• ActionFilter
• AliveCheck
• Damage
• EnslaveCharacter
• FacialAnim
• FacialExpression
• GetNearestActor
• GrabObject
• HealthCheck
• HealthGet
• HealthSet
• KillInfo
• KillPlayer
• LocalPlayer
• LocalPlayerMovementParameter
• OnDeath
• PlayMannequinFragment
• PlayerCinematicControl
• PlayerDropObject
• PlayerInteractiveAnimation
• PlayerIsInAir
• PlayerKnockDown
• PlayerLink
• PlayerLookAt
• PlayerOnPickUpAmmo
• PlayerStaging
• PlayerStumble
• RagdollizeCharacter
• Sensor
• SetPlayerModel
• VisualDetector

ActionFilter

FlowNode to enable/disable ActionFilters. If a player is not proved as input entity, the filter will affect all players.

```
Actor:ActionFilter

Choose Entity

Enable
Disabled

Filter:
```

AliveCheck

Check the death/alive status of an entity (actor).

```
Actor:AliveCheck

Choose Entity

Trigger
Status
Alive
Dead
```

Damage

Damages attached entity by [Damage] when [Trigger] is activated.

```
Actor:Damage

Choose Entity

Trigger
Damage=0
DamageRelative=0
Position=0,0,0
```

The Damage node (formerly Game:DamageActor) is used to damage the AI characters or the player.
The target of the node has to be an actor, it does not work with entities that are not based on an actor class. The amount of damage as well as the position where the damage will be applied can be defined.

In the above example, when the player is inside the trigger box it will apply 1 damage to the player every second at the start. As the Interpol:Float counts down to 0.05, this will reduce the time between each damage strike being applied.

**EnslaveCharacter**

Enslave one character to another.

**FacialAnim**

Play facial sequences on Actors.

**FacialExpression**

Play facial expressions on Actors.

**GetNearestActor**

Outputs the nearest actor to a given position. WARNING: potentially performance heavy, don't trigger it every frame.
**GrabObject**

Target entity will try to grab the input object, respectively drop/throw its currently grabbed object.

**HealthCheck**

The HealthCheck (formerly Game:ActorCheckHealth) node checks the health of an actor and outputs the result on a Boolean output port.

In the above example, we have setup a flow graph to test the players health value when he enters a trigger box, then if it is above 50, spawn all three of the enemies. If the players health is below 50, only spawn two of the enemies. The health of an actor is checked by using the Game:ActorCheckHealth node and checking the Boolean output port using a Math:FromBoolean node. The input to the Math:FromBoolean node will be true if the health of the target entity is between 0 and 50. The Math:FromBoolean node then splits the output to be passed on to either 2 or 3 enemies.

**HealthGet**
Formerly "Game:ActorGetHealth". Get health of an entity (actor).

This node outputs the health of the target entity as an integer.

In this example, the health of the target entity is compared using a Math:Less node to check if the value is below 50.

If it is true, it will pass through the true port & into the HealthSet node then carry on with the flowgraph. If the health is above 50, do nothing and pass on to the next step of the flowgraph.

**HealthSet**

Formerly "Game:ActorSetHealth". Set health of an entity (actor).

This node can be used to set the health amount on a specified entity.

In the above example, we are testing the players health level with the Game:ActorSetHealth node and if it is below a certain level we use the Game:ActorSetHealth node to set it to 75.

**KillInfo**

Tracks Kills on Actors. All input conditions (KillerId, VictimId) must be fulfilled to output. If a condition is left empty/not connected, it is regarded as fulfilled.

**KillPlayer**

Instantly kills the local player.
LocalPlayer

Outputs the local players entity id - NOT USABLE FOR MULTIPLAYER WITHOUT UPDATING BY HAND AFTER GAMESTART.

In this example the ID of an entity entering a proximity trigger is compared to the local player ID using the Math:Equal node.

LocalPlayerMovementParameter

Set/get actor parameters during runtime. Output always includes multipliers!

OnDeath

Triggers when an actor dies.

PlayMannequinFragment

Play a Mannequin Fragment on a given entity with given Tags.
PlayerCinematicControl
Restrict player controls during cinematics.

PlayerDropObject
Force the player to drop any currently held object/enemy.

PlayerInteractiveAnimation
Plays interactive animation for Player.

PlayerIsInAir
Check the InAir/NotInAir status of the player. InAir = jumping or falling.

PlayerKnockDown
Knock down local player.
PlayerLink

Linking the Player to an Entity (with FreeLook).

The PlayerLink node (formerly Game:PlayerLink) can be used to dynamically attach the player to an entity with free look.

KeepTransformDetach means when the unlink happens, keep the orientation from the linked position.

In the above example, after a two second delay we are linking the player to the entity Grunt1. After six seconds we are unlinking the player again.

PlayerLookAt

Enable/Disable look at target option for the player.

PlayerOnPickUpAmmo

Triggers when the specified ammo is picked up by the local player (picking up is NOT just taking from ammoboxes or similar.)

PlayerStaging

Formerly “Game:PlayerStaging”. The PlayerStaging node is used to limit the controls of the player - ViewLimits and Position Lock.
The TryStance input port tries to place the character into a specific stance. The ViewLimit inputs are scaled from 0-100, 100=180 degrees, 50=90 degrees.

**PlayerStumble**

Causes the player to stumble with varying strengths and dampening of impulses.

**RagdollizeCharacter**

(un)ragdollize a character with a defined impulse.

**Sensor**

Tracks the attached Entity and its Vehicle-related actions.
When enabled the Sensor node (formerly Game:ActorSensor) outputs events on the output ports when the target entity of this node performs certain actions.

The node has only two input ports for enabling and disabling it. When enabled the output ports will be triggered whenever a certain action happens. If the outputs are not needed, the node should be disabled to prevent any unnecessary outputs.

Each port outputs an entity id or other information that can be used to determine which objects the target entity interacted with.

In the above example, we are testing if the player has entered a vehicle, then when he has entered we complete the objective and then 1 second later enable the second objective.

Then we are testing if the player dies or has exited the vehicle, if so fail the second objective. When you have finished your test, it is a good idea to Disable the Actor:Sensor.

SetPlayerModel

Set local player's model.
VisualDetector

Sets a visibility detector for actors. The Node's entity is only used to provide position and orientation, it can be any entity type.

<table>
<thead>
<tr>
<th>Actor:VisualDetector</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Choose Entity</strong></td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>seenEntity</td>
</tr>
<tr>
<td>disable</td>
<td>notSeen</td>
</tr>
<tr>
<td>observerFOV=20</td>
<td></td>
</tr>
<tr>
<td>sightRange=50</td>
<td></td>
</tr>
<tr>
<td>entityToLookFor=0</td>
<td></td>
</tr>
</tbody>
</table>