Index

[1.º Input information 2](#_Toc62393231)

[2º Project information 2](#_Toc62393232)

[2.1 C++ classes explanation 3](#_Toc62393233)

[AndrianCharacter 3](#_Toc62393234)

[AreaClass 3](#_Toc62393235)

[ARPG\_Controller 3](#_Toc62393236)

[HealthSystemComponent 3](#_Toc62393237)

[MechanicsTestCharacter 4](#_Toc62393238)

[MechanicsTestGameMode 4](#_Toc62393239)

[ProjectileArea 4](#_Toc62393240)

[ProjectileClass 4](#_Toc62393241)

[3º Design patterns 4](#_Toc62393242)

# 1.º Input information

WASD & Left stick -> Movement

Space & Face button down -> Jump

Mouse movement & right stick -> camera movement

Left mouse & Right shoulder -> Fire ball

Q & Left button -> Area buff

E & Right button -> Acid Orb

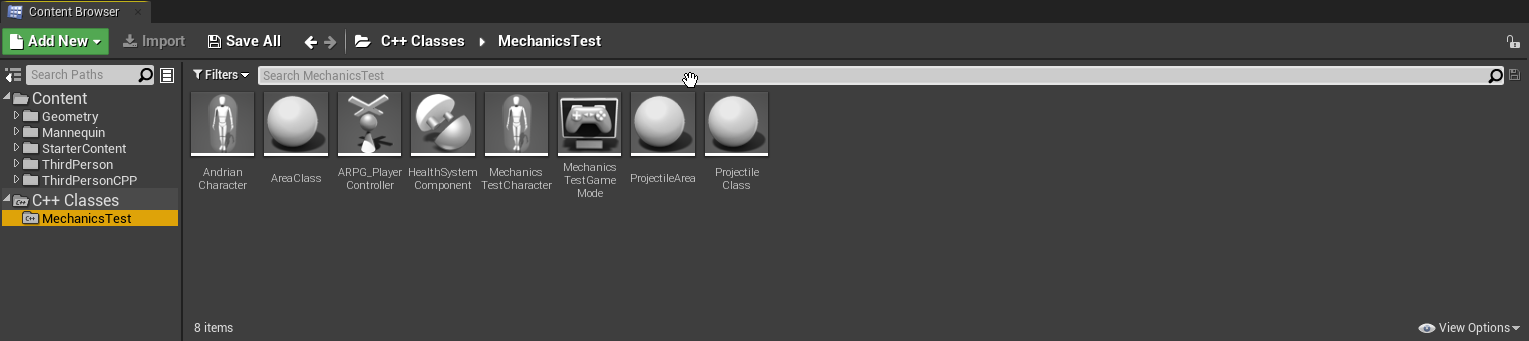
R & Top Button -> Homing buff

T & Left Shoulder -> Teleports

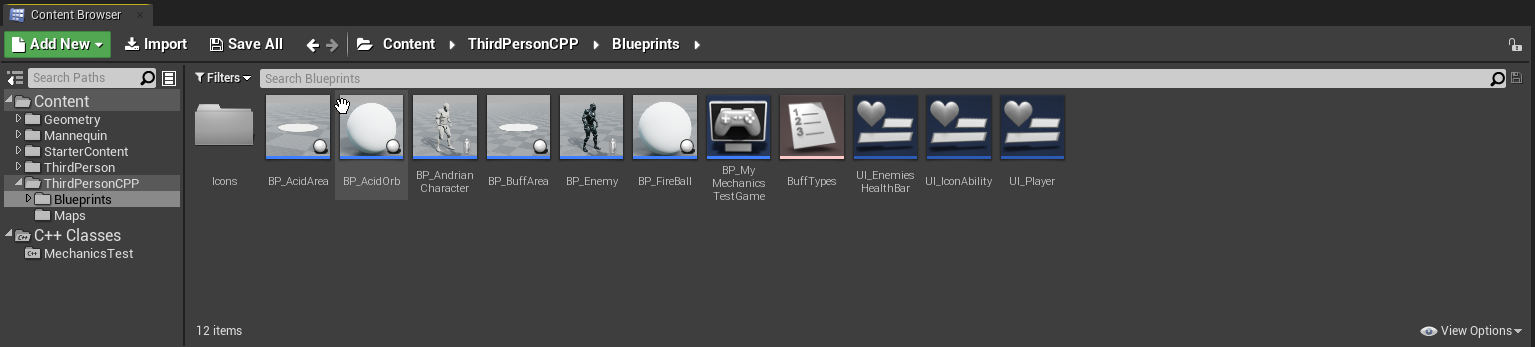
Left Ctrl & Left Trigger-> Invisibility

# 2º Project information

All C++ classes are located under next folder:



And BP classes are located here:



I put all BP on the same folder for make more easier the correction of this exercise, but this should be organize on a folder structure where BP characters will be located with UI that they are using with mesh, animations and anything else that they use.

## 2.1 C++ classes explanation

### AndrianCharacter

Here is located information from playeable character with the abilities implementent. This class inheritance from **MechanicsTestCharacter** where have those methods declared.

Also here I declarated more scene components for this character like spawn points and VFX effects.

Buff effects on character are prepared for escalation purposes if you create different types of buffs.

### AreaClass

Class created for BP purposes. Using this class I created differents areas: buff and damage areas, that other classes will spawn later.

Contain basic info like: how time will live, which value of damage or buff will apply… ect

Areas logic are implemented on BP and use an ENUM for knowing which class of area is.

### ARPG\_Controller

Here is the default controller that player will use where have all the inputs defined.

Controlled pawn it is casted to **MechanicsTestCharacter** where all inputs logic functions are implemented.

### HealthSystemComponent

Basic component that controls health for the character and they status. Added by default on **MechanicsTestCharacter**

### MechanicsTestCharacter

Basic character class where player class inherits and BP enemy too. Have basic info for character movement, character mesh, abilities without implementing like this will be for child class.

BP enemy inherits from this class and it logic are implemented there like it doesn’t have any AI implemented and it is only a dummy test enemy where you can test your abilities.

### MechanicsTestGameMode

Basic game mode that have a BP with default classes to be used.

### ProjectileArea

Projectile child class that have the capacity for spawn an area tan can be do damage to enemy.

There is a BP where this information is declared for use it on characters.

### ProjectileClass

Basic projectile where have implemented the damage that will do, projectile component information and can be a homing projectile if Homing buff is on use, like setup the necesary information that homing projectile needs.

# 3º Design patterns

* I create dummy enemies for time purposes. I think main core of this exercise it is making a mechanics test with a correct structure, so I focus on that from the beggining.
* The basic character structure starts from: **MechanicsTestCharacter** where player class and enemies inherit. This structure could be upgrade it adding some new specs like: structure with stats info from each entity on this game, separete movement logic for input and camera components on another class, so enemies will not have that information like they will move using a navmesh and going to specific points on the map or follow the player if it is on its field of view.
* I create a UI where players will see each ability coldown and an icon for each ability: 

Icons are from Diablo 2 game.

* There is a “bug” where when you cast Acid Orb, games frozes on Unreal viewport, but only occurs on first time game is executed. After that, game runs with normality. I’m investigating for which reason that is happening, but I don’t find it.
* This is a first implementation for this kind of mechanics and later will be improve changing variables, making algorithms more optimized and adding more classes.