Gauntlet Controller

Learn how to drive runtime functional tests.

Gauntlet Controllers are C++ objects that drive automated tasks outside the Automation Test Framework. They are intended for runtime functional tests, especially when networking is involved.

You can create a Gauntlet Controller by reimplementing the class `UGauntletTestController`, typically in a custom plugin.

`UGauntletTestController` has several methods you can reimplement to control the flow of the test, including:

- `OnInit()` - Called when the controller initializes.
- `OnPreMapChange()` - Called before a map change.
- `OnPostMapChange(UWorld* World)` - Called after a map change. `GetCurrentMap()` returns the new map.
- `OnTick(float TimeDelta)` - Called periodically to let the controller check and control state.

- `OnStateChange(FName OldState, FName NewState)` - Called when a module's state changes. States are game-driven.

Call `EndTest(ExitCode)` when the test finishes to pass its state to the Game instance. The UAT Gauntlet picks up the result of the controller and promotes it to the test.

Gauntlet Roles

For a Gauntlet test to use a Gauntlet Controller, the controller's name needs to be attached to the Gauntlet Role. You can do this with the following code, assuming the name `UMyControllerName:

```
UnrealTestRole ClientRole = Config.RequireRole(UnrealTargetRole.Client);
ClientRole.Controllers.Add("MyControllerName");
```

Copy full snippet

Several Roles can have different controllers.