Overview

Visual Sandbox
User Guide

Topics

Introduction

Visual Sandbox is a *free-to-use* complete project for Unreal Engine 5, for a fast prototyping that includes a collection of pre-made Blueprints with cars, guns, items and more, featuring a character capable of running, shooting, and driving from an over-the-shoulder perspective. The entire template was built 100% with Blueprint, with a clean and simple design, fully accessible to be explored and modified without writing a single line of code.

Requirements

User Prerequisites

- Good English skills
- Good knowledge of Unreal Engine and Blueprint programming
- Unreal Engine version 5.5 or higher installed
- Chaos Vehicles Plugin Activated in Unreal Engine 5

Minimum Hardware Requirements

- Operating System: Windows 10 (64-bit)
- Processor: Intel or AMD quad-core, 2.5 GHz or higher

- RAM: 8 GB
- Graphics Card: DirectX 11 or 12 compatible
- Storage: SSD with at least 256 GB of free space

Recommended Hardware Requirements

- Operating System: Windows 11 (64-bit)
- Processor: Intel i7/i9 or AMD Ryzen 7/9
- RAM: 32 GB or more
- Graphics Card: NVIDIA RTX 2080 or AMD Radeon RX 5700 XT or better
- Storage: NVMe SSD with at least 1 TB of free space

Downloading and Installation

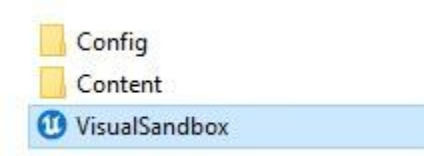
Downloading from Gumroad Store

After getting the Visual Sandbox from the gumroad store, you will be redirected to the VS content page on gumroas

1. Choose the desired project version and click the download button.

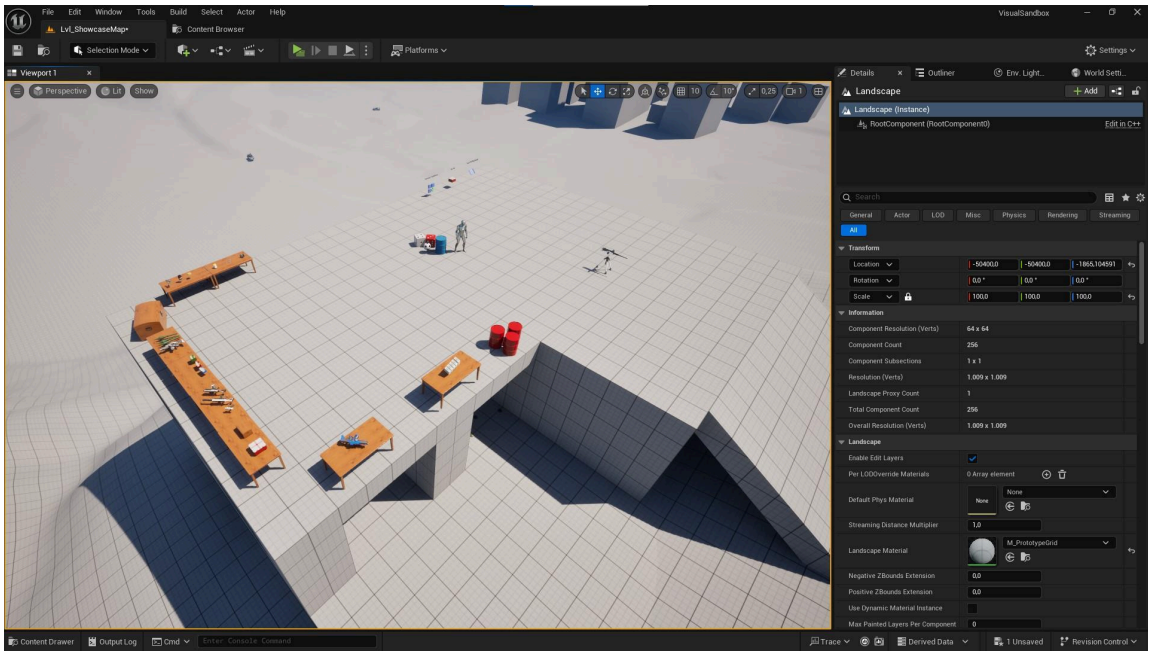
2. A Zip file of the Visual Sandbox Project will be downloaded. extract the contents of the Zip file to the desired folder

3. Open the Visual Sandbox folder and run the VisualSandbox.exe file and the unreal engine 5 editor will automatically open the project



You can also rename the VisualSandbox.exe file to any name you desire.

4. The default showcase map will be loaded, and it's done!



Downloading from Fab.com

After getting the Visual Sandbox from Fab.com

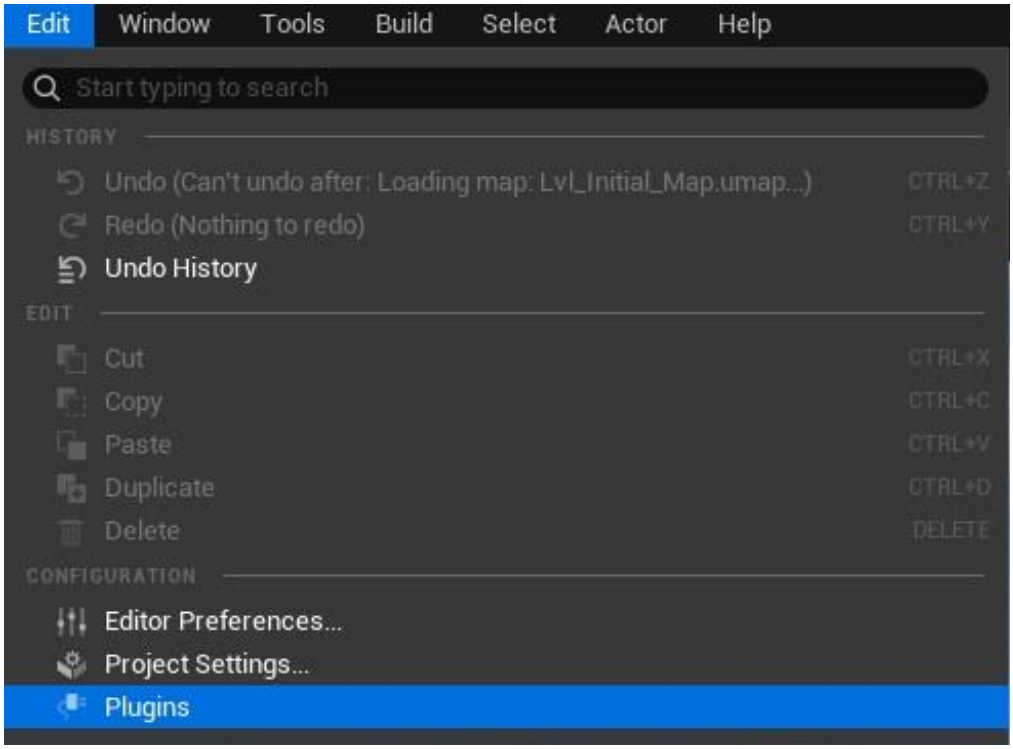
1. Open the Epic Games Launcher
2. Find Visual Sandbox in your Library and click **Create Project**
3. Choose the Installation location and click Create

Installing Chaos Vehicles Plugin.

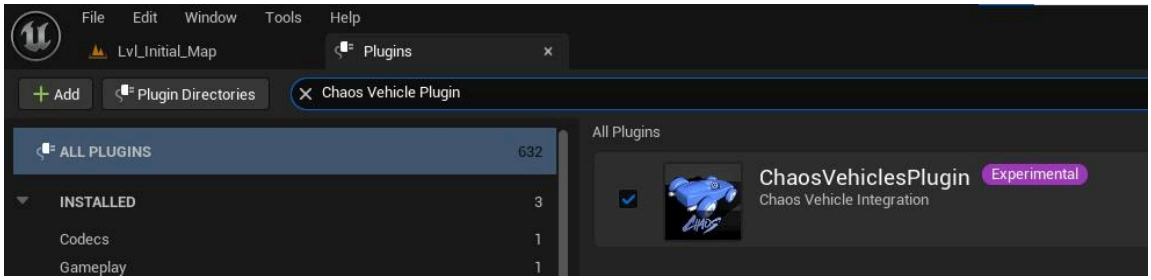
Once the project launches, it will automatically load the default showcase map, and you may need to activate the **Chaos Vehicles Plugin.** a native plugin of the unreal engine to unlock wheeled vehicle class for cars. Without this plugin activated, cars will not appear on the editor.

Activating the **Chaos Vehicles Plugin.**

1. On the top of the editor window, click on **Edit** > **Plugins**



2. On the Plugins panel search for **Chaos Vehicles Plugin** and click on the check box to activate it.



3. Once activated, restart the editor to apply the change.

The Third Person Shooter game mode is already set as the default Game Mode in the project settings. This means no manual configuration is required after installation.

About The Author

My name is Silas Santos, I'm a 29 years old Brazilian, currently living and working in Japan.

I have a lot of experience in Unreal Engine, Blueprint, and C++ programming, as well as 3D modeling, texturing, and optimization techniques, acquired over years of intensive practice using different types of software. I also have strong skills in graphic design, video editing and music production, developed over decades of study and practice since my childhood. All these skills combined have given me a broad understanding of game development with an artistic touch.

My recent works

Warwolf Project - (Castle builder game) Alpha Trailer 2023

Top Down Shooter Pro v1.4 - Showcase

Quick Tutorials

Quick Tutorials

Player

- How to change the HUD Widget?
- How to add New Input Actions?
- How to customize the Third Person Camera?

Items

- How to add a new Firearm?
- How to add a new item?
- How to change Item Widget?

Storage System

- How to make a global inventory?

Sentry Gun

- How do I set the sentry gun to rotate 360 degrees?

NPCs

- How to make NPC kill each other?
- How to change NPC initial weapons?
- How to remove NPCs Health bar?

Quick Questions

1. **Is it replicated?**
   Visual Sandbox is designed for single player experience

2. **Is it possible to implement replication?**
   It is totally possible to implement replication, however you will need some extra work.

3. **Can I use Visual Sandbox for commercial purposes?**
   It's not recommended for commercial use in this version, because many improvements still need to be made. however nothing prevents you from using it commercially as long as you know what you are doing.

4. **Where can I find character movement input actions?**
   All input actions for character movements and actions are within the **BP_ShooterCharacter** blueprint.

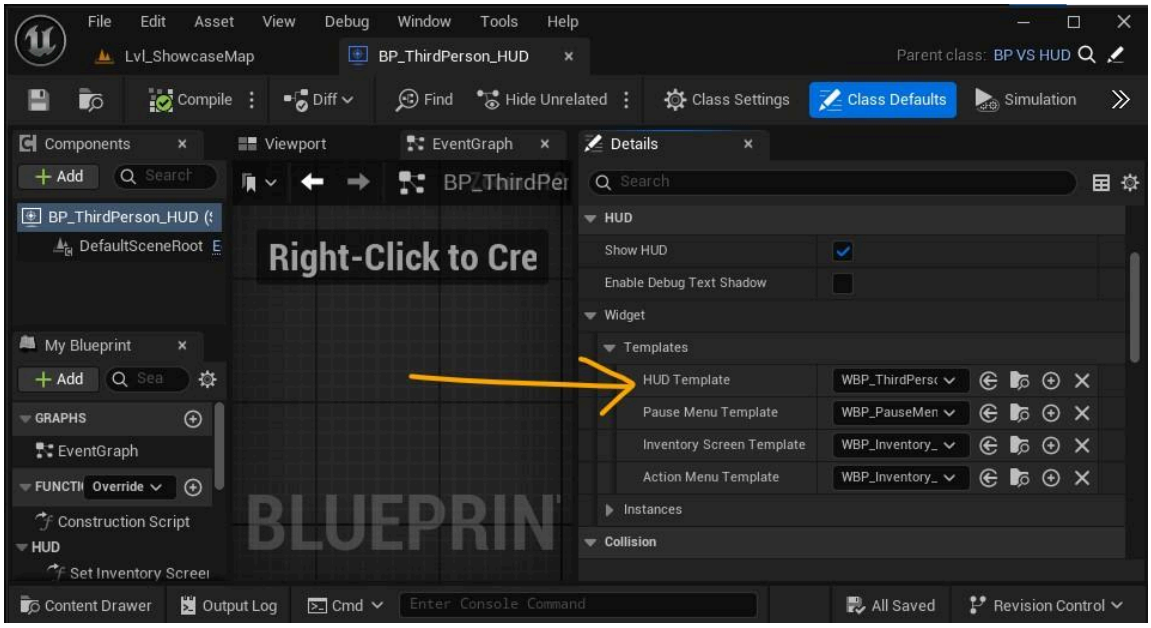
5. **How does the respawn system work?**
   See, When the **BP_ShooterCharacter's** health points reach 0, the **Dying Event** of the Shooter Character is triggered. The **Dying Event**, in turn, notifies the player controller that the pawn is dead through the **Report Pawn Death Interface event**. The **BP_ThirdPerson_PC** sends a respawn request to **BP_VS_GameMode**, which in turn searches for **BP_RespawnPoint** closest to the player's death location and spawn a new **BP_ShooterCharacter** pawn to be possessed by the **BP_ThirdPerson_PC** again. ending the respawning cycle.

New tutorials and answers coming soon, in the meantime you can send your question directly to the New Visual Sandbox server on discord
Server Link: https://discord.gg/NYZ8HDn3U5

Changing the HUD widget

1. Locate and open the **BP_ThirdPerson_HUD** on the Content browser

2. Change the **HUD Template** value to another desired User Widget.



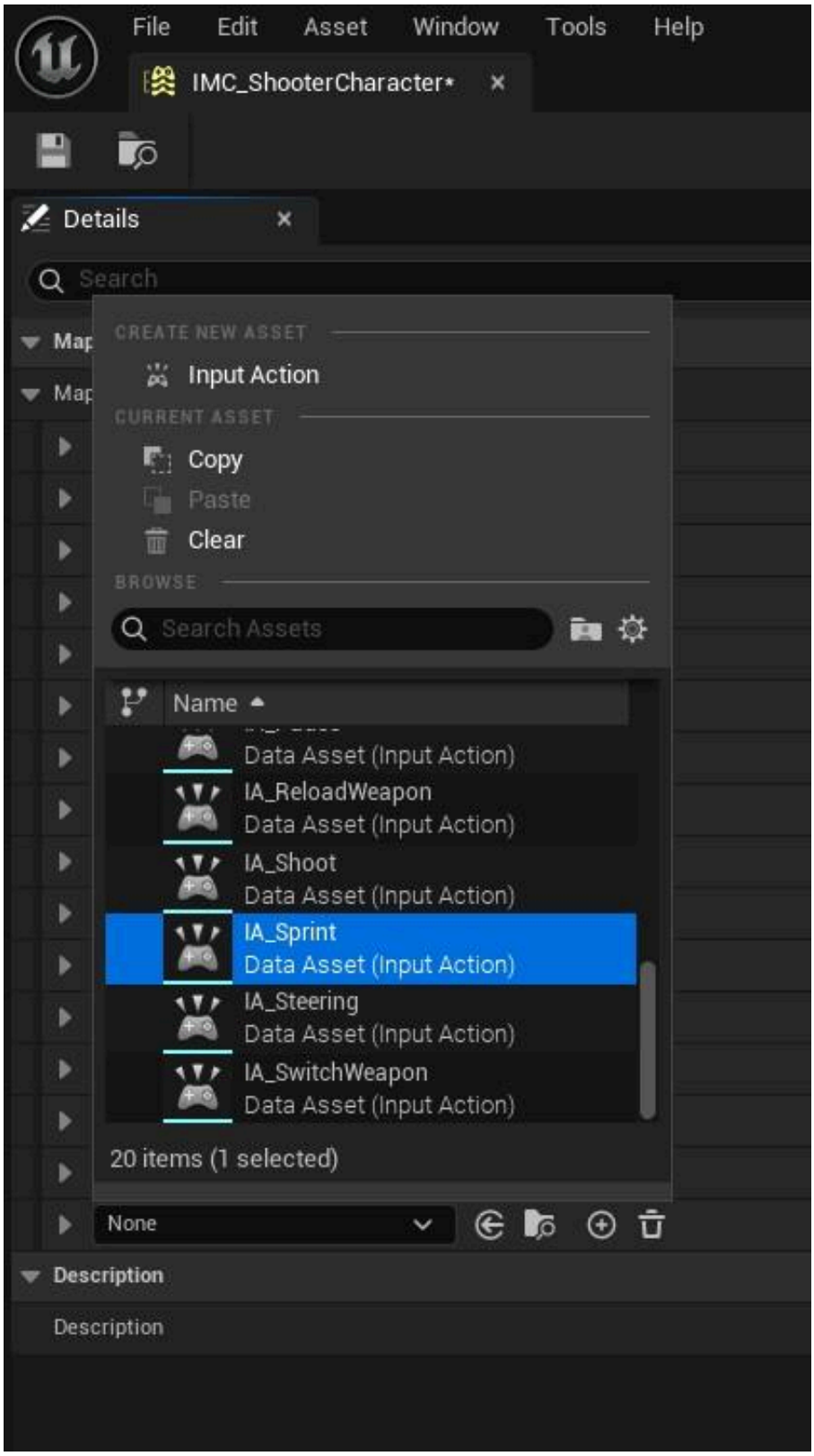
3. Hit compile, and it’s done.

