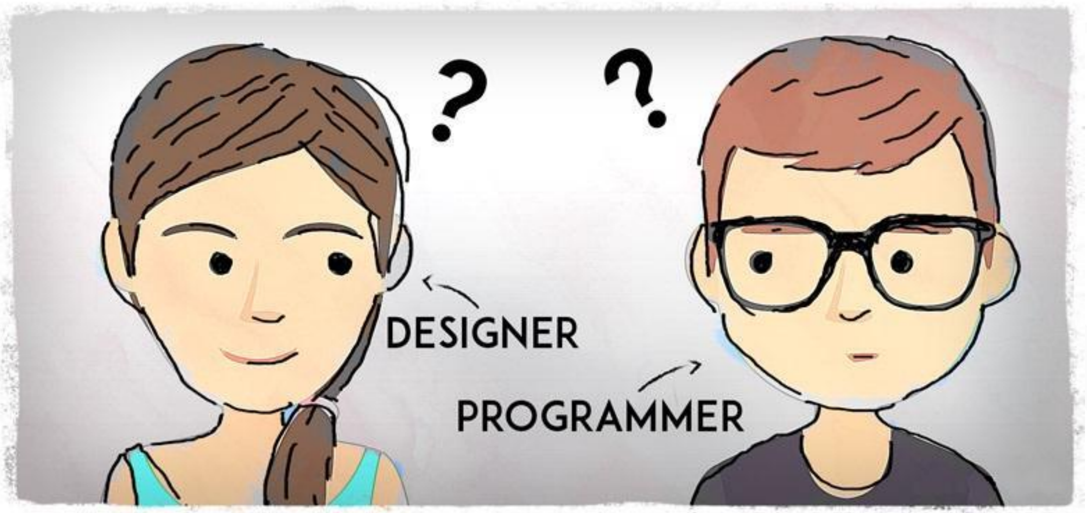
- Need quick iterations of gameplay logic

Issues with compiled language

- Need recompilation with even a little modification
- Program can easily get crashed with incorrect codes

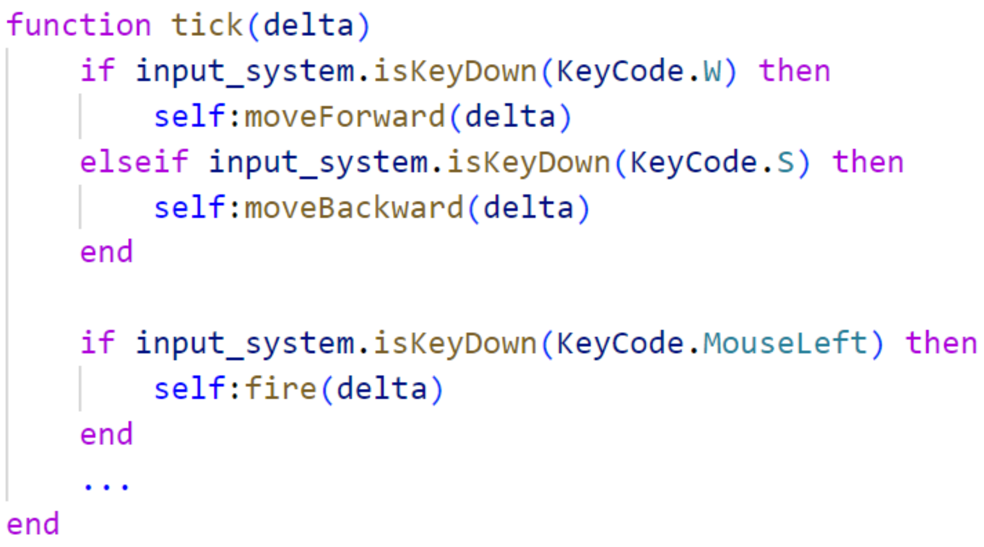
Glue Designers and Programmers

- Get rid of inefficient communication between designers and programmers
- Designers need direct control of gameplay logic
- Artists need to quickly adjust assets at the runtime environment



Scripting Languages

- Support for rapid iteration
- Easy to learn and write
- Support for hot update
- Stable, less crash by running in a sandbox

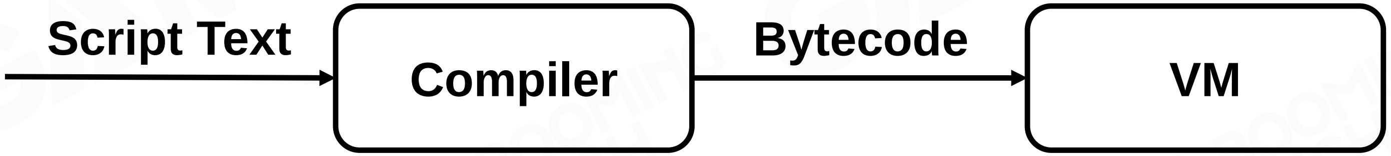


```lua
function tick(delta)
    if input_system.isKeyDown(KeyCode.W) then
        self:moveForward(delta)
    elseif input_system.isKeyDown(KeyCode.S) then
        self:moveBackward(delta)
    end

    if input_system.isKeyDown(KeyCode.MouseLeft) then
        self:fire(delta)
    end
    ...
end
```

Lua Script Example

How Script Languages Work

Script is converted to ***bytecode*** by a ***compiler*** first, then run on a ***virtual machine***



Instruction	Opcode	Description
NUM	0x00	Push a literal number
ADD	0x01	Pop two numbers and push the result of addtion
PRT	0x02	Pop a value and print

Instruction Set Example

Script: print(36 + 15)

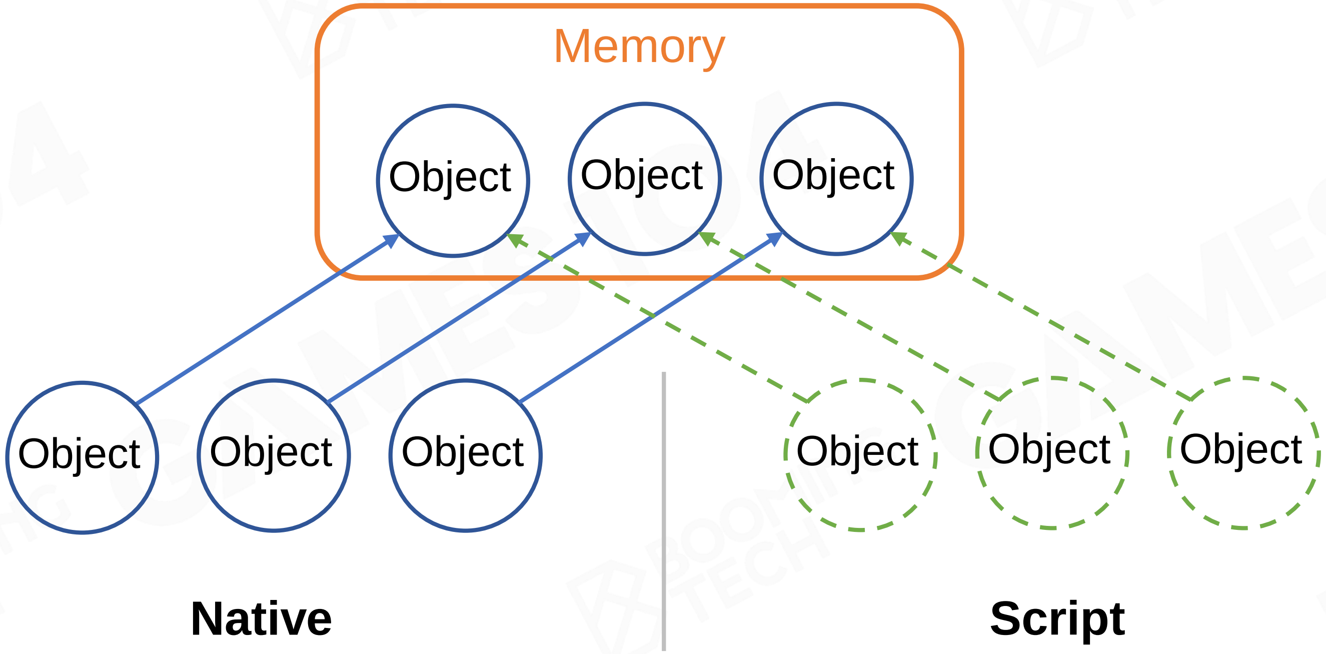
Bytecode:	0x00	0x24	0x00	0x0F	0x01	0x02
Instruction:	NUM	36	NUM	15	ADD	PRT

Bytecode Example

Object Management between Scripts and Engine (1/2)

Object lifetime management in ***native engine code***

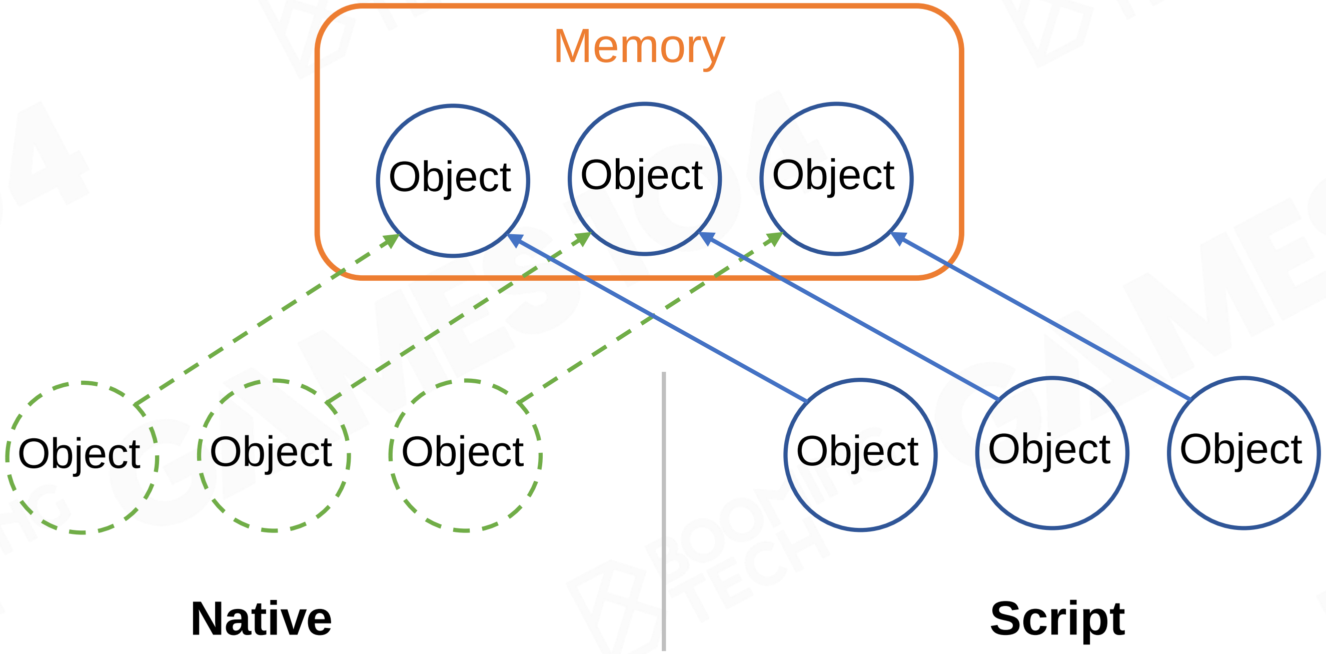
- Need to provide an object lifetime management mechanism
- Not safe when script uses native objects (may have been destructed)



Object Management between Scripts and Engine (2/2)

Object lifetime management in ***script***

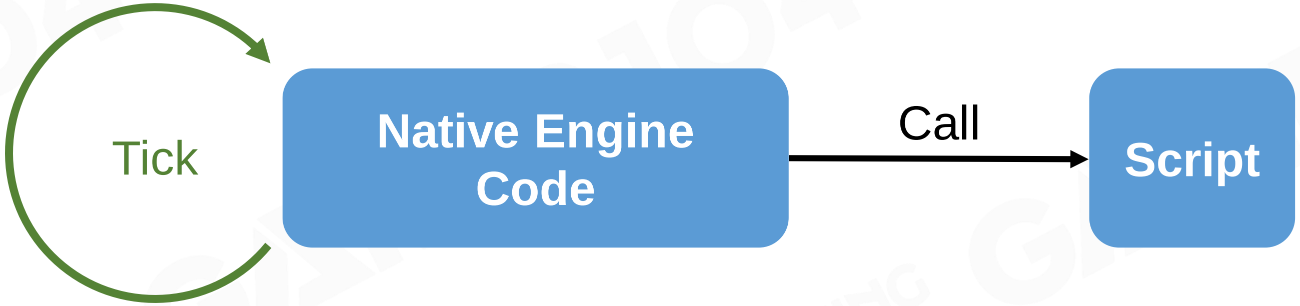
- The lifetime of objects are auto managed by script GC
- The time when object is deallocated is uncontrolled (controlled by GC)
- Easy to get memory leak if reference relations get complex in script



Architectures for Scripting System (1/2)

Native language dominants the game world

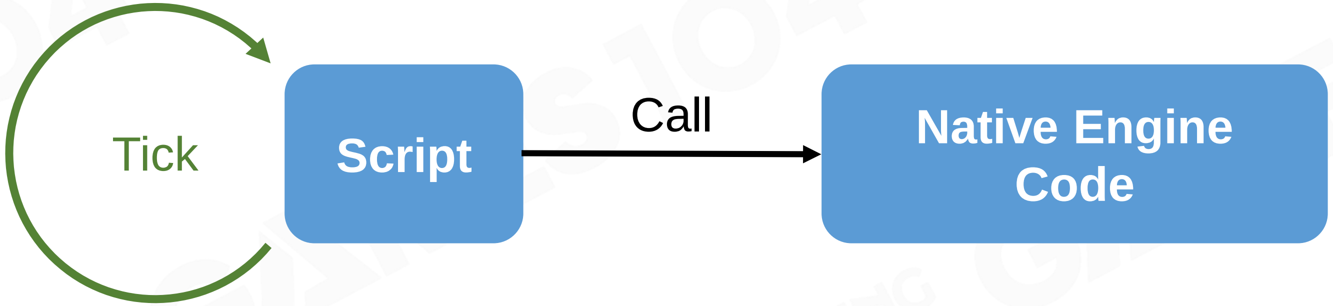
- Most gameplay logic is in native code
- Script extends the functionality of native engine code
- High performance with compiled language



Architectures for Scripting System (2/2)

Script language dominants the game world

- Most gameplay logic is in script
- Native engine code provides necessary functionality to script
- Quick development iteration with script language



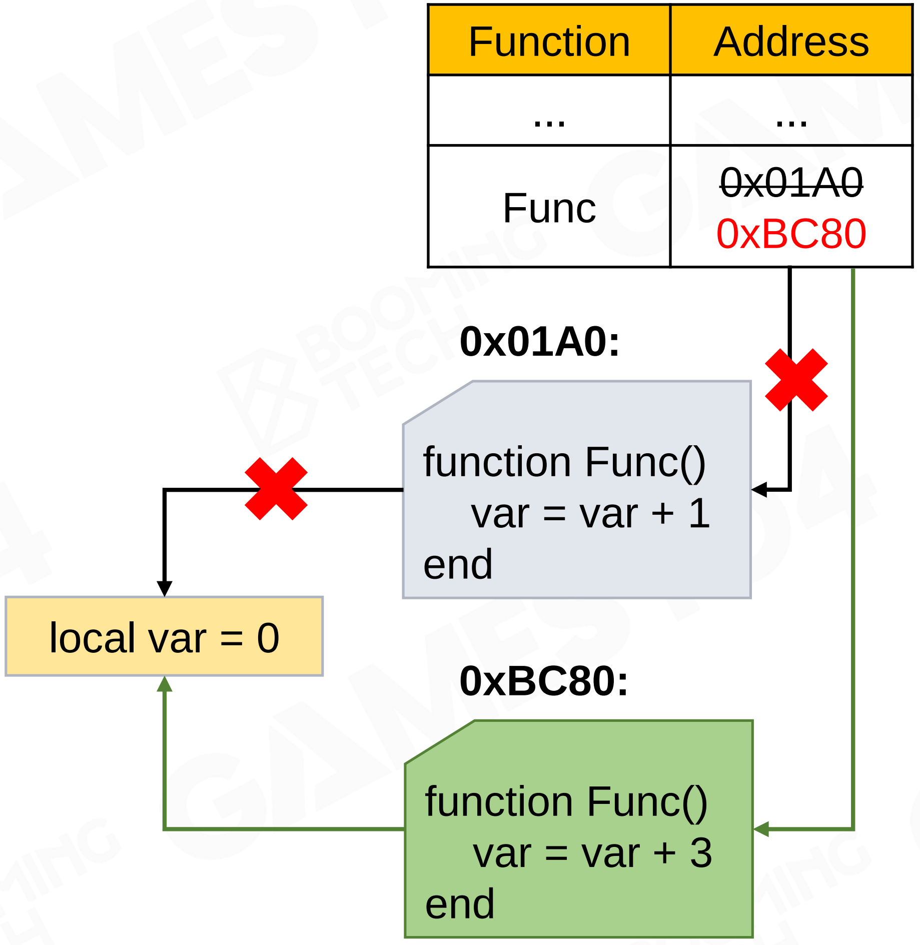
Advanced Script Features - Hot Update

Allow modifications of script while game is running

- Quick iteration for some specific logic
- Enable to fix bugs in script while game is online

