Technical Manual for the Personality Demonstrator

Robert Kraaijeveld, Cees-Jan Nolen, Steven Schenk

February 8, 2017

Contents

1 Introduction

The application to which this manual is attached, the 'Personality Demonstrator', was made by the three authors of this document for their 3rd year internship at the University of Tilburg; this internship was supervised by mr. Abbadi, PhD Student and dr.ir. P.H.M. Spronck.

It was made specifically as a demonstration of the capabilities of the Experience Room (A four-wall Mixed Reality chamber) located at Tilburg University. Instructions for the usage of this package within the context of the Experience Room can also be found within this document.

Robert Kraaijeveld, Cees-Jan Nolen, Steven Schenk

2 Brief features overview

2.1 Customizable characters, actions and events

Each character and its' actions are customizable through XML. Adding actions and the like in XML will be explained more thoroughly later.

2.2 Action selection through genetic algorithm and DotProduct

What actions and reactions a given NPC carries out is determined by a genetic algorithm which computes a dotProduct of the NPC's personality values and the personality values of the currently reachable actions/reactions in the graph; the action whose DotProducts' value matches the closest gets chosen.

When an NPC has finished carrying out an action, its' personality values get summed with the values of the action that was just carried out. This reflects how an NPC's previous deeds would influence its' future decision-making.

2.3 3D and Experience Room compatible

This package can be easily reconfigured to be used within the Tilburg University Experience Room, using standard scripts already available at the University.

3 Prerequisites

3.1 Unity

Naturally, you need the Unity environment to develop games using this package. The entire package was tested on Unity 5.3.5, so we can only verify that the package will work without problems on this specific version of Unity.

3.2 Casanova

Our project uses the Casanova programming language, but unless you want to change details like the exact time it takes for an event to be completed it is not necessary to install the Casanova Compiler yourself.

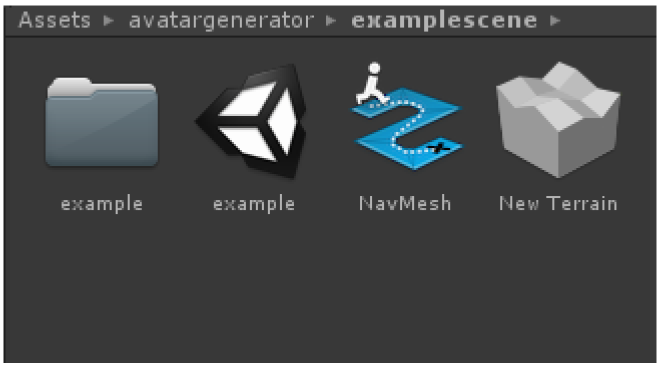
If you do want to customize these details, please follow the instructions at https://github.com/vs-team/casanova-mk2/wiki/Workshop. Also, make sure your project directory looks exactly(!) like this:

```
Casanova template and binaries
├── CasanovaCompiler
├── ResetCNV
└── Samples
    └── The folder containing your Unity Project
```

4 Setting up

This section of the manual will explain to you how to get the Personality Demonstrator working in your project.

IMPORTANT: If you want to start a project with the Personality demonstrator present from the beginning, you can simply open the Example Scene and begin working from there!



