ExtremeAI Quick Start Guide

Unity Setup

It's easy to set up the ExtremeAI engine! There are two ways, depending on how you downloaded the engine--through the asset store or as a zip file.

If you bought ExtremeAI in Unity's asset store, simply follow the import instructions there. The dlls and resources will be imported into their appropriate folders automatically. NOTE: The **TaiPE_Lib.dll** (located in Assets/ExtremeAI/Scripts) must be in the same folder as any scripts that use it. You should copy it into whichever folder you're using for your project's script files.

Make sure to read the notes after the "That's it!" paragraph below.

If you have the zip file:

1. First, unzip the ExtremeAI file (if you haven't already).

2. Next, copy the ExtremeAI folder into your Assets folder. The ExtremeAI folder contains three subfolders: Editor, Resources, and Scripts.

3. Finally, copy the contents of ExtremeAI's Scripts subfolder (**TaiPE_Lib.dll**) into whichever folder you're using for your project's script files.

That's it!

Actually, that's almost it ... you'll need to click on the menu bar in Unity to get the Tools/ExtremeAI menu option to show up for the first time.

Also, to successfully build a project using ExtremeAI, you'll need to change the PlayerSettings in Unity (Edit->Project Settings->Player, or click Player Settings from the Build Settings dialogue box) setting for API Compatibility Level (at the bottom, under Optimization) to .NET 2.0 (NOT the default ".NET 2.0 Subset"). The subset lacks the functionality necessary to run the engine after building.

And that's really it! You're ready to build personalities into your NPCs!

Create a Character

1. Click on Tools->ExtremeAI->Character Editor.
2. Type in a character name (such as "Buttons"), then click the "Create/Edit Character" button.
3. Edit the Stimuli/Response types to your heart's content. You have to click the checkbox next to the stimulus/response before you can edit it. Drag the slider to whatever value you wish.
4. There will be a slight pause while the engine recalculates the character's overall personality based on the changes you've made (every stimulus change alters the rest of the personality to keep the character "human-like").
5. Click "Save Character".
6. Click "Done! (Close, no save)" (don't worry, if you've already saved, your character will still exist).

You now have a personality for your NPC!

Connect the Mental to the Physical

Now let’s take that personality and attach it to a “physical” character!

1. Create a new script (for example, a C# script called “NPCInterface”). Remember to place it in the same folder as **TaiPE_Lib.dll**!
2. In the script, add “using TaiPE_Lib;” to the other “using …” statements at the top
3. Add “public AICharInfo myAIChar = new AICharInfo();” to your variables
4. Add “string myName = [your character’s name]” to your variables. For example, this could be string myName = “Buttons”;
5. In the Start method, initialize the engine by adding “myAIChar.Init (this.tag,myName);”
6. Save the script
7. Add the script to the GameObject representing your NPC. (Make sure the GameObject has a tag so the Init call above works!)

Ask Your NPC for an Opinion

Now that he/she can give you his/her opinions, let’s ask for one!

Let’s say we’ve created a guard NPC, and now the player’s about to encounter him. We want to know if he is going to try to be intimidating, so we check (but without changing his personality):

```
int am_I_Intimidating = myAIChar.AIReturnResult ("Player","intimidating",false);
```

Note that if you’re calling this from outside the NPC’s interface script (in this case, NPCInterface), you’ll need to have something like:

```
int am_I_Intimidating = GameObject.Find("NPCGameObject"). GetComponent
        <NPCInterface>().myAIChar.AIReturnResult ("Player","intimidating",false);
```

“Player” is the name of a registered player; see the User Manual for more on this. (The name “Player” comes preregistered with the engine, so you don’t need to run off and do this now.) “intimidating” is one of the Stimuli/Response Types listed in the Character Editor (note that these are case-sensitive and are all lower case). “false” tells the engine not to alter the NPC’s personality, just return a result.

This will return an integer from 0 to 4; 0 would be the opposite of intimidating, whereas 4 would be extremely intimidating. Also, this result is specific to this player; the guard may feel less intimidating toward someone else (say, someone he knows well and likes).

Just for fun, create the following Start method in your NPCInterface script:

```
void Start () {
        myAIChar.Init (this.tag,myName);
        int am_I_Intimidating = myAIChar.AIReturnResult ("Player","intimidating",false);

        Debug.Log ("value of am_I_intimidating: " + am_I_Intimidating);

        string well_am_i = "";

        switch(am_I_Intimidating) {
        case 0: //Nope! Just the opposite
            well_am_i = "I shiver even to think of it!";
            break;
        case 1: //Not so much
            well_am_i = "Not really";
            break;
        case 2: //Average
            well_am_i = "Sometimes, but sometimes not ...";
            break;
        case 3: //Yeah, kinda intimidating
            well_am_i = "Yeah, I am. What of it?";
            break;
        case 4: //You betcha!
            well_am_i = "Are you lookin' at me? Are you lookin' at me?!";
            break;
        }


        Debug.Log ("Do I feel intimidating? " + well_am_i);
    }
```

Now start your game. Depending on the values you chose when creating your NPC, he'll be more or less intimidating. Note that the result will not always be the same; even an average person might get his dander up sometimes.

And On …

So after the encounter, maybe the guard does indeed feel like being less intimidating toward this player; we can change this attitude by:

```
    myAIChar.AINoResult ("PlayerName","intimidating",false);
```

where the "false" means the attitude is made less strong.

Note that each encounter will also alter the NPC's attitude toward everyone (in addition to the specific player or NPC in the encounter), but to a lesser extent. If the guard begins to like enough people, he'll feel less like being intimidating overall, but just liking one person is unlikely to be that life-changing.

See the User Manual for complete usage of these methods.