A troublesome problem with hot update

- All variables reference to old functions should be updated too



Function	Address
...	...
Func	~~0x01A0~~ 0xBC80

0x01A0:

```
function Func()
    var = var + 1
end
```

0xBC80:

```
function Func()
    var = var + 3
end
```

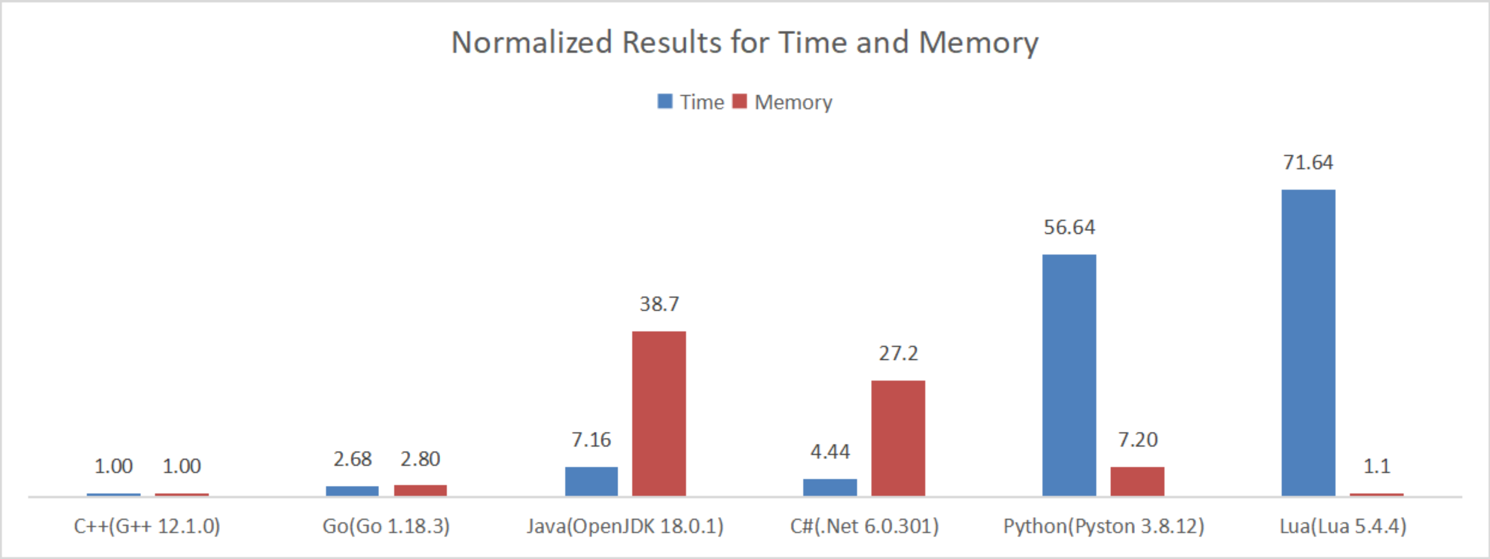
Hot update workflow example

Issues with Script Language

The **performance** is usually lower than compiled language

- Weakly typed language is usually harder to optimize when compile
- Need a virtual machine to run the bytecode
- JIT is a solution for optimization

Weakly typed language is usually harder to refactor



N-body problem benchmark of popular languages

Make a Right Choice of Scripting Language

Things need to be considered

- Language performance
- Built-in features, e.g. object-oriented programming support

Select the proper architecture of scripting

- Object lifetime management in native engine code or script
- Which one is dominant, native language or script

Popular Script Languages (1/2)

Lua (used in *World of Warcraft*, *Civilization V*)

- Robust and mature
- Excellent runtime performance
- Light-weighted and highly extensible

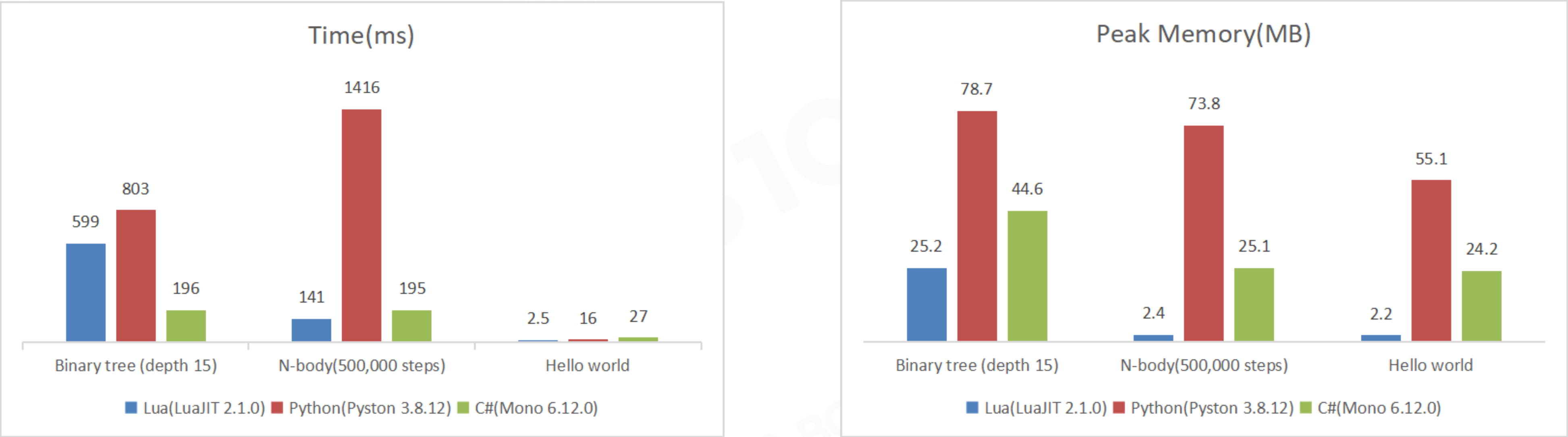
Python (used in *The Sims 4*, *EVE Online*)

- Reflection support
- Built-in object-oriented support
- Extensive standard libraries and third-party modules

Popular Script Languages (2/2)

C# (to bytecode offline, used in *Unity*)

- Low learning curve, easy to read and understand
- Built-in object-oriented support
- Great community with lots of active developers



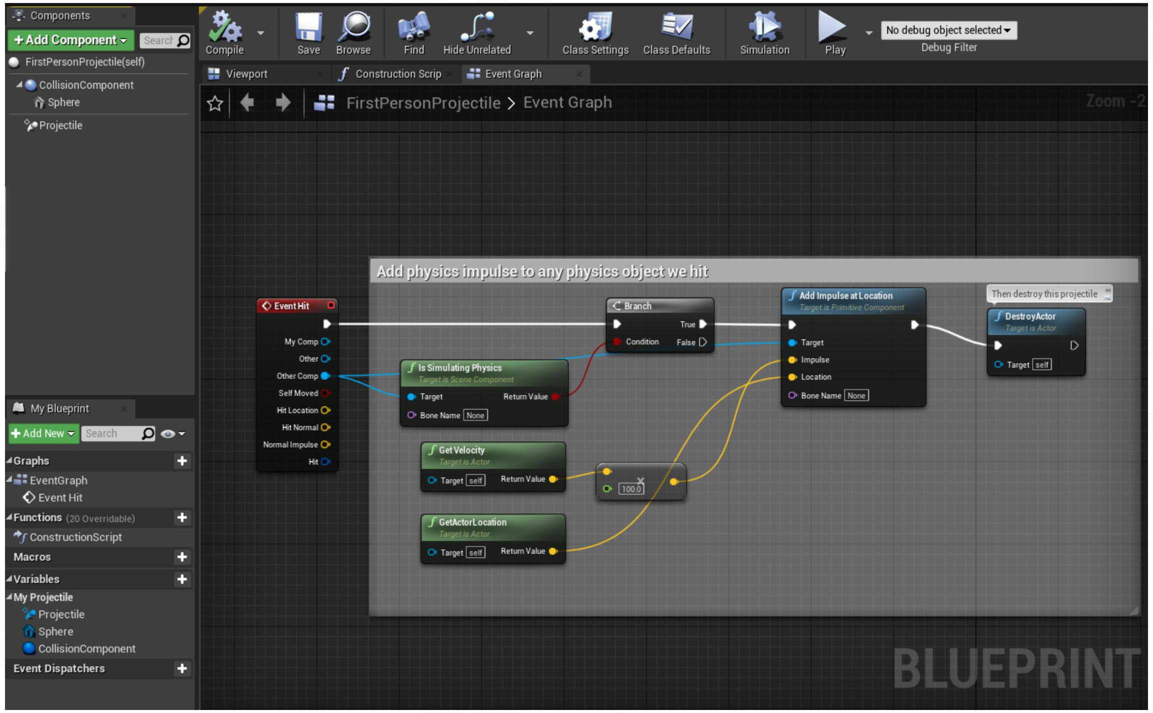
Benchmark of 3 popular script languages

GamePlay

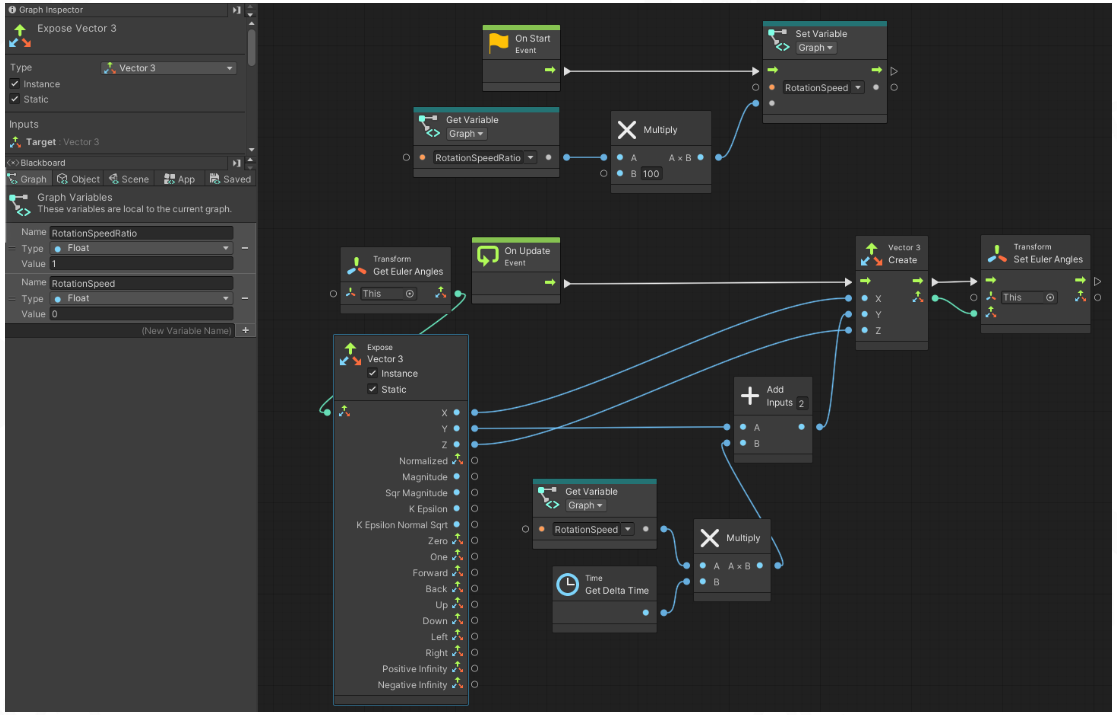
Visual Scripting

Why We Need Visual Scripting

- Friendly to non-programmers, especially designers and artists
- Less error-prone with drag-drop operations instead of code writing



Unreal Blueprint

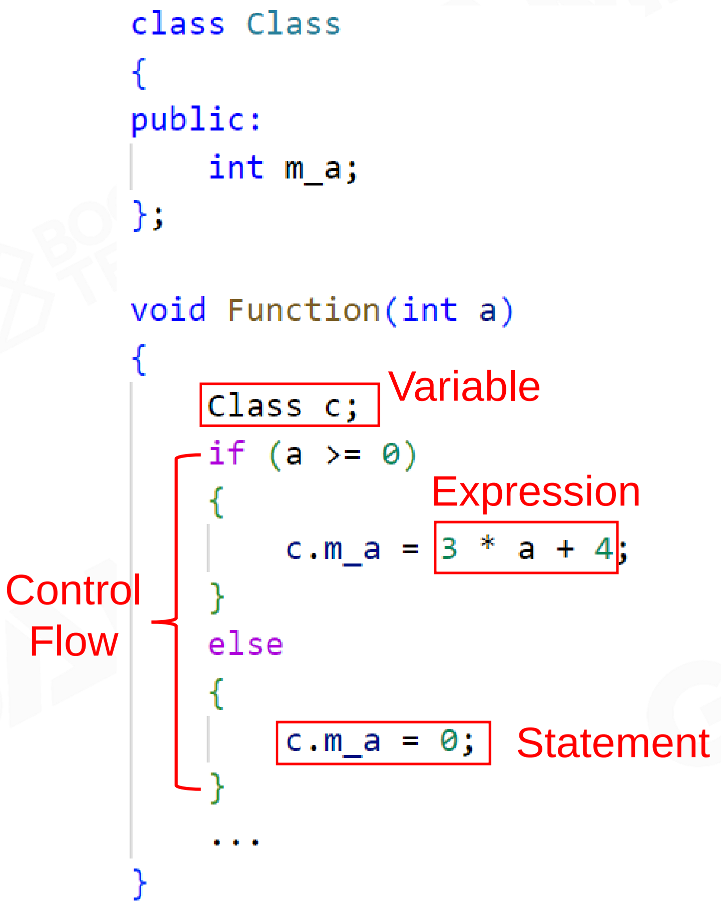


Unity Visual Scripting

Visual Script is a Program Language

Visual script is also a programming language, which usually needs

- Variable
- Statement and Expression
- Control Flow
- Function
- Class (for object-oriented programming language)

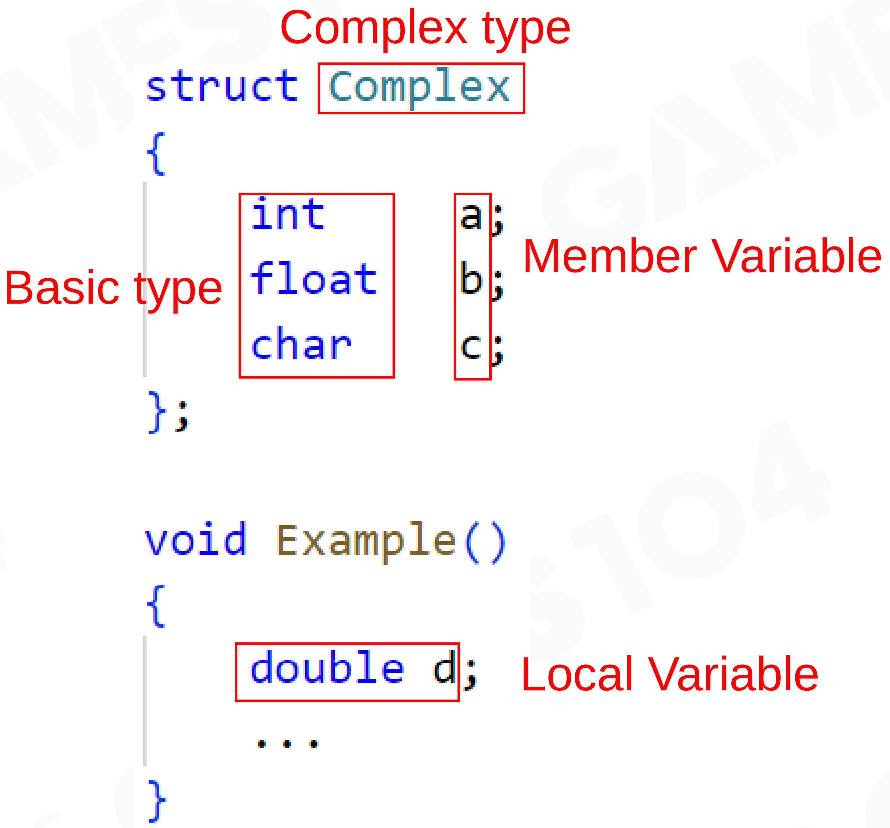


```
class Class
{
public:
    int m_a;
};

void Function(int a)
{
    Class c;
    if (a >= 0)
    {
        c.m_a = 3 * a + 4;
    }
    else
    {
        c.m_a = 0;
    }
    ...
}
```

Variable

Preserve the data to be processed or output

- Type
  - Basic type, e.g. integer, floating
  - Complex type, e.g. structure
- Scope
  - Local variable
  - Member variable
  - ...