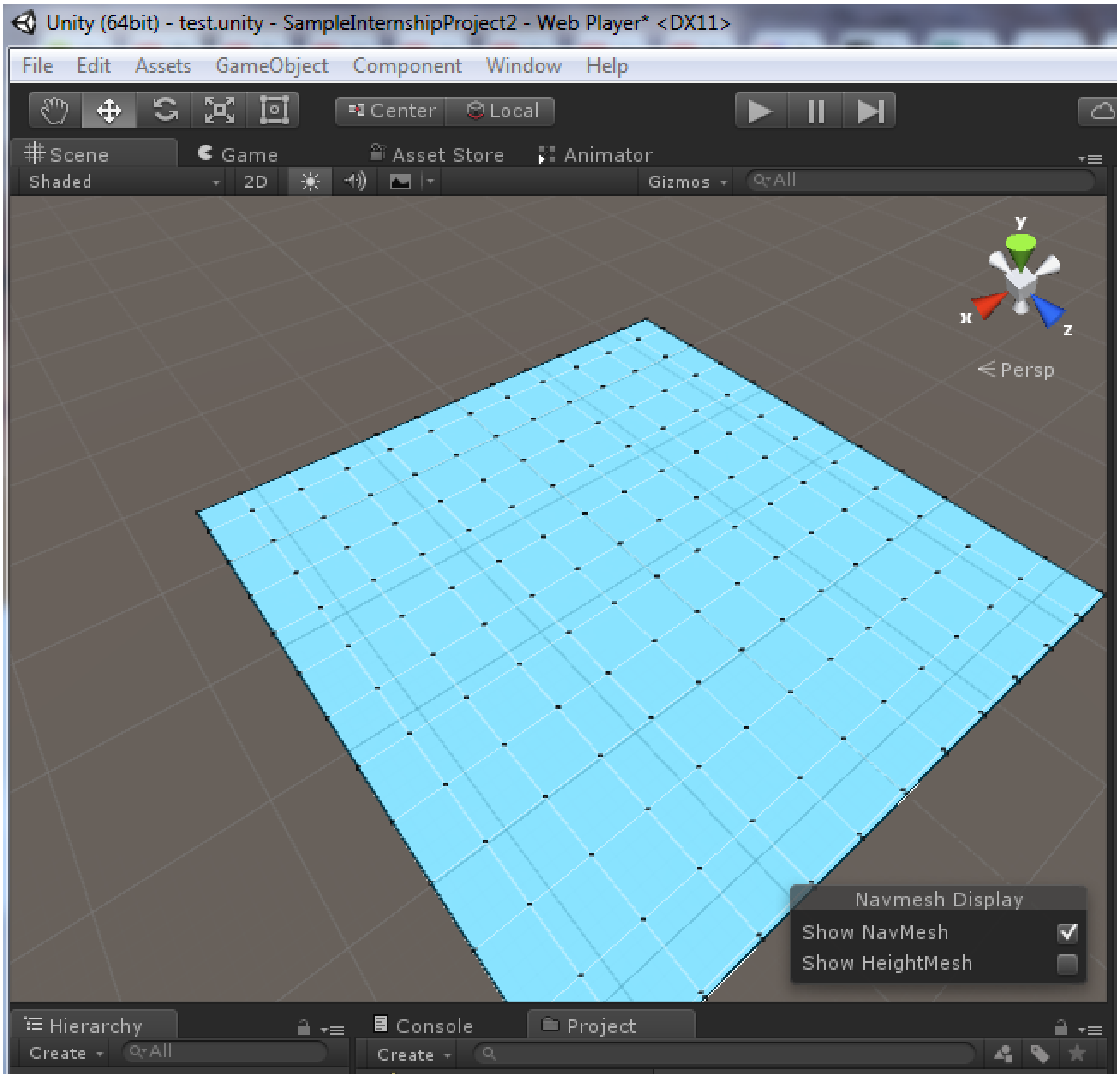
In case you want to add the personality demonstrator later on, or do not wish to restructure your entire scene, follow these steps.

4.1 Import the Unity Package

Naturally, the first step is to import the Personality Demonstrator Unity Package. You can choose to leave out our boilerplate models and animations if you want use your own instead.

4.2 Create a navmesh

Our NPC's use Unity's Navmesh functions to move around your scene, so you need to define a terrain including a navmesh for them to walk around on. (A tutorial for this can be found here, in case you are not familiar with it: https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html)

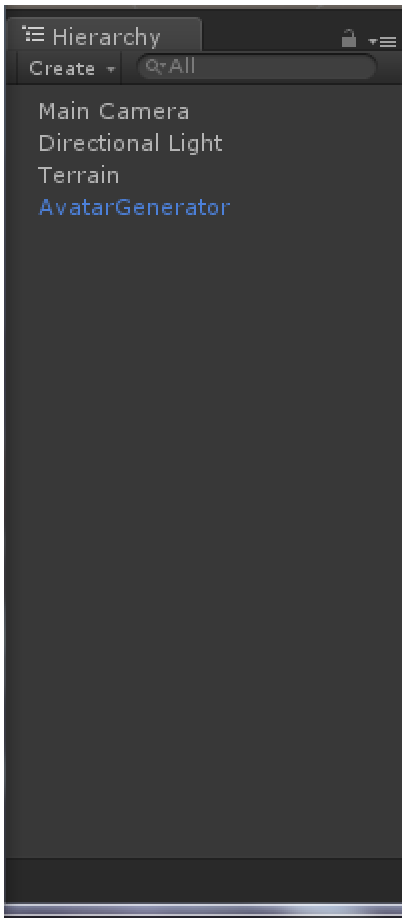

Unity (64bit) - test.unity - SampleInternshipProject2 - Web Player* <DX11>
File Edit Assets GameObject Component Window Help
Center
Local
Scene
Game
Asset Store
Animator
Shaded
2D
Gizmos
All
y
x
z
Persp
Hierarchy
Create
All
Console
Project
Create

