Setting Up a Monster

To set up a monster in your custom story, you need to place it in the level editor. Monsters can be found when you click the “Entities” tab and then go to the subsection “enemy”. For this tutorial, we'll choose the monster that is called “servant_grunt”. Select it and then place it wherever you want. I'll assume you have a least part of your level constructed and know how to make a custom story entirely. Once you placed it, go to the select tool and select the monster you just placed. Under it's name, there will be a checkbox called “Active”. Click the checkbox to set the monster inactive. You will notice that the monster is partially “faded” in a sense. We do this so that the monster doesn't activate when you start the level.

Now go to the tab called “Areas”. The area type should say “Script”. Click in the level editor on where you want the player to touch for when you want the monster to spawn. Go to the select tool and click on the area where it says “AREA”. You will see that it's name is `ScriptArea_1`. It is possible that it could be a different number at the end. Now click on where it says it's name and change it to “PlayerCollide” without quotes. Press enter. Then near the top right corner of the level editor you'll see that there are 3 buttons. From top to bottom they say “Translate”, “Rotate”, and “Scale”. Click on the bottom one that says “Scale”. Now scale the script area box to fit whatever hallway or path the player will touch to set the monster active.

Adding Path Nodes

We will now add path nodes to the map so the monster walks from one place to another. When testing/playing your custom story, the monster will find you if you go too close to it and it sees you. If the path nodes aren't near you, then the monster can't look for you over where you are, but for now we will have the monster walk from one place to another.

Go click on the tab where it says “Areas” and then go to the subsection called “PathNode”. Now make a path you want the monster to go to. Try not to put too many of them; Perhaps keep them a meter or two apart. Make sure the monster doesn't have to cut corners which could get him stuck. The monster will automatically disappear if it reaches the final path node given to it and the player isn't looking at the monster and the player is at least 10 meters away from the monster.

Scripting It All Together

Now you got the monster and path nodes set up. Now it's time to script. We will have the player collide with an area called `PlayerCollide`, which will activate the mosnter.

Let's assume that we have over 10 path nodes in our map, leading to each other, that we want the monster to follow. `PathNodeArea_1` will be the starting path node, `PathNodeArea_5` is at the center of an intersection where the monster will make a left turn towards `PathNodeArea_10`, which is where we want the monster to come to an end. Each of these path nodes have other path nodes leading to them, each separated no further than 2 meters apart. From `PathNodeArea_1` to `PathNodeArea_5` is a straight path, and from `PathNodeArea_5` to `PathNodeArea_10` is also a

straight path.

With that assumption, we would write our script like this:

```
void OnStart()
{
    AddEntityCollideCallback("Player", "PlayerCollide", "MonsterFunction",
true, 1);
}

void MonsterFunction(string &in asParent, string &in asChild, int alState)
{
    AddEnemyPatrolNode("servant_grunt_1", "PathNodeArea_1", , "");
    AddEnemyPatrolNode("servant_grunt_1", "PathNodeArea_5", 0.001, "");
    AddEnemyPatrolNode("servant_grunt_1", "PathNodeArea_10", , "");
    SetEntityActive("servant_grunt_1", true);
}
```

Notice that we have 0.001 for the 5th path node. This is because a wait time of 0 does not mean 0, and 0.001 secs is practically 1 millisecond! We don't want the monster to be spending any time at the intersection, and so we have the game waste the least amount possible on the path node.

The parameter for monster animation is practically useless, so we leave it blank.

In the Script Functions Page, this is what it says about the `AddEnemyPatrolNode` function:

```
void AddEnemyPatrolNode(string& asName, string& asNodeName, float
afWaitTime, string& asAnimation);
```

Adds a patrol node to the enemy's path.

asName - internal name of the enemy
asNodeName - Name of path node
afWaitTime - Time in seconds that the enemy waits at the path node before continuing (remember, 0 does not mean 0!)
asAnimation - The animation the enemy uses when reaching the path node (this, however, provides no results, so just leave it blank)

Infinite Loops

It is possible to make the monster to follow a set of patrol nodes and never stop repeating them until the monster is disabled manually by `SetEntityActive` or until the player is out of the monster's activation distance. By default, monsters will automatically loop through the path nodes assigned to them, but there are many factors that would cause them to auto-disable before then. Before the release of Justine, people had relied on continuously adding path nodes to the monster while having increased the monster's activation distance to a large amount, therefore the monster never reaches

the final path node in its memory and so never given a chance to auto-disable and the player is always within its activation distance. Since the release of Justine, we've had the option to disable “auto-disable on final path node” for the monster. By default, the enemy suitors `Basile` and `Malo` have auto-disable disabled for them. In order to do the same for other monsters, you need to manually edit the ENT file for the monster.

Never modify the original Amensia content without first making a separate copy which you can safely modify for your own purposes!

Open the monster ENT in a plain text editor and add the following line under the `UserDefinedVariables` element:

```
<Var Name="AutoRemoveAtPathEnd" Value="false" />
```

Also, look for the variable `ActivationDistance` and change its value to a large number (e.g. 1000).

Save the changes, and then add the monster to your level (or replace a previous monster).

Updating the script

Let's assume we've replaced the monster with one that doesn't auto-disable when reaching the final path node. We would update the script like so:

```
void OnStart()
{
    AddEntityCollideCallback("Player", "PlayerCollide", "MonsterFunction",
true, 1);
}

void MonsterFunction(string &in asParent, string &in asChild, int alState)
{
    AddEnemyPatrolNode("servant_grunt_1", "PathNodeArea_1", , "");
    AddEnemyPatrolNode("servant_grunt_1", "PathNodeArea_5", 0.001, "");
    AddEnemyPatrolNode("servant_grunt_1", "PathNodeArea_10", , "");
    AddEnemyPatrolNode("servant_grunt_1", "PathNodeArea_5", 0.001, "");
    SetEntityActive("servant_grunt_1", true);
}
```

Since the monster will automatically patrol the path nodes, we don't have to duplicate the line for PathNodeArea_1

Tips

Here is some tips and tricks that can help when trying to avoid disaster when it comes to monsters and path nodes.

1. Monsters have difficulty navigating the map without path nodes. Be sure to place path nodes all over the map, even if you're not going to reference them in your script file.
2. Be careful when placing path nodes for a monster on stairs because if there is one that is missing or not placed right, it'll mess up completely.
3. Monsters have difficulty turning corners, so always place a path node at every intersection (and lead the monster there if need be).
4. The Script Function page is your friend! Under “Enemies”, there are many things that can be used to “spice it up”.
5. Any monster can bash through doors if their path nodes go through it and the box is unchecked for “DisableBreakable” for it.