Adding New Action Input

1. Locate and open the **IMC_ShooterCharacter** on the Content Browser

2. Click on plus sign to add a new Action Mapping

3. Select the desired Action Input Asset from the dropdown menu, and bind an input key on it.

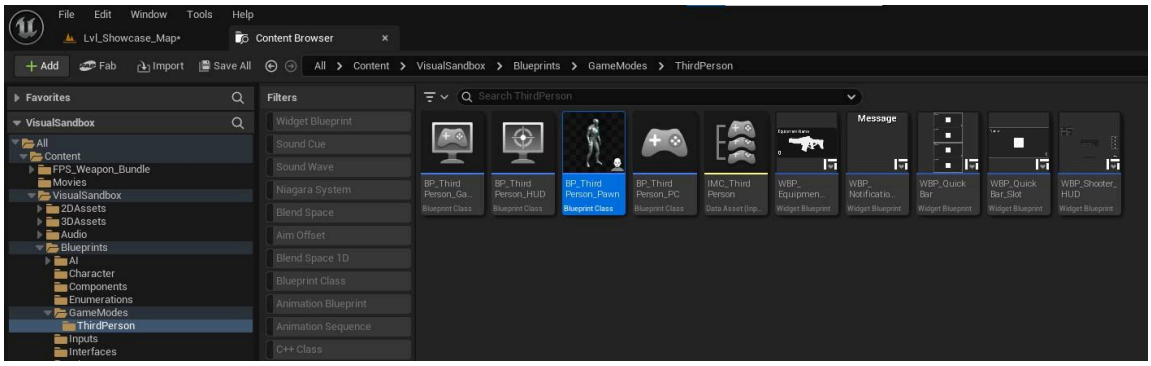


You can also change the input key of pre-existing Input Actions from here.

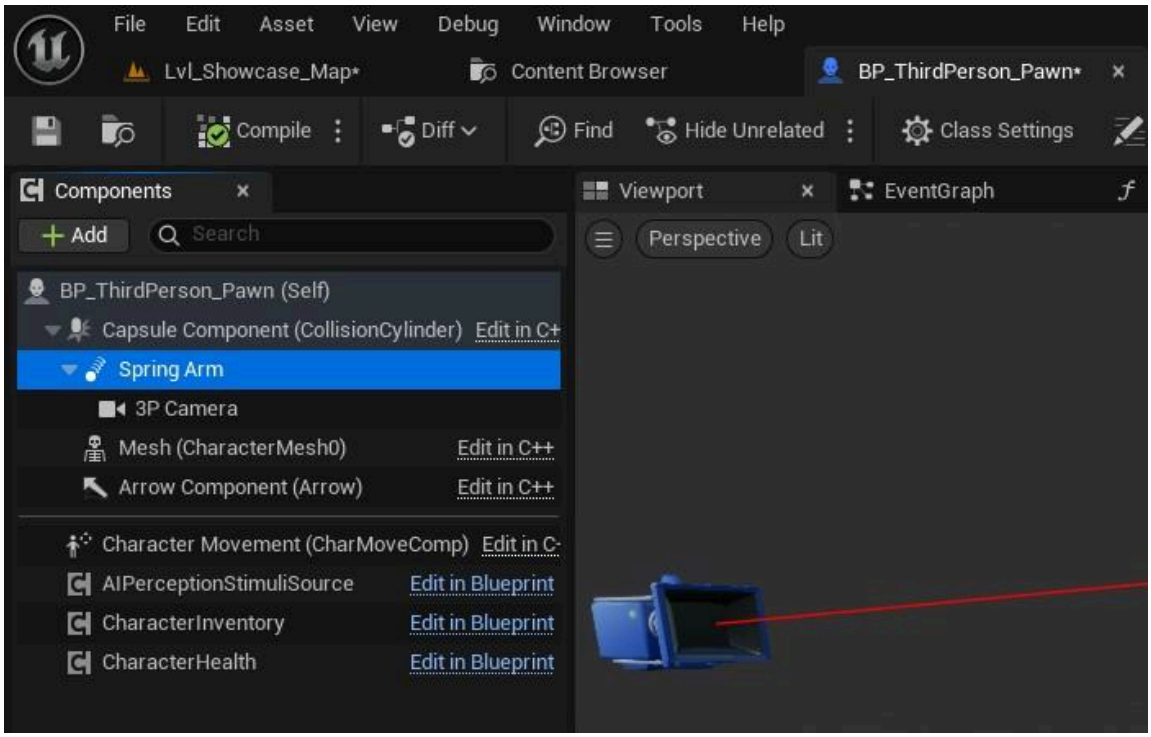

File
Edit
Asset
Window
Tools
Help
IMC_ShooterCharacter*
Details
Search
Mappings
Mappings
IA_Move
W
S
D
A
Up
Down
Gamepad Left Thumbstick 2D-Axis
IA_Look
IA_Jump
IA_Crouch
IA_Shoot
IA_Aim
IA_DropWeapon
IA_OpenInventory
IA_ReloadWeapon
IA_BulletTime
IA_Interact
IA_Exit_Vehicle
IA_Pause
IA_SwitchWeapon
IA_HolsterWeapon
Description
Description

Customizing the Third Person Shooter Camera

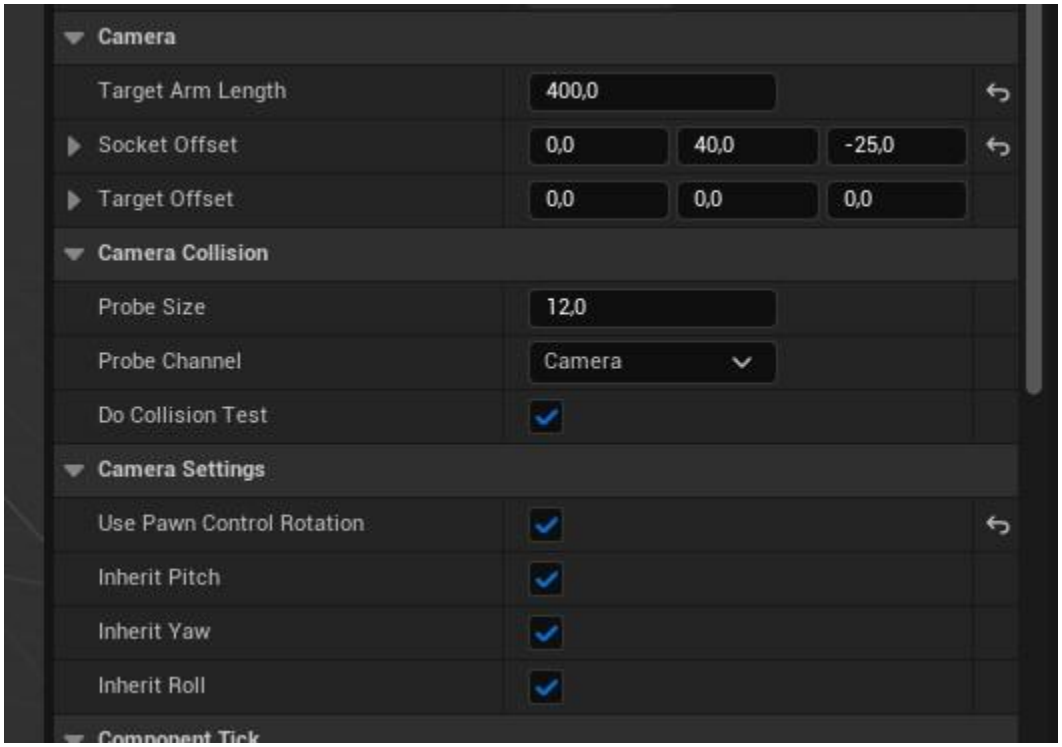
1. Locate and open the **BP_ThirdPerson_Pawn**



2. On the **Component** tab select the Spring Arm component

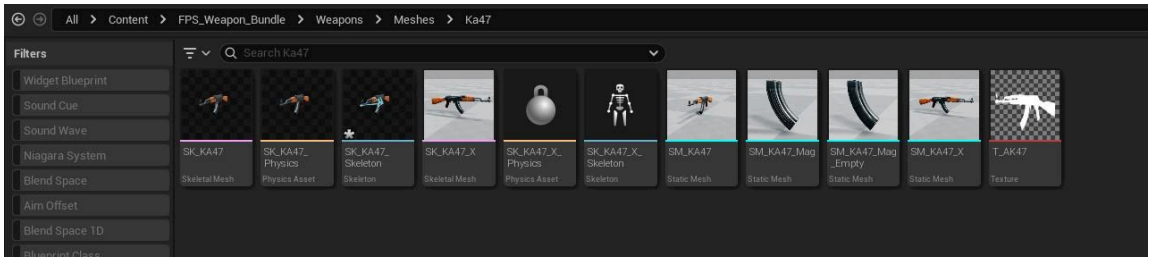


and change the default values on the **Detail** tab to desired Camera distance and offset



Creating a new Gun

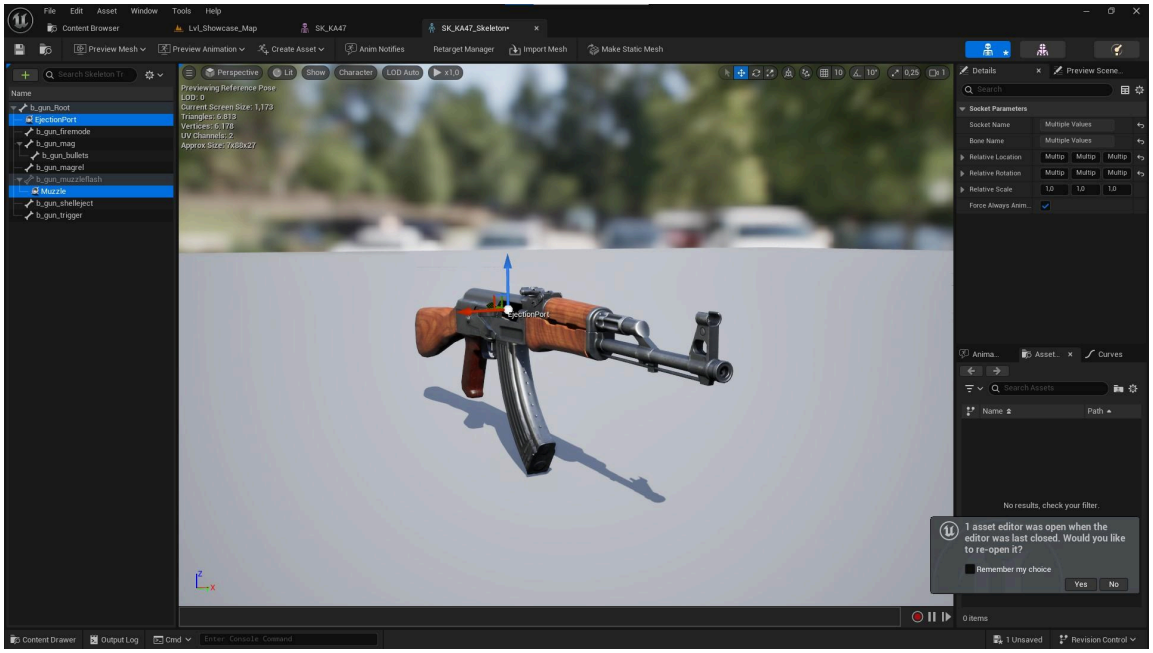
1. Import the Weapon Assets



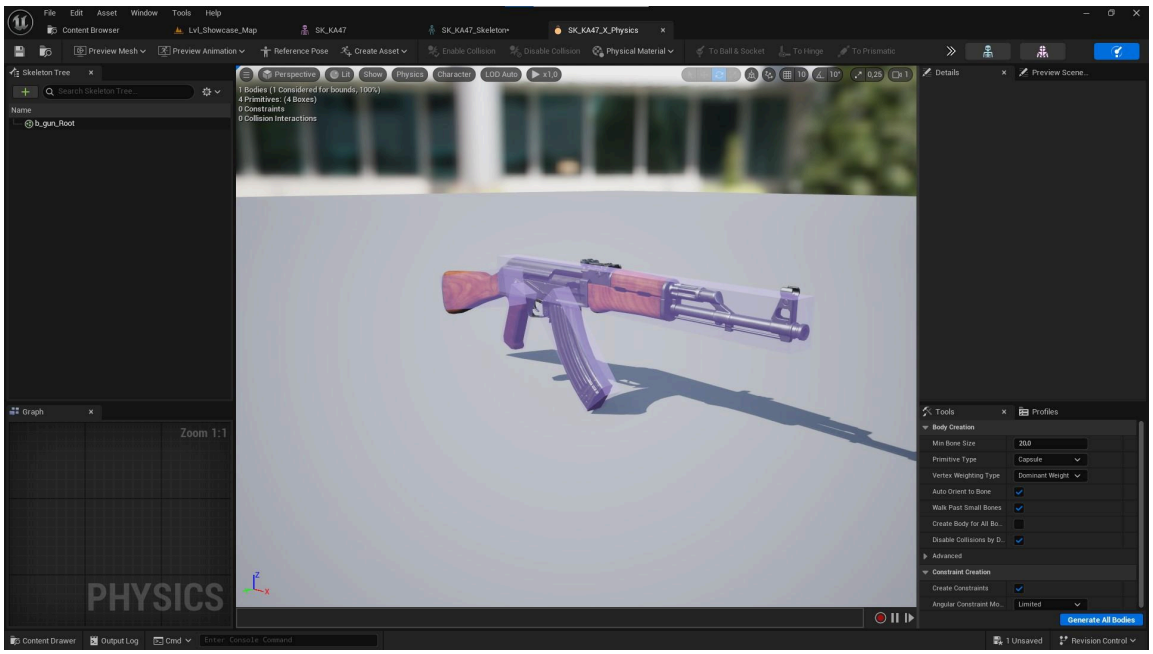
This AK model is a free Asset from [Fab.com](Fab.com)

2. Open the Weapon Skeletal Mesh and add two additional sockets, one called “Muzzle” where the projectile will be shooted attached on the tip of the Gun, and another one

called “EjectionPort” attached to the capsule ejection port.

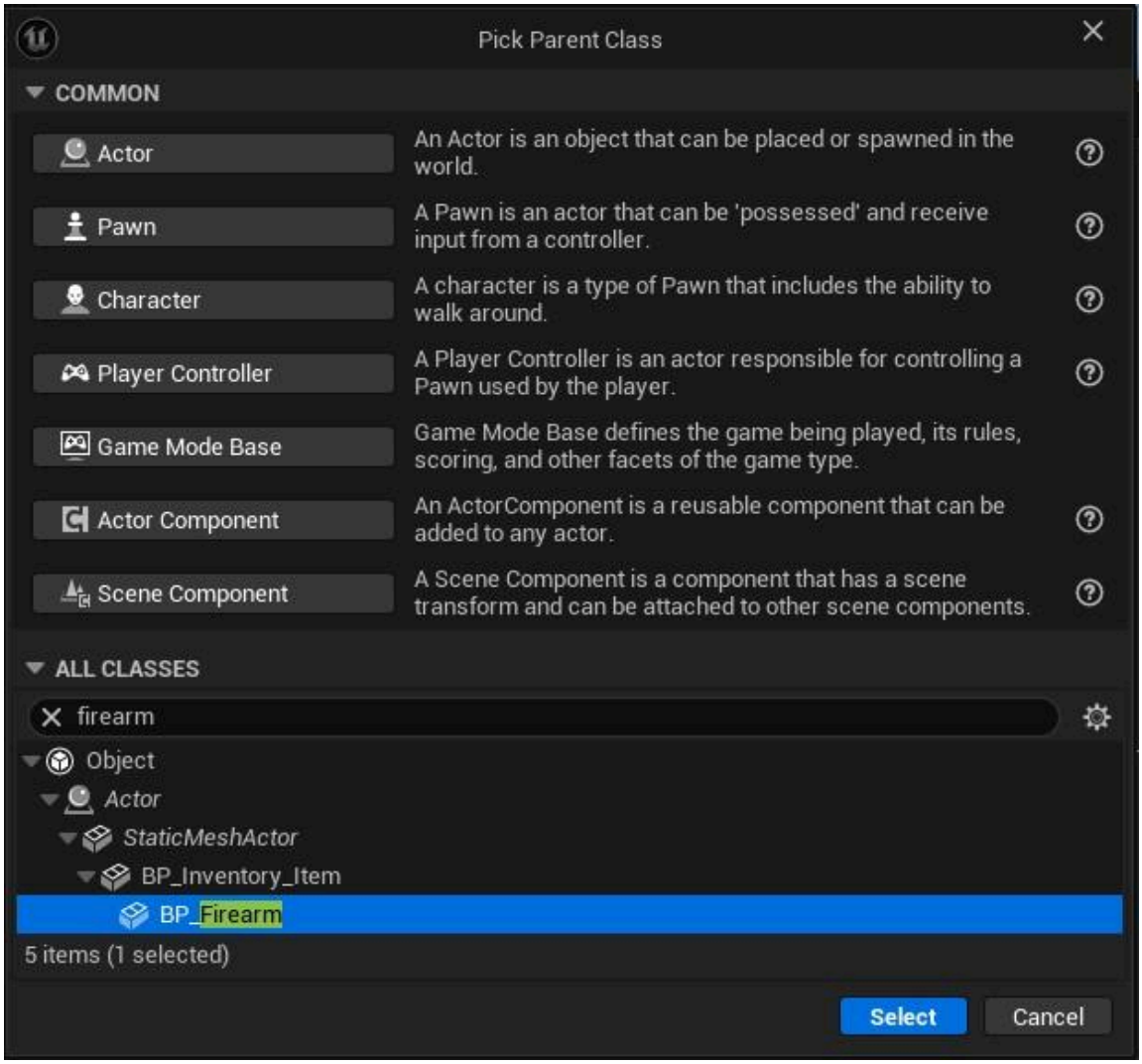


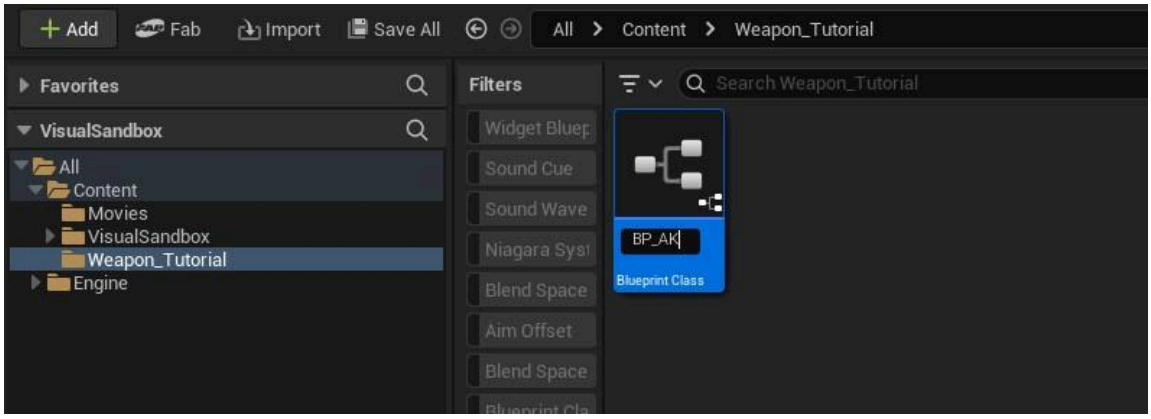
3. Create or assign a physics asset to the weapon mesh.



