Inventory System / Equipping Weapons and Armor

04/17/2020

Documentation shows how to implement an inventory system (not the UI part) and equipping weapons and armor to your character. This documentation is a work in progress and might be out of date depending on any bugs/issues we run into. However, this documentation will be a guide and help grasp and understand more of how we can improve this system.

To begin, we need to know how to equip a weapon and armor. First, the weapon itself needs to be spawned into the world and attach the item to the player’s bone. Second, the armor is a skeletal mesh that will replace the body part of the player.

There are three ***enumeration*** that we need create and understand:

* E\_ItemCategory : This will determine what item is in which category.
  + Consumable
  + Primary Weapon
  + Secondary Weapon
  + Armor
  + Material
  + Furniture
  + –None—
* E\_ArmorType : This enum will specify which Armor piece it is
  + Head
  + Chest
  + Gloves
  + Pants
* E\_SecondaryWeaponType : This enum will determine which type of secondary weapon it is in order to correctly attach the item to certain bones.
  + Shield
  + Dagger
  + Arrow

Now that we have an understanding of the **enumerations,** we can go and understand the ***structures***.

* S\_Item : This is our main structure that will be storing any main details and anything that is important.
  + Name
  + Description
  + Icon
  + Quantity
  + MaxQuantity
  + Category (E\_ItemCategory) : this will determine how items will react
  + Granted Ability (Gameplay Ability) : used for GAS in order to activate certain abilities.
  + bIsConsumable
  + Blueprint: used to spawn weapons and retrieve the skeleton mesh for the armor.
* S\_PlayerEquipment: This will be our current equipment slots, anything that is equipped in our body will be stored in here for reference.
  + Primary Weapon Instance (BP Weapon Base)
  + Secondary Weapon Instance (BP Weapon Base)
  + Head Class (BP Amor Base)
  + Chest Class (BP Armor Base)
  + Arms Class (BP Armor Base)
  + Legs Class (BP Armor Base)
* S\_Inventory: This stores all the items being picked up and the max capacity of the inventory.
  + Inventory (Array of S\_Item)
  + MaxCapacity

After understanding the enumeration and structures, we are able to create components and blueprints that connects with each other and spawn weapons / and armor.

We will look at the ***InventoryComponent,*** as this is simple and straight forward for now. It only contains the S\_Inventory.

Regards to attach weapons and armor we will only focus on **BP\_ItemBase, BP\_WeaponBase, and BP\_ArmorBase.**

# BP\_ItemBase

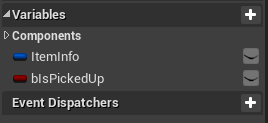
BP\_Itembase will contain ***ItemInfo*** *(S\_item)* and a Boolean variable ***bIsPickedUp*** which determines if it has been picked up on the floor yet. It will also have a SphereCollision to determine the pickup radius

Components:

* PickupRadius (Sphere collision): To determine pickup radius for player.

Variables:

* bIsPickedUp (Bool): To determine if the item has been picked up the ground yet.

A screenshot of a cell phone

Description automatically generated

BP\_ItemBase also implements an Interface(I\_Pickup) to add items to the inventory.

A screen shot of a computer

Description automatically generated

# BP\_WeaponBase

The BP\_WeaponBase is a child of BP\_ItemBase with additional variables/components to accommodate some parameter.

Components:

* WeaponMesh (Static Mesh): To display the weapon
* HitBoxCollision (Cube Collision): To determine if the weapon is hitting any enemies.

Variables:

* A screenshot of a cell phone

  Description automatically generatedSecondaryType (E\_SecondaryWeapon): To determine where to place mesh if it is a secondary weapon.

# BP\_ArmorBase

BP\_ArmorBase is a child of BP\_itemBase. It hold addition variables and components.

Components:

* Mesh(Static Mesh): Just to display onto the world.

Variables:

* ArmorType(E\_ArmorType): To determine the specific armor type being equipped.
* SkeletalMesh(Skeletal Mesh): Used to replace the skeletal mesh of the player
* Material\_01(Material): Used to properly materialized the skeletal mesh being used.
* Material\_02(Material): Used to properly materialized the skeletal mesh being used.A screenshot of a cell phone

  Description automatically generated

# BP\_BaseCharacter

Here we will start implementing adding the weapon / armor onto the player. For our variables we have a ***Equipment(S\_PlayerEquipment)***. We then will create 4 functions that will help us equip our player:

* EquipPrimaryWeapon()
* EquipSecondaryWeapon()
* EquipArmor()
* Equip() : Universal, used in all cases and will decide to use one of the three functions above.