Navmesh Display
Show NavMesh ✓
Show HeightMesh

Above you can see what the end result should look like.

4.3 Add the AvatarGenerator Prefab to the project hierarchy

This speaks for itself; This prefab needs to be added to the hierarchy in order to allow several important scripts, as well as the 'World' script, consisting of compiled Casanova code, to run.



Main Camera
Directional Light
Terrain
AvatarGenerator

5 Configuration

5.1 Creating custom NPC's

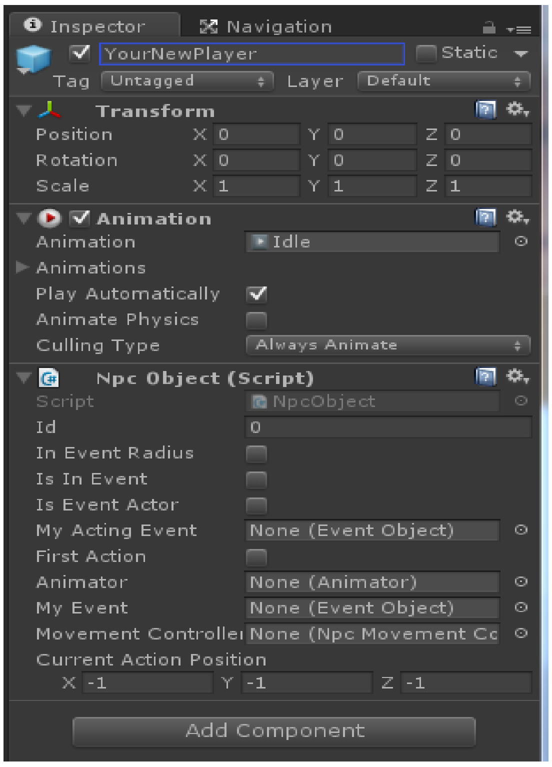
5.1.1 Adding a fixed-value NPC

If you want to add an NPC with fixed instead of random personality values, follow these steps:

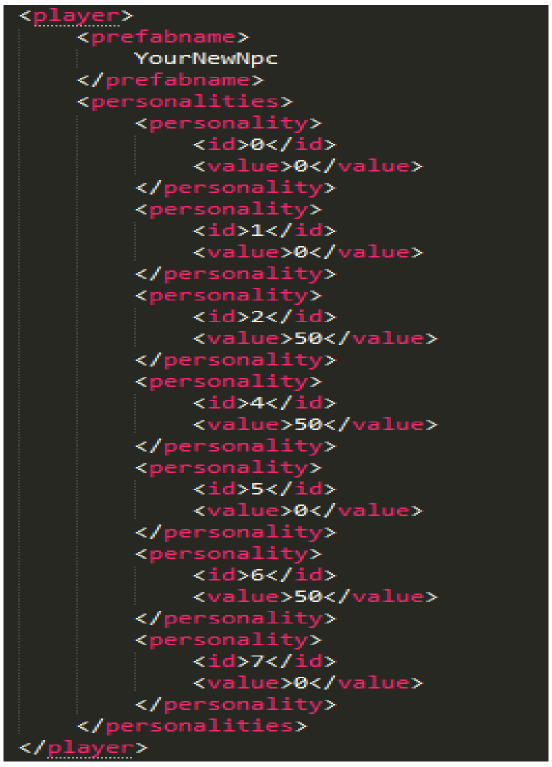
1. Create a new Prefab and add the NpcObject script and an Animation component, containing a standard idle animation and a list of the animations that you want your NPC to be able to execute (More on this later, when we explain how to add custom animations.)