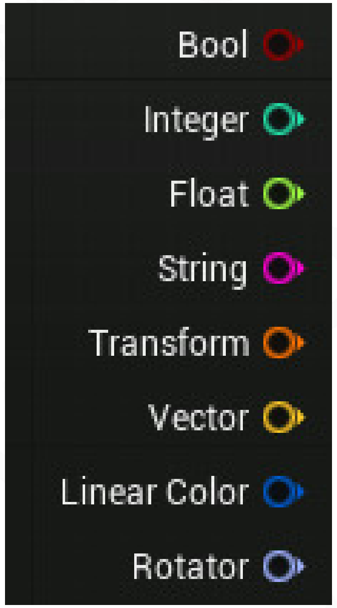


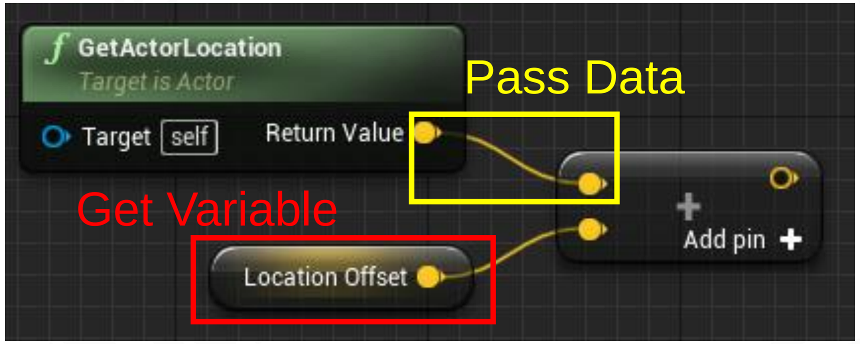
```
struct Complex
{
    int   a;
    float b;
    char  c;
};

void Example()
{
    double d;
    ...
}
```

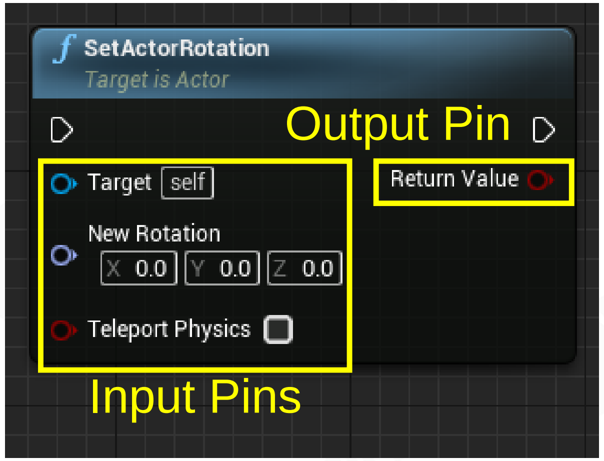
Variable Visualization - Data Pin and Wire

Use ***data wires*** through ***data pins*** to pass ***variables*** (parameters)

- Each data type uses a unique pin color



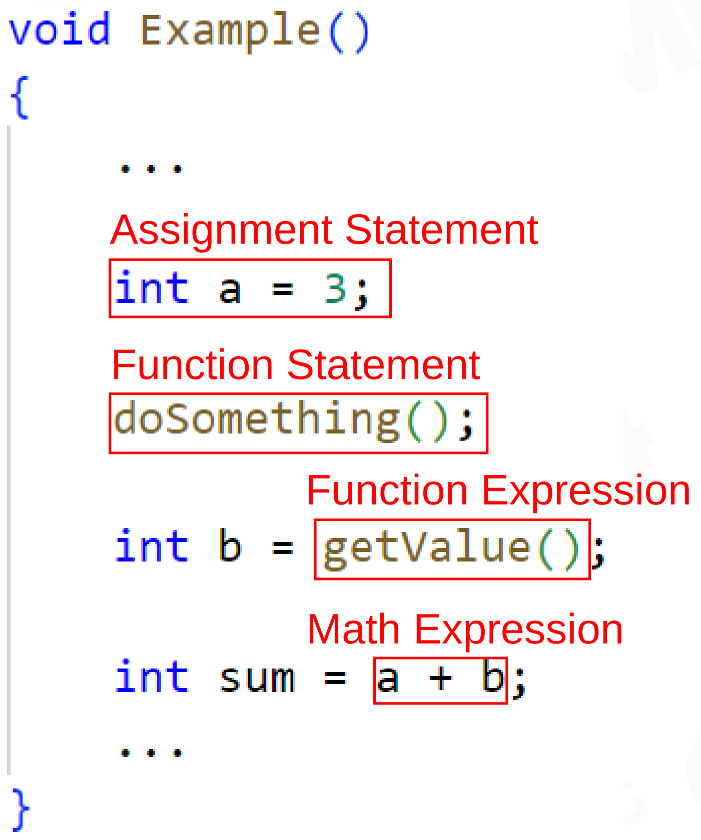




Statement and Expression

Control how to process data

- Statement: expresses some action to be carried out
    - Assignment Statement
    - Function Statement
    - ...
- Expression: to be evaluated to determine its value
    - Function Expression
    - Math Expression
    - ...

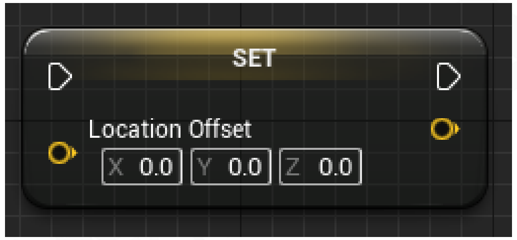


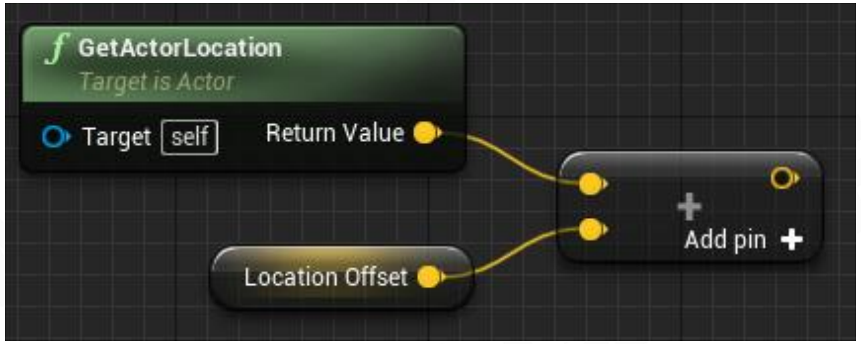
```
void Example()
{
    ...
    int a = 3;
    doSomething();
    int b = getValue();
    int sum = a + b;
    ...
}
```

Statement and Expression Visualization - Node

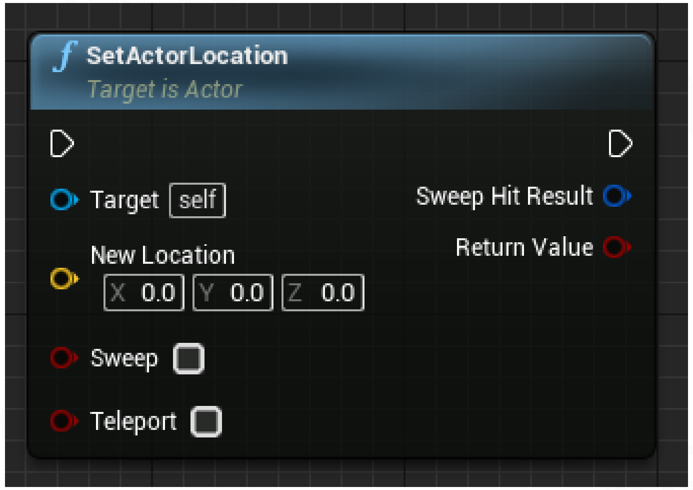
Use ***nodes*** to represent ***statements*** and ***expressions***

- Statement Node
- Expression Node





Expression Nodes

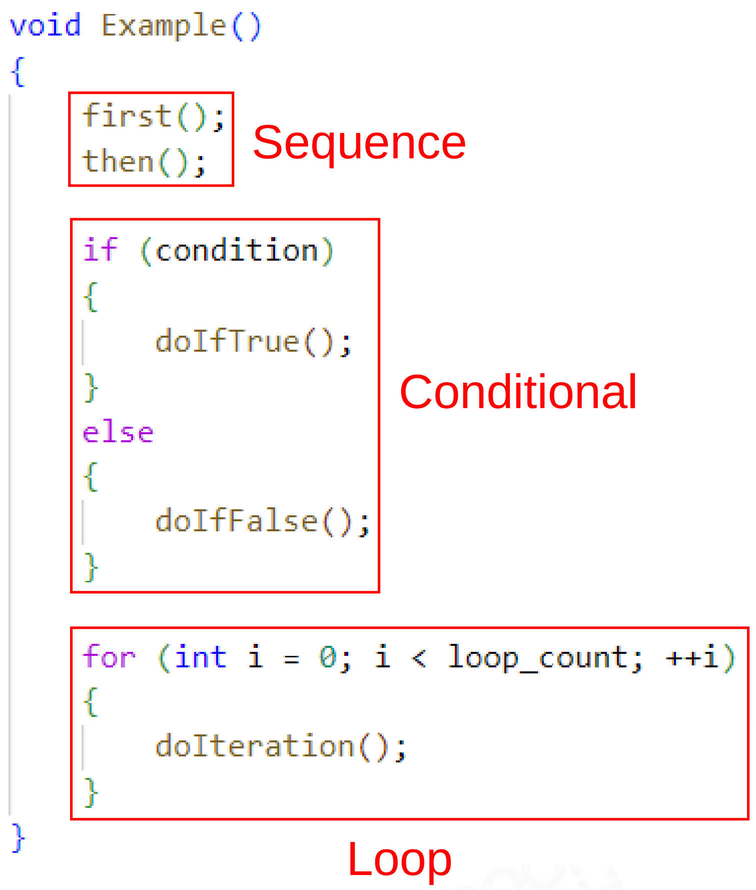


Statement Nodes

Control Flow

Control the statement execution order

- Sequence
  - By default statements are executed one by one
- Conditional
  - Next statement is decided by a condition
- Loop
  - Statements are executed iteratively until the condition is not true



```
void Example()
{
    first();
    then();

    if (condition)
    {
        doIfTrue();
    }
    else
    {
        doIfFalse();
    }

    for (int i = 0; i < loop_count; ++i)
    {
        doIteration();
    }
}
```

Sequence

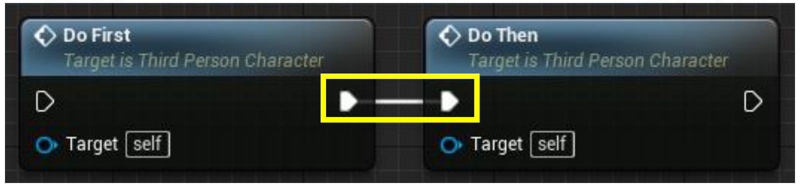
Conditional

Loop

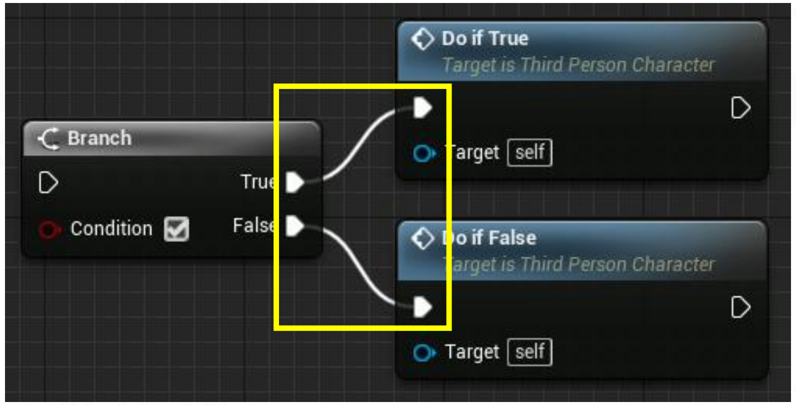
Control Flow Visualization - Execution Pin and Wire

Use ***execution wires*** through ***execution pins*** to make ***statements*** sequence

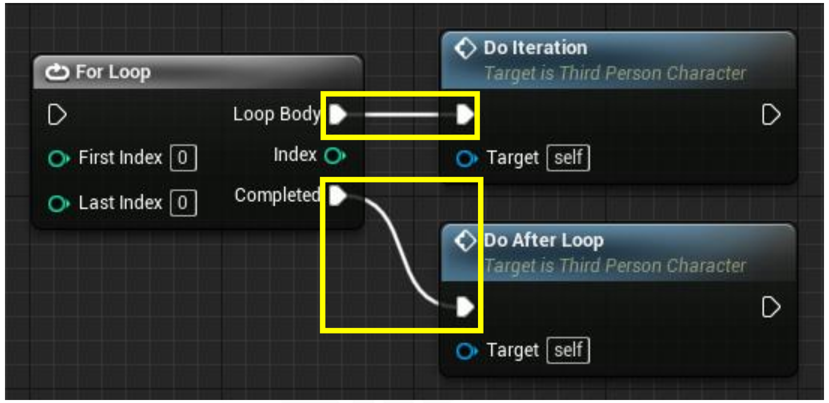
- Use control statement nodes to make different control flow


Do First
Target is Third Person Character
Target self
Do Then
Target is Third Person Character
Target self

Sequence



Conditional

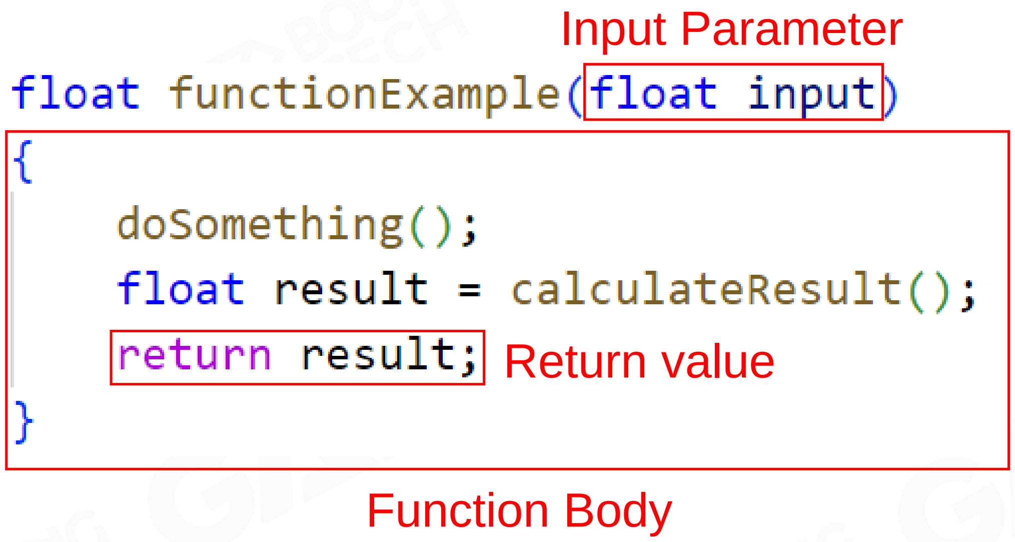


Loop

Function

A logic module which take in data, process it and return result(s)

- Input Parameter
  - The data required input to be processed
- Function Body
  - Control how to process data
- Return value(s)
  - The data to be returned



Input Parameter

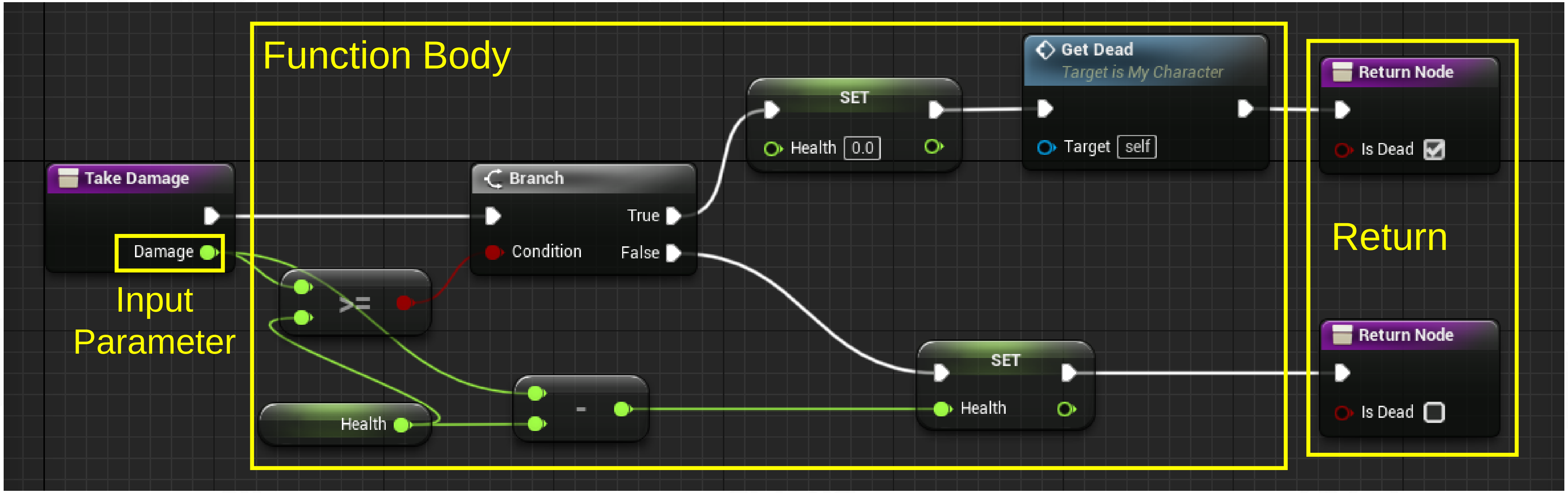
```
float functionExample(float input)
{
    doSomething();
    float result = calculateResult();
    return result;
}
```

Return value

Function Body

Function Visualization - Function Graph

Use a ***graph*** with connected nodes to make a function

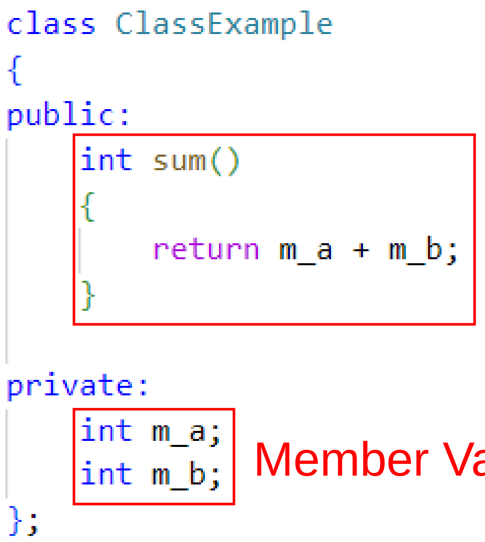


Example: Define a Function with Graph

Class

A prototype for a kind of objects

- Member Variable
  - The lifetime is managed by the object instance
- Member Function
  - Can access member variables directly
  - Maybe overrided by derived classes



```
class ClassExample
{
public:
    int sum()
    {
        return m_a + m_b;
    }

private:
    int m_a;
    int m_b;
};
```

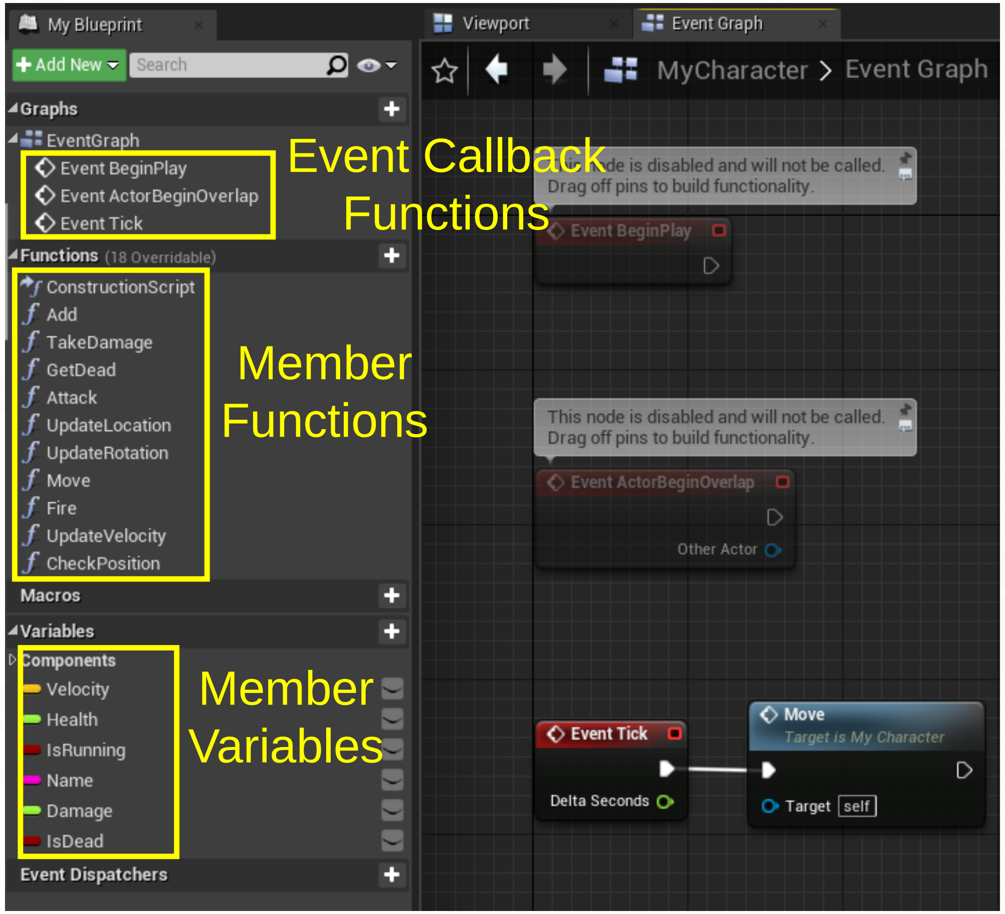
Member Function

Member Variables

Class Visualization - Blueprint

Use ***blueprint*** to define a class that inherits from a native class

- Event Callback Functions
- Member Functions
- Member Variables
- ...

