First exposure to triggers and scripted actions.

Introduction

Triggers are some of the most useful features of UnrealEd. You'll need to use them for opening doors, calling lifts, starting videos, controlling AI Behaviour and lots more besides.

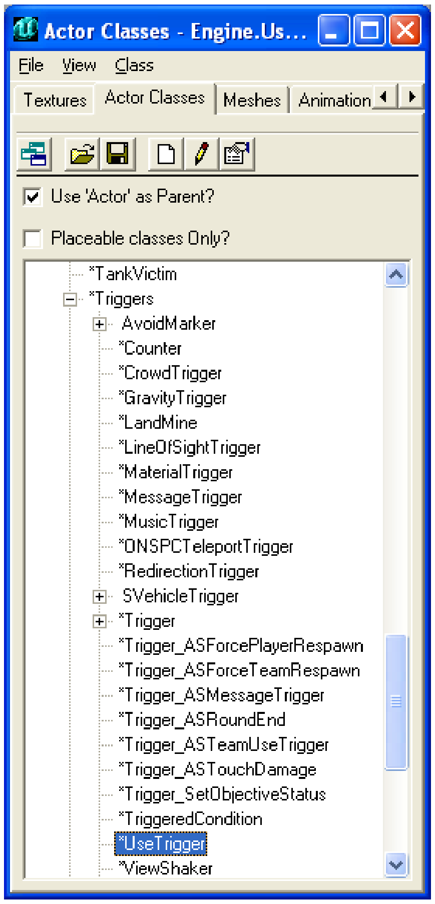
Event Basics

Unreal is EVENT based. You've seen this set up before – think back to your Gamemaker days (the joys of a drag and drop interface). You associated Events with Actions. I'm sure you can remember:

- an Event might be the user pressing the space bar and the associated action is to create a new bullet object and have it fly across the screen.
- another event would be the collision of this bullet with an enemy and the associated action is to remove the enemy from the screen and increase your score.

The more common name for these two stages are Event and Event Handler (what actions are performed when the event is triggered)

This short tutorial shows you an example of using this process within Unreal. It's not as quite as straight-forward in UT as in was in GameMaker, but it is very powerful and actually quite flexible once you get the hang of it. You'll be able to implement a lot of your game using these in-engine scripts. For more advanced sections of our game we'll definitely have to get our hands dirty in the UnrealScript code. Personally, I find that it's often easier to do most things at the UnrealScript level, but I'll leave you to make your own mind up on that ☺ It's a similar to GameMaker – create using the drag and drop interface, or use the scripting language GML.


Actor Classes - Engine.Us...
File View Class
Textures Actor Classes Meshes Animation

[✓] Use 'Actor' as Parent?
[ ] Placeable classes Only?

```
*TankVictim
*Triggers
  AvoidMarker
  *Counter
  *CrowdTrigger
  *GravityTrigger
  *LandMine
  *LineOfSightTrigger
  *MaterialTrigger
  *MessageTrigger
  *MusicTrigger
  *ONSPCTeleportTrigger
  *RedirectionTrigger
  SVehicleTrigger
  *Trigger
  *Trigger_ASForcePlayerRespawn
  *Trigger_ASForceTeamRespawn
  *Trigger_ASMessageTrigger
  *Trigger_ASRoundEnd
  *Trigger_ASTeamUseTrigger
  *Trigger_ASTouchDamage
  *Trigger_SetObjectiveStatus
  *TriggeredCondition
  *UseTrigger
  *ViewShaker
```