2. Add a new entry in Resources/players.xml for the new player you just made.

Your player object and players.xml ought to look something like this when you are done:



```
Inspector    Navigation
YourNewPlayer    Static
Tag Untagged    Layer Default
Transform
Position    X 0    Y 0    Z 0
Rotation    X 0    Y 0    Z 0
Scale       X 1    Y 1    Z 1
Animation
Animation           Idle
Animations
Play Automatically  ✓
Animate Physics
Culling Type        Always Animate
Npc Object (Script)
Script              NpcObject
Id                  0
In Event Radius
Is In Event
Is Event Actor
My Acting Event     None (Event Object)
First Action
Animator            None (Animator)
My Event            None (Event Object)
Movement Controllei None (Npc Movement Cc
Current Action Position
  X -1    Y -1    Z -1
Add Component
```



```
<player>
    <prefabname>
        YourNewNpc
    </prefabname>
    <personalities>
        <personality>
            <id>0</id>
            <value>0</value>
        </personality>
        <personality>
            <id>1</id>
            <value>0</value>
        </personality>
        <personality>
            <id>2</id>
            <value>50</value>
        </personality>
        <personality>
            <id>4</id>
            <value>50</value>
        </personality>
        <personality>
            <id>5</id>
            <value>0</value>
        </personality>
        <personality>
            <id>6</id>
            <value>50</value>
        </personality>
        <personality>
            <id>7</id>
            <value>0</value>
        </personality>
    </personalities>
</player>
```

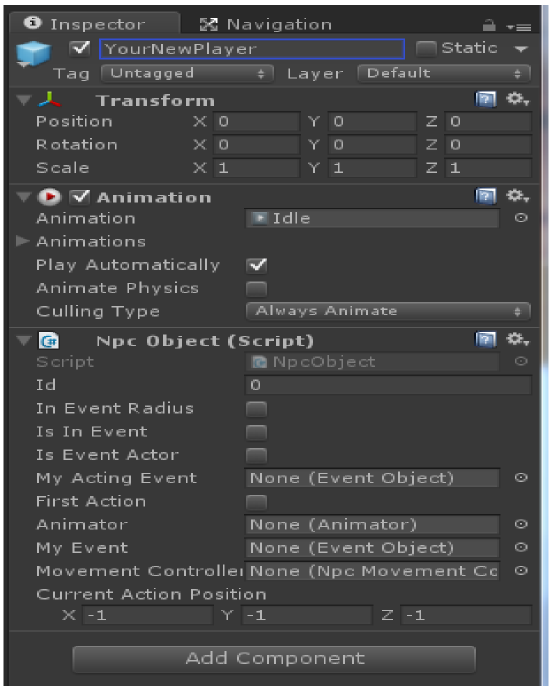
5.1.2 Adding a random-value NPC

Adding an NPC whose personality values will be determined randomly is very easy; Simply follow these 2 steps.

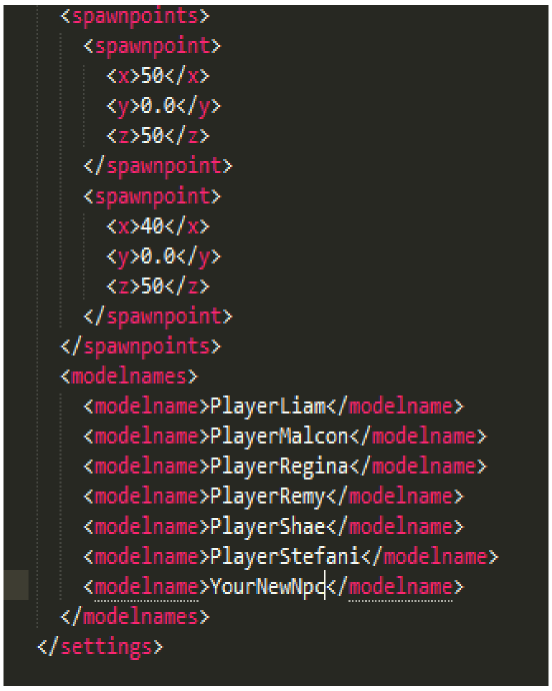
1. Create a new Prefab [1] and add the NpcObject script and an Animation component, containing a standard idle animation and a list of the animations that you want your NPC to be able to execute (More on this later, when we explain how to add custom animations.) [2]