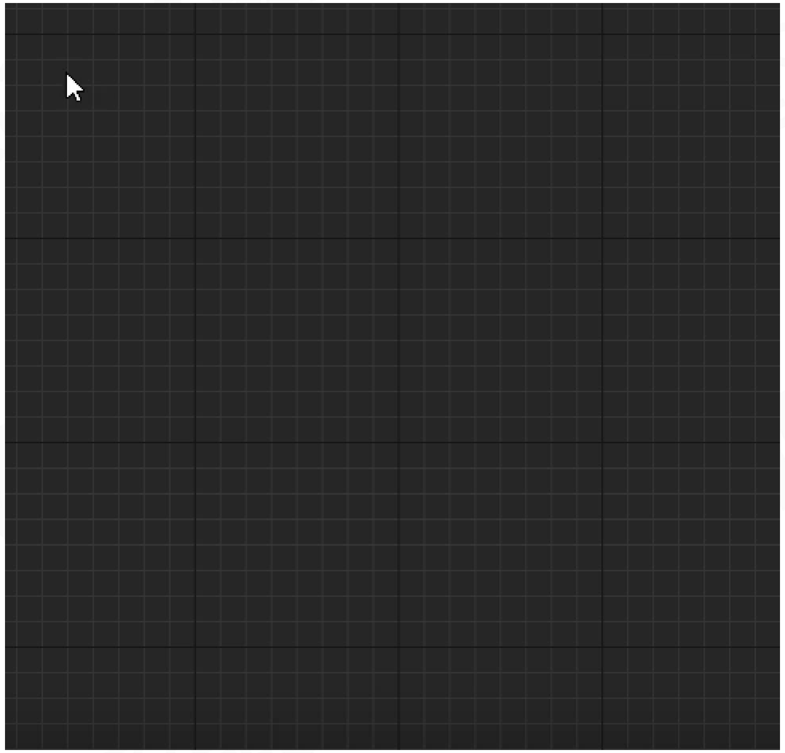


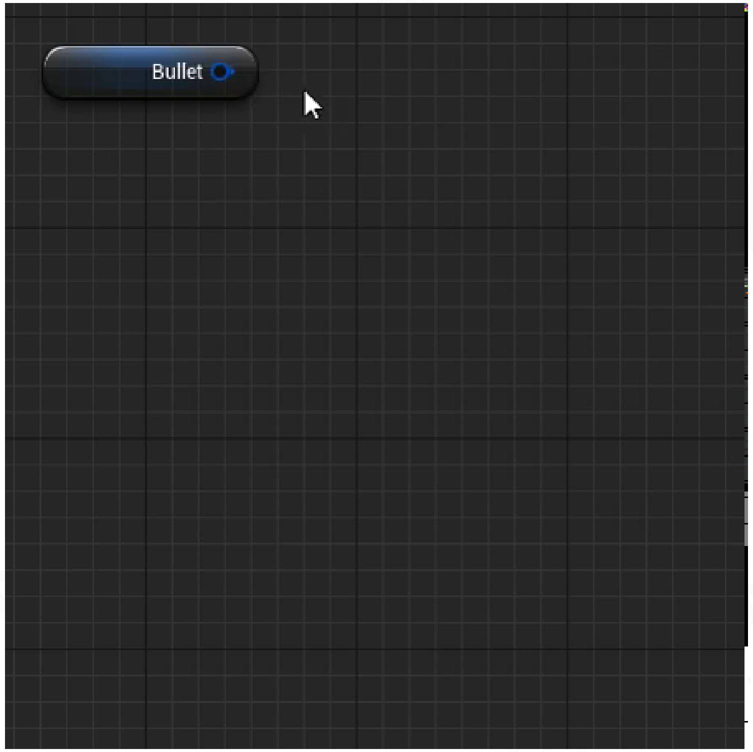
This node is disabled and will not be called. Drag off pins to build functionality.

This node is disabled and will not be called. Drag off pins to build functionality.

Make Graph User Friendly

- Fuzzy finding
- Accurate suggestions by type

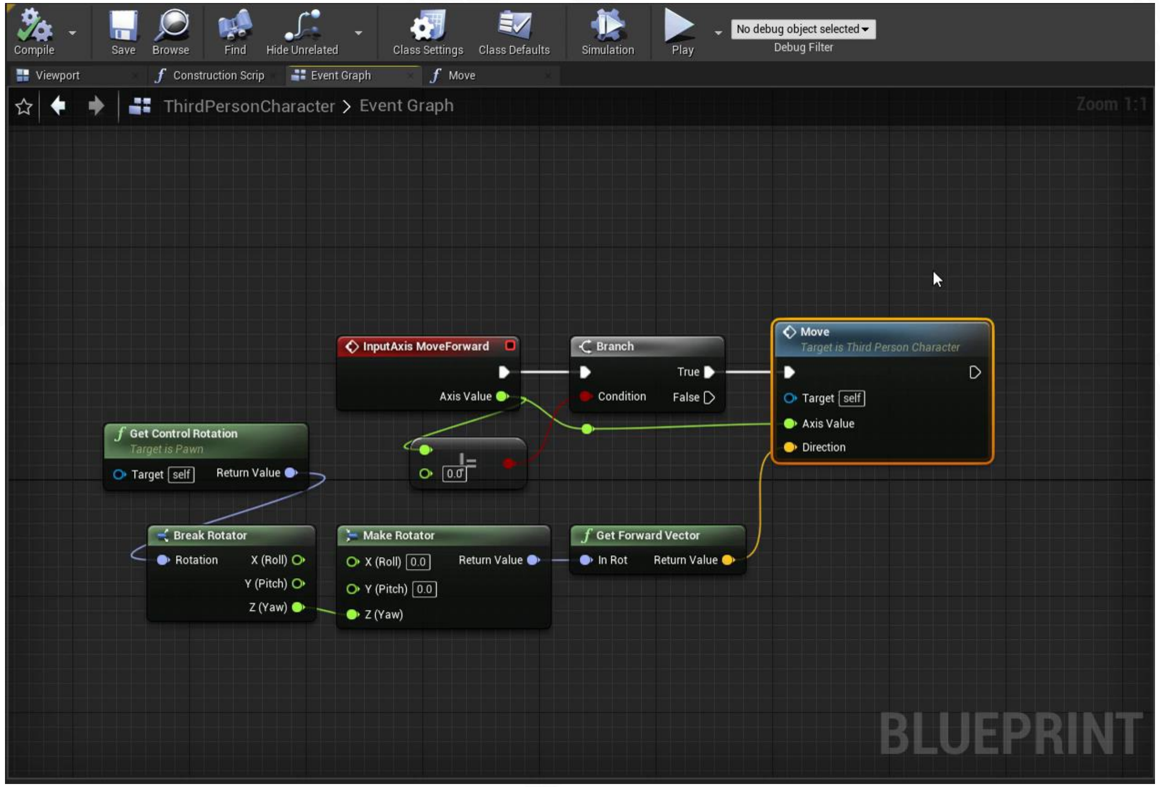




Visual Script Debugger

Debug is an important step among development

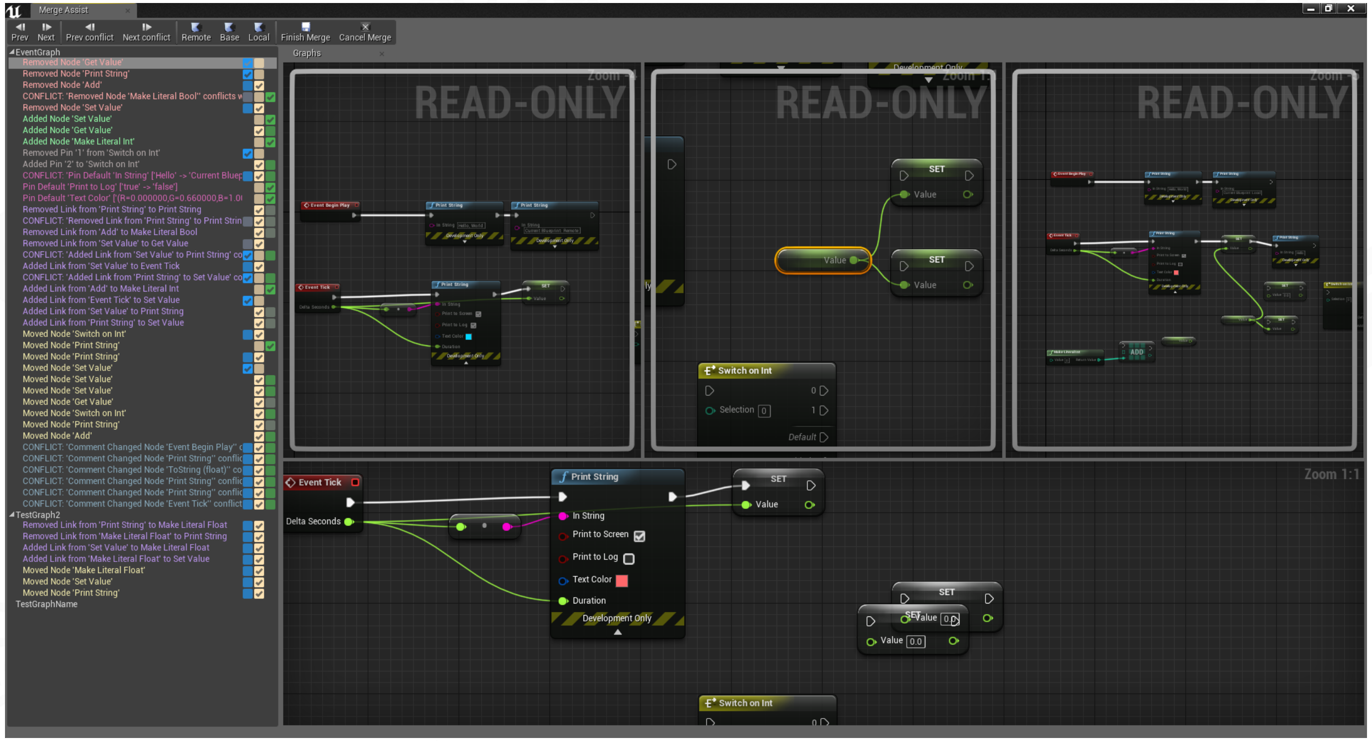
- Provide user-friendly debug tools for visual scripting



Issues with Visual Scriping (1/2)

Visual script is hard to merge for a team work

- Usually a visual script is stored as a binary file
- Manually reorder script graph is inefficient and error-prone even with a merge tool

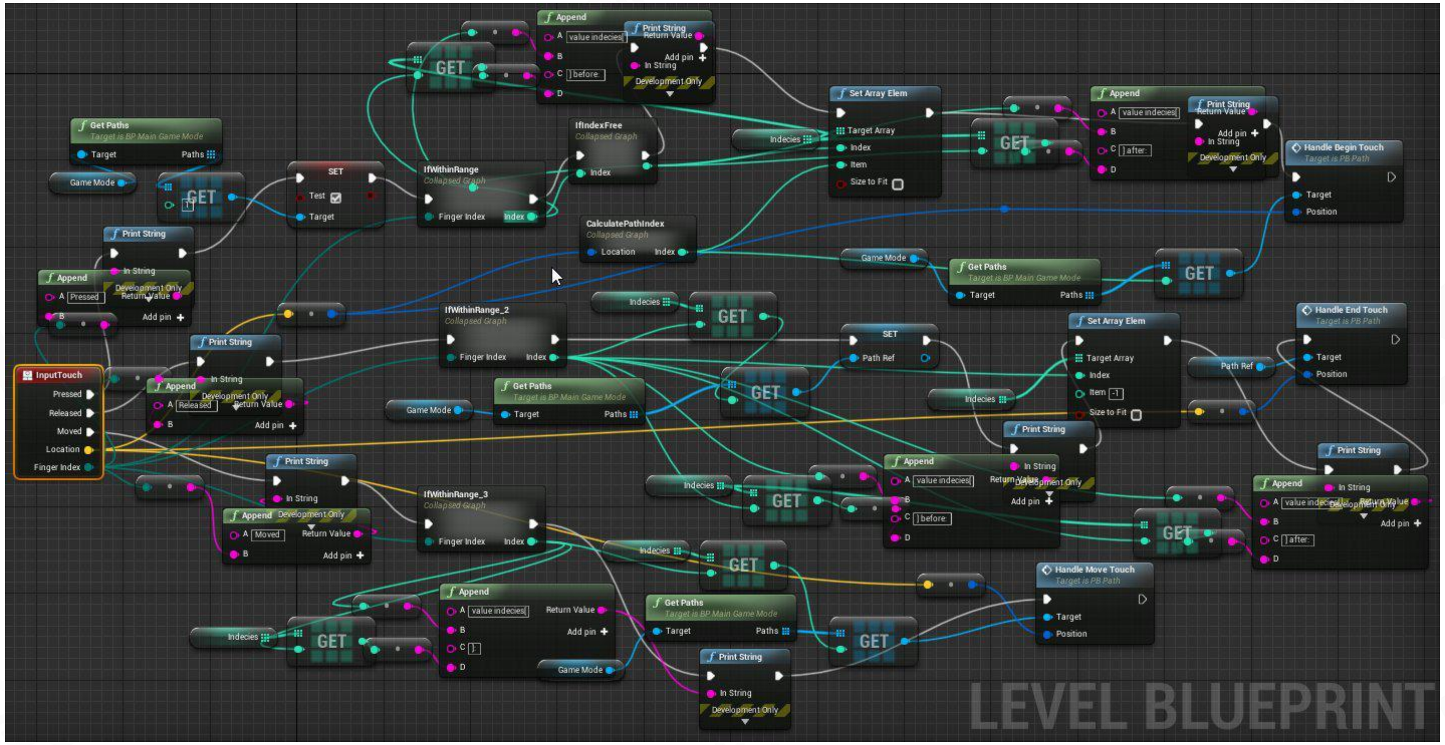


```
EventGraph
  Removed Node 'Get Value'
  Removed Node 'Print String'
  Removed Node 'Add'
  CONFLICT: 'Removed Node 'Make Literal Bool'' conflicts w
  Removed Node 'Set Value'
  Added Node 'Set Value'
  Added Node 'Get Value'
  Added Node 'Make Literal Int'
  Removed Pin '1' from 'Switch on Int'
  Added Pin '2' to 'Switch on Int'
  CONFLICT: 'Pin Default 'In String' ['Hello' -> 'Current Bluep
  Pin Default 'Print to Log' ['true' -> 'false']
  Pin Default 'Text Color' ['(R=0.000000,G=0.660000,B=1.0(
  Removed Link from 'Print String' to Print String
  CONFLICT: 'Removed Link from 'Print String' to Print Strin
  Removed Link from 'Add' to Make Literal Bool
  Removed Link from 'Set Value' to Get Value
  CONFLICT: 'Added Link from 'Set Value' to Print String' co
  Added Link from 'Set Value' to Event Tick
  CONFLICT: 'Added Link from 'Print String' to Set Value' co
  Added Link from 'Add' to Make Literal Int
  Added Link from 'Event Tick' to Set Value
  Added Link from 'Set Value' to Print String
  Added Link from 'Print String' to Set Value
  Moved Node 'Switch on Int'
  Moved Node 'Print String'
  Moved Node 'Print String'
  Moved Node 'Set Value'
  Moved Node 'Set Value'
  Moved Node 'Set Value'
  Moved Node 'Get Value'
  Moved Node 'Switch on Int'
  Moved Node 'Print String'
  Moved Node 'Add'
  CONFLICT: 'Comment Changed Node 'Event Begin Play'' c
  CONFLICT: 'Comment Changed Node 'Print String'' conflic
  CONFLICT: 'Comment Changed Node 'ToString (float)'' co
  CONFLICT: 'Comment Changed Node 'Print String'' conflic
  CONFLICT: 'Comment Changed Node 'Print String'' conflic
  CONFLICT: 'Comment Changed Node 'Event Tick'' conflict
TestGraph2
  Removed Link from 'Print String' to Make Literal Float
  Removed Link from 'Make Literal Float' to Print String
  Added Link from 'Set Value' to Make Literal Float
  Added Link from 'Make Literal Float' to Set Value
  Moved Node 'Make Literal Float'
  Moved Node 'Set Value'
  Moved Node 'Print String'
TestGraphName
```