Setting up the Change Level Trigger

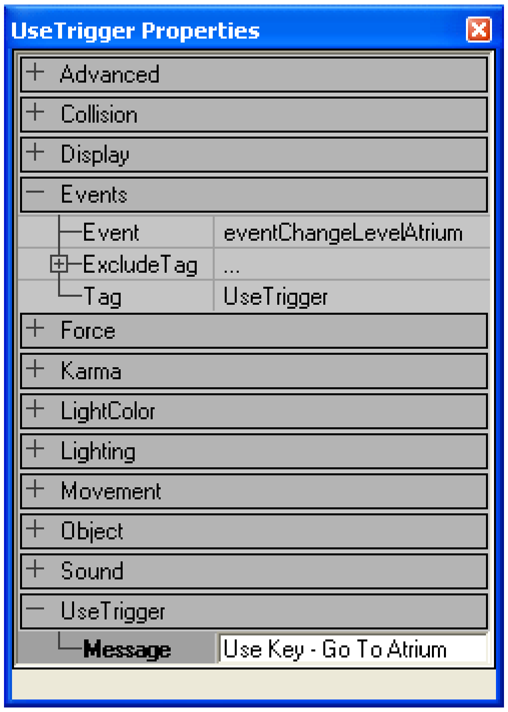
This example will use the Event model to allow our player to change between levels. The player will not automatically move to the new level on collision with an exit, but will be given the choice as to whether they want to teleport to the next arena or stay in the current one.

When the player collides with the exit, a message will be displayed on the screen:

```
"Press USE to teleport to the Atrium"
```

If at this point the player presses their USE button (defaults to E key) they will be teleported to the new arena.

In order to achieve this we need both a `UseTrigger` (the event trigger) and a `ScriptedTrigger` (the event handler).



UseTrigger Properties

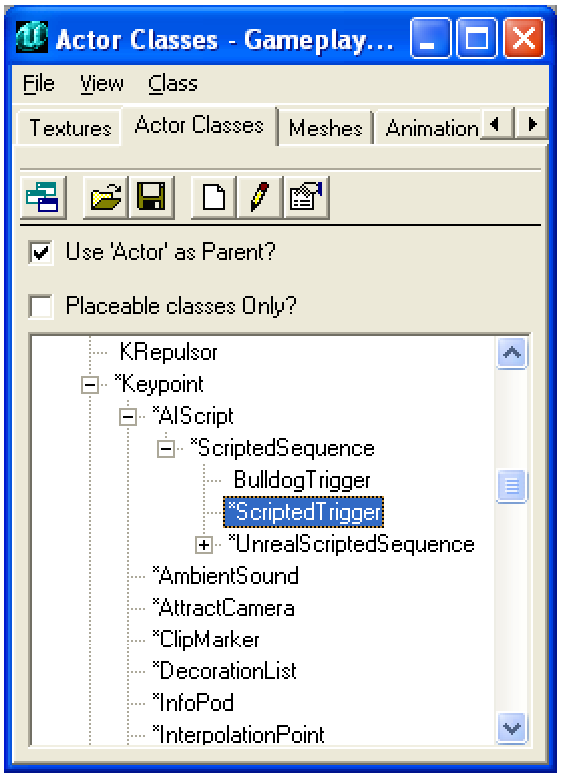
+ Advanced	
+ Collision	
+ Display	
− Events	
Event	eventChangeLevelAtrium
⊞ ExcludeTag	...
Tag	UseTrigger
+ Force	
+ Karma	
+ LightColor	
+ Lighting	
+ Movement	
+ Object	
+ Sound	
− UseTrigger	
Message	Use Key - Go To Atrium

Setting up the Event Trigger

Begin by placing a `UseTrigger` down in your level. It can be found in the Actor Browser at `Actor`→`Triggers`→`UseTrigger`.

The location of the `UseTrigger` in the level is the position where a player will be given the option to teleport. In my level I've placed it by a doorway that the player will teleport through.

Next, bring up the properties for this `UseTrigger` (double-click on the `UseTrigger` in the 3D view). The property window allows us to alter all the viewable properties. When we start writing UnrealScript you'll be given the option to 'expose' properties within the UnrealEd. You'll see a number of different property categories.


Actor Classes - Gameplay...
File View Class
Textures Actor Classes Meshes Animation

☑ Use 'Actor' as Parent?