Without a Physical Asset assigned, Unreal Engine cannot simulate physics and the weapon will float when dropped.

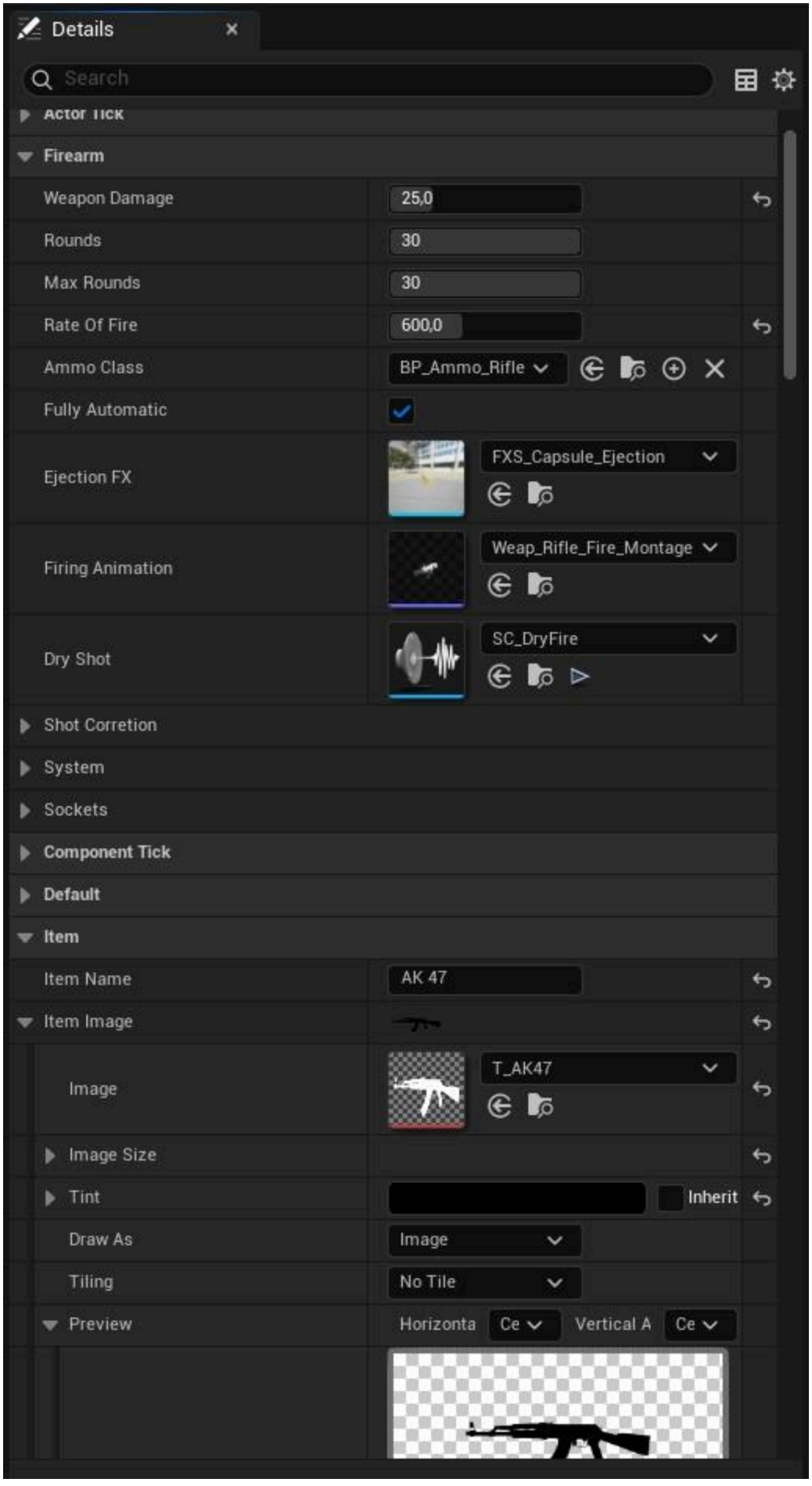
4. Create a new Blueprint picking **BP_Firearm** as parent class.



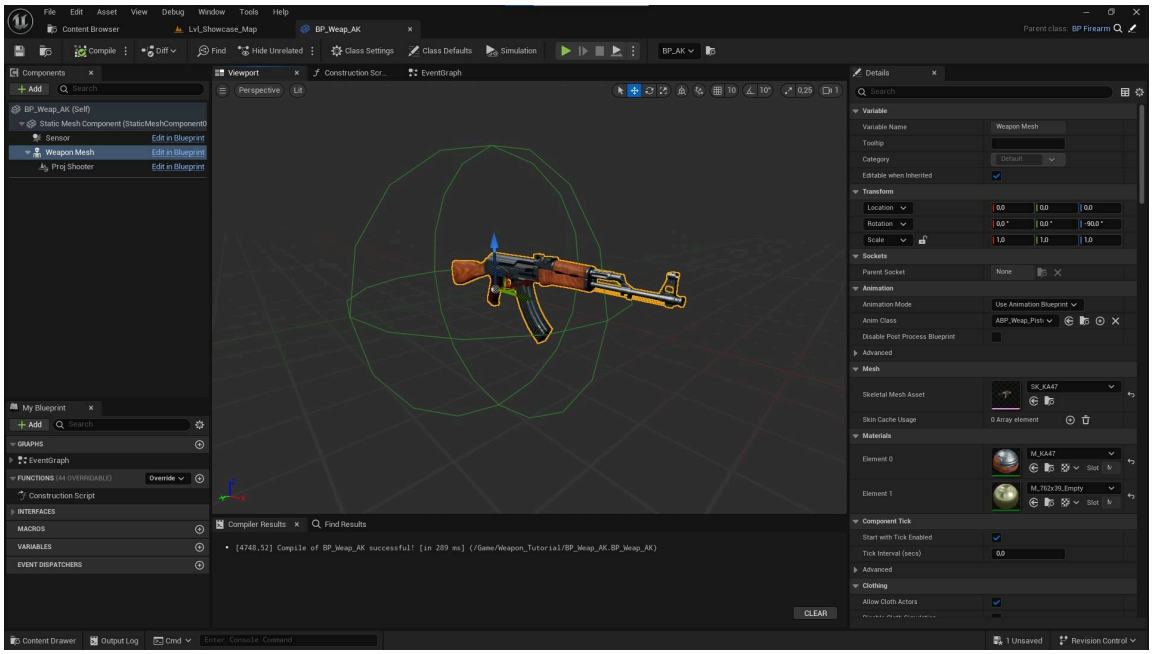


Name it.

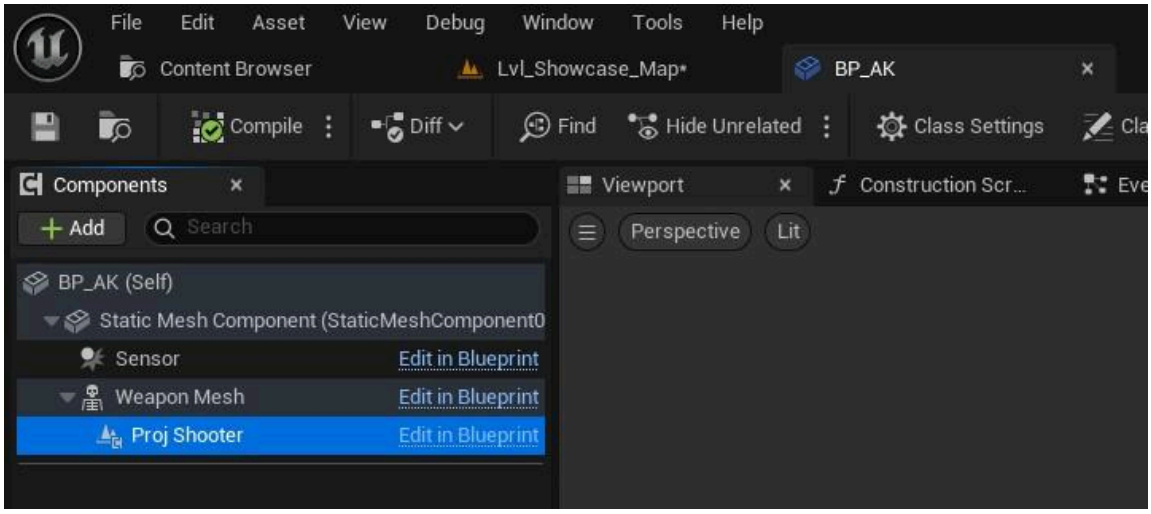
5. Open the new blueprint, and set all the default properties of the weapon such as number of rounds, max rounds, rate of fire, weapon, icon on the **Detail** tab.


Details
Search
Actor Tick
Firearm
Weapon Damage
25,0
Rounds
30
Max Rounds
30
Rate Of Fire
600,0
Ammo Class
BP_Ammo_Rifle
Fully Automatic
Ejection FX
FXS_Capsule_Ejection
Firing Animation
Weap_Rifle_Fire_Montage
Dry Shot
SC_DryFire
Shot Corretion
System
Sockets
Component Tick
Default
Item
Item Name
AK 47
Item Image
Image
T_AK47
Image Size
Tint
Inherit
Draw As
Image
Tiling
No Tile
Preview
Horizonta
Ce
Vertical A
Ce

6. Go to the viewport and set the weapon skeletal mesh by replacing the Skeletal Mesh slot of the WeaponMesh Component.

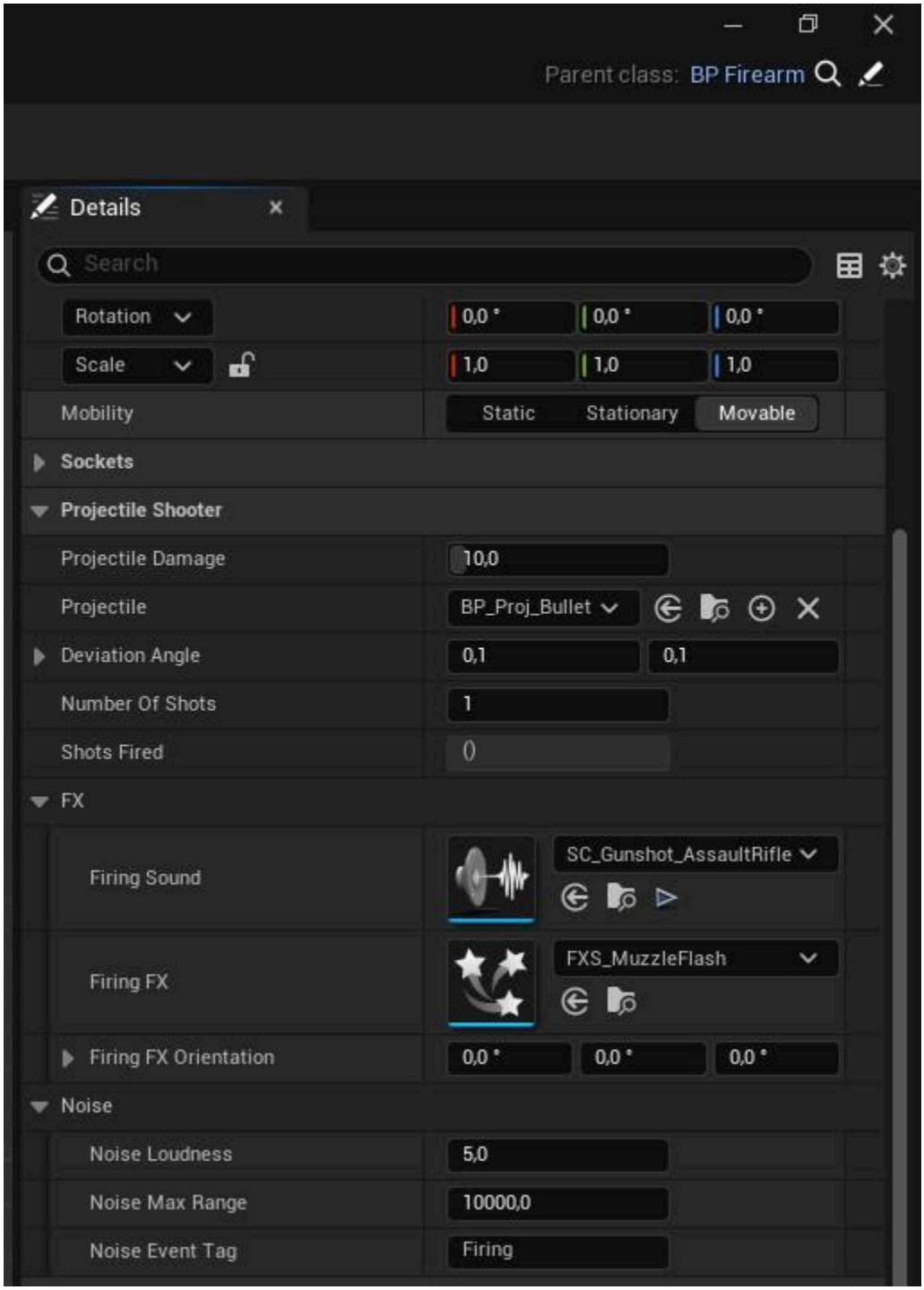


7. Go to Component tab and select on the Muzzle Component

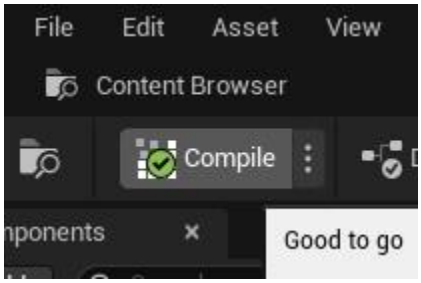


And back to detail tab and set the muzzle properties such as Projectile Class to shoot,

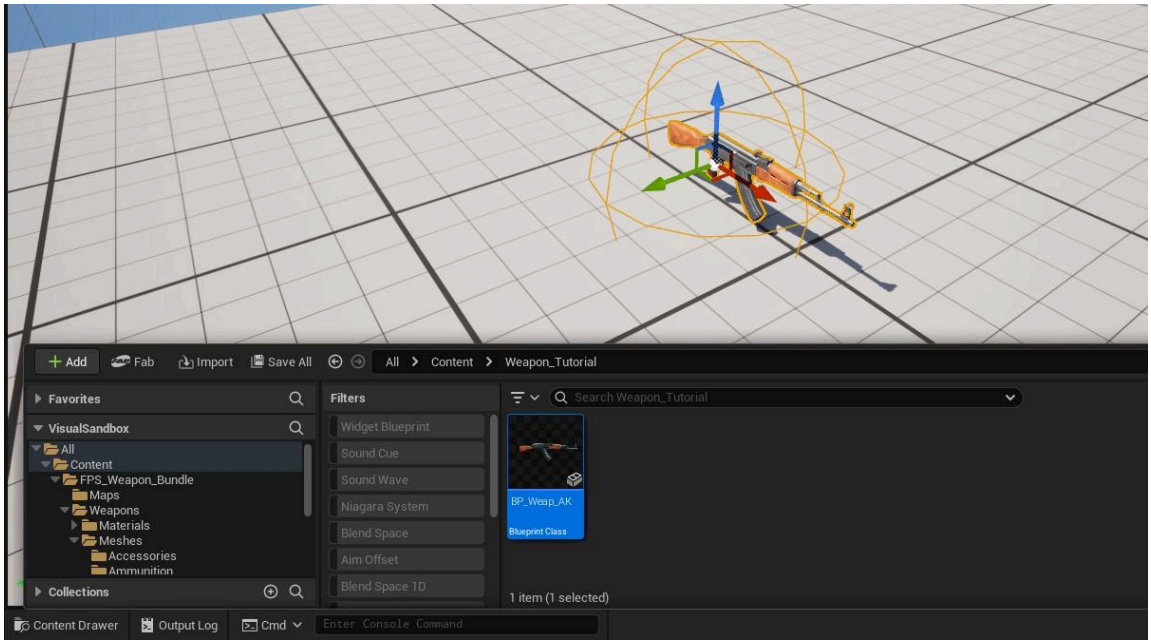
Deviation Angle to simulate inacuracy, Firing sound and visual effect