Issues with Visual Scriping (2/2)

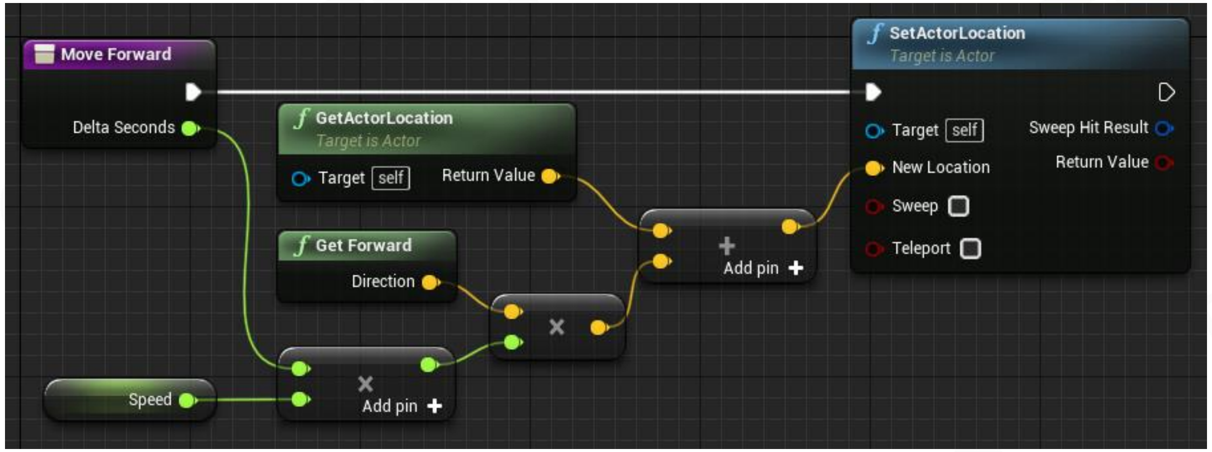
The graph can get pretty messy with complex logic

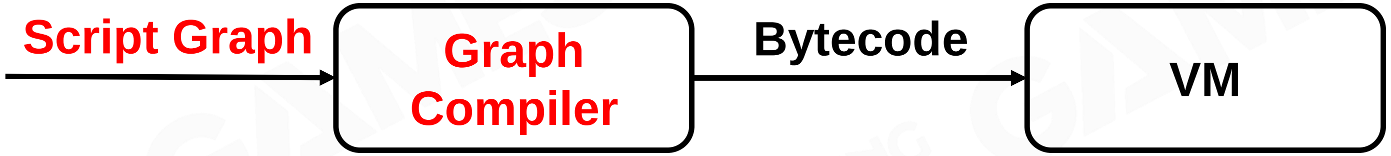
- Need uniform graph layout rules for a team work



Script and Graph are Twins

```
function moveForward(delta_seconds)
    local location = self:getLocation()
    local direction = self:getForward()
    local speed = self.speed
    local movement = delta_seconds * speed * direction
    self:setLocation(location + movement)
end
```





Game Play

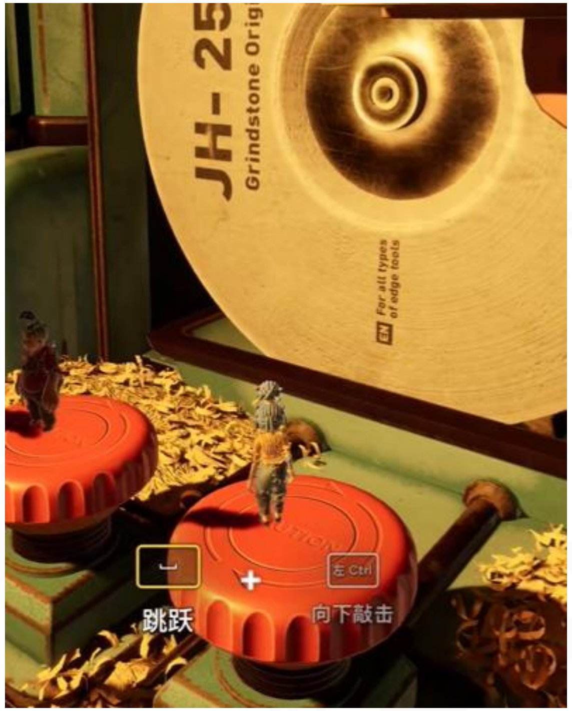
“3C” in Game Play

It takes two

What is 3C?

3C: Character, Control & Camera

3C is the primary element that determines the gameplay experience



Character

In-game character, both player and npc.

Include character movement, combat, health, mana, what skills and talents they have, etc.

One most basic element of a character is **movement**.

Character: Well-designed Movement

Movement looks simple, but it's hard to do well.

In AAA games, every basic state of action needs to be broken down into detailed states.

several state changes in a few seconds

Idle

Start

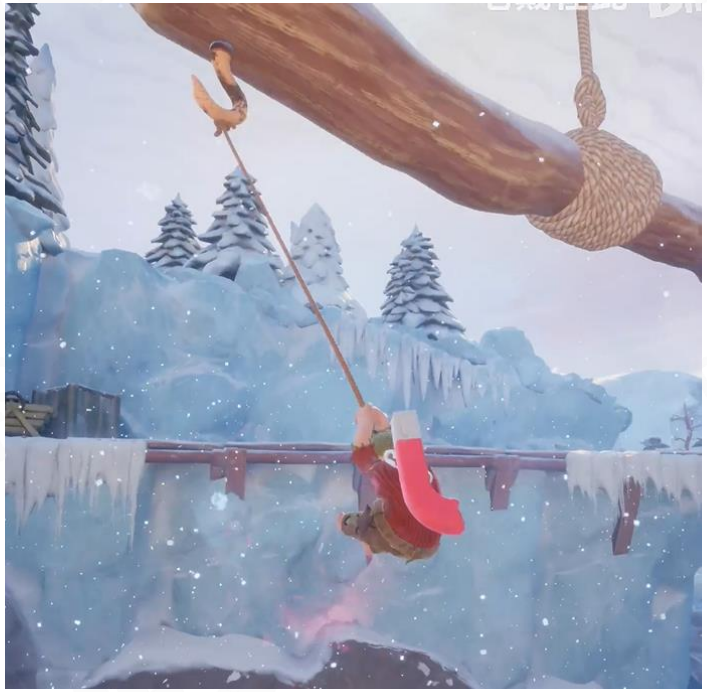
Walk

Accelerate

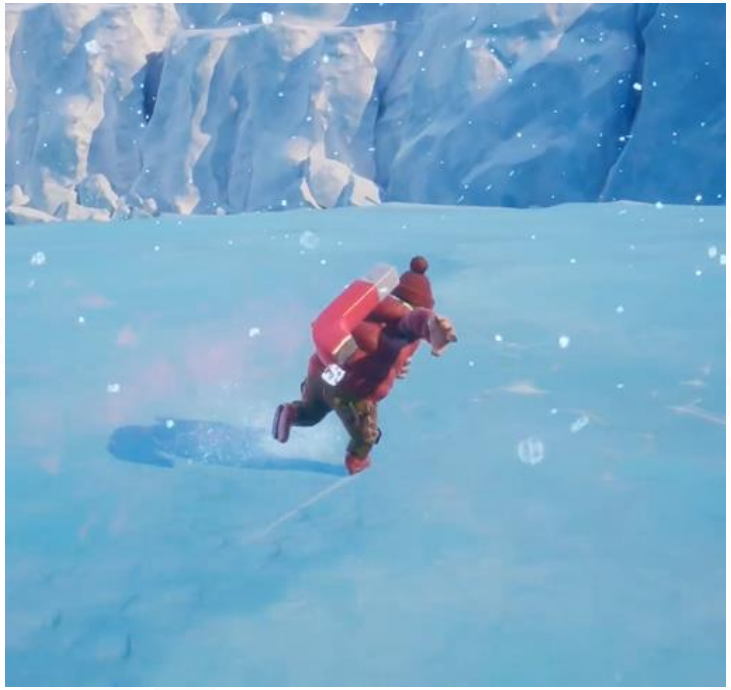
Run

Brake

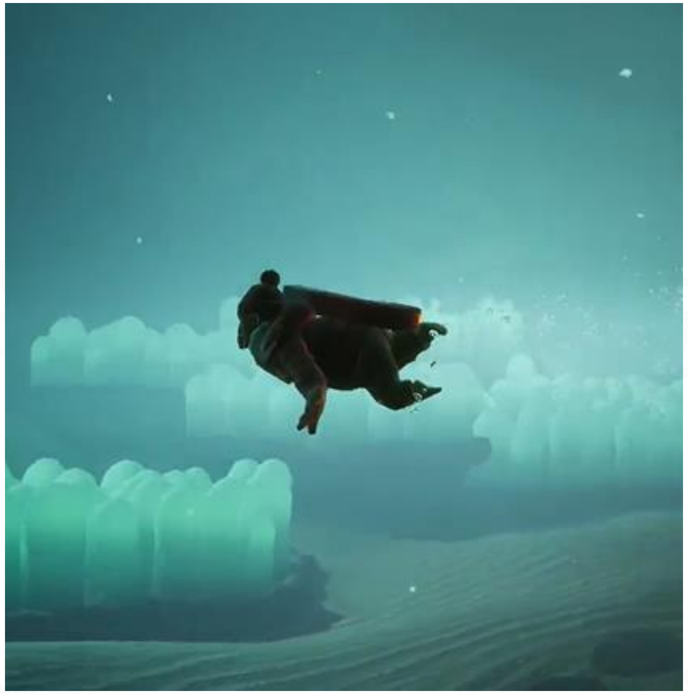
Extended Character：More complex and varied states



Hanging



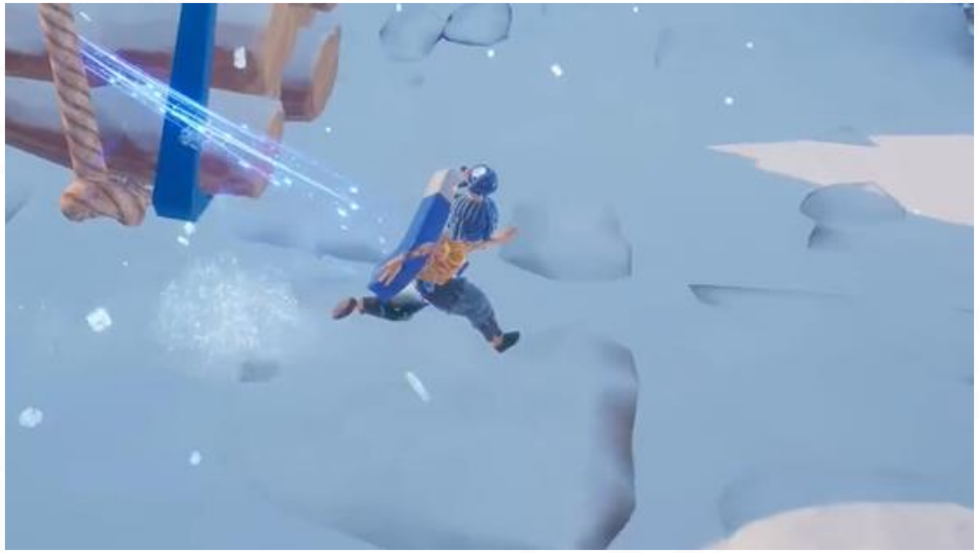
Skating

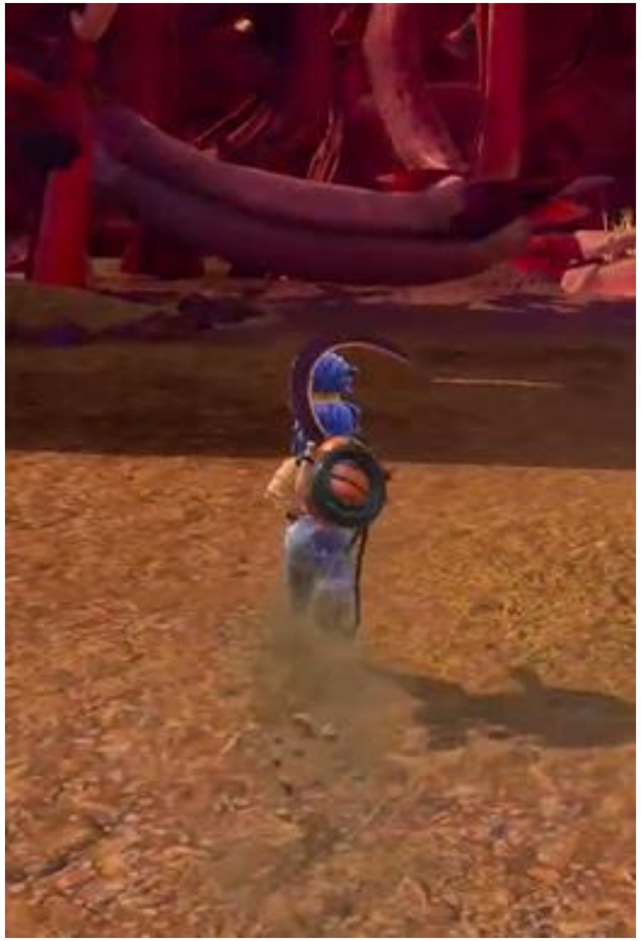


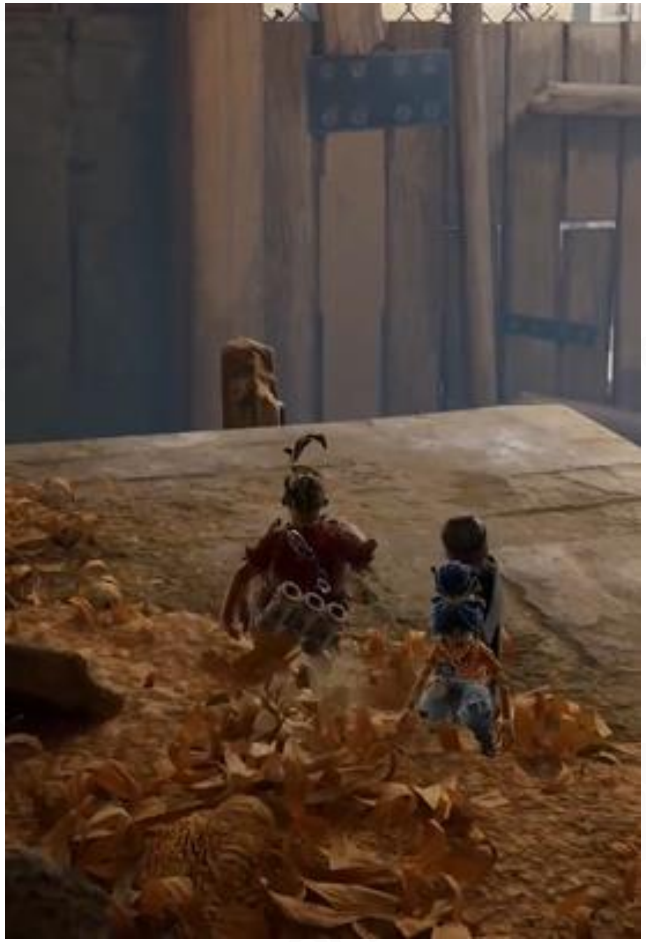
Diving

Extended Character：Cooperate with other systems

Game effects, sound, environment interaction.



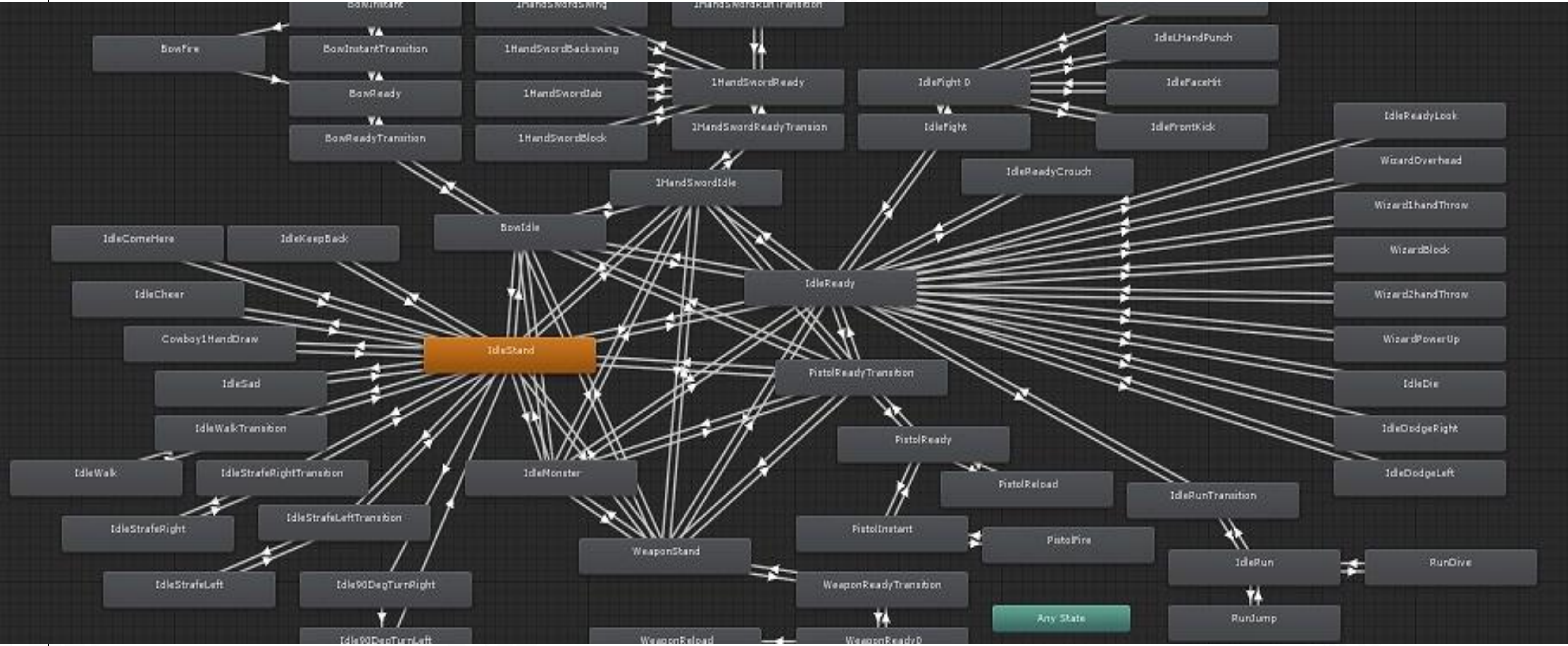




Extended Character：More realistic motion with Physics

- Airflow
- Inertia tensor
- Torque
- ...