☐ Placeable classes Only?

```
KRepulsor
⊟ *Keypoint
  ⊟ *AIScript
    ⊟ *ScriptedSequence
        BulldogTrigger
        *ScriptedTrigger
      ⊞ *UnrealScriptedSequence
    *AmbientSound
    *AttractCamera
    *ClipMarker
    *DecorationList
    *InfoPod
    *InterpolationPoint
```

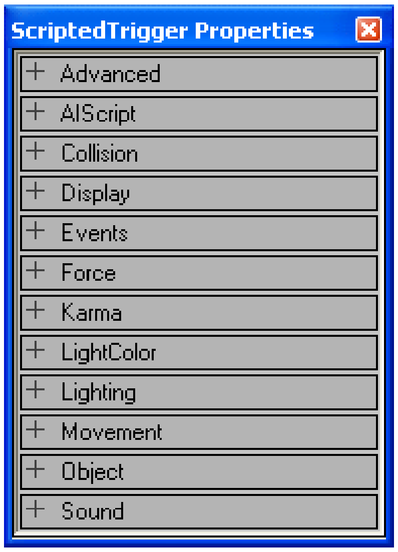
In order to change the text displayed to the user we need to alter the text in `UseTrigger`→`Message`. Do this now and change it to read something appropriate.

At this point we need to agree on a naming convention. We need to give this event a name so that we can refer to it in the Event Handler.

I'm going to call my event: `eventChangeLevelAtrium`

Add this to the properties under `Events`→`Event`.

Okay, that's out Event Trigger all set up – now we need to set up the Event Handler.



Setting up the Event Handler

Again, this isn't too difficult to get working. We'll need to use two commands, one that tells the script to wait for our event and the second that actually does the teleport.

Start by placing down the `ScriptedTrigger` in the level. This can be found `Keypoint→AIScript→ScriptedTrigger`. It actually doesn't matter where in the level it goes but it makes sense to stick it down somewhere near our `UseTrigger`.

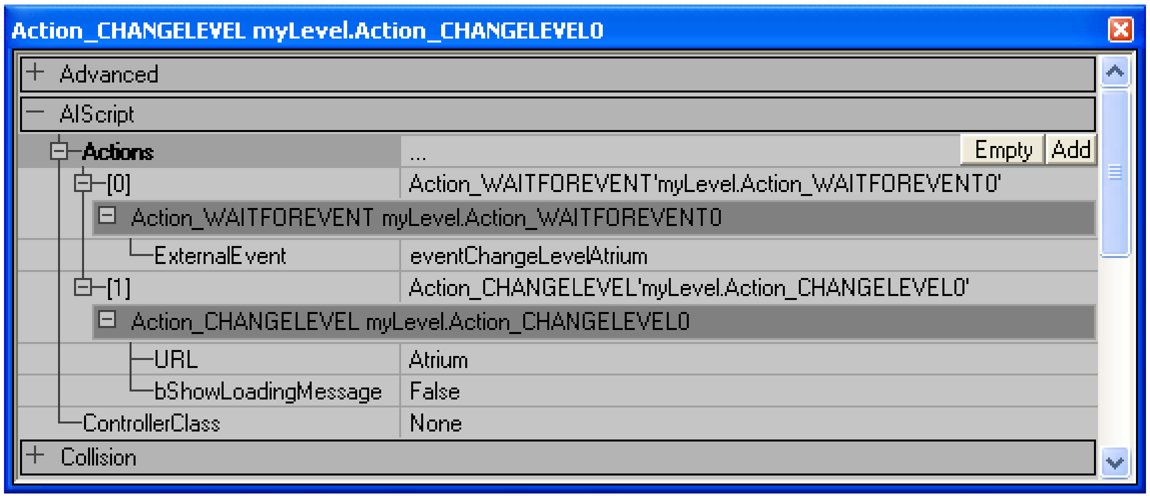
The `ScriptedTrigger` (and other child classes of `AIScript`) contain the `AIScript` category in its property list. This is essential for adding our own scripts. If you've got the Mastering Unreal book handy – flick to page 505 onwards to read the full list of Actions we can include in our script.

Just to remind you – using the AIScript interface isn't really related to UnrealScript – it's just a simple way of performing a few built-in actions. It's still pretty useful but be prepared for a lot more coding.