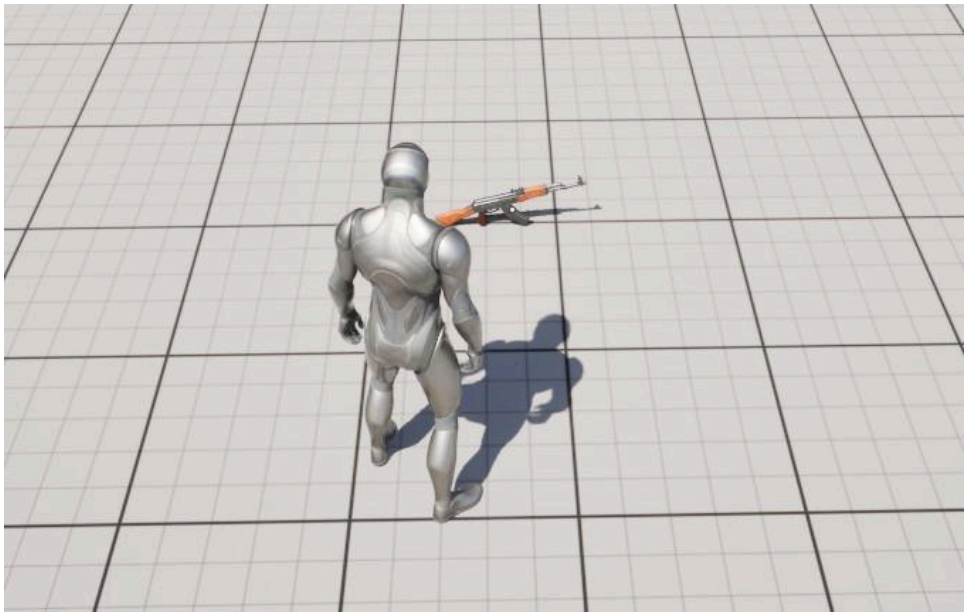


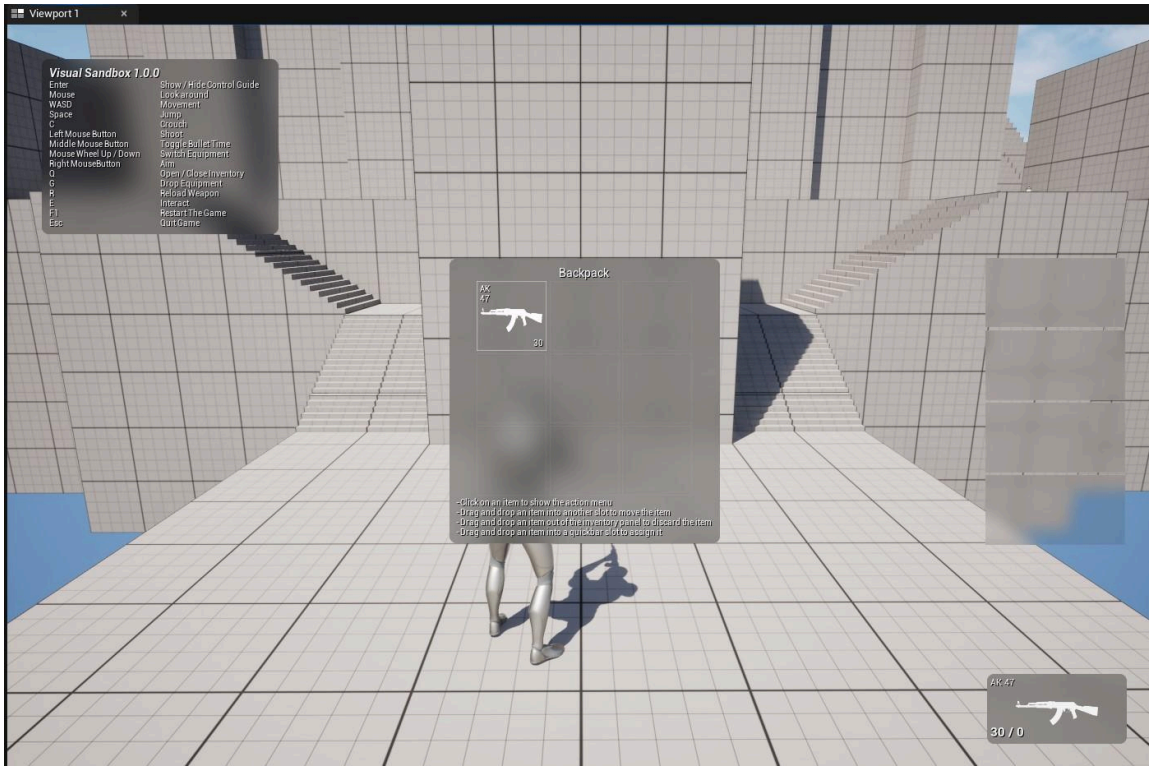
8. Hit compile and it's done



Now there are two ways to equip the weapon that you just created, the first one is just drag and drop the weapon blueprint from the content browser straight into level, and pick up during the game by touching it. and open the inventory pressing ‘Q” and click on the weapon and click on Equip button from the drop down menu to equip.



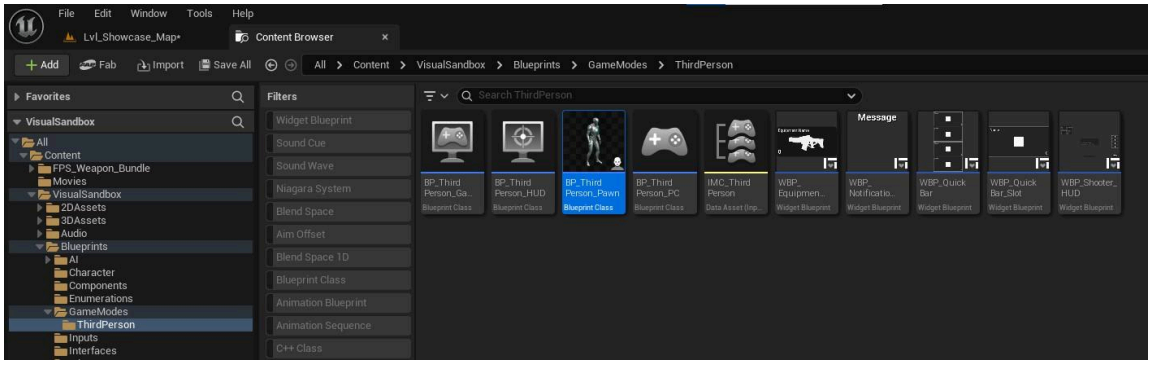



Viewport 1
Visual Sandbox 1.0.0
Enter Show / Hide Control Guide
Mouse Look around
WASD Movement
Space Jump
C Crouch
Left Mouse Button Shoot
Middle Mouse Button Toggle Bullet Time
Mouse Wheel Up / Down Switch Equipment
Right MouseButton Aim
Q Open / Close Inventory
G Drop Equipment
R Reload Weapon
E Interact
F1 Restart The Game
Esc Quit Game
Backpack
AK 47
30
-Click on an item to show the action menu
-Drag and drop an item into another slot to move the item
-Drag and drop an item out of the inventory panel to discard the item
-Drag and drop an item into a quickbar slot to assign it
AK-47
30 / 0

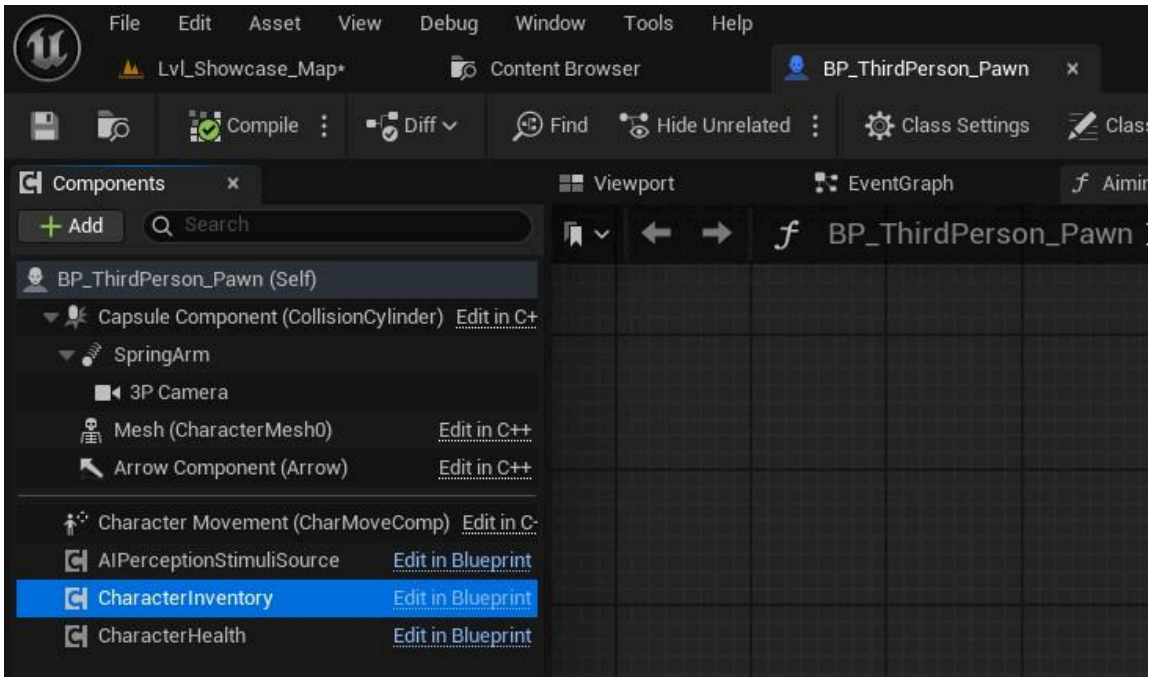
Now you can shoot around

The second way is adding directly on the Inventory of Third Person for that:

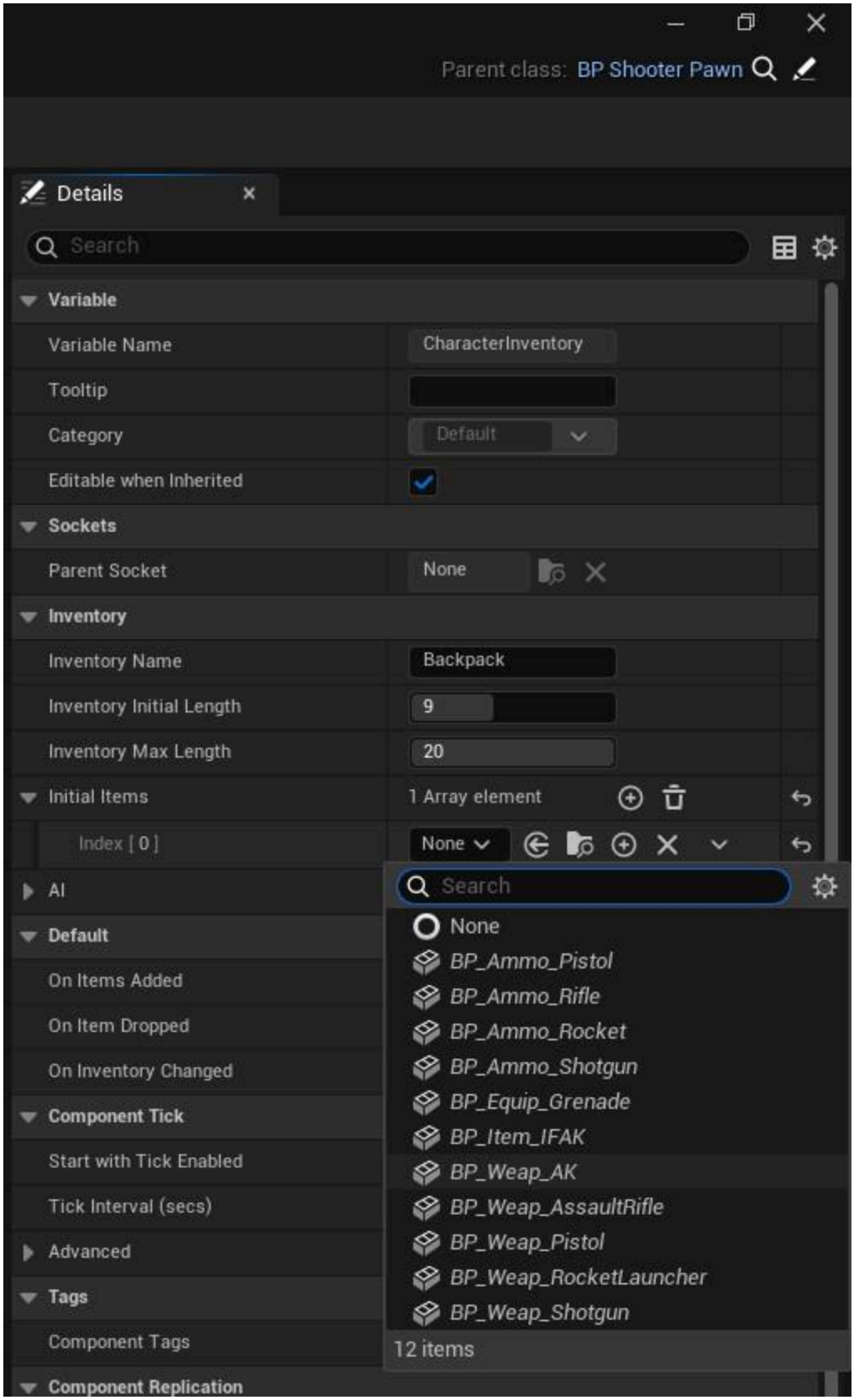
1. Locate and open the BP_ThirdPerson_Pawn.



2. On the Component tab click on the Character Inventory Component, and go to the details tab and find the “Initial Items” array



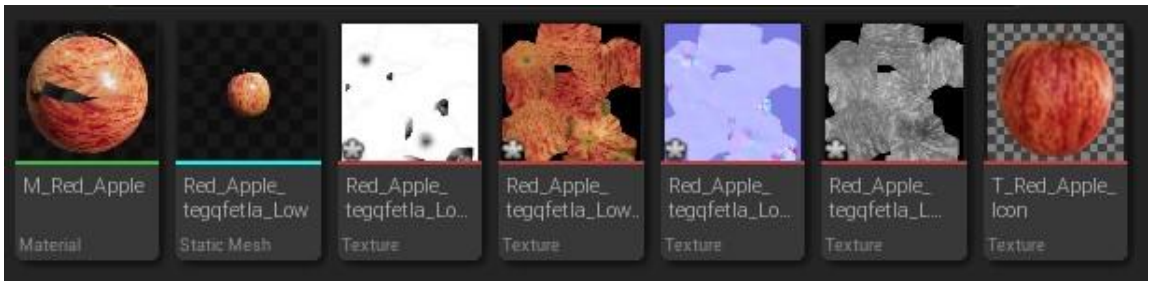
3. On the “Initial Items” click on plus sign and choose the desired Weapon class that you want to be spawned within the inventory as initial weapon.