Movement State Machine

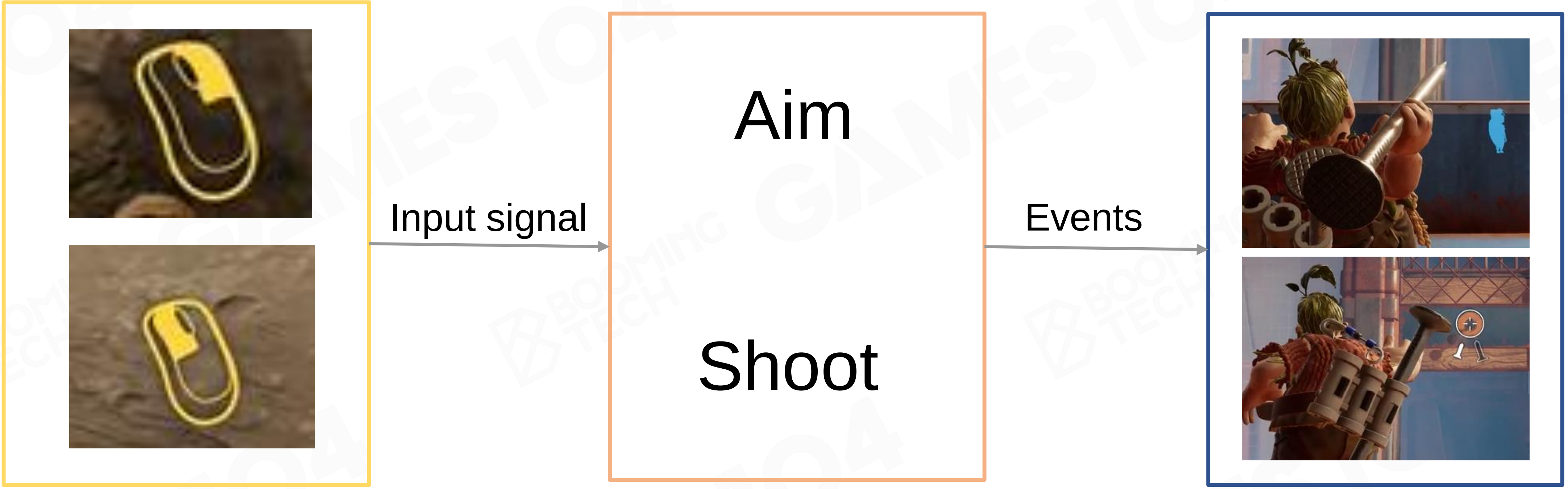


Control

Different input device
Different game play

A Good Example of Control

From Input to Game Logic



Input signal

Aim

Shoot

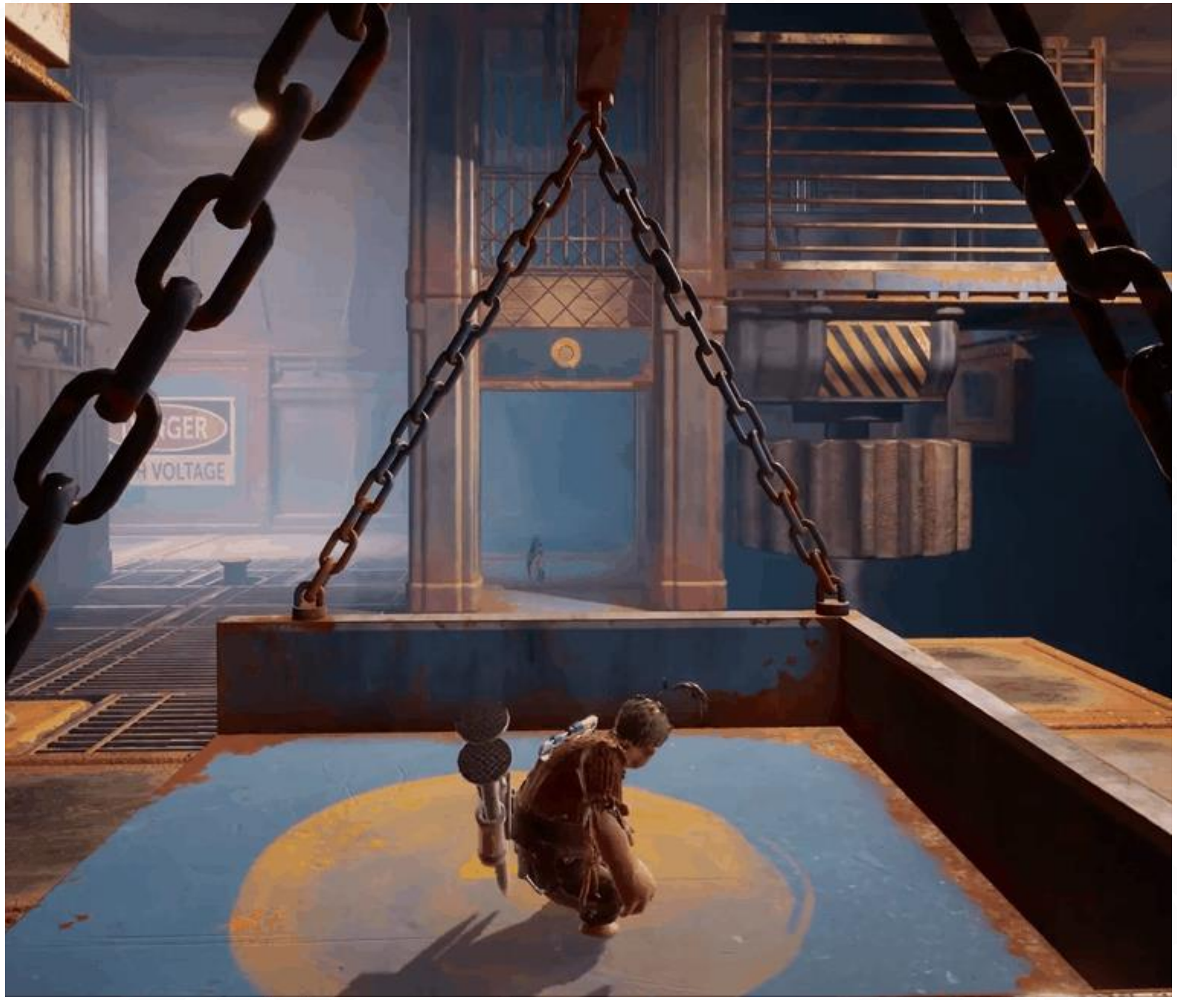
Events

Input device

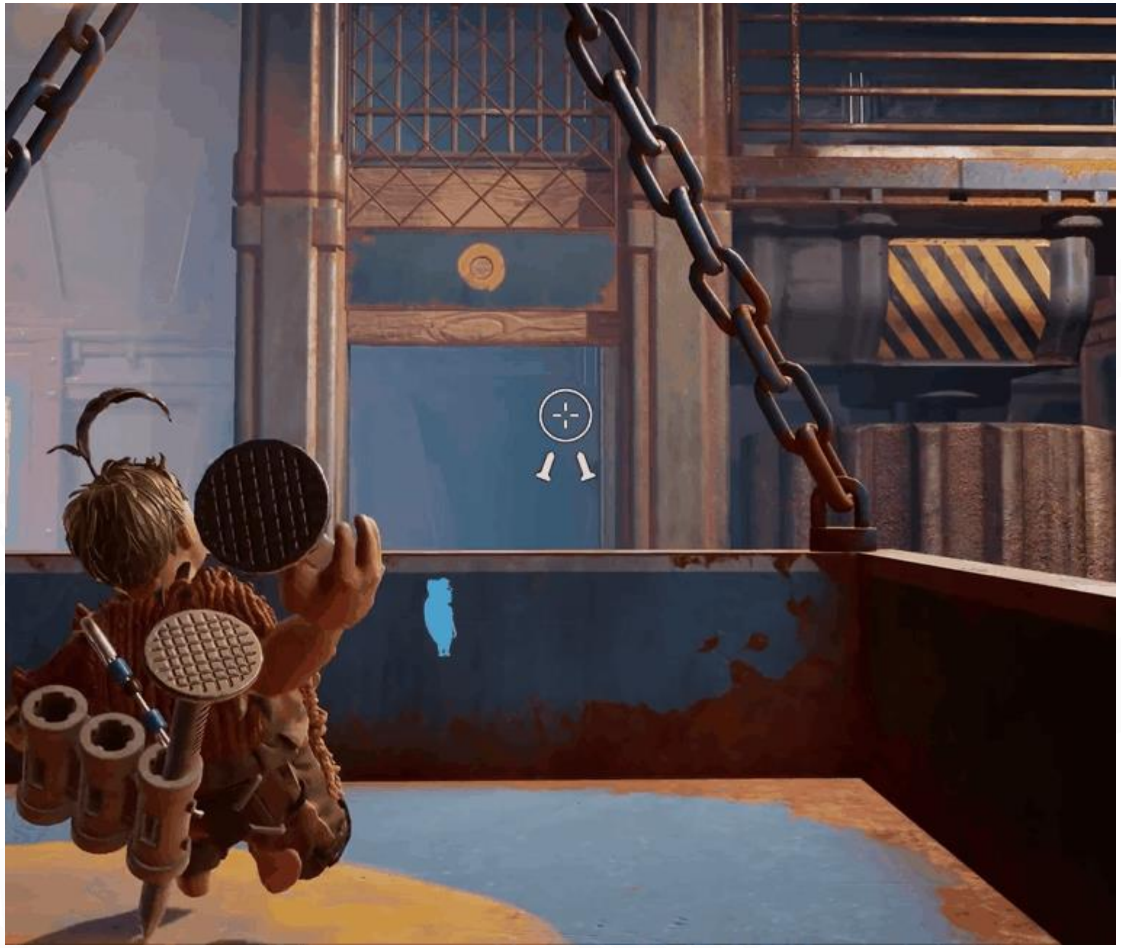
Action

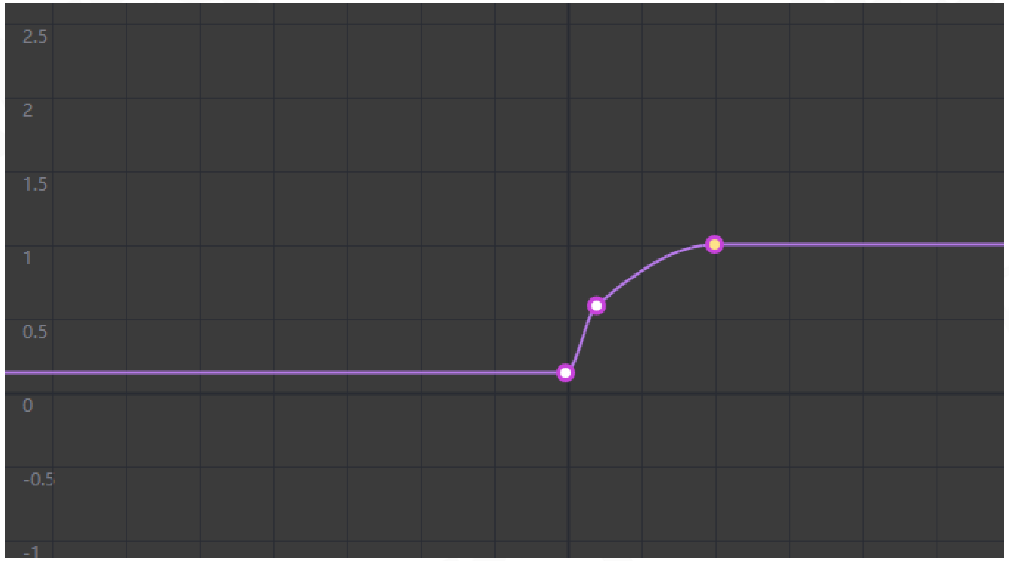
Game logic

Control: Zoom in and out

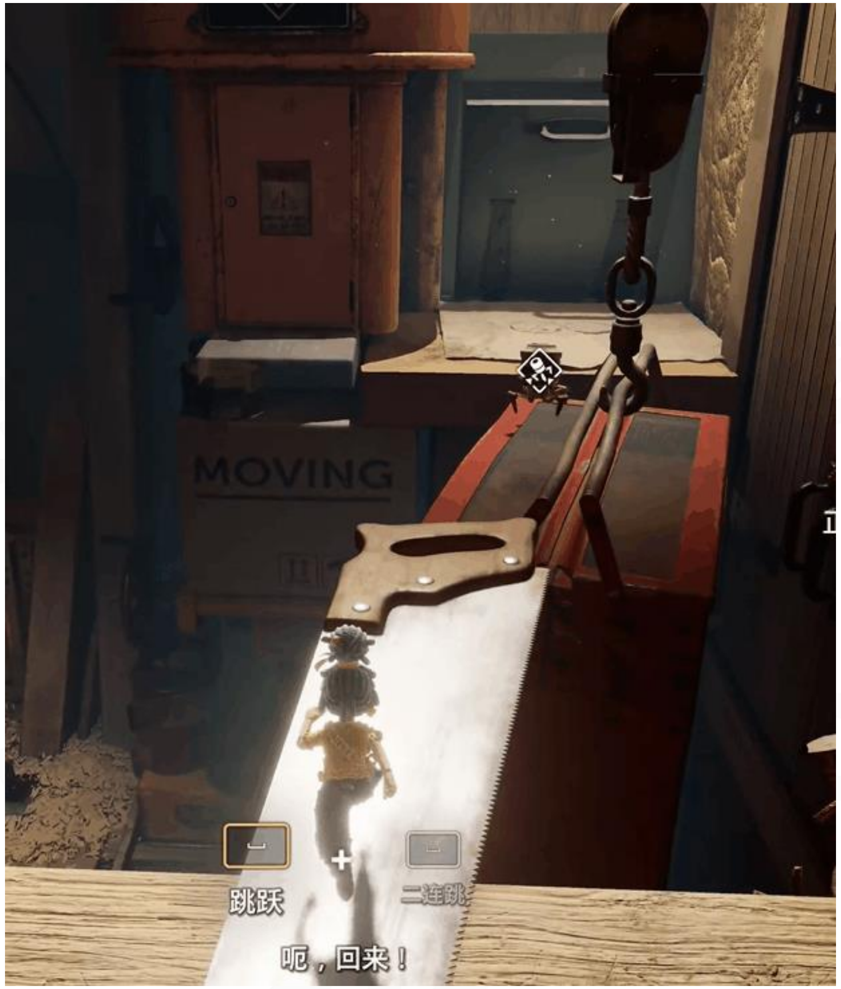


Control: Aim Assist





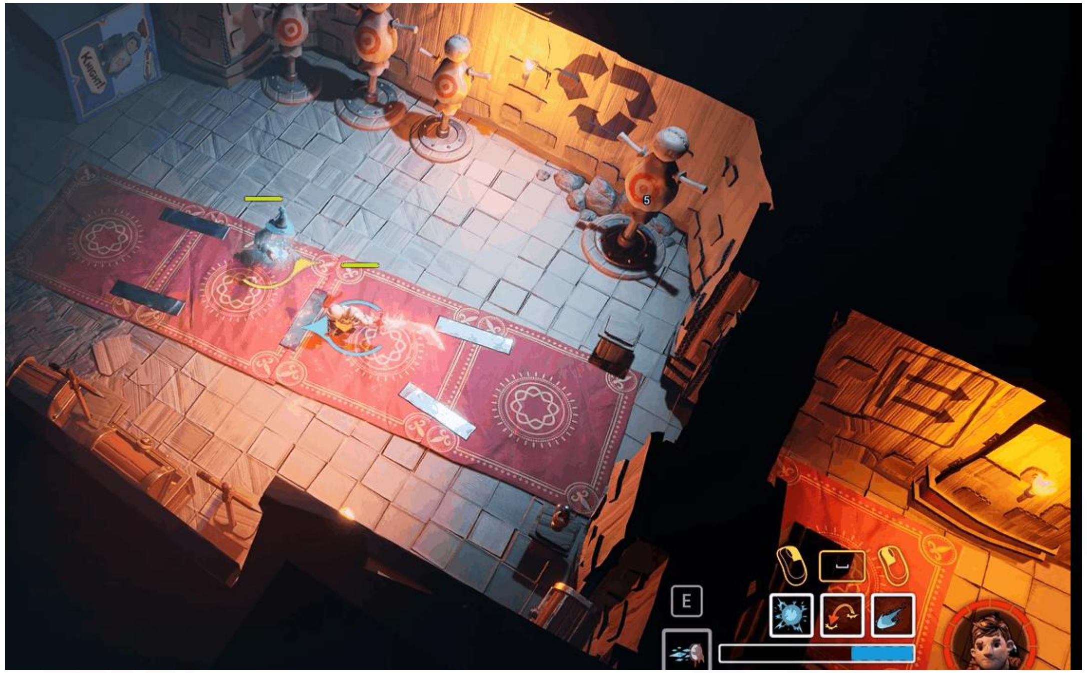
Control: Feedback



Control: Context Awareness

Context-sensitive controls

- The same input button produces different effects in different game scenarios.