The actions that we'll be using for our Event Handler are:

- `Action_WAITFOREVENT` – this sets-up the synchronous wait.
- `Action_CHANGELEVEL` – actually changes the level

I find the interface a bit cumbersome – but I'll leave you to explore this yourself. Add those two actions and set up the properties that go with them. For example, the `ExternalEvent` property in the `Action_WAITFOREVENT` should be set to your event name.


Action_CHANGELEVEL myLevel.Action_CHANGELEVEL0

Advanced	
AIScript	
Actions	...
[0]	Action_WAITFOREVENT'myLevel.Action_WAITFOREVENT0'
Action_WAITFOREVENT myLevel.Action_WAITFOREVENT0	
ExternalEvent	eventChangeLevelAtrium
[1]	Action_CHANGELEVEL'myLevel.Action_CHANGELEVEL0'
Action_CHANGELEVEL myLevel.Action_CHANGELEVEL0	
URL	Atrium
bShowLoadingMessage	False
ControllerClass	None
Collision	

Testing

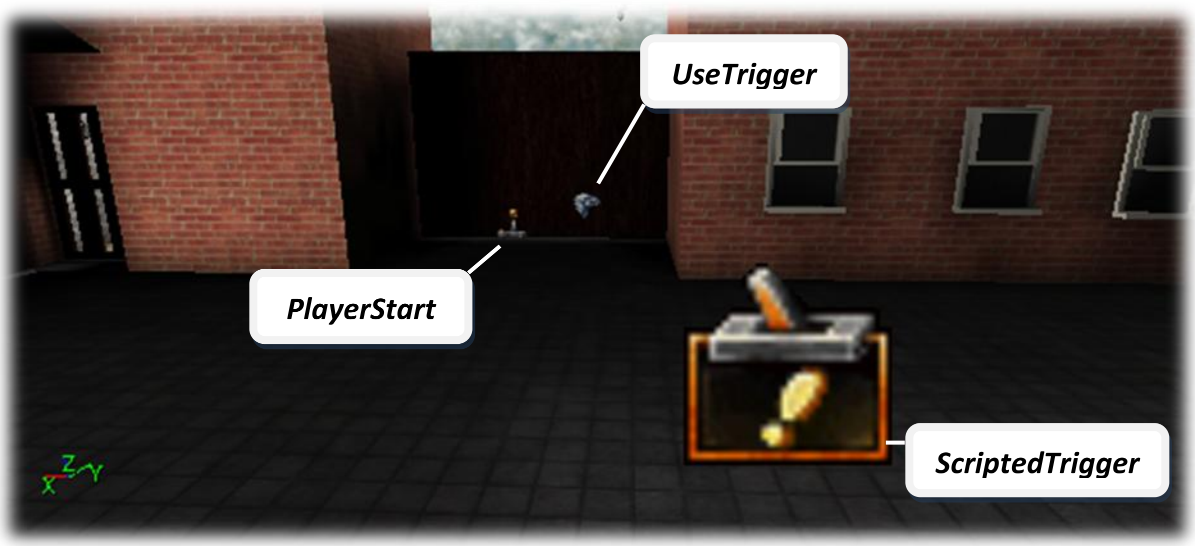
It's a fairly simple script, so hopefully you can start up your level and it will work first time.

Note – You don't need to rebuild your level if you're just changing scripts or properties in UnrealEd. You only need to rebuild your level if you're altering lights / geometry / textures etc.

If things aren't working, we have two options to try:

- Use the console command `CAUSEEVENT`.
  `CAUSEEVENT <eventname>` allows us to trigger off an event without using the `UseTrigger`. This will help us to decide if only one part of the Event and Event Handler are working.
- Look in the logs for any error messages:
  `C:\Program Files\UT2004\System\UT2004.log`

If you can't spot any problems with either of these then you'll need to get some help off a tutor or go through the steps again.



What Next?

Now it's time to start exploring what else you can use the UnrealED scripts for. If you get a good understanding of this early you might find you don't need to write as much UnrealScript for your final game.