ExtremeAI User Manual

Unity-specific version
rev 1.4, April 2015

Table of Contents

Introduction

Welcome to ExtremeAI, Unity edition! ExtremeAI provides a simple but complete add-on to create and change non-player personalities both before and during a game. With such power, your characters will react more realistically and will change their opinions of players depending on how they are treated. You can create different personalities for as many NPCs as you wish, and each will develop independently of all the others, with potentially different reactions depending on interactions with a player, with other players and NPCs, and even with world events.

Through a set of menus in the Unity editor, you can easily create and edit character personalities and assign them to NPCs. And through scripting, you can access these personalities in-game and have your NPCs react, grow, and change throughout the game’s run, giving every play-through a different feel, giving your players that much more incentive to return to your game and play it again, wondering what would happen if they had treated those soldiers, or shopkeepers, or henchmen a little differently …

Please feel free to contact us at Quantum Tiger Games with questions, or even just to share your experiences using ExtremeAI, at *support@quantumtigergames.com*.

Thank you!
Jeff Georgeson
Quantum Tiger Games, LLC

1.0 Installation and Overview

1.1 System requirements

The Unity-specific version of the ExtremeAI engine works in Unity 4.x (there is also a Windows exe version available). The disk space, memory, etc. requirements are minimal.

1.2 Initial setup for Unity

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** (or **TaiPE_Lib_Light.dll**, if you bought the light version; both are 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 a zip file, first, unzip the ExtremeAI file (if you haven't already; if you're reading this manual, you probably have, come to think of it ...).

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

Finally, copy the contents of ExtremeAI's Scripts subfolder (**TaiPE_Lib.dll** or **TaiPE_Lib_Light.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 read some of the dlls used after compiling.

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

2.0 Using the Engine

2.1 Setting up and editing characters

The various submenus of the ExtremeAI menu (found in Tools->ExtremeAI) allow you to create and edit the base personalities for your NPCs. This is all pregame stuff; the characters will begin your game in whatever configuration you choose here.

2.1.2 Background

You can create personalities for as few or as many NPCs as you wish; there is no requirement that every NPC in your game use the ExtremeAI engine. We have endeavored to make the creation process as simple as possible, however, giving you the option of using preset base personalities or tweaking as many as 37 different aspects of your characters' personalities. (Note that in ExtremeAI Light, you can tweak only 9 aspects.)

These aspects are the same stimuli/responses you will be able to choose from your game code. For instance, you can adjust the initial intensity of an NPC's "kind"ness—that is, his likelihood of responding to kindness with kindness, or to perform a kind act—using the character editor. Note that you are not directly changing the underlying facets of the NPC's personality (there is no "kindness" facet, for instance), but are affecting the underlying facets that go into making us more prone to kindness. Because of this, when you change kind, you affect the other stimuli/response types that have similar facets as components.

Also note that the numerical values you see are for your guidance only; the underlying personality facets are the core of the engine, and are ultimately used (along with our personality algorithms) to choose/change levels of response in the actual game situation.