2. Add a new entry in settings.xml for the new NPC's you just made, supplying only the name of the NPC's gameObject.

Your player object and players.xml ought to look something like this when you are done:



```
Inspector        Navigation
YourNewPlayer                         Static
Tag  Untagged            Layer  Default
Transform
Position          X 0        Y 0        Z 0
Rotation          X 0        Y 0        Z 0
Scale             X 1        Y 1        Z 1
Animation
Animation                 Idle
Animations
Play Automatically        ✓
Animate Physics
Culling Type              Always Animate
Npc Object (Script)
Script                    NpcObject
Id                        0
In Event Radius
Is In Event
Is Event Actor
My Acting Event           None (Event Object)
First Action
Animator                  None (Animator)
My Event                  None (Event Object)
Movement Controlle        None (Npc Movement Cc
Current Action Position
    X -1          Y -1          Z -1
Add Component
```



```
  <spawnpoints>
    <spawnpoint>
      <x>50</x>
      <y>0.0</y>
      <z>50</z>
    </spawnpoint>
    <spawnpoint>
      <x>40</x>
      <y>0.0</y>
      <z>50</z>
    </spawnpoint>
  </spawnpoints>
  <modelnames>
    <modelname>PlayerLiam</modelname>
    <modelname>PlayerMalcon</modelname>
    <modelname>PlayerRegina</modelname>
    <modelname>PlayerRemy</modelname>
    <modelname>PlayerShae</modelname>
    <modelname>PlayerStefani</modelname>
    <modelname>YourNewNpc</modelname>
  </modelnames>
</settings>
```

[1]The easiest way to do this is to drag one of our pre-made Npc's into the hierarchy, changing what you want, and then dragging the finished NPC back to the Resources/ folder.

[2]This is done in the same way as with a fixed-value NPC

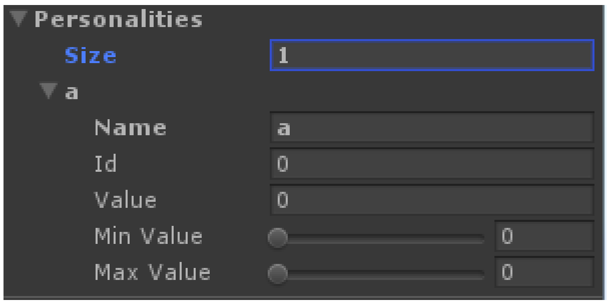
5.2 Customizing available NPC traits

You can create your own personality traits, and determine how they can be set in the Unity Editor.

5.2.1 Defining personality traits in XML or via the Unity Interface

You can either define new personality traits in settings.xml, or via the Unity Interface.

If you uncheck the checkbox under ’Use default value’ you can create new personalities in the following way: Increase the amount of personalities; then, set the values of the new personality.



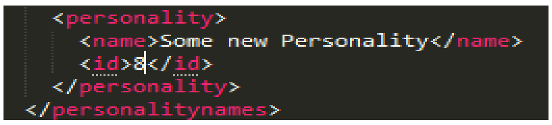
```
▼ Personalities
    Size          1
  ▼ a
      Name        a
      Id          0
      Value       0
      Min Value   0
      Max Value   0
```

5.2.2 Creating your own personality traits via XML

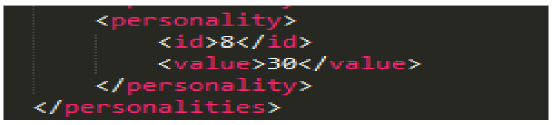
If you check the checkbox under ’Use default value’ you can create new personalities in this way: Add a new personality tag and it’s body in settings.xml. Note that the new trait has to have a unique ID.

Also, note that the Personality’s name tag is not used directly by the program; its’ only use is to make it easier for you to see which personality id is representing which real-life trait.