Parent class: BP Shooter Pawn
Details
Search
Variable
Variable Name
CharacterInventory
Tooltip
Category
Default
Editable when Inherited
Sockets
Parent Socket
None
Inventory
Inventory Name
Backpack
Inventory Initial Length
9
Inventory Max Length
20
Initial Items
1 Array element
Index [0]
None
AI
Default
On Items Added
On Item Dropped
On Inventory Changed
Component Tick
Start with Tick Enabled
Tick Interval (secs)
Advanced
Tags
Component Tags
Component Replication
Search
None
BP_Ammo_Pistol
BP_Ammo_Rifle
BP_Ammo_Rocket
BP_Ammo_Shotgun
BP_Equip_Grenade
BP_Item_IFAK
BP_Weap_AK
BP_Weap_AssaultRifle
BP_Weap_Pistol
BP_Weap_RocketLauncher
BP_Weap_Shotgun
12 items

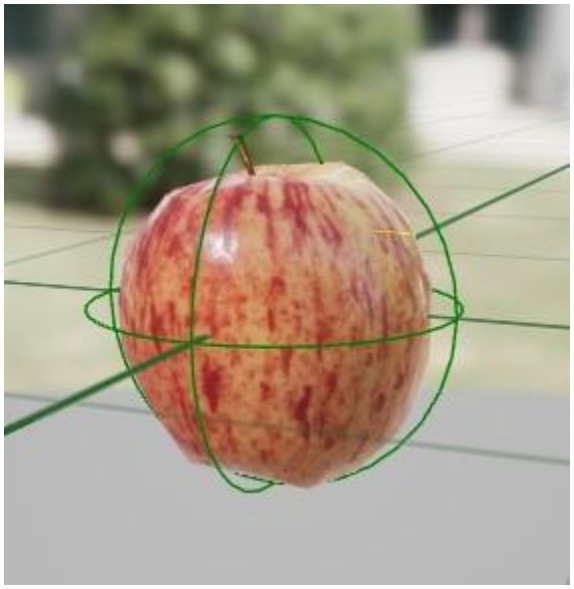
4. Hit compile and press play and open the Inventory by pressing “Q” and you will find the weapon there
5. Click on the weapon slots and select equip from the drop down menu to equip

Creating a new item in 7 steps

1. Download and import all assets related to your item, including mesh, textures, materials and 2D icon, into the editor



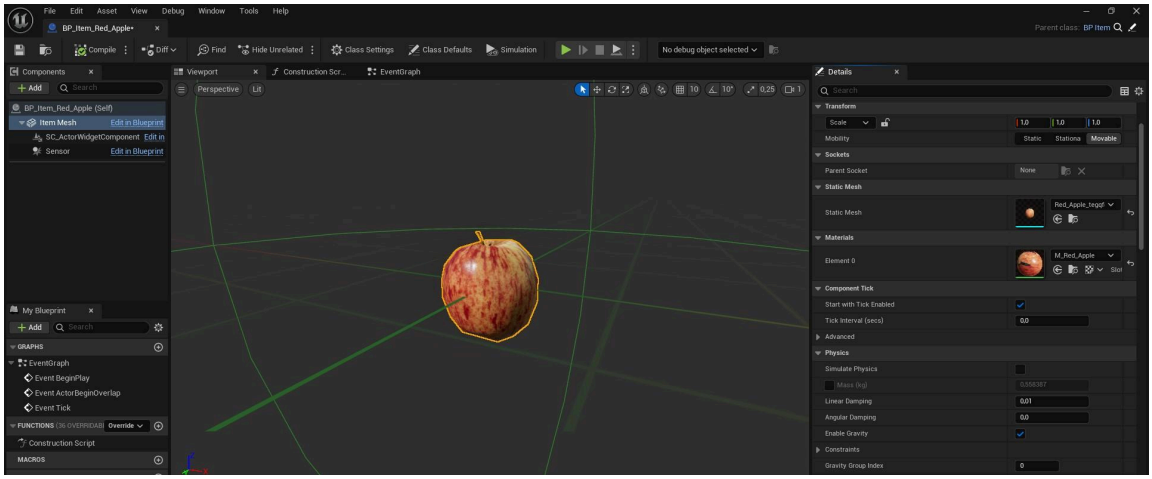
This Red Apple was downloaded from FAB



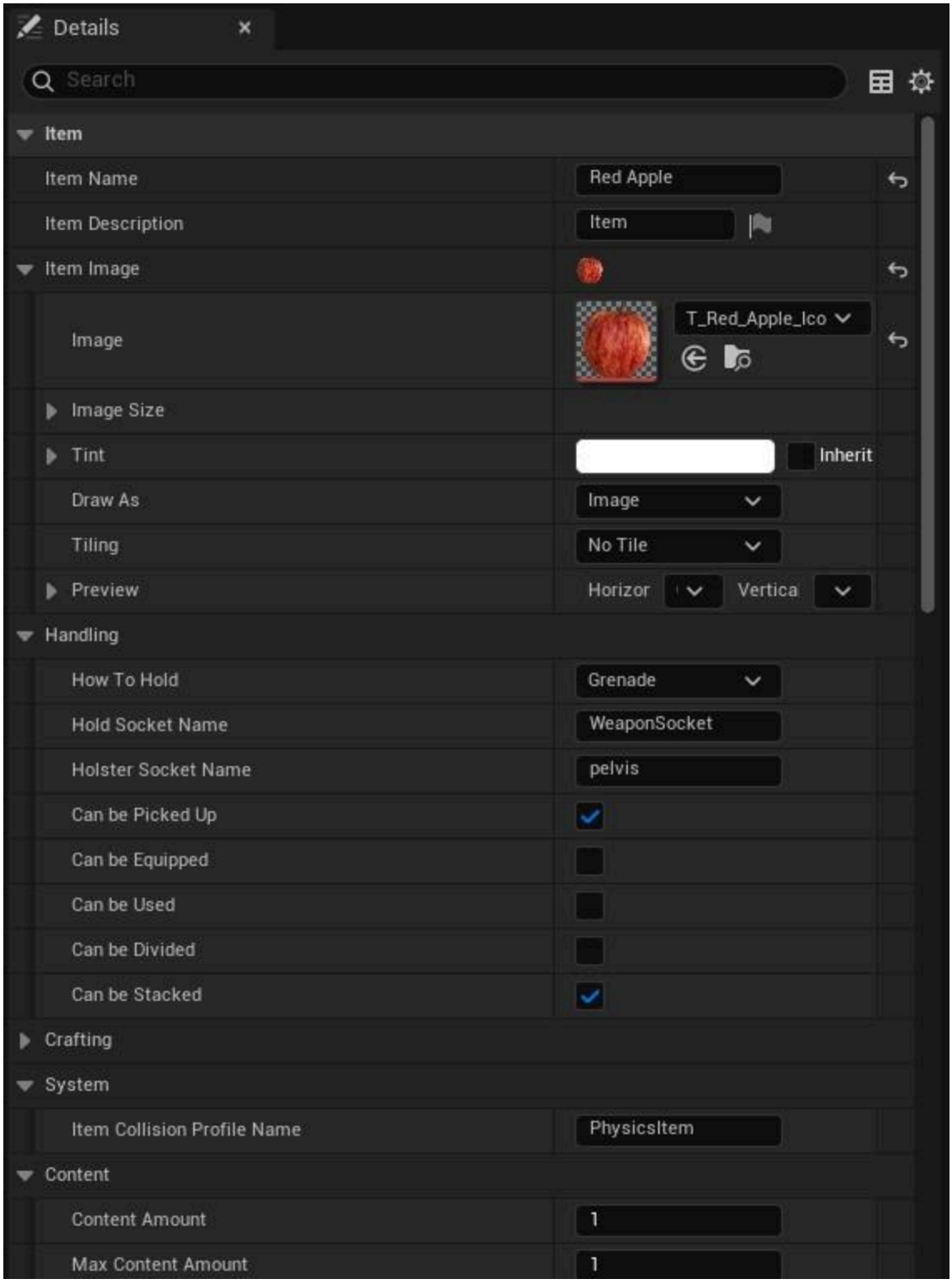
NOTE: Make sure to add collision to your item. Without collision the unreal engine cannot simulate the physics of the item, and it will be floating in the air during the gameplay.

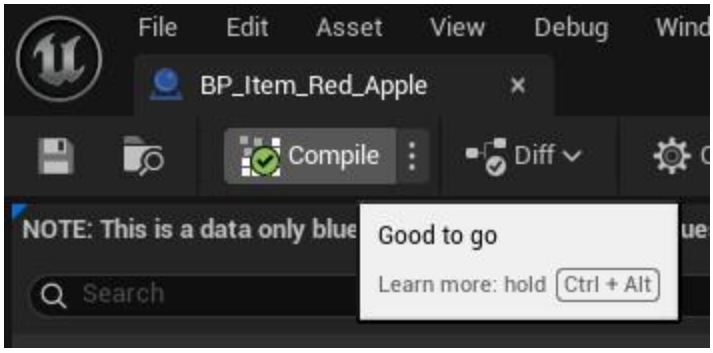
2. Create a new blueprint class picking **BP_Item** as parent class.

3. Open the item blueprint you just created, and go to the Viewport.
4. Select the Item Mesh component from the Components tab, then go to the Details tab and replace the Static Mesh slot with your desired static mesh asset.



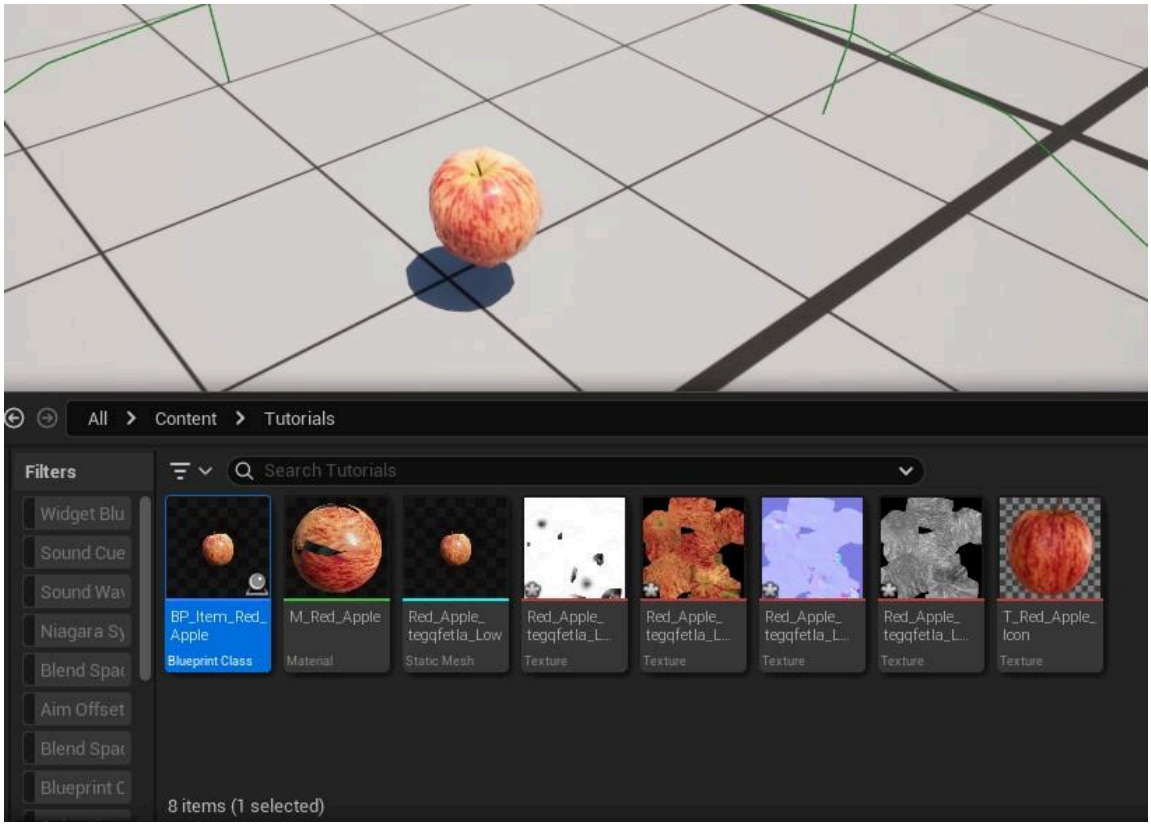
5. Go back to the Class Defaults' Details tab and set all the specifications for your item, such as the item name, description, icon, and so on.


Details
Search
Item
Item Name
Red Apple
Item Description
Item
Item Image
Image
T_Red_Apple_Ico
Image Size
Tint
Inherit
Draw As
Image
Tiling
No Tile
Preview
Horizor
Vertica
Handling
How To Hold
Grenade
Hold Socket Name
WeaponSocket
Holster Socket Name
pelvis
Can be Picked Up
Can be Equipped
Can be Used
Can be Divided
Can be Stacked
Crafting
System
Item Collision Profile Name
PhysicsItem
Content
Content Amount
1
Max Content Amount
1

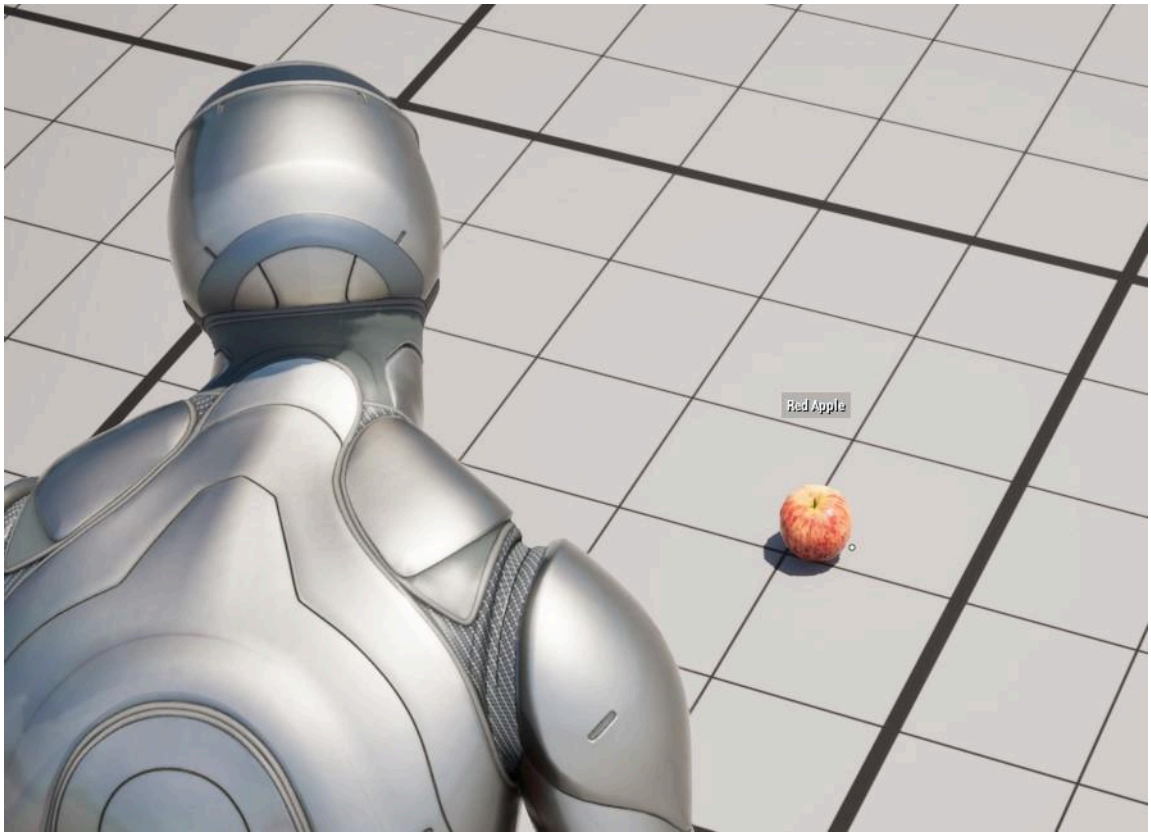


Hit compile, and it's done.

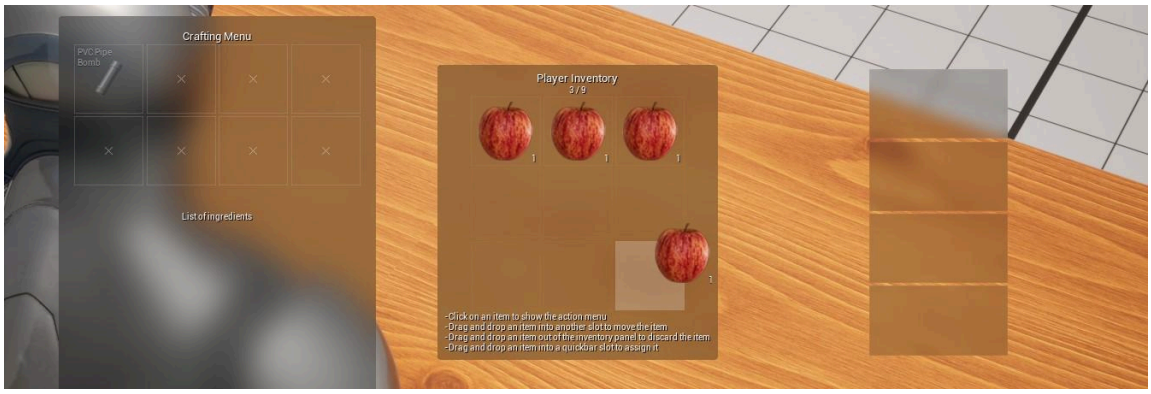
6. Go back to the Content Browser and drag the item blueprint directly into the level.



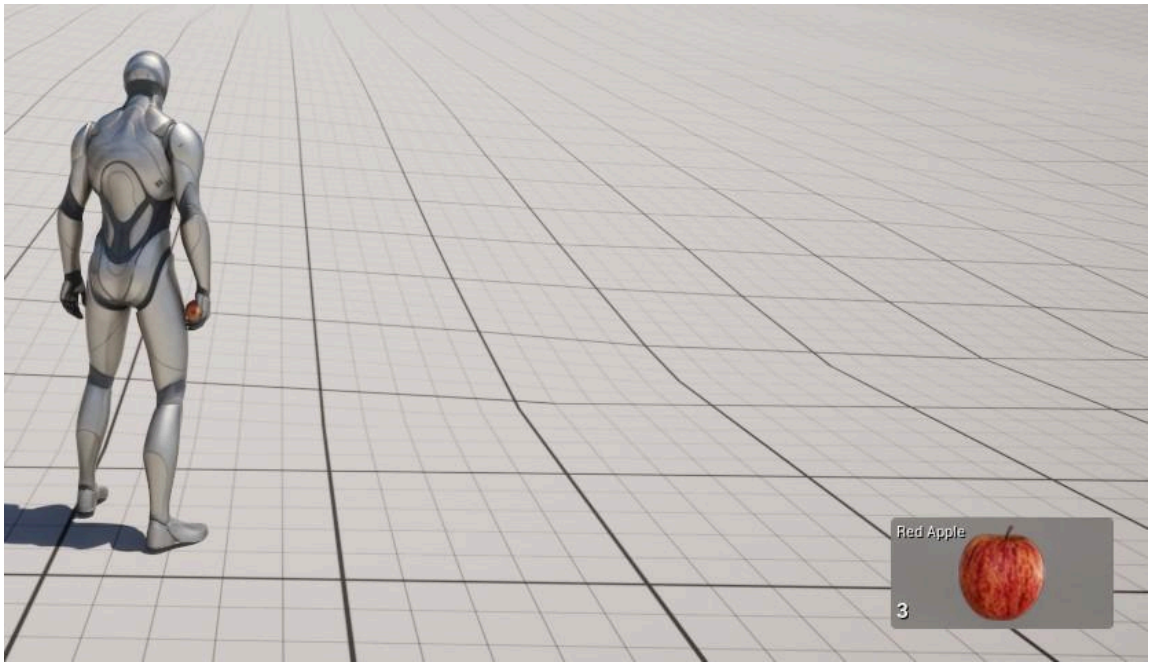
7. Hit Play and pick up the item by pressing the interaction button during gameplay.



