Unreal Engine Terminology

Actor

An **Actor** is any object that can be placed into a level, such as a Camera, static mesh, or player start location. Actors support 3D transformations such as translation, rotation, and scaling. Creating and deleting new instances of `AActor` classes is called **spawning/destroying**.

Component

A **Component** is a piece of functionality that can be added to an Actor. When you add a Component to an Actor, the Actor can use the functionality that the Component provides. For example:

- An Audio Component will give your Actor the ability to play sounds.
  Components must be attached to an Actor and can't exist by themselves.

World

The World is the top level object representing a map or a sandbox in which Actors and Components will exist and be rendered.
所有3d视窗都是一个world

Level

A Level is a collection of Actors (lights, volumes, mesh instances etc.). Multiple Levels can be loaded and unloaded into the World to create a streaming experience.

Pawn

Pawns are a subclass of Actor and serve as an in-game avatar or persona (for example, the characters in a game). Pawns can be controlled by a player or by the game's AI, as non-player characters (NPCs).This control fuction is called Possess.

Character

A **Character** is a subclass of a Pawn Actor that is intended to be used as a player character. The Character subclass includes a collision setup, input bindings for bipedal movement, and additional code for player-controlled movement.

PlayerController

A **Player Controller** takes player input and translates it into interactions in the game. Every game has at least one Player Controller in it. A Player Controller often possesses a Pawn or Character as a representation of the player in a game.

AIController

Just as the Player Controller possesses a Pawn as a representation of the player in a game, an **AI Controller** possesses a Pawn to represent a non-player character
Only pawn possessed by AIController can start/control a behavior tree.

GameMode

The **Game Mode** sets the rules of the game that is being played. These rules can include:

- How players join the game.
- Whether or not a game can be paused.
- Any game-specific behavior such as win conditions.
  多人游戏的规则

GameState

A **Game State** is a container that holds information you want replicated to every client in a game. In simpler terms, it is 'The State of

the Game' for everyone connected.
多人游戏的状态

Basic of Blueprint Programming

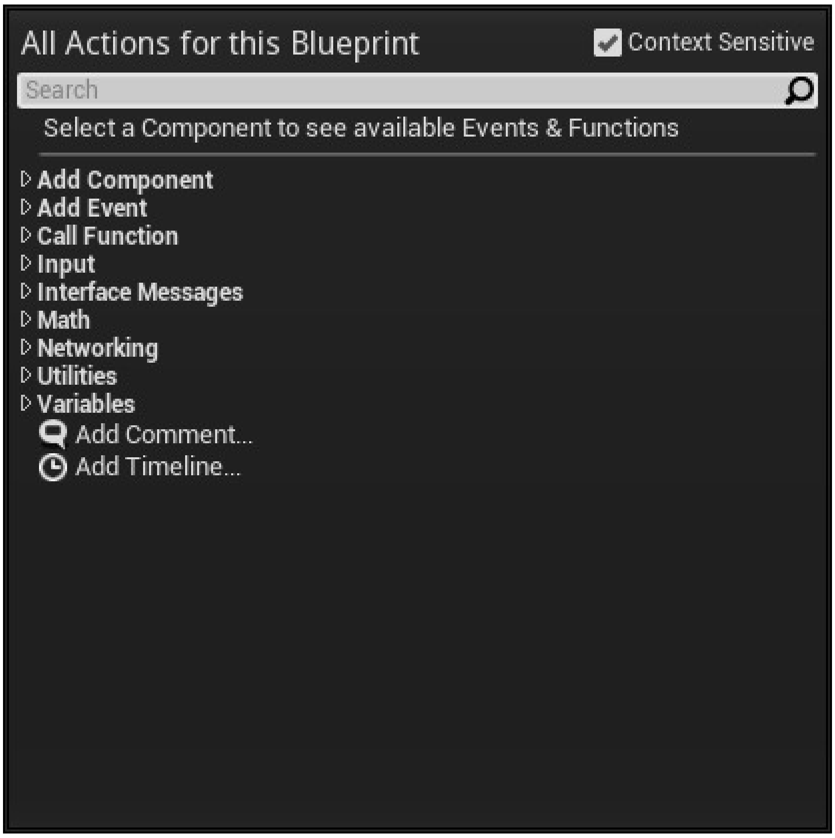
Variables && DataType

basic variables

int
float
string
structure

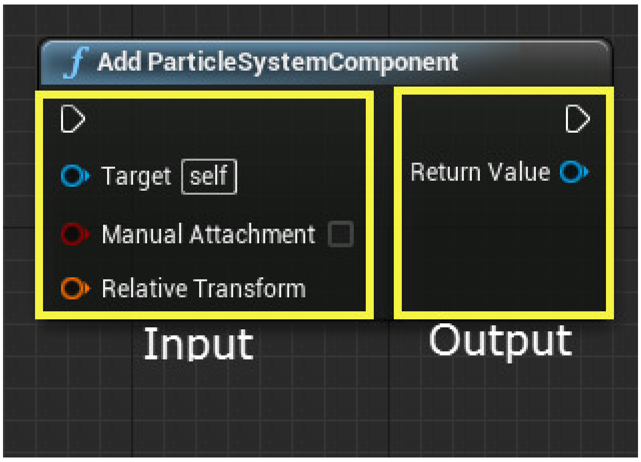
get/set

create Node



All Actions for this Blueprint ☑ Context Sensitive
Search
Select a Component to see available Events & Functions
▷ Add Component
▷ Add Event
▷ Call Function
▷ Input
▷ Interface Messages
▷ Math
▷ Networking
▷ Utilities
▷ Variables
Add Comment...
Add Timeline...