Of course, in order for the personality to be actually used, you have to include it when creating a new personality for an NPC or when you are setting the personality values of an action. For example:



```
  <personality>
    <name>Some new Personality</name>
    <id>8</id>
  </personality>
</personalitynames>
```



```
    <personality>
        <id>8</id>
        <value>30</value>
    </personality>
</personalities>
```

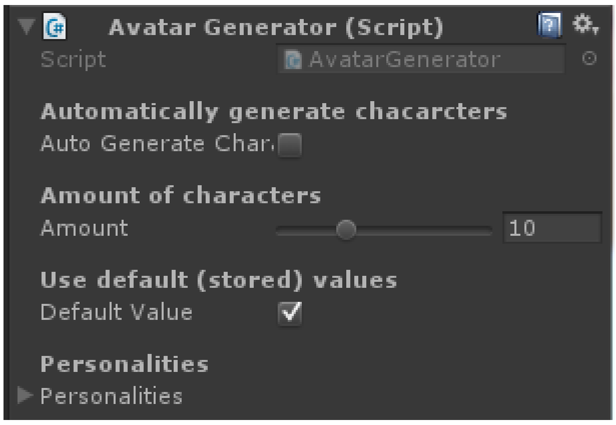
You can create, add and use as many personality values as you like!

Personality values always range from 0 to 100.

5.2.3 Switching between random and fixed-value NPC's

You can easily switch between using a fixed amount of fixed-value NPC's or using a variable amount of NPC's with random personality values in the Unity Editor.

Just check the AvatarGenerator's 'Auto Generate Characters' value in order to use the fixed NPC's, or uncheck it and set the amount of random-personality NPC's that you want to create.



Avatar Generator (Script)