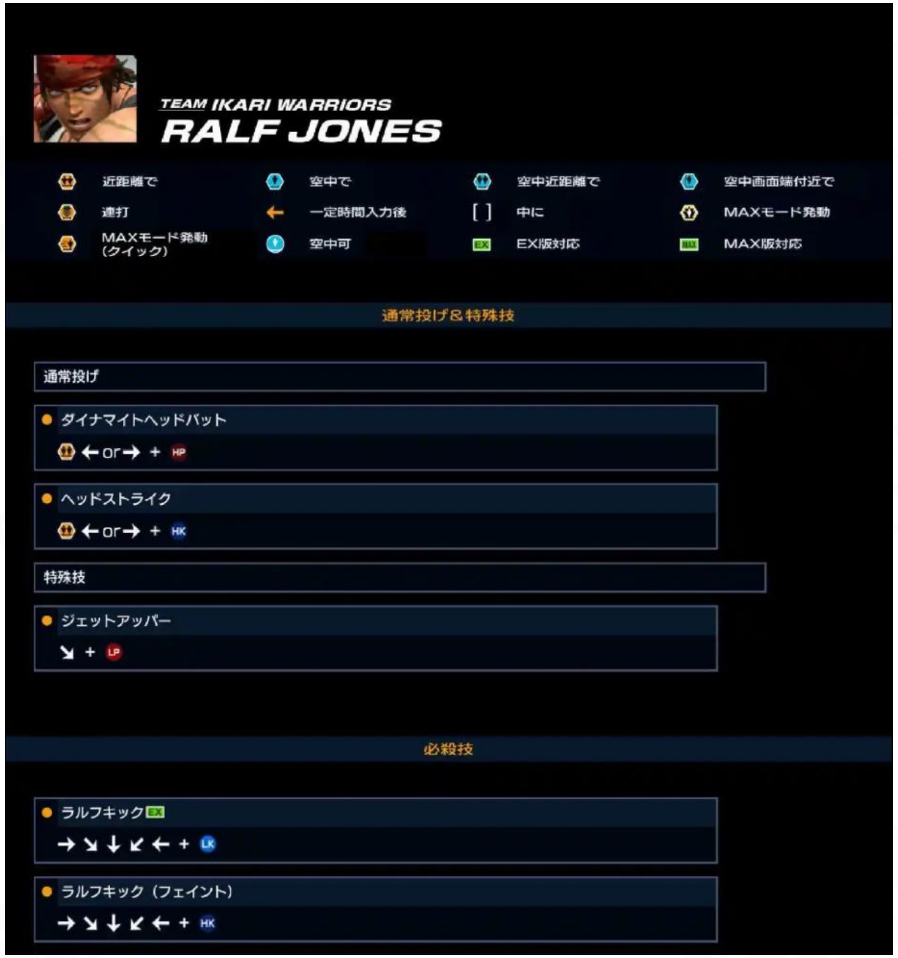
Control : Chord &Key Sequences

Chords

- when pressed at the same time, produce a unique behavior in the game

Key Sequences

- Gesture detection is generally implemented by keeping a brief history of the HID actions performed by the player



TEAM IKARI WARRIORS
RALF JONES

近距離で	空中で	空中近距離で	空中画面端付近で
連打	← 一定時間入力後	[ ] 中に	MAXモード発動
MAXモード発動（クイック）	空中可	EX EX版対応	MAX MAX版対応

通常投げ＆特殊技

通常投げ

- ダイナマイトヘッドバット
  ←or→ + HP
- ヘッドストライク
  ←or→ + HK

特殊技

- ジェットアッパー
  ↘ + LP

必殺技

- ラルフキックEX
  → ↘ ↓ ↙ ← + LK
- ラルフキック（フェイント）
  → ↘ ↓ ↙ ← + HK

Camera: Subjective Feelings

Camera Basic：POV & FOV

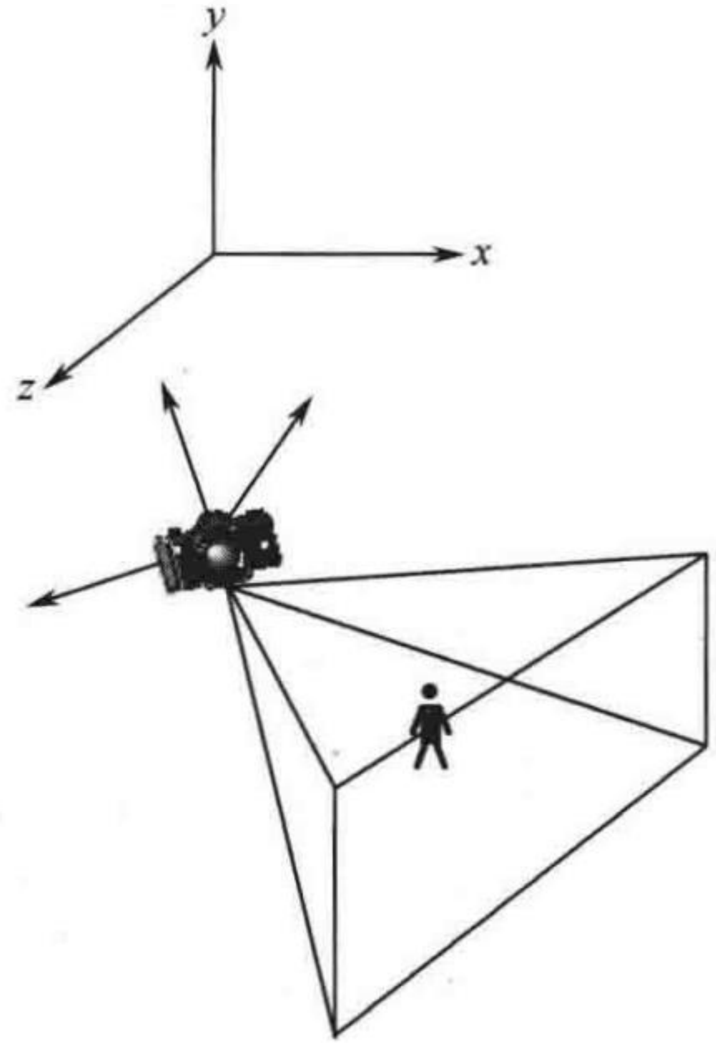
POV (point of view)

- determines the position of the player to observe

FOV (field of view)

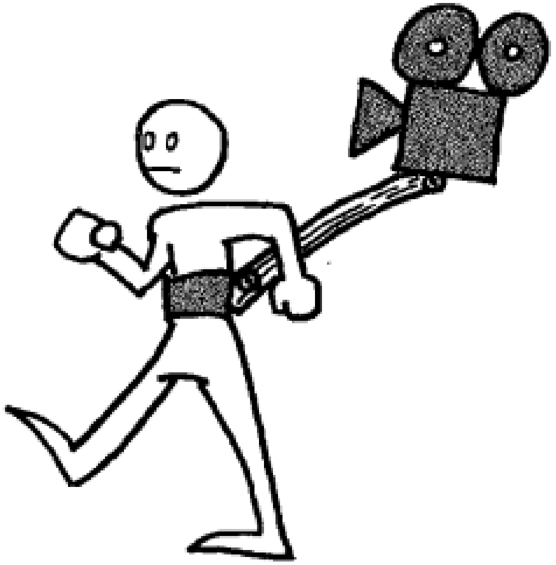
- determines the size of the player's viewing Angle

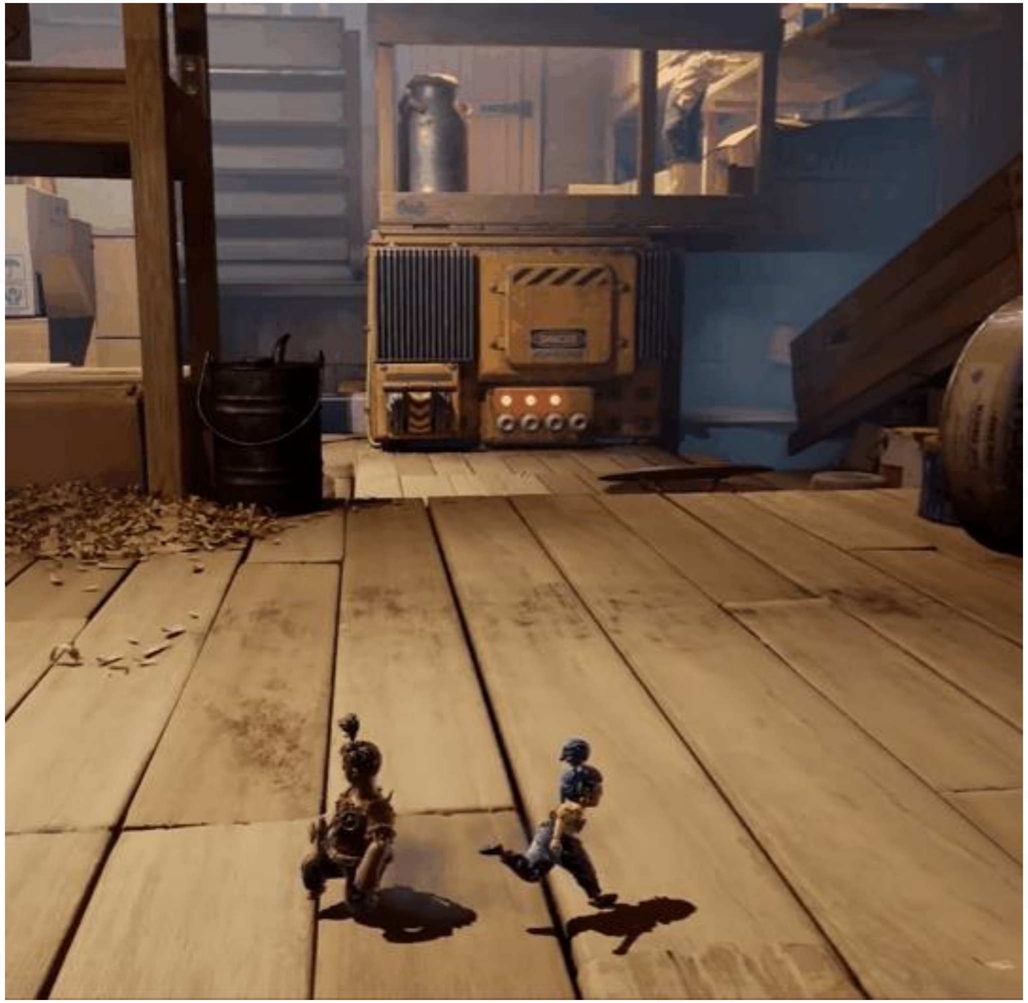
world space



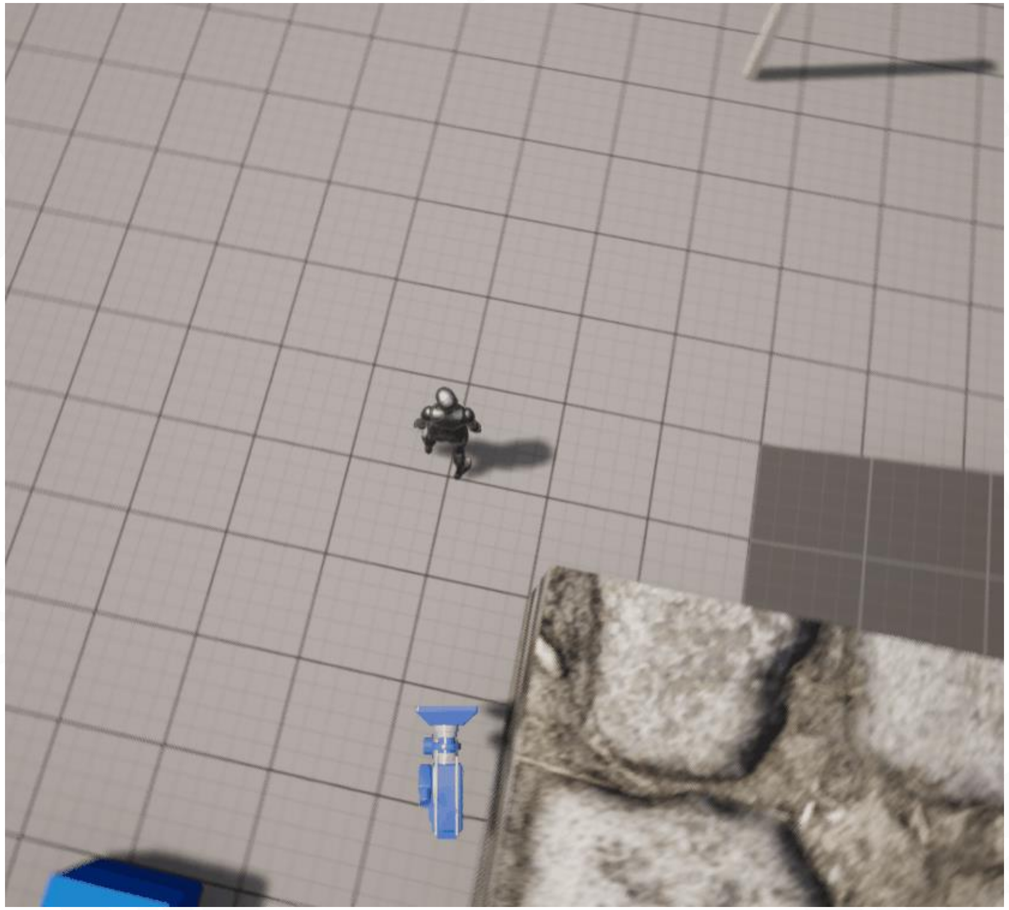
Camera Binding

Using POV and rotation to bind.