## Equip Primary Weapon

1. Our EquipPrimaryWeapon function will take in an input of ***Item(S\_Item)****.* This will give us all the information such as the name, category, ability, and blueprints needed to spawn our armor/weapon.
2. We then use a ***Switch on E\_ItemCategory*** to determine if the piece we are equipping is a Primary Weapon, Secondary Weapon or Armor. If is not, a primary weapon it will lead to the correction equipment function.
3. After checking that it is certainly a Primary Weapon, we can cast the ***Blueprint*** that is in the **Item Info** given as input to ***BP\_WeaponBase***so we can access more specific data.
4. Before we spawn the new Primary Weapon actor we have to delete any old primary weapon that is being equipped in our ***Equipment*** *variable* in our BP\_BaseCharacter, if there is any.
5. After destroying any old primary weapon we then spawn the Primary Weapon. We then set the new Primary Weapon Instance to our ***Equipment*** variable so we can track it.
6. We then set the PrimaryWeapon’s bool variable ***bIsPickedUp*****to true** so we prevent picking it up while we are wearing it already.
7. We then set the Primary Weapon Instance onto the Player’s Right Hand Socket (Hand\_RSocket).
8. Then we set the Primary Weapon’s owner to the Character so we can backtrack and destroy both of them if needed.
9. Lastly we use a function implemented in C++ called ***Equip Primary Ability***for our gameplay ability, if any.**A screenshot of a computer

   Description automatically generated**

## Equip Secondary Weapon

This is very similar to the Primary Weapon.

1. Our EquipSecondaryWeapon function will take in an input of ***Item(S\_Item)***. This will give us all the information such as the name, category, ability, and blueprints needed to spawn our armor/weapon.
2. We then use ***Switch on E\_ItemCategory*** to determine if the piece we are equipping is a Secondary Weapon. If it is not a secondary weapon it will lead the item information to the correct equipment function.
3. After checking that it is certainly a Secondary Weapon, we can cast the ***Blueprint***that is in the ***Item Info*** given as the input to ***BP\_WeaponBase*** so we can access more specific data.
4. Before we spawn the new Secondary Weapon actor we have to delete any old secondary weapon that is being equipped in our ***Equipment*** variable in our BP\_BaseCharacter, if there is any.
5. After destroying the previously secondary weapon, we then spawn the Secondary Weapon. Then set the new Secondary Weapon Instance to our ***Equipment*** variable so we can track it.
6. We then set the Secondary Weapon’s bool variable ***bIsPickedUp*** to True so we prevent picking it up while we are wearing it already.
7. We then create ***Switch on E\_SecondaryWeaponType*** to compare with the variable ***SecondaryType(E\_SecondaryWeaponType)***in our Secondary Weapon Instance (This is why we cast the Blueprint as a BP\_WeaponBase, so we can access the SecondaryType variable).
8. After determining which type of secondary weapon it is, we can determine which Player’s socket we can attach it. This is important because the shield and dagger mesh requires different socket placement to have it place at the correct location.
9. We then set the Secondary Weapon’s owner to the player so we can backtrack and destroy both of them if needed.
10. lastly, we use a function implemented in C++ called ***Equip Secondary Ability*** for our gameplay ability, if any.

## Equip Armor

This function will equip either helmet, chest, pants, or arms.

1. Our EquipArmor function will take in an input of ***Item(S\_Item)***. This will give us all the information such as the name, category, ability, and blueprints needed to spawn our armor/weapon.
2. We then use ***Switch on E\_ItemCategory*** to determine if the piece we are equipping is a Armor Type. If it is not a Armor Type it will lead the item information to the correct equipment function.
3. After checking that it is certainly a Secondary Weapon, we can cast the ***Blueprint***that is in the ***Item Info*** given as the input to ***BP\_ArmorBase*** so we can access more specific data.
4. After casting the **Blueprint as an BP\_ArmorBase** we use the node ***Get Class Default*** to access the ArmorType, Skeletal mesh, and Materials needed to change the players armor.
5. We then create ***Switch on E\_ArmorType*** to compare the ArmorType in the class default to determine what armor part is being equipped.
6. All armor parts will have the same exact nodes except replacing the skeletal mesh of the component of the player. Example, if it is a helmet being equipped we replace the HeadMesh (Skeletal Mesh) and if a pants are being equipped we replace the LegMesh(Skeletal Mesh) instead. ***We will use Gloves as an example.***
7. We will then set the Player’s ArmMesh(Skeletal Mesh) to the new Skeletal Mesh.
8. We will also set the Player’s ArmMesh Materials to the the new Materials given.
9. After setting up the new Skeletal Mesh and Materials, we then set the ***Blueprint given in the Item Input*** to the ***Equipment*** variable so we can track it.
10. Last, we use the function implemented in C++ called ***Equip Arms Slot*** for our gameplay ability, if any.

# PC\_Player

Lastly but the most importantly is the PC\_Player which will be managing the inventory, picking up the items, and equipping the items.

PC\_Player will have the InventoryComponent attached to it.

## PickUp Action

1. We have a pickup action to check if the player is overlapping any actors (based on the pickup radius).
2. If there is actors that are in range, it will check if the actors has the ***I\_PickUP*** interface so we can be sure that it is an Item that we are picking up.
3. After checking that it does have the interface and it is an Item, we cast the actor as BP\_ItemBase.
4. After, We then the Boolean variable ***bIsPickedUp.*** If it is not picked up, we can send a pickup message to the item that we are going to pick up the item.

# Equipping Items on the floor.

To test if the inventory / equipping worked. I’ve placed equipment actors onto the map so I can pickup. After picking them up I have debug tools to equip items based on the index it is in the inventory component.

A picture containing indoor, computer, black, sitting

Description automatically generated