After you pick up the item, it will automatically be added to the inventory.



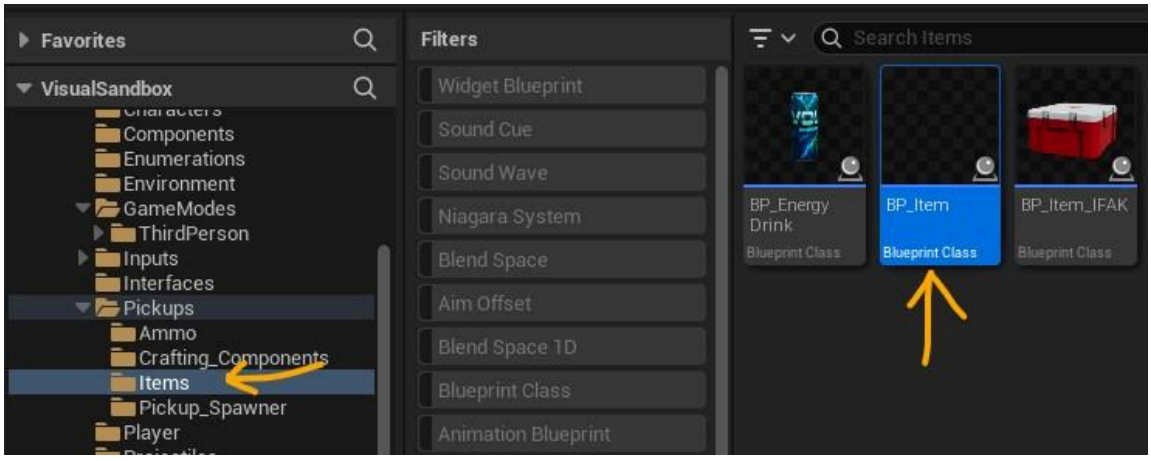
When you open the inventory, you will be able to drag and drop items to move or discard.



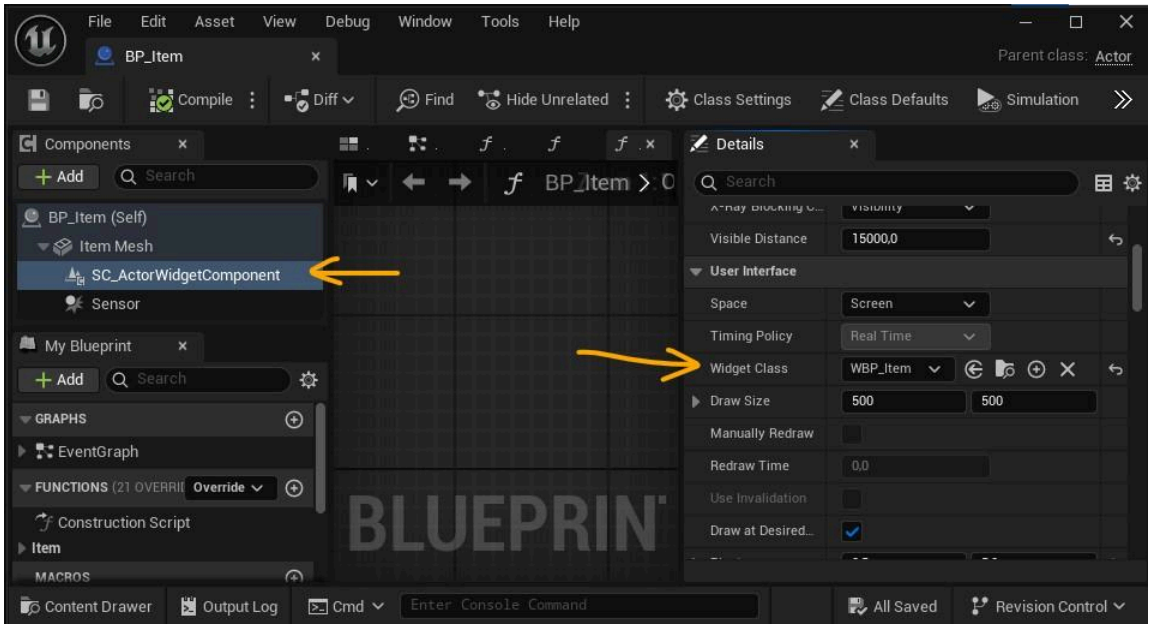
As of version 1.4, you can equip any item from your inventory, currently you can only equip weapons and grenades

Changing Item Widget

1. Locate and open the **BP_Item** blueprint on the Content Browser



2. On the components tab select the **AC_ActorWidgetComponen**

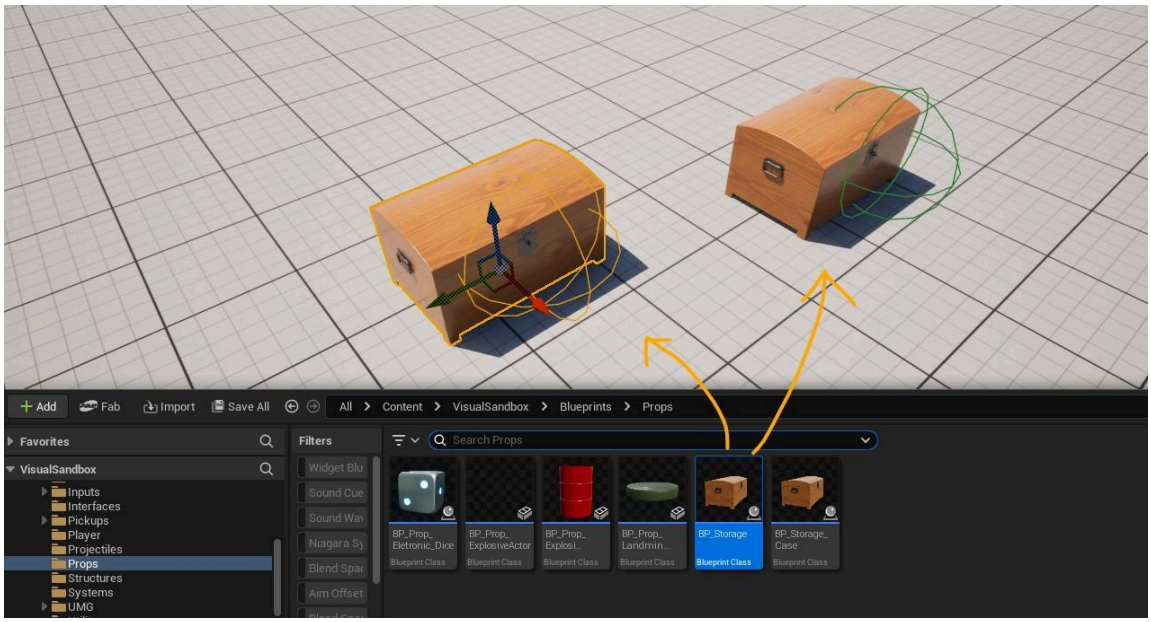


3. With the **AC_ActorWidgetComponent** still selected, look for **Widget Class** on the details tab, and change the widget class value.

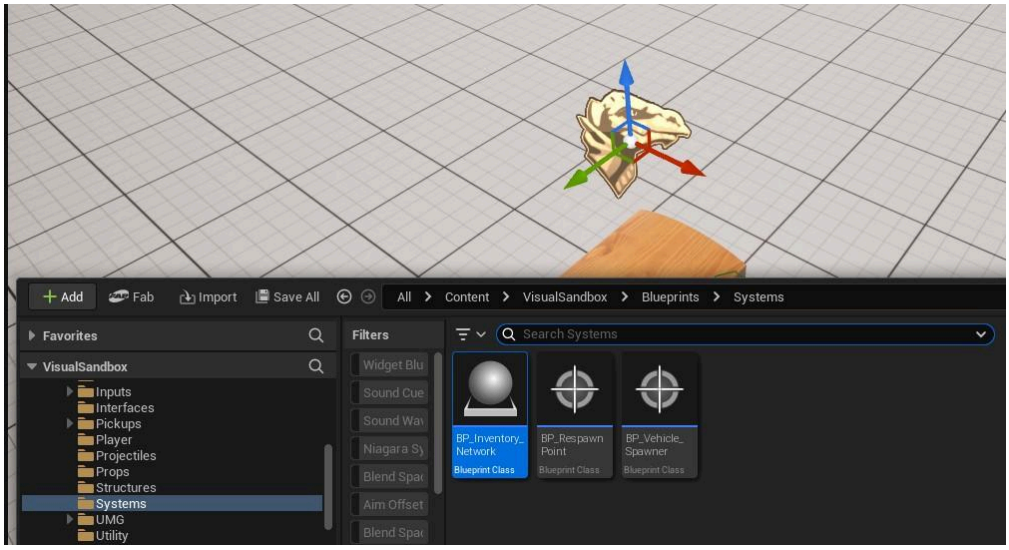
If you don't want any widget to appear, just delete the SC_ActorWidgetComponent and you'll be fine.

Making Global Storage System

1. Add two **BP_Storage** blueprint into the level

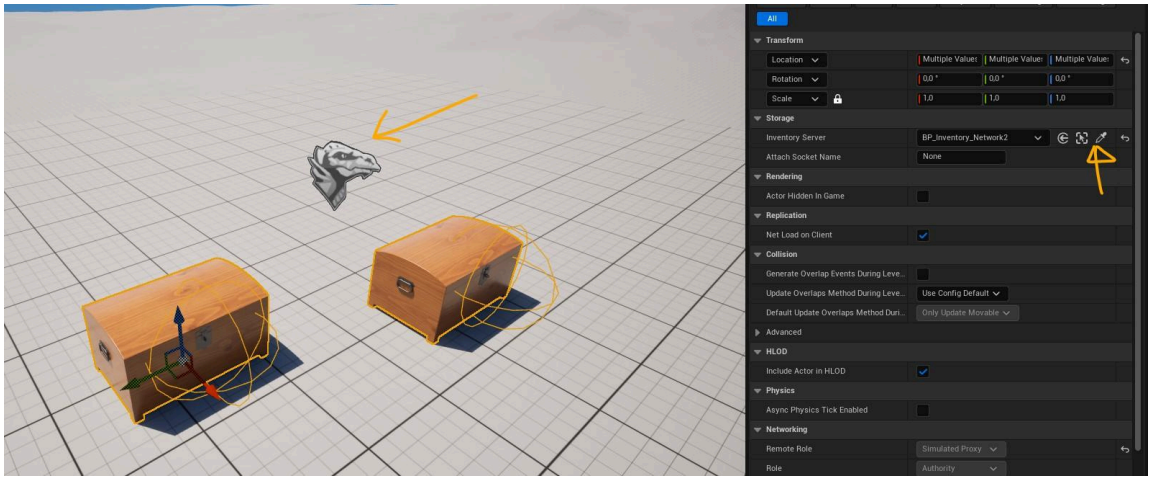


2. Add another blueprint called **BP_Inventory_Network**



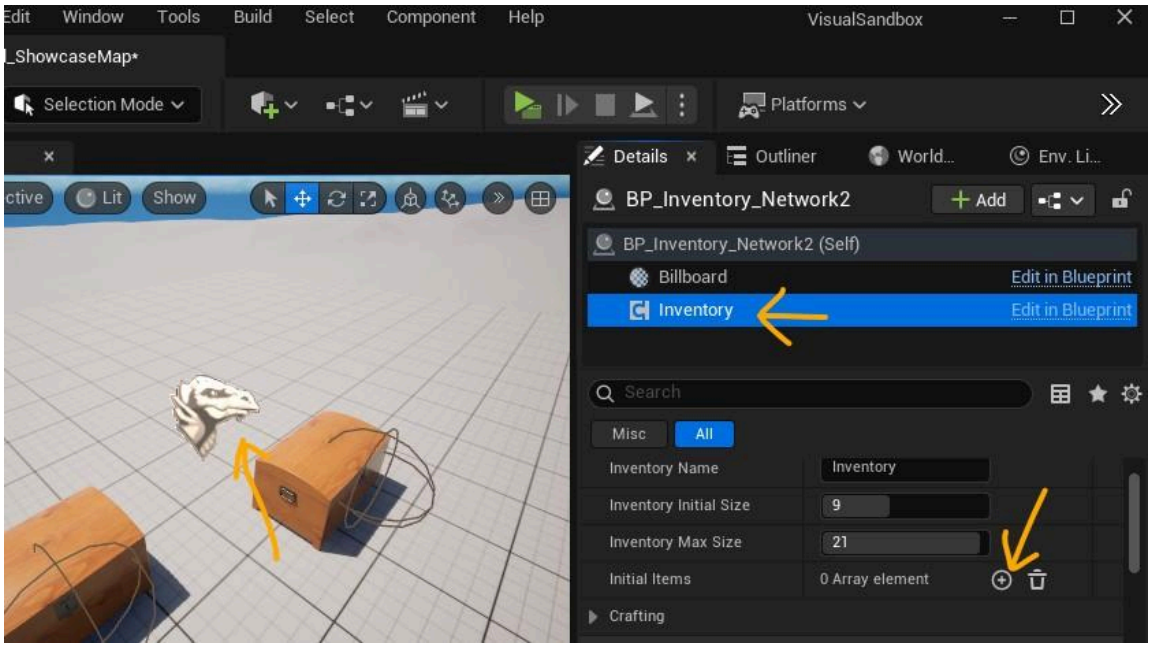
The **BP_Inventory_Network** is an Inventory server that can be accessed from anywhere on the map

3. Select the two chests**.** go to the details tab, and look for the **Inventory Server** variable.
4. Click on the dropper icon  and pick the **BP_Inventory_Network** placed on the level (Dinosaur Sprite)

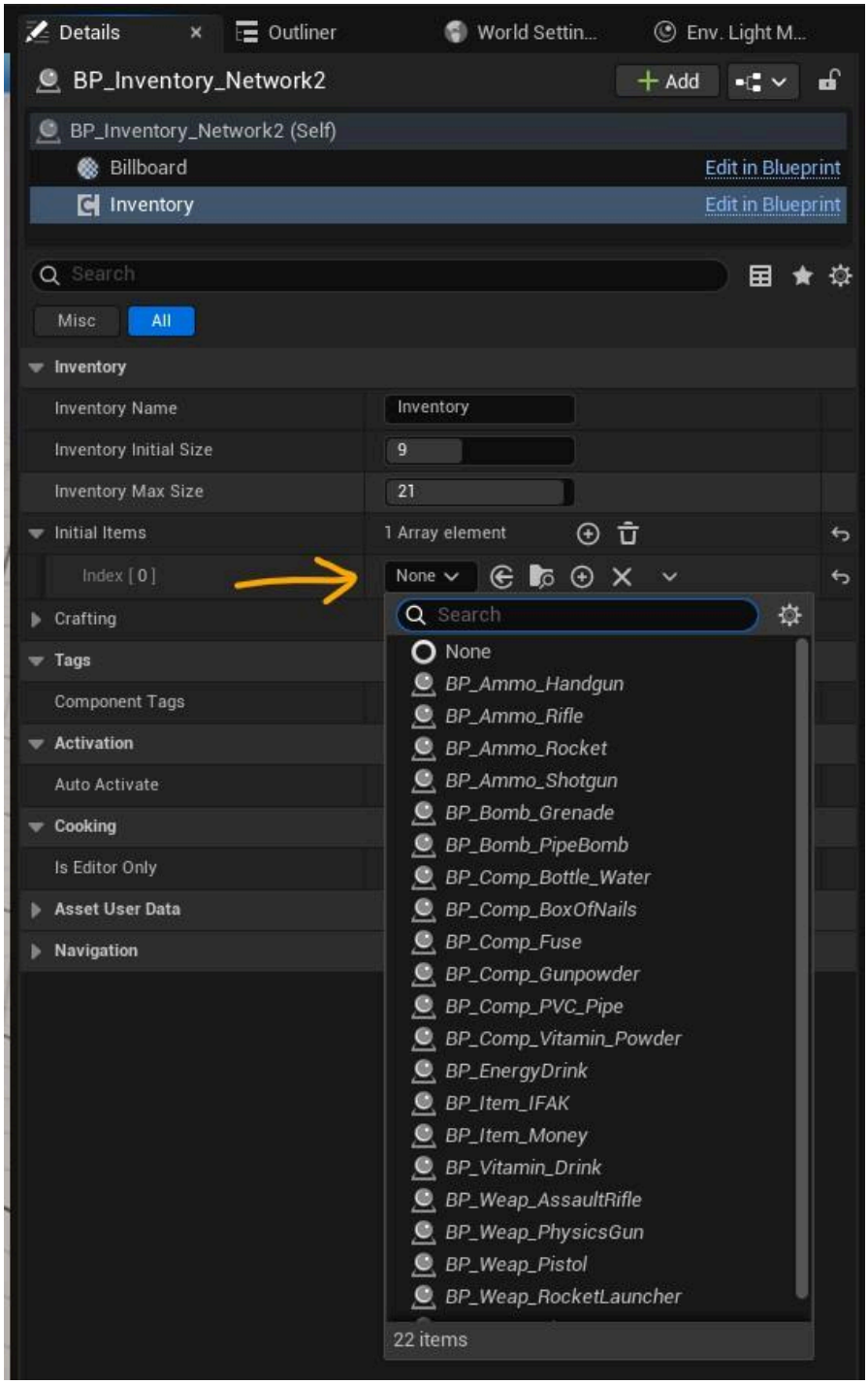


By doing this, both chests are now connected to the same inventory network, and can be accessed by one or the other.