Pins

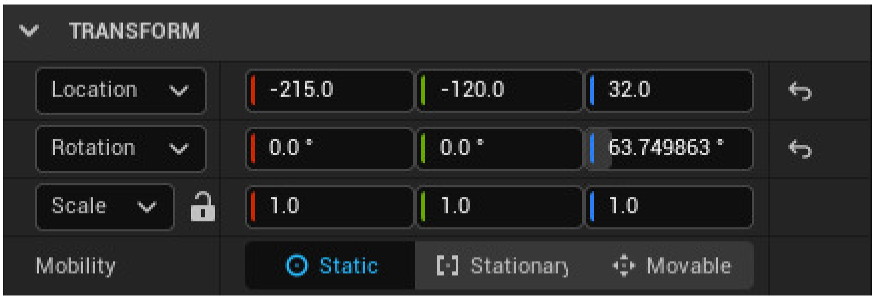


Important Blueprint Nodes

Event BeginPlay : execute when the game begin.
Event Tick : execute every frame.
Print String : print text to log file,usually used in debugging.
SetActorTransform : Set the location,scale,rotation of an Actor.
FlowControlNodes:
Sequence,Branch,SwitchOnInt

Basic of 3D Math

When you select one or more Actors in the **Level Viewport**, you can view and edit their **Location**, **Rotation**, and **Scale** in this section.



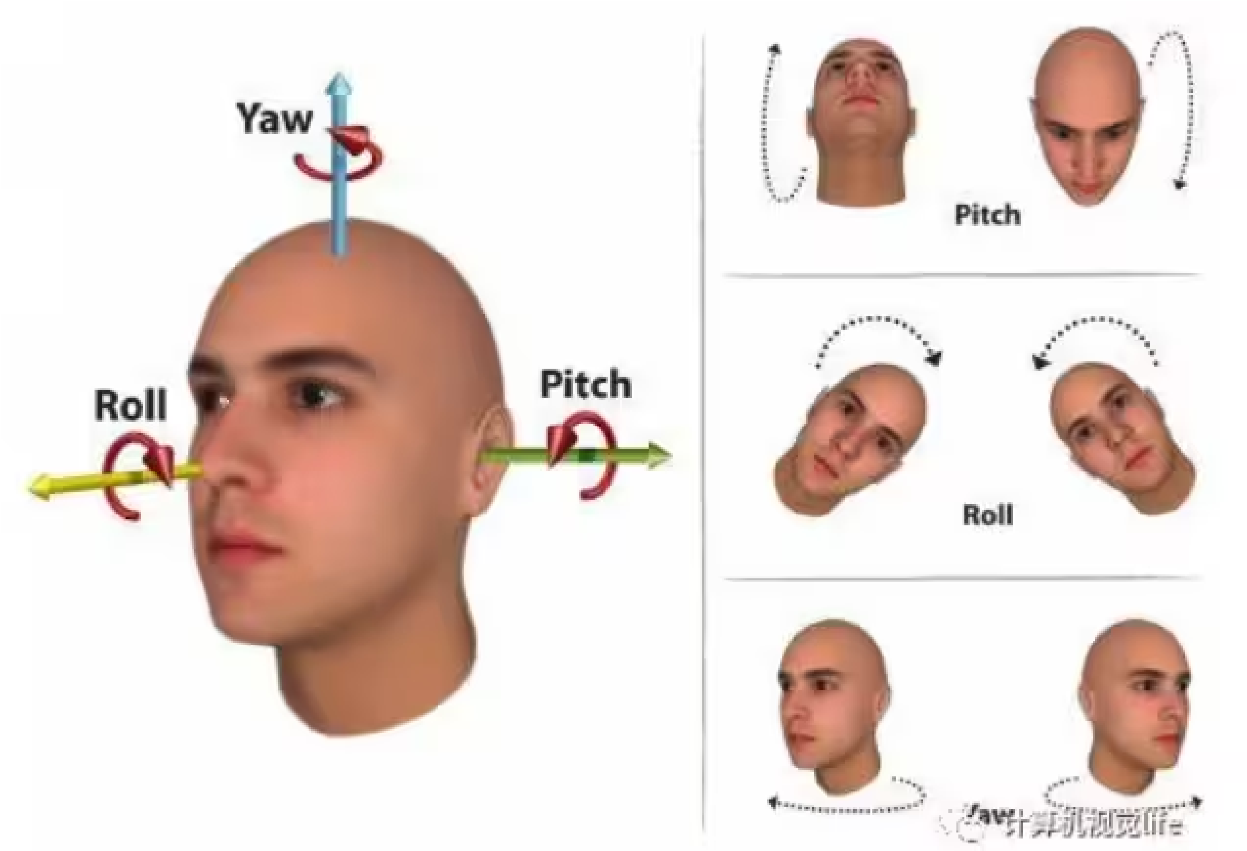
TRANSFORM

Location	-215.0	-120.0	32.0
Rotation	0.0 °	0.0 °	63.749863 °
Scale	1.0	1.0	1.0
Mobility	[✓ Static]	Stationary	Movable

Vector

x : forward/back
y : left/right
z : up/down

Rotator



Basic of Git

https://git-scm.com/book/zh/v2/Git-%E5%9F%BA%E7%A1%80-%E8%8E%B7%E5%8F%96-Git-%E4%BB%93%E5%BA%93

git clone

git checkout

git pull

git commit

git push

git branch -a

git reset --soft/--hard

git clean -f -x

Basic of Markdown

https://markdown.com.cn/intro.html

https://markdown.com.cn/basic-syntax/headings.html

https://obsidian.md/

Reference

https://docs.unrealengine.com/5.2/en-US/unreal-engine-terminology/

https://docs.unrealengine.com/5.2/en-US/transforming-actors-in-unreal-engine/

https://docs.unrealengine.com/5.2/en-US/transforming-actors-in-unreal-engine/