Overview

A scope (also known as an Action Scope) is a high level animation *channel*.

Fragments are played on scopes. (and more technical: actions control scopes)

A scope represents where fragments get triggered. Scopes can be used to drive:

- Different parts of the body (Base body, Torso, Legs)
- Different entities that are synchronized to the main entity (Weapon, Attachments, Target Player)

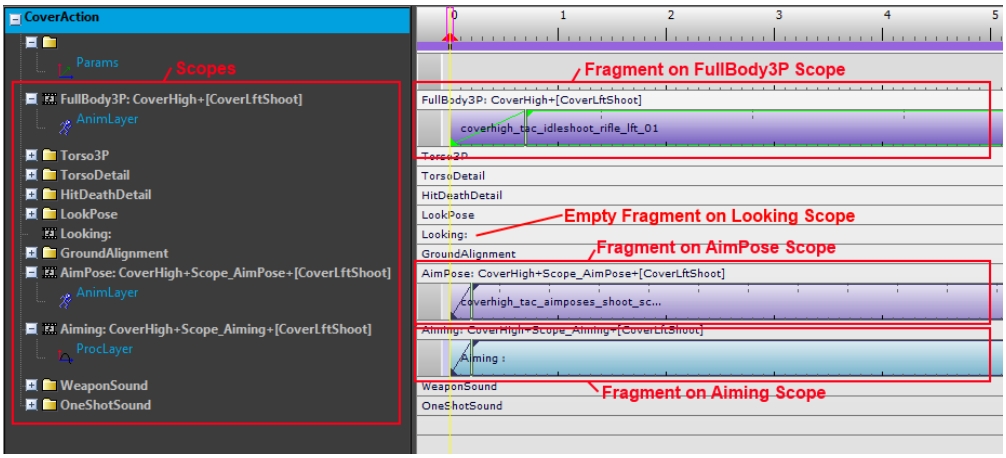
They can also represent more abstract channels:

- Systemic animation (ProceduralLooking, ProceduralWeaponMovement, ...)
- Systems not directly related to animation (ForceFeedback, WeaponSound, ...)

To do this, each scope has:

- A scope context, which contains:
  - an Entity, the entity this scope plays on. Typically a scope plays on a main entity, but it can also play on attached entities like weapons or other entities or characters that this character interacts with.
  - (optional) a Character in that Entity (entities can have multiple characters attached to them, so you need to specify which one)
  - an Animation Database, containing the fragments to play on this scope.
- A range of animation layers to play animations on. Any animation clips in fragments that are playing on this scope will end up on those layers. You can also specify that this scope refers to no layers at all, which is useful for the more abstract scopes that only use procedural clips.
- (optional) Scope Tags, to make sure only fragments with those tags are played on this scope (needed to play different fragments on different scopes that use the same scope context)

Example



Here we see a screenshot of the Mannequin Fragment Editor.

To the left you see a list of the scopes available in this mannequin setup. The four scopes with the movie icon (FullBody3P, Looking, AimPose, Aiming) are part of the scope mask, and we can edit the fragments on those scopes.

We opened up the fragmentID "CoverAction" (see top left), with tags "CoverHigh" and fragtags "CoverLftShoot". This opened up **four** fragments. The ones on the FullBody3P, AimPose and Aiming scope are clearly visible. The one on the Looking scope is also there, but empty. This also means that when an action plays this combination of fragmentID & tags, there will be nothing playing on the Looking scope.

To understand this example setup, read the tutorial Controlling Looking (and Aiming) for AI in Mannequin.

Creating & Editing Scopes

Scopes are created and edited by editing the Controller Definition File (xxxControllerDefs.xml).