Unreal Engine Complex Automation Test

Creating a Complex Automation Test Class

1. Open Unreal Engine.
2. Select "C++ Classes" folder in the Content Browser.
3. Right Click and add a new C++ class.
4. Select None for the parent class.
5. Select "Private".
6. Input the class name with the following format "*FeatureBeingTested*Test.cpp".

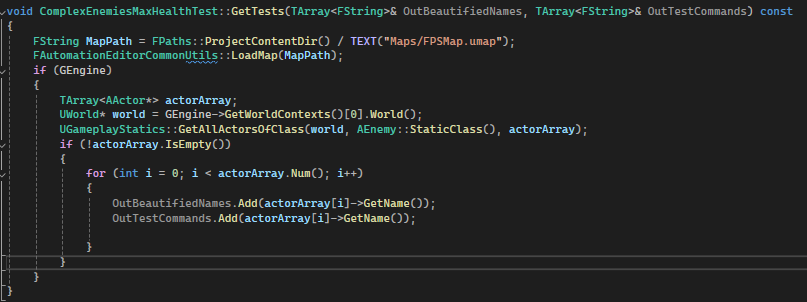
e.g. EnemiesBaseHealthTest.cpp

1. Add "Tests/" to the end of the file path.
2. Once compiling has finished close Unreal Engine.
3. Open the projects source folder and navigate to the new class.
4. Delete the header file and rename the cpp file to Return to the project folder and delete the following folders/file:
5. .vs
6. Binaries
7. Intermediate
8. Saved
9. "*ProjectName*.sln"
10. Right click the "*ProjectName*.uproject" file and select "Generate Visual Studio Project Files".
11. Once completed open the "*ProjectName*.sln" file.
12. Build the vs project by pressing F5.
13. Open the "*FeatureBeingTested*Test.cpp" file in visual studio.

Writing a Complex Automation Test

1. Delete everything that in the "*FeatureBeingTested*Test.cpp" file.
2. At the top of the file, you will need to add any includes that are required for the test to run.
3. The Class you are testing, e.g. #include “Enemy.h”
4. Start with the IMPLEMENT\_COMPLEX\_AUTOMATION\_TEST Macro.
5. The first section of the Macro is the test class name.
6. The second section is where the test will be located in the session front end.
7. The third section is for the [EAutomationTestFlags](https://docs.unrealengine.com/4.26/en-US/API/Runtime/Core/Misc/EAutomationTestFlags__Type/).



1. Write a "GetTests" with the parameters: "TArray<FString>& OutBeautifiedNames” and “TArray<FString>& OutTestCommands". 
2. Inside the "GetTests" function we gather together everything that will have test run on, for this example we will be getting all the enemies in a map, but it could also be something like get all the map names to run tests on each map.    
   For the enemies example we will load the map that has the enemies in using the “FAutomationEditorCommonUtils::LoadMap” function which takes the maps path. Next we get the world via the World Context and pass the world into the “UGameplayStatics::GetAllActorsOfClass” function to find all the actors that are of the type enemy and add them to an array. When then check the array isnt empty and then loop through the array and add the Actor’s name to both the BeautfiedNames and OutTestCommands Arrays that are passed into the GetTests function.    
      
   The BeautfiedNames Array is the name of the test that will be displayed in the session front end in this case were just using the Actor’s name but for the map example this could be the map names.    
      
   The OutTestCommands Array is what the test runs through, think of the test as a foreach loop that’s run on each element of the OutTestCommands Array. Therefore, we need to add something to this array that we can use in the test, for this enemies example we will be using the Actor’s name again but for the map example this could be the paths to each map. 

Next we are going to write the RunTest function which takes single parameter “const FString& Parameters” and returns a bool.   
   
Inside this function will be the actual test code. For the enemies example we will load the map again get all the actors and then see which matches the one we are testing. We will then test our enemies health value. 