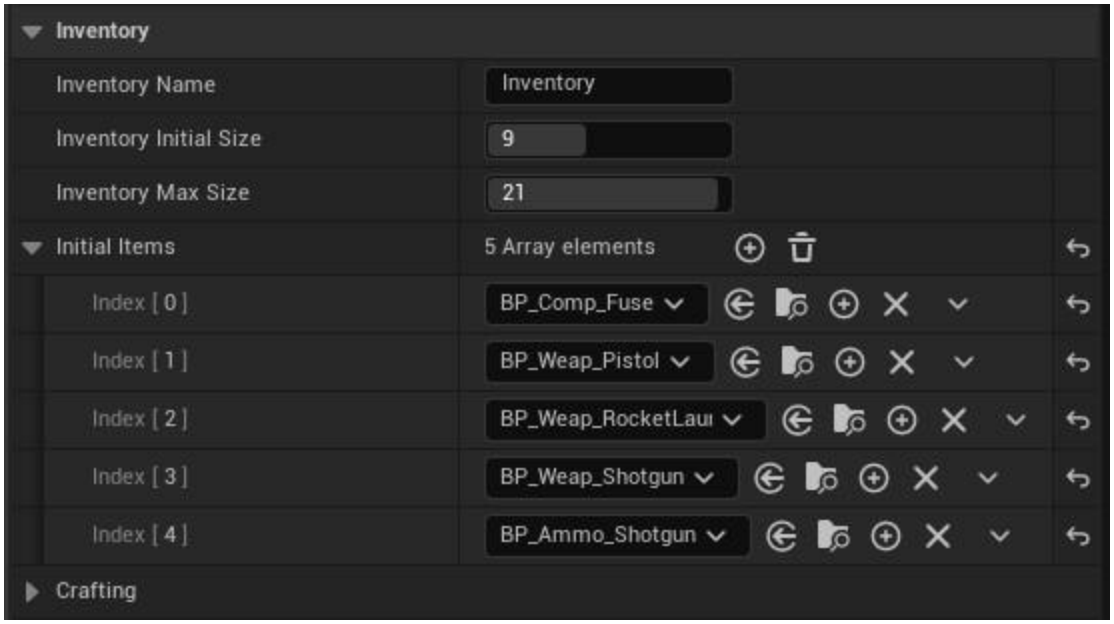
5. To add Items to the inventory, Select the Inventory Network, go to the Details tab, and click on **Inventory** Component
6. Going further down, look for the **Initial Items** array.



7. Click on plus sign  and choose the desired item from the drop down menu

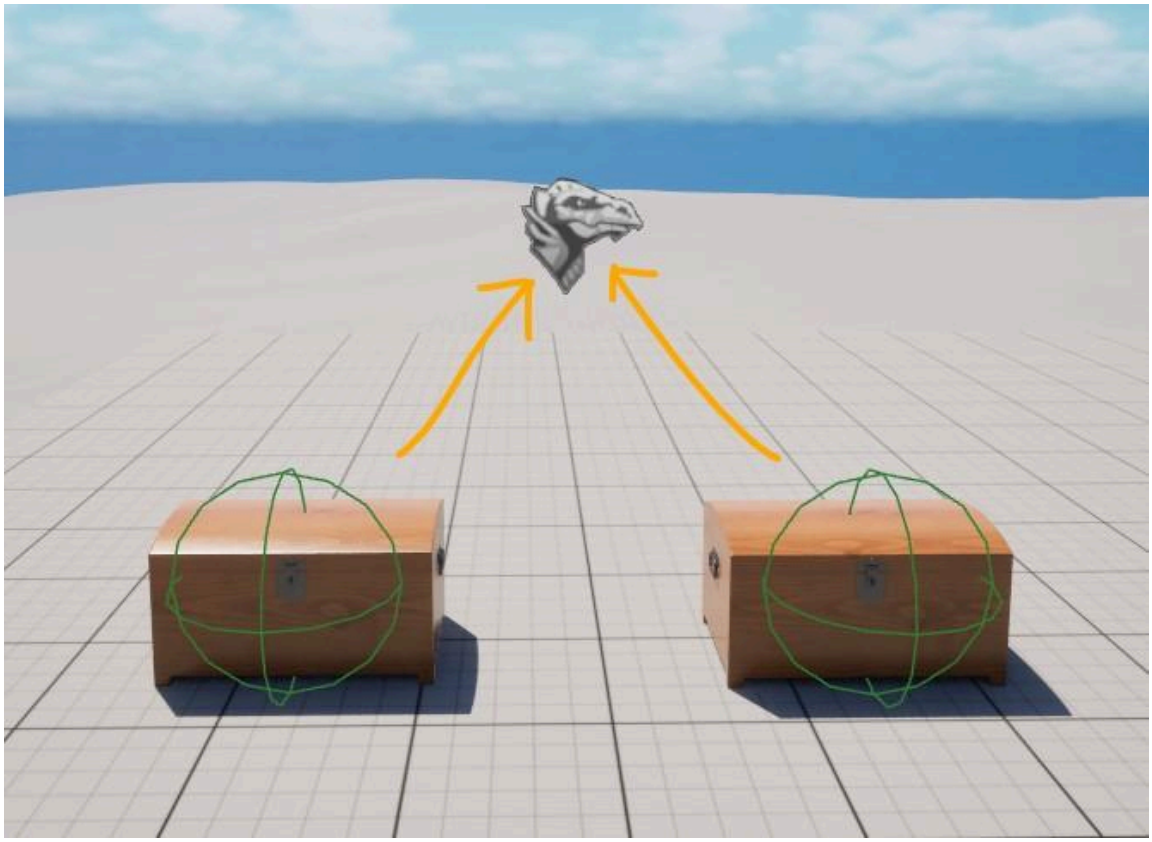


add as many items as you want



8. Press play, and open the chest.

You will notice that the items you added will appear there, and you can just drag and drop them into the player's inventory.



See, both chests are now connected to the same inventory server, this means that you can place each chest in different places on the map and they will still access the same inventory.

Changing Sentry Gun Rotation Angle

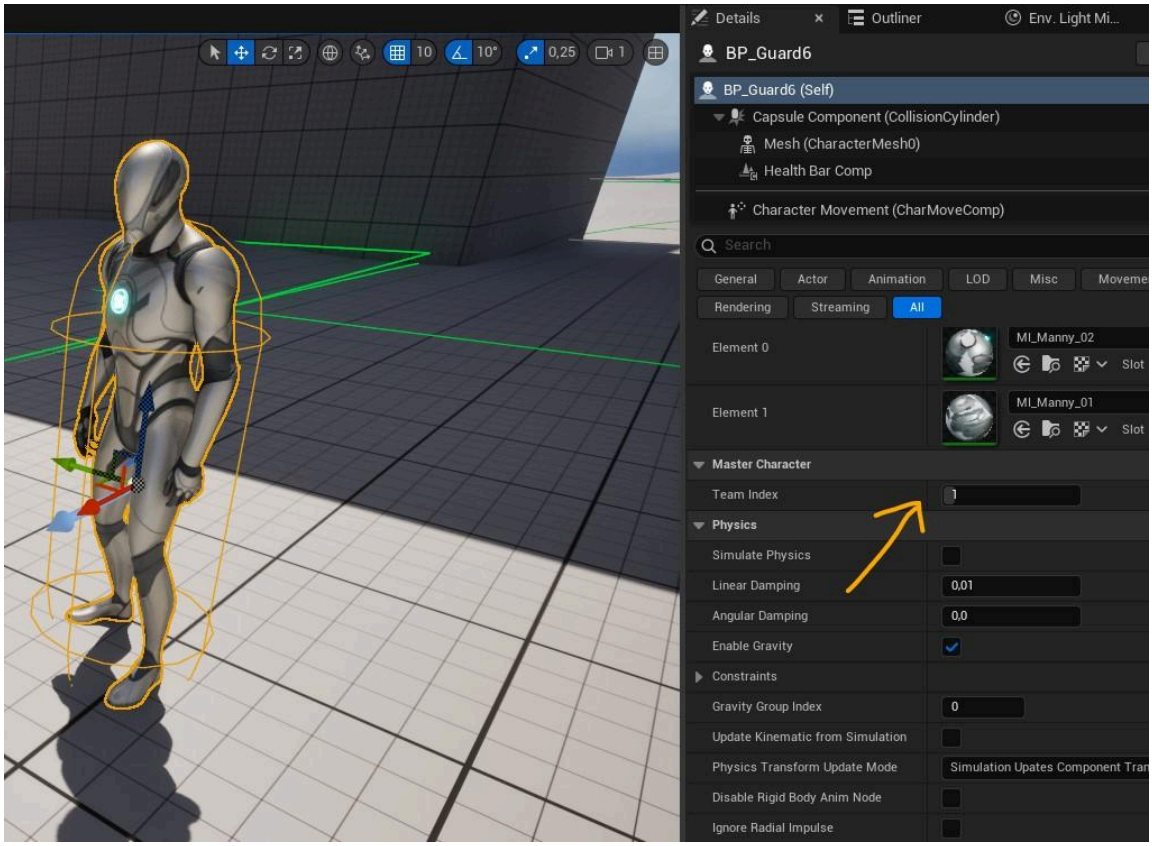
See, the rotation of the turret gun is limited by the angle of sight perception of the AI Controller. If you want to increase the turret's rotation angle, you just need to:

1. Open the **BP_SentryGunController** blueprint.
2. Navigate to the components tab, and select **AI Perception Component**
3. With the **AI Perception Component** still selected, go to the details tab and look for **PeripheralVisionHalfAngleDegrees**. Increase it to 180, and it’s done.
   This will enhance the turret's sight perception, allowing it to detect enemies in all directions.

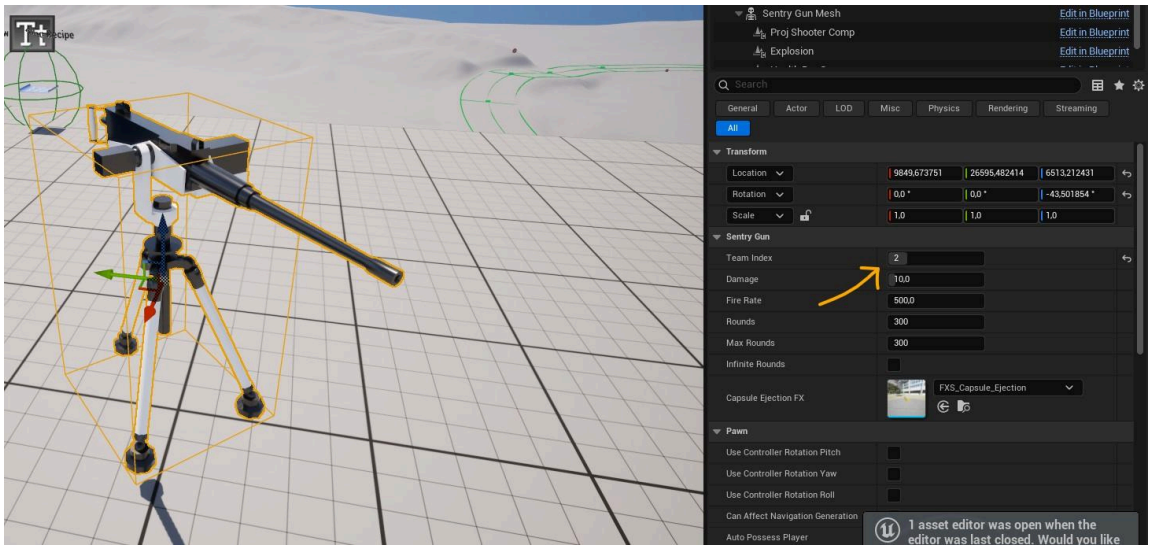
Changing NPC Team Index

1. Select the NPC whose team index you want to change

2. Go to the details tab and look for **Team Index** value.



Each NPC has a team index including the player, if each one has a different team index number they will start killing each other.



The same goes for the sentry guns.

Changing NPC initial weapon

Method 1

1. In the Content Browser, find and open the **BP_Guard** Blueprint.
2. In the Components tab, select the **Character Inventory** component.
3. With Character Inventory still selected, go to the Details tab and find the **Initial Items** array.
4. Click the plus icon and select your desired weapon from the dropdown menu.
5. Compile the Blueprint, and you're done!

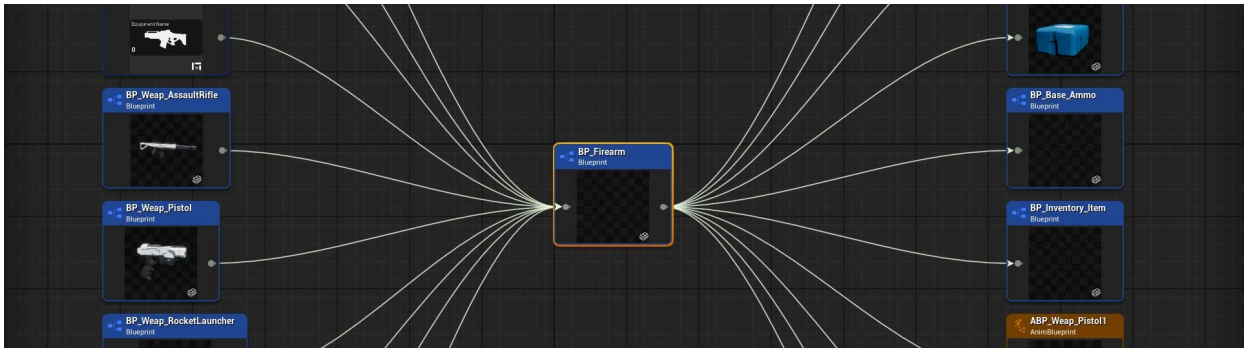
Method 2

1. Find and select the NPC you want to edit in the level.
2. With the NPC selected, go to the Details tab and locate the Character Inventory component.
3. Look for the Initial Items array.
4. Click the plus icon and select your desired weapon from the dropdown menu.

Removing NPC Health Bar

1. Locate and open the **BP_Guard** blueprint on the content browser.
2. Go to the component tab, and simply delete the **Health Bar Comp**.
3. Hit Compile, and it’s done.

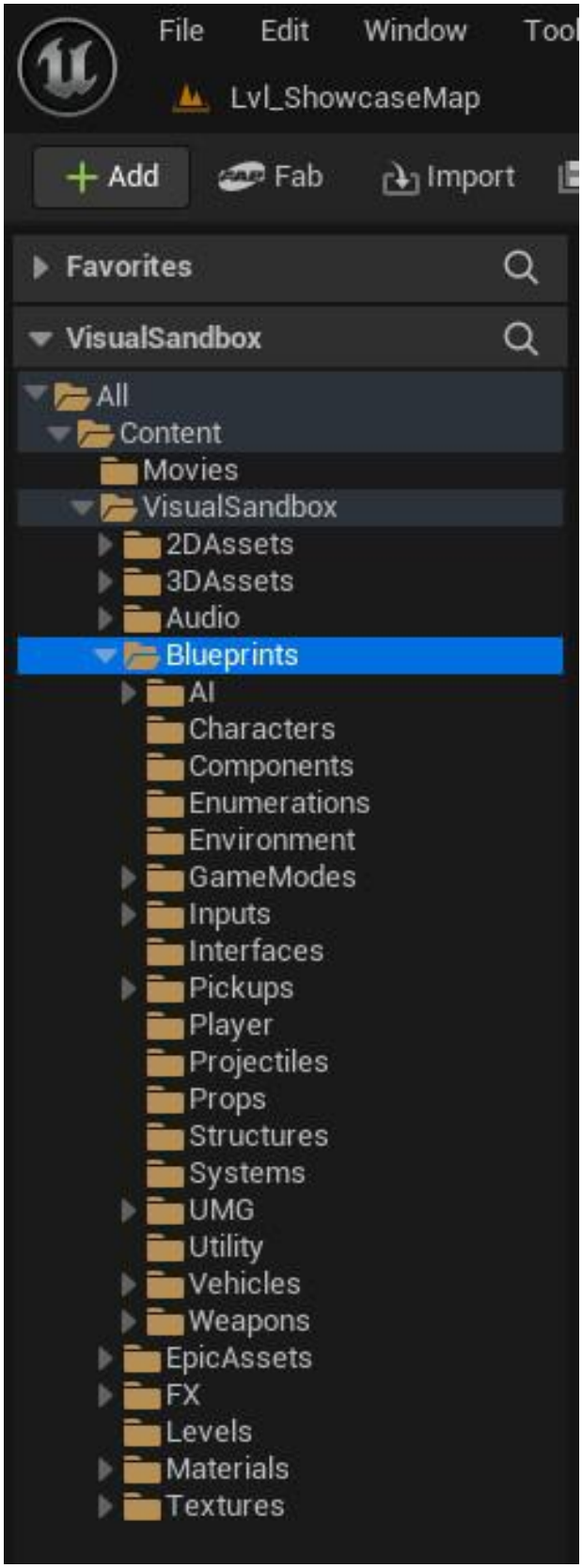
Blueprint Structure



Blueprint Structure

Number Of Blueprints: 98

The Visual Sandbox blueprints are organized by category within the **Blueprints** folder.