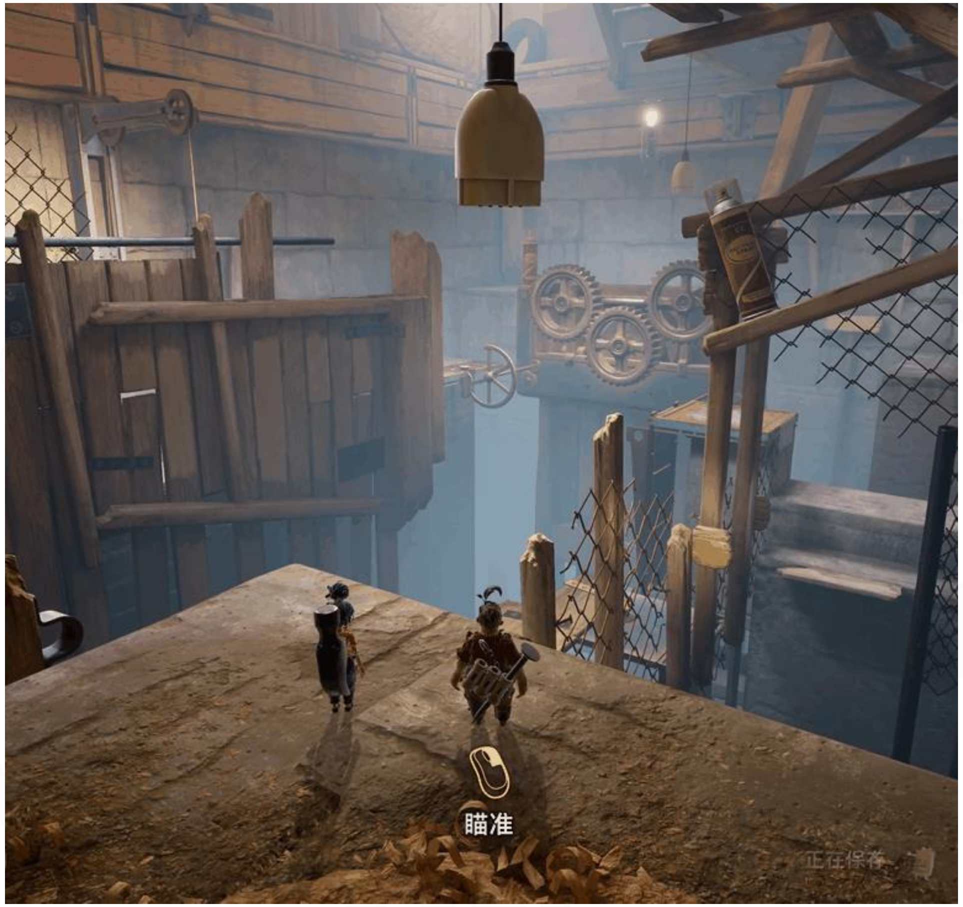




Camera Control



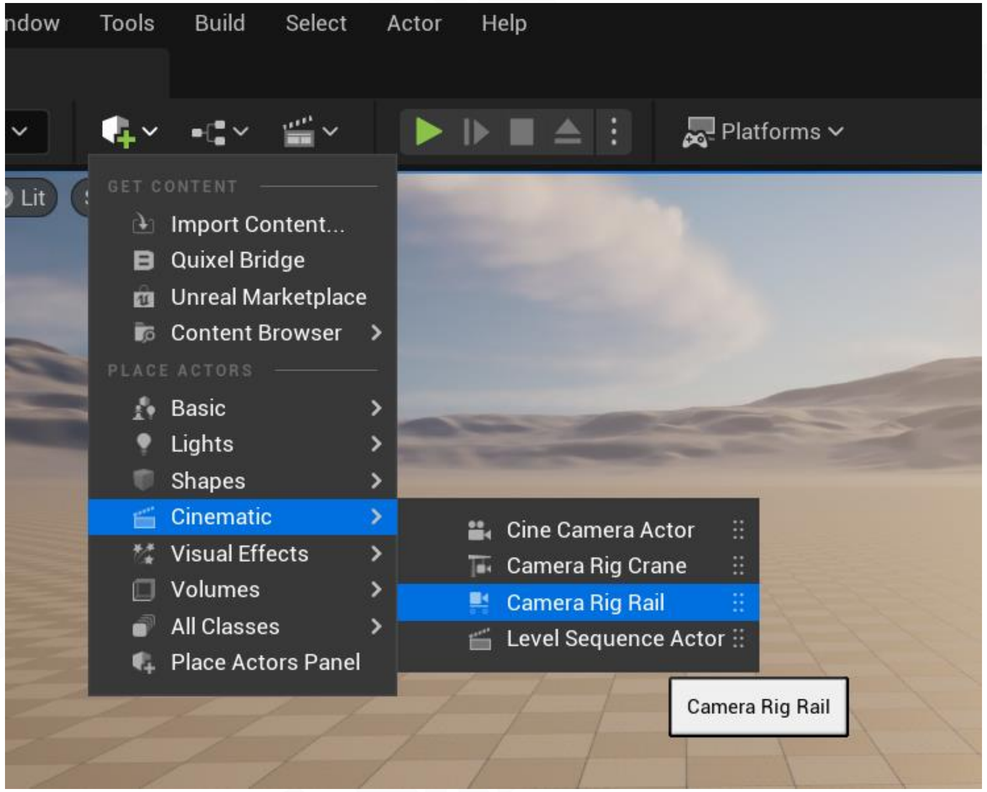
Spring Arm



focusing
FOV&distance Curve

Camera Track

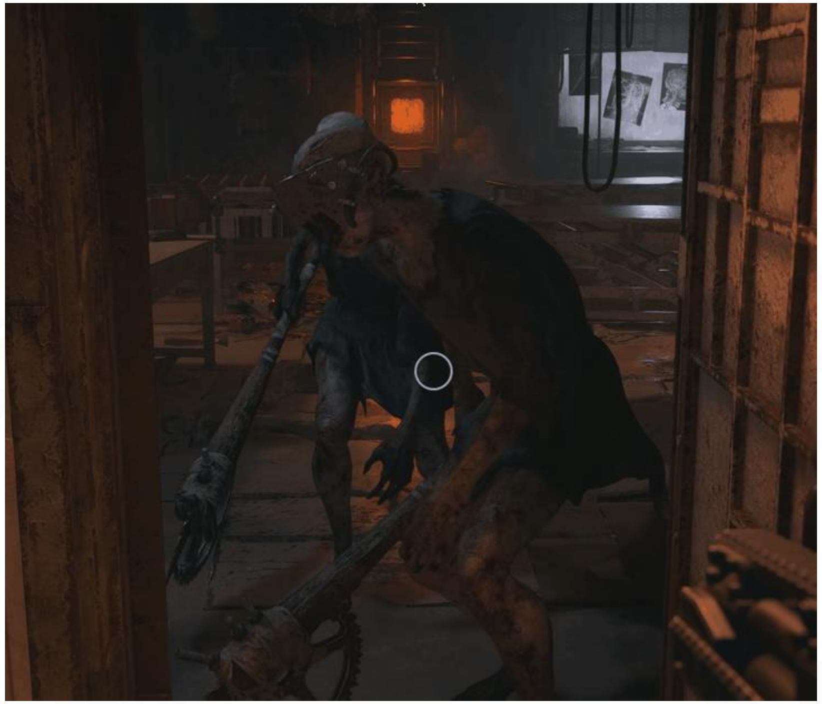
Camera Track



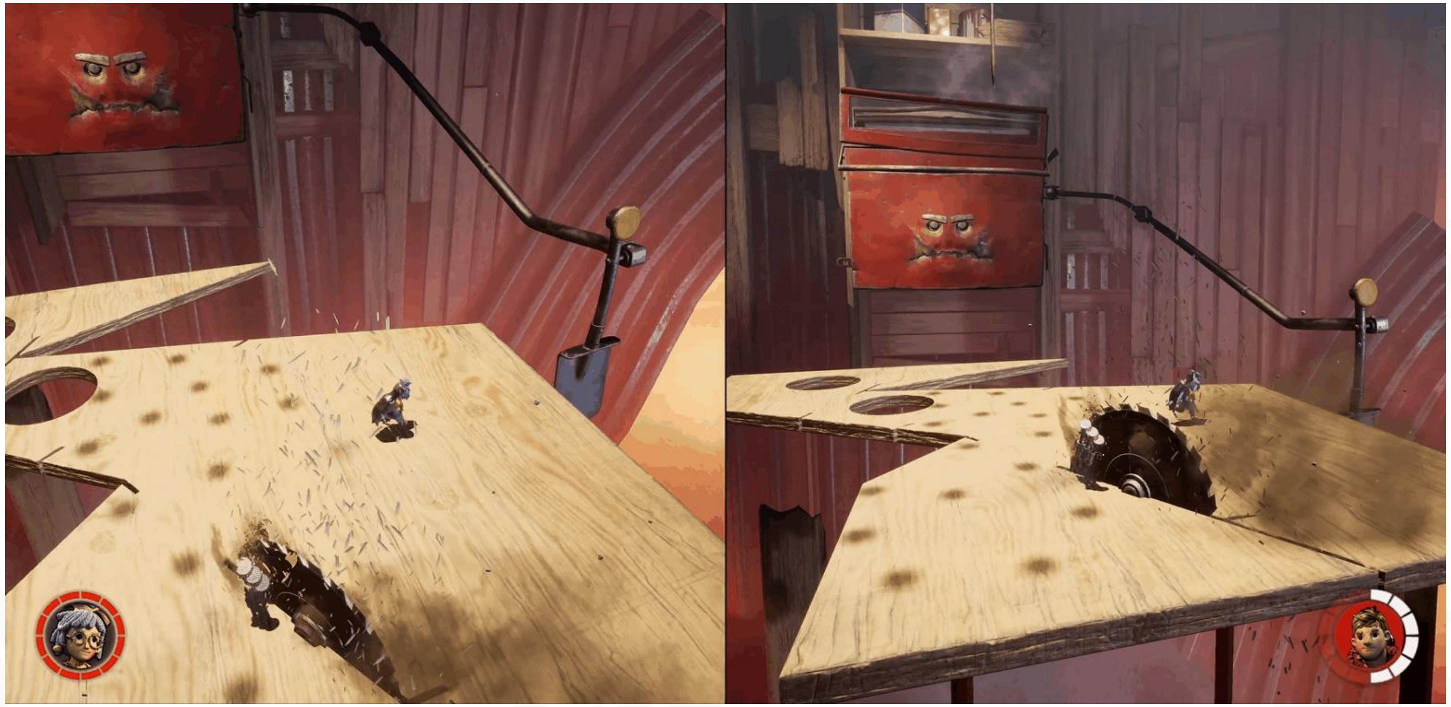
Scene Editor

Camera Effects

Provide the camera with more post-visual effects, such as filters and shake.

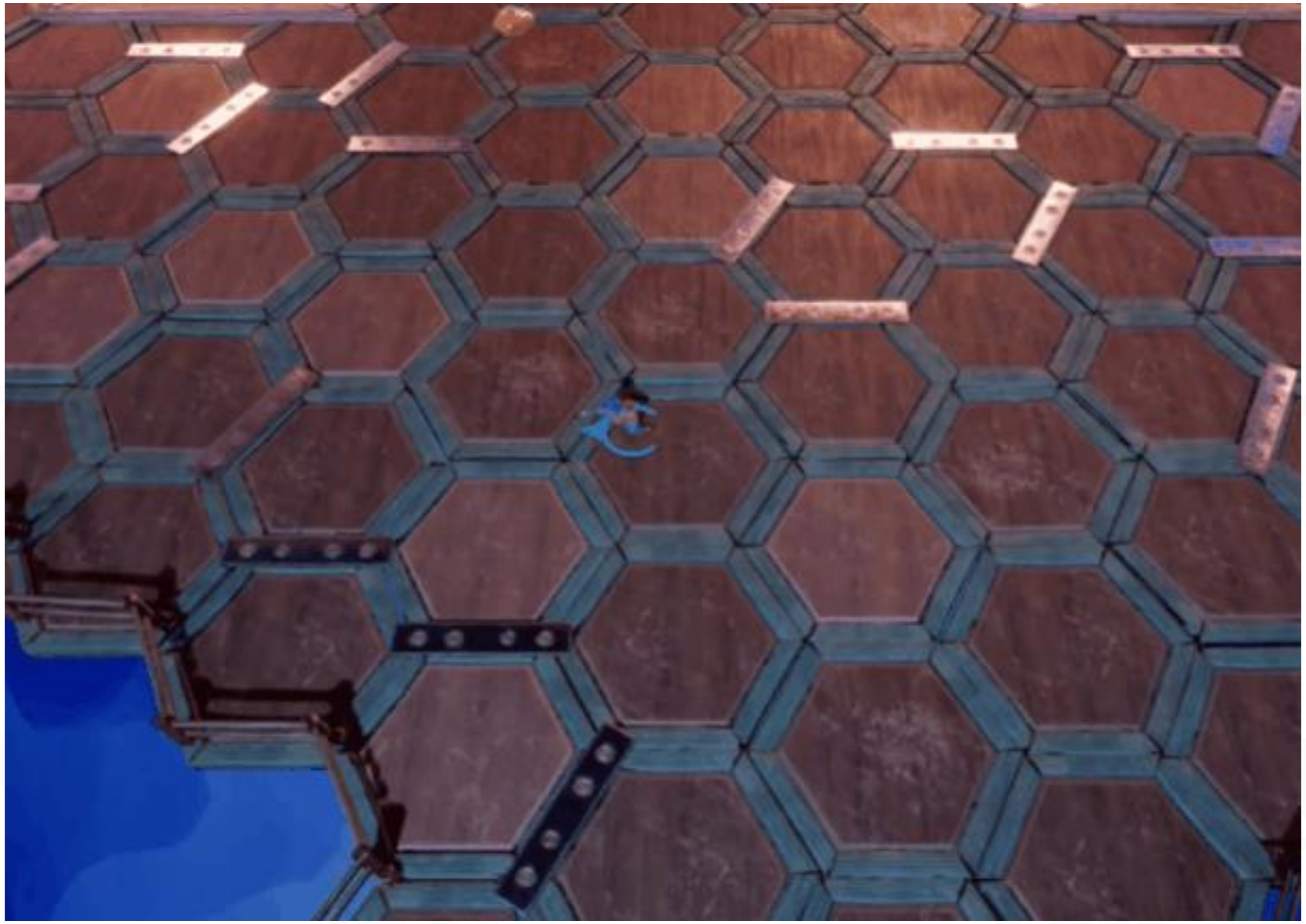


Camera Shake



Camera Filter

Many Cameras : Camera Manager



Camera Switch

Camera: Subjective Feelings

Complex effects are often achieved by multiple base adjustments.
To create a sense of speed as an example, we can do:

- Add lines in the speed direction
- The character falls backwards
- The dynamic fuzzy
- Zoom in FOV (to speed up changes in screen content)

Speed
Motion blur, magnify FOV

Camera: Subjective Feelings

loose feeling

- Relax camera movement

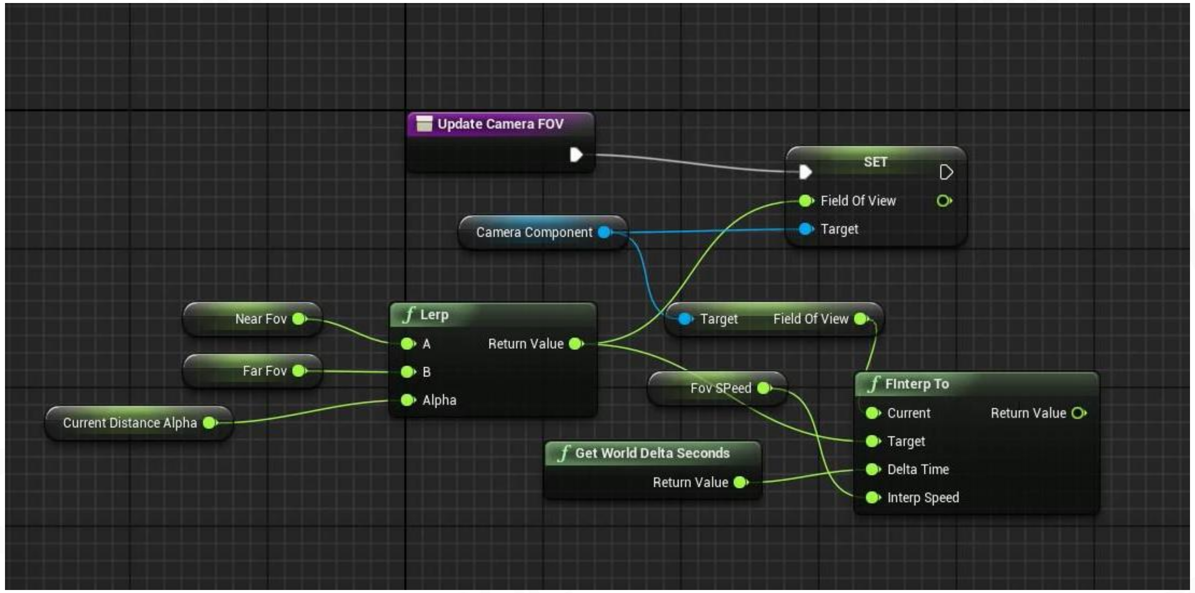
Cinematic

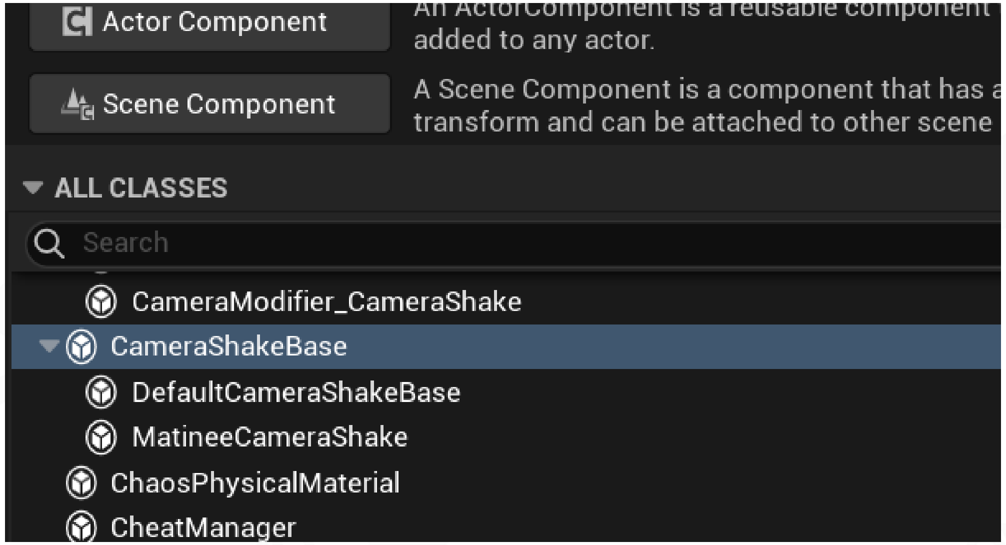
- filter, motion, sound, narrator, model, animation, camera movement,...

Camera

For artists and designers to optimize the effect:

- Inheritable classes
- Function that can be accessed by Blueprint
- Adjustable parameters



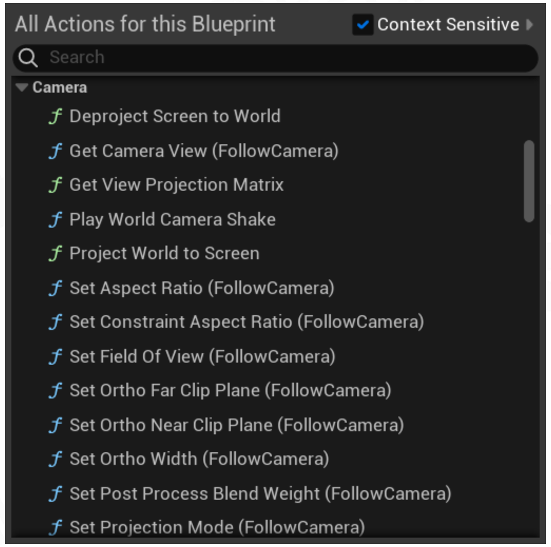

Actor Component | An ActorComponent is a reusable component added to any actor.

Scene Component | A Scene Component is a component that has a transform and can be attached to other scene

ALL CLASSES

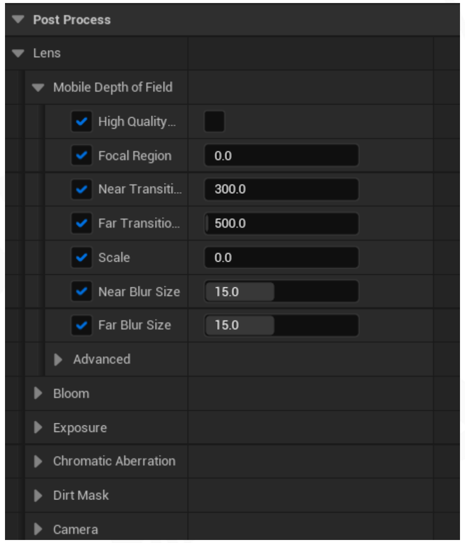
Search

- CameraModifier_CameraShake
- CameraShakeBase
  - DefaultCameraShakeBase
  - MatineeCameraShake
- ChaosPhysicalMaterial
- CheatManager


All Actions for this Blueprint ☑ Context Sensitive

Search

Camera
- Deproject Screen to World
- Get Camera View (FollowCamera)
- Get View Projection Matrix
- Play World Camera Shake
- Project World to Screen
- Set Aspect Ratio (FollowCamera)
- Set Constraint Aspect Ratio (FollowCamera)
- Set Field Of View (FollowCamera)
- Set Ortho Far Clip Plane (FollowCamera)
- Set Ortho Near Clip Plane (FollowCamera)
- Set Ortho Width (FollowCamera)
- Set Post Process Blend Weight (FollowCamera)
- Set Projection Mode (FollowCamera)


Post Process
- Lens
  - Mobile Depth of Field
    - ☑ High Quality... □
    - ☑ Focal Region 0.0
    - ☑ Near Transiti... 300.0
    - ☑ Far Transitio... 500.0
    - ☑ Scale 0.0
    - ☑ Near Blur Size 15.0
    - ☑ Far Blur Size 15.0
    - Advanced
  - Bloom
  - Exposure
  - Chromatic Aberration
  - Dirt Mask
  - Camera

Everything is Gameplay

Lecture 15 Contributor

- 一将
- 蓑笠翁
- 炯哥
- 玉林
- 小老弟
- 建辉
- Hoya
- 爵爷
- Jason
- 砚书
- BOOK
- MANDY
- Unicorn
- 灰灰
- 喵小君
- 果蝇
- 梨叔
- Shine
- 邓导
- Judy
- Leon
- QIUU
- C佬
- 阿乐
- 阿熊
- CC
- 大喷
- 金大壮

Q&A

Enjoy ;)
Coding

Course Wechat

Follow us for
further information