Script: AvatarGenerator

Automatically generate chacarcters

Auto Generate Char: ☐

Amount of characters

Amount: 10

Use default (stored) values

Default Value: ✓

Personalities

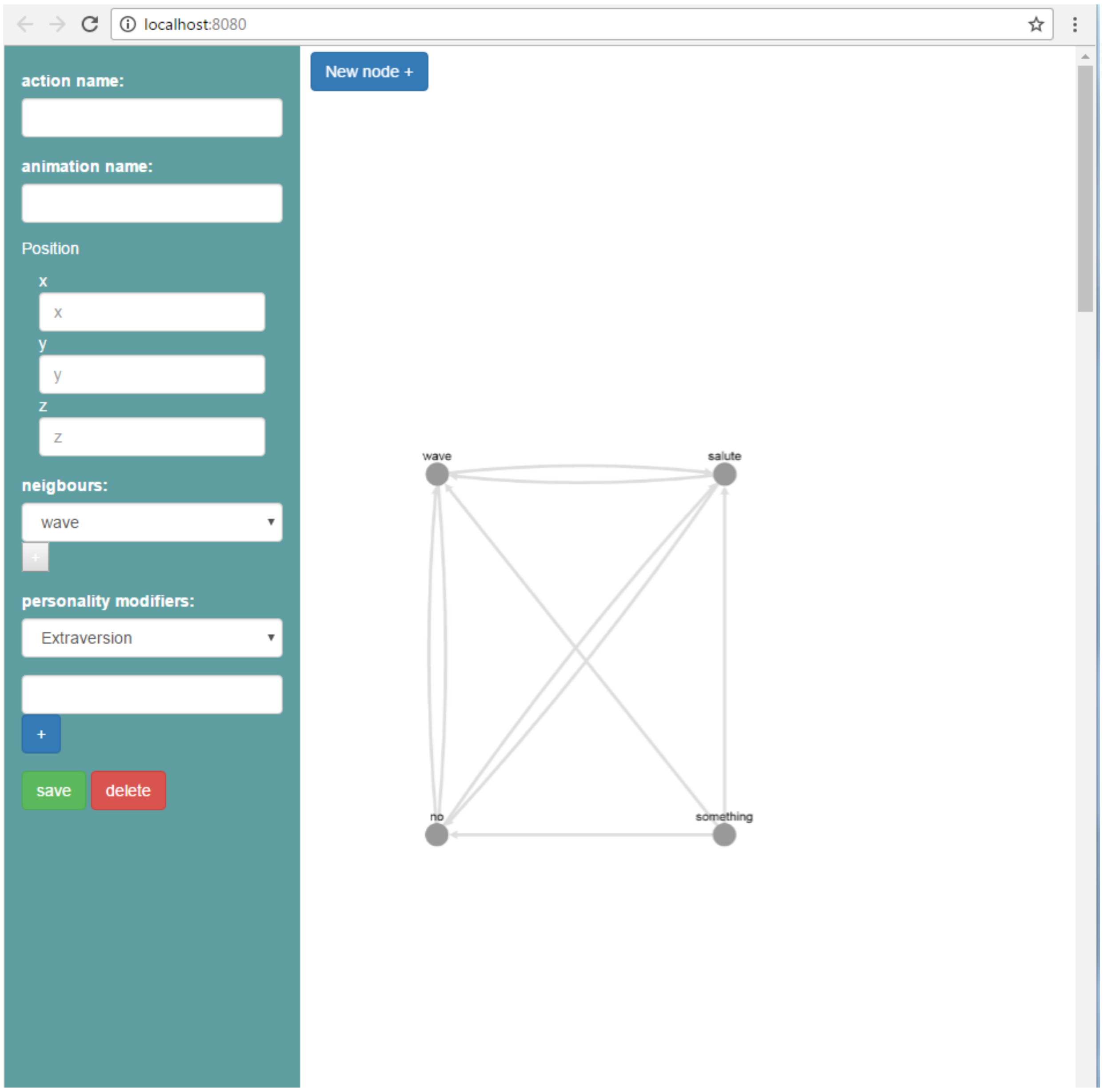
▶ Personalities

5.3 Creating custom actions and importing animations

5.3.1 Creating actions using the editor

We created a simple actions editor in Python, which makes it very easy to create your own actions in a visual interface and to define their relationship to eachother.

You can find the executable at Assets/avatargenerator/grapheditor.