Furthermore, Enumerations, Structures, Input Actions and User Widget, which have a direct link to blueprint, are inside the same folder.

Naming convention prefix

Prefix	Asset Type	Example
BP	Blueprint	BP_Firearm
WBP	User Widget	WBP_HealthGauge
E	Enumerations	EFiringMode
F	Structures	FPlayerResources
IA	Input Actions	IA_Shoot

Blueprints Class Hierarchy

- **Actor (41 subclasses)**
    - BP_Item
        - BP_Item_Skeletal
            - BP_Firearm
                - BP_Weap_AssaultRifle
                - BP_Weap_PhysicsGun
                - BP_Weap_Pistol
                - BP_Weap_RocketLauncher
                - BP_Weap_Shotgun
        - BP_Throwable_Bomb
            - BP_Bomb_Grenade
            - BP_Bomb_PipeBomb
        - BP_Item_IFAK
            - BP_Vitamin_Drink
            - BP_Energy_Drink
        - BP_Ammo
            - BP_Ammo_Handgun
            - BP_Ammo_Rifle
            - BP_Ammo_Rocket
            - BP_Ammo_Shotgun
        - BP_Item_Money
        - BP_Comp_Bottle_Water
        - BP_Comp_BoxOfNails
        - BP_Comp_Fuse
        - BP_Comp_Gunpowder

        - BP_Comp_PVC_Pipe
        - BP_Comp_Vitamin_Powder
    - BP_Pickup_Base
        - BP_Health_Spawner
        - BP_InventoryExpansion_Spawner
        - BP_Crafting_Recipe_Spawner
    - BP_Portal
    - BP_Projectile
        - BP_Proj_Bullet
        - BP_Proj_Rocket
        - BP_Proj_Missile
    - BP_Storage
        - BP_Storage_Case
    - BP_Prop_Eletronic_Dice
    - BP_Inventory_Network
    - BP_Button
    - BP_Ceiling_Lamps
- **AICotnroller (3 subclasses)**
    - BP_BotController
        - BP_ShooterController
        - BP_SentryGunController
- **BTDecorator Blueprint Base (3 subclasses)**
    - BP_Shooter_IsEquipped
    - BP_CheckAmmo
    - BP_IsAlive
- **BTService Blueprint Base (1 subclasses)**
    - BP_FindRandomLocation
- **BTTask Blueprint Base (13 subclasses)**
    - BP_CheckTargetOnSight
    - BP_LookAt
    - BP_ClearBlackboardValues
    - BP_Shooter_Task
        - BP_Shooter_Task_Aim
        - BP_Shooter_Task_EquipWeapon
        - BP_Shooter_Task_Overwatch
        - BP_Shooter_Task_ReloadWeapon
        - BP_Shooter_Task_Shoot
    - BP_SentryGunTask
        - BP_SentryGun_CheckTargetOnSight
        - BP_SentryGun_LockOn

        - BP_SentryGun_Shoot
- **Actor Component (2 subclasses)**
    - AC_HealthComponent
    - AC_InventoryComponent
- **Scene Component (3 subclasses)**
    - SC_ActorWidgetComponent
    - SC_HealthBarComponent
    - SC_ExplosionComponent Removed
    - SC_ProjectileShooterComponent
- **Character (4 subclasses)**
    - BP_MasterCharacter
        - BP_ShooterChatacter
            - BP_Guard
            - BP_ThirdPerson_Pawn
- **TargetPoint (2 subclasses)**
    - BP_RespawnPoint
    - BP_VehicleSpawner
- **StaticMeshActor (3 subclasses)**
    - BP_Prop_ExplosiveActor
        - BP_Prop_ExplosiveBarrel
        - BP_Prop_Landmine_Trap
- **Pawn (2 subclasses)**
    - BP_SentryGun
    - BP_Vehicle
- **WheeledVehiclePawn (Chaos Vehicles Plugin) (2 subclasses)**
    - BP_Vehicle_Car
        - BP_Vehicle_Car_Custom
- **ChaosVehicleWheel (2 subclasses)**
    - SportsCar_WheelsFront
    - SportsCar_WheelsRear
- **BlueprintInterface (6 subclasses)**
    - BPI_Bot
    - BPI_Game
    - BPI_HUD
    - BPI_Physics
    - BPI_Useful
    - BPI_Vehicle
- **Macro Library (1 subclasses)**
    - BP_UsefulMacros
- **Function Library (2 subclasses)**

    - BP_UsefulFunctions
    - BP_Physics_Calculation
- **GameMode (2 subclasses)**
    - BP_VS_GameMode
        - BP_ThirdPerson_GameMode
- **PlayerController (2 subclasses)**
    - BP_VS_PlayerController
        - BP_ThirdPerson_PC
- **HUD (2 subclasses)**
    - BP_VS_HUD
        - BP_ThirdPerson_HUD
- **PlayerCameraManager (1 subclasses)**
    - BP_VS_Camera_Manager
- **DragDropOperation (1 subclasses)**
    - BP_Inventory_DragDropOperation

Third Person Game Mode

Third Person Shooter Game Mode

By default, Visual Sandbox (VS) includes a third-person shooter game mode that features a character capable of running, shooting, and driving from an over-the-shoulder perspective.

This game mode was essentially built by using and combining different blueprints from the Visual Sandbox Blueprints library.

Third Person Shooter Classes:

Game Mode Class	BP_ThirdPerson_GameMode
Player Controller Class	BP ThirdPerson PC
HUD Class	BP_ThirdPerson_HUD

Default Pawn Class	BP Third Person Pawn

These are the only custom classes created exclusively for the Third Person game mode. All other features like Shooter Character, Inventory System, Guns and NPC are part of the VSB library

BP_ThirdPerson_GameMode
Parent Class: BP_VS_GameMode
Path: Content/VisualSandbox/Blueprints/GameModes/ThirdPerson

It all starts with this simple subclass that defines all the TPS game classes, such as player controller, HUD and player pawn. It does not contain any custom logic, functions, or events. It just defines which classes will be loaded in TPS mode at the beginning of the game.
This game mode is already set as the Default Game Mode in the Project Settings, meaning that any newly created map will automatically load this mode in the World Settings. Pressing Play will launch the third-person mode by default.

BP_ThirdPerson_PlayerController
Parent Class: BP_VS_PC
Path: Content/VisualSandbox/Blueprints/GameModes/ThirdPerson

This is the Controller class of the TPS game that will control the player's character, configuring the Input Mapping Context to receive player input actions. It handles the camera control rotation via IA_Look input action mapped to mouse movement.
It also includes inputs for:

- Activate / Deactivate Bullet Time
- Switch between Default Look Sensitivity and Aiming Sensitivity
- Requesting respawn upon death
- Restarting the game
- Exiting the game

On the Details tab, you can adjust look sensitivity and invert the mouse Y-axis if needed.

BP_ThirdPerson_HUD
Parent Class: BP_VS_HUD
Path: Content/VisualSandbox/Blueprints/GameModes/ThirdPerson

This class is responsible for creating and projecting the all HUD widget elements onto the screen.
At the beginning of the game, this class creates the HUD widget, Inventory Widget individually and receives inputs to display on the player's screen.

BP Third Person Pawn
Parent Class: BP_ShooterCharacter
Path: Content/VisualSandbox/Blueprints/GameModes/ThirdPerson

This is the player-controlled character, inheriting core systems from BP_ShooterCharacter, such as:

- Movement
- Weapon and equipment management
- Inventory
- Health system

Additionally, it implements a third-person over-the-shoulder camera using a Spring Arm attached to the character.
This class is ideal for configuring camera behavior, player stats, and initial inventory items.

As you can see, the classes used in the Third Person Gamemode are simple and purpose-driven. Each one inherits functionality from its parent class and organizes the necessary elements to deliver a cohesive third-person gameplay experience.
This modular approach makes the structure easy to understand and highly customizable, allowing users to tweak or replace specific parts without disrupting the overall system.

Changelog

Changelog

Version: 1.5.8 **Helicopter & framework struct improvement**
Release date: coming soon
List of change:

- Flying vehicles
- Save and load system
- Checkpoint system
- Main Menu
- Door system
- UI change and improvement
- Blueprint optimization
- Framework struct change

—----------

Version: 1.4.7 **Item Storage System Implemented**
Release date: Aug 7, 2025
List of change:

- **Item Storage System** - I implemented a storage system similar to Resident Evil 2. And there are two types of storage, local and global. Local storage stores items only in a specific chest, while global storage shares the same items in different chests.
- **Framework Structure Changes -** major changes to the framework structure. Some blueprints have been changed and improved, and others were deprecated.
- **New Functions** - I implemented two new functions within the Useful Function Library: Spawn Explosion and Hitscan. These two functions can be called by any blueprint that is a subclass of Actor.

- **Pause Menu Screen added**
- **Inventory screen UI Redesign**
- **New Respawning System**

—---

Version: 1.3.5 - **Crafting System implemented**
Release date: Jun 16, 2025
List of change:

- **The crafting system was implemented** within the **AC_InventoryComponent**, and a new user widget class for the Crafting Panel was added to the **WBP_Inventory_Screen** to display the crafting menu widget.
- **New pickable items added**
- **Pickable crafting recipes added:** now there are a blueprint class called **BP_Crafting_Recipes_Spawner**, which generates the crafting recipe for an item defined in the blueprint details tab
- **Blueprint Changes:** previously the **BP_ShooterCharacter** could pick up items by overlapping them, now it is necessary to press the interaction button to pick up items.
- Items icons was update
- Ammo pack mesh changed

—---

Version: 1.2.4 - **Drivable Car added**
Release date: May 28, 2025
List of change:

- **Drivable car added:** now the Shooter Character has gained the ability to drive car around and run over NPCs
- **New Showcase map added**
- **Blueprint fixes and improvements**

—---

Version: 1.1.2 - **Physics Gun added**
Release date: May 9, 2025

List of change:

- **Physics Gun Added:** I implemented a new gun similar to the Gravity Gun from Half Life 2, which I called Physics Gun to avoid legal problems. With it, you can play around by grabbing any static mesh or skeletal mesh that is simulating physics and throwing it in any direction. The result was a lot of fun and the Physics Gun blueprint is fully accessible for you to explore and see how it was made.

—---

Version: 1.0.0 - **Asset Released**
Asset release date: May 6, 2025

Developer Note

Over 7 years working with Unreal Engine, I have noticed some limitations and inconsistencies in this engine, which end up requiring laborious and bureaucratic solutions. In this section, I have documented some of the challenges I have encountered when using it.

- **AI Control Rotation:** By default, Unreal Engine doesn't pitch view to the AI Control Rotation, so the AI only looks along the horizontal axis and ignores the vertical axis. This can be an issue for projects where the AI needs to look up or down. There's actually a piece of code in **AIController.cpp** file that tells the AI to ignore pitch rotation unless a focus actor is set.

AIController.cpp

UE5 — AAIController

```cpp
void AAIController::UpdateControlRotation(float DeltaTime, bool bUpdatePawn)
{
    APawn* const MyPawn = GetPawn();
    if (MyPawn)
    {
        FRotator NewControlRotation = GetControlRotation();

        // Look toward focus
        const FVector FocalPoint = GetFocalPoint();
        if (FAISystem::IsValidLocation(FocalPoint))
        {
            NewControlRotation = (FocalPoint - MyPawn->GetPawnViewLocation()).Rotation();
        }
        else if (bSetControlRotationFromPawnOrientation)
        {
            NewControlRotation = MyPawn->GetActorRotation();
        }

        // Don't pitch view unless looking at another pawn
        if (NewControlRotation.Pitch != 0 && Cast<APawn>(GetFocusActor()) == nullptr)
        {
            NewControlRotation.Pitch = 0.f;
        }

        SetControlRotation(NewControlRotation);

        if (bUpdatePawn)
        {
            const FRotator CurrentPawnRotation = MyPawn->GetActorRotation();

            if (CurrentPawnRotation.Equals(NewControlRotation, 1e-3f) == false)
            {
                MyPawn->FaceRotation(NewControlRotation, DeltaTime);
            }
        }
    }
}
```

See, this is the part of the AI Controller code that clearly says to ignore the pitch view for Control Rotation. I haven't yet found a definitive solution to solve this issue using Blueprint, but with C++ you can create a new C++ class picking **AIController.h** as parent class, and override the **UpdateControlRotation()** function and copy the code (from the Image above) except for the part that blocks the pitch view and paste it into the new AI Controller Class.

- **Physics simulation:** Unreal Engine frequently shows inconsistencies in physics simulation, especially when there’s interaction with the player. Sometimes, physics simulated objects go flying with just the slightest touch from another object, the physics system may flicking, or objects may pass through walls, and sometimes the character simply does not respond to the physics collision, some features that are difficult to emulate with unreal engine are realistic impact force, and this is very frustrating for new projects involving realistic physics simulation. Additionally, the engine struggles to accurately replicate physics on skeletal meshes, requiring specific techniques to better handle these cases. Despite these limitations, Unreal Engine’s physics simulation is

quite impressive and can be effectively applied in games and simulations, as long as you're mindful of its constraints.

- **Run Over:** When you place a character in the level and try to run them over with a vehicle, you'll notice that the car stops abruptly upon impact, as if it had hit a concrete wall, meanwhile, the character remains unaffected. This happens because the vehicle is colliding with the character's collision capsule, which does not simulate physics at all and cannot be pushed or affected by external forces. There are a few ways to implement run-over in Unreal Engine. One common method is to set the character's capsule to overlap with vehicles and apply full damage as soon as the vehicle passes through the capsule. However, this approach is inefficient; even a light touch from the vehicle would instantly kill the character, as if it had been hit at full speed. Even if you implement logic to scale damage based on the vehicle's speed, there's another problem—since the collision is set to overlap, the vehicle will pass through the character mesh without any physical interaction, making it feel unrealistic and visually incorrect. One method I've implemented is using a capsule trace around the character to detect nearby vehicles. When a vehicle is detected, the system checks its speed. If the vehicle is moving at high speed, the character's collision capsule is temporarily set to overlap with vehicles. If the vehicle then overlaps the capsule, full damage is applied to the character, resulting in instant death. However, if the detected vehicle is moving at low speed, the collision capsule remains in block mode. This way, if the vehicle bumps into the character gently, it won't cause any damage, as the overlap doesn't occur and the collision remains physically reactive without triggering the run-over logic. However, this is a provisional method of running over characters and I am studying more efficient and definitive techniques for running over.

- **Terrain texture tiling repetition:** When you apply a grass texture to a huge landscape, you quickly notice the repeating pattern, which makes the terrain looks ugly and unsuitable for games where the terrain is seen from sky (RTS games or flying simulators). There are several common approaches to handle texture tiling issues, such as using texture mosaics, macro noise, color and size variation, or texture blending. But all of these techniques just mask the problem without truly solving it and can be unnecessarily complicated. I believe there's still a much simpler solution out there waiting to be discovered, one that could be extremely valuable.

- **UI Icon Rendering:** In Unreal Engine, HUD icons tend to get jagged and lose quality when their size is changed. A practical solution is to keep the icons at their original size, even when the screen resolution changes

- **Fab**: As a producer and seller, I've encountered numerous issues with Fab.com Epic Games' new asset platform that replaced the Unreal Marketplace in October 2024.
  The original Unreal Marketplace was entirely focused on Unreal Engine. Customers and sellers had full confidence that everything listed there was designed specifically for Unreal. It was a reliable, specialized platform.
  In 2023, however, Epic Games announced plans to unify several of its content stores — including the Unreal Engine Marketplace, Sketchfab Store, Quixel Megascans, and ArtStation Marketplace into a single platform called Fab.
  What initially sounded like a promising idea turned out to be a complete disaster in practice.
  The transition to Fab.com was rushed, poorly communicated, and the platform launched prematurely with missing features and dysfunctional with a confusing, unintuitive, and disorganized user interface. It had a clean visual design, but functionally it was inefficient.
  Fab ended up mixing assets from all these different platforms into a single space. Previously, customers browsing the Unreal Marketplace knew they were purchasing content for Unreal Engine. Now, on Fab, everything is jumbled together, leaving customers confused and uncertain whether the assets are for Unreal Engine, Unity, Blender, or other platforms.
  And to complete this big mess, Fab.com was flooded with AI-generated assets. All content created with the help of AI is supposed to be labeled with the CreatedWithAI tag to indicate its origin. However, despite this requirement, moderation is largely automated and ineffective. Sellers can mislabel their own items or choose not to label them at all and in most cases, these assets are neither removed nor manually reviewed.
  The result has been a wave of frustration and loss of trust in the platform, from both customers and sellers. Some producers have reported sharp drops in sales, which has discouraged the development of new assets.
  In summary, the launch of Fab was rushed, premature, dysfunctional, and quickly flooded with AI generated content. This led to customer dissatisfaction and drastic drops in sales since the platform's release. In the end, the whole situation left many customers and sellers feeling orphaned, unsure of where to buy or sell. I'm not here to criticize Fab, I'm simply sharing what happened based on public reports and personal experience. In fact, I hope Fab recovers from this big mess. Meanwhile, many sellers and customers are moving to alternative platforms, such as Gumroad, which offers more control, features and freedom.