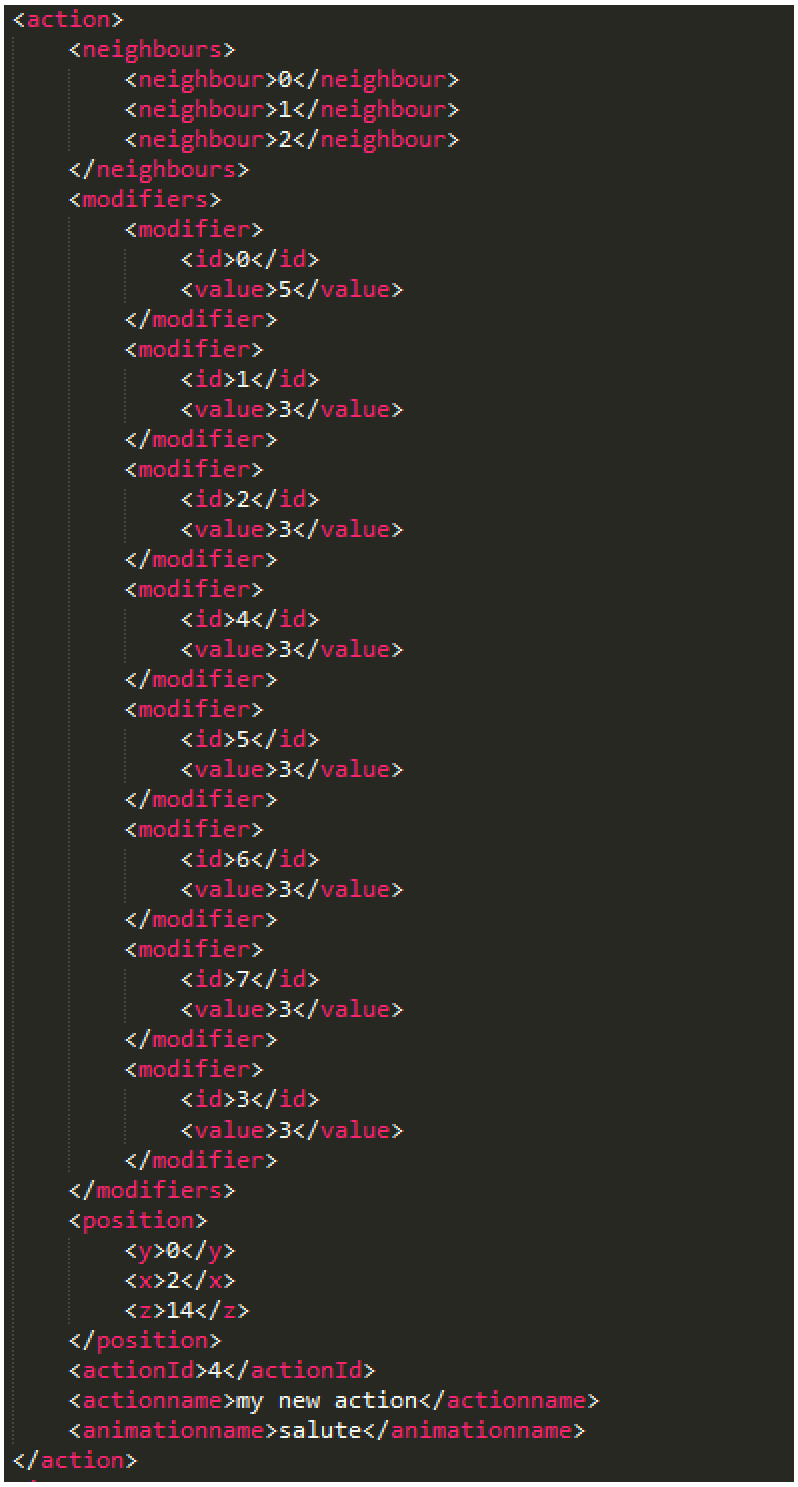
5.3.2 Creating actions manually

A new action is easily defined using actions.xml.

Each action contains the following values in xml:

1. The actions that an NPC can choose when he/she is done with the selected action
2. The personality values of this action: The more these match with the NPC's personality values, the more likely the NPC will choose to perform this action.
3. The position of the action.
4. The id of the action.
5. The name of the action, which is only used for debugging purposes and to make reading this XML file easier for you.
6. The name of the animation tied to this action.

An example:



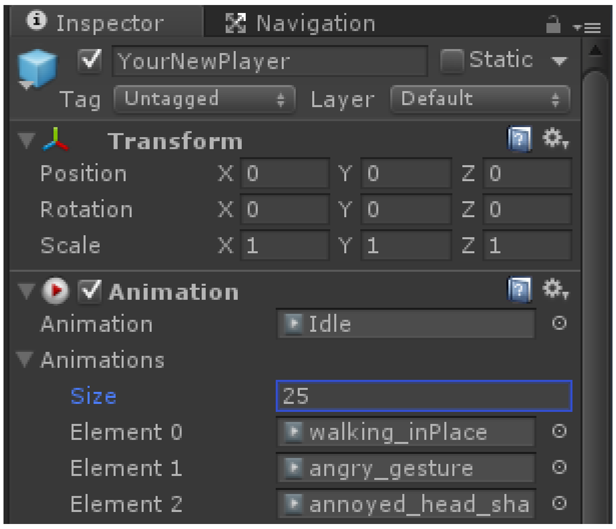
```xml
<action>
    <neighbours>
        <neighbour>0</neighbour>
        <neighbour>1</neighbour>
        <neighbour>2</neighbour>
    </neighbours>
    <modifiers>
        <modifier>
            <id>0</id>
            <value>5</value>
        </modifier>
        <modifier>
            <id>1</id>
            <value>3</value>
        </modifier>
        <modifier>
            <id>2</id>
            <value>3</value>
        </modifier>
        <modifier>
            <id>4</id>
            <value>3</value>
        </modifier>
        <modifier>
            <id>5</id>
            <value>3</value>
        </modifier>
        <modifier>
            <id>6</id>
            <value>3</value>
        </modifier>
        <modifier>
            <id>7</id>
            <value>3</value>
        </modifier>
        <modifier>
            <id>3</id>
            <value>3</value>
        </modifier>
    </modifiers>
    <position>
        <y>0</y>
        <x>2</x>
        <z>14</z>
    </position>
    <actionId>4</actionId>
    <actionname>my new action</actionname>
    <animationname>salute</animationname>
</action>
```

5.3.3 Importing animations for actions

Of course, you can add your own animations to use in your custom actions. There are some prerequisites however;

1. Every animation must be in the .FBX format
2. Every animation has to have its' rig-¿Animation Type set to Legacy.
3. The animation has to be added to in XML to the action that you wish to associate with it, as explained on the previous 2 pages
4. Last but not least, the animation has to be added to each NPC's list of possible animations. (The screenshot below shows where this can be done since it is not very obviously visible in the Unity Editor)

Simply expand the animations tab, increment the animations number and select your animation in the new slot. Make sure your animations' name and the animation name that is used in your action in xml are exactly the same!


Inspector
Navigation
YourNewPlayer
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Animation
Animation Idle
Animations
Size 25
Element 0 walking_inPlace
Element 1 angry_gesture
Element 2 annoyed_head_sha