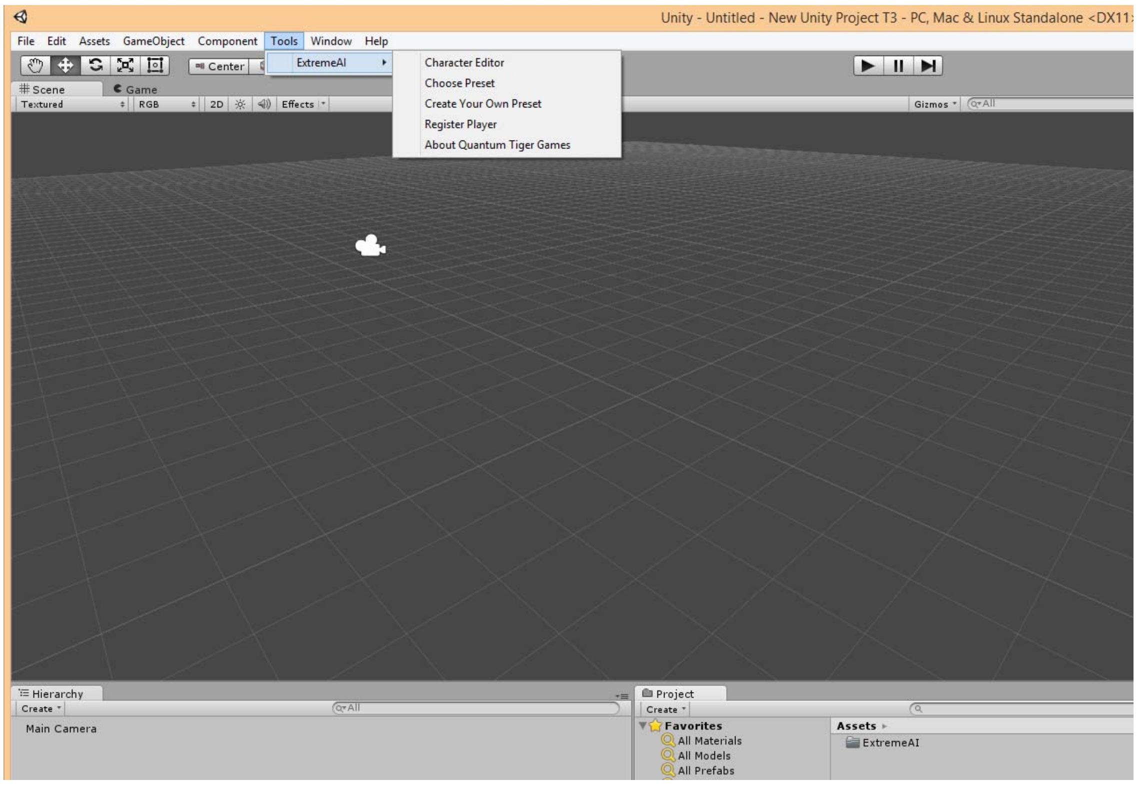


Figure 1 Unity editor screen, including ExtremeAI menu.

2.1.3 What are these “stimulus/response types”?

Personality theory in general posits that there are certain base elements that go into creating a person’s personality, and that these factor in various ways into the sorts of decisions and actions we make in our lives. In the Five Factor Model, a well-respected theory (see, for instance, Goldberg, 1993; Costa & McCrae, 1995; John et al, 2010; McCrae & Costa, 2010; DeYoung, 2010*), these elements are the 30 personality facets listed in Table 1, which are grouped into five overall categories (hence the ‘Five’ in the name of the model).

Rather than having the developer try to check the individual personality facets and figure out how these might apply to various everyday stimuli (such as another person being nice to you) or response types (like whether an NPC will go up to a stranger and be nice without provocation), the engine provides 37 possible stimuli/responses (see Table 2) that have already been keyed into the various facets (using our own extensive research and applying this to game-type situations). Thus the developer merely has to query whether a stimulus will elicit a strong reaction, and the engine will take care of not only finding that answer but figuring out how much (if at all) having that interaction affects the personality of the NPC in future interactions, either with the same player or with other players (or NPCs or, even, world events). Thus, just as in real life, the very act of interacting with others can have an effect on how a character deals with the future, and on how similar stimuli are dealt with.

Also as in real life, having high tendencies toward a certain reaction does not mean the character will react in the same exact way each and every time; our algorithms take this into account and return a reaction intensity from which you can make your own judgment as to what the NPC says or does.

2.1.4 ExtremeAI menu

See again Figure 1. Pretty self-explanatory. From here you choose:

Openness	**Extraversion**	**Neuroticism**
Fantasy	Warmth	Anxiety
Aesthetics	Gregariousness	Angry Hostility
Feelings	Assertiveness	Depression
Actions	Activity	Self-Consciousness
Ideas	Excitement Seeking	Impulsiveness
Values	Positive Emotions	Vulnerability
Conscientiousness	**Agreeableness**	
Competence	Trust	
Order	Straightforwardness	
Dutifulness	Altruism	
Achievement Striving	Compliance	
Self-Discipline	Modesty	
Deliberation	Tender-Mindedness	

Table 1 The Five Categories and Their Facets (categories in bold)

sadness	deceitfulness	assertiveness
happiness	condescension	humour
anger	quarrelsome	intellectual
sternness	helpfulness	demanding
jealousy	selfishness	productive
anxiety	affectionate	ambitious
impulsive	dependability	orderly
stubborn	efficiency	kind
guilt	moodiness	annoying
standoffish	wittiness	intimidating
reluctance	excitability	gregarious
conformity	imaginative	
distrustful	talkative	

Table 2 Possible Stimuli/Response Types (to be queried by the developer)

Character Editor to create or edit an existing character using the full set of possible stimuli/response types—gives you access to the 37 different stimuli/response types (or 9 in the Light version). Use this to fine-tune a character.

Choose Preset shows you a list of preset personality types—allowing you to quickly make a "type" of character without having to adjust anything. Note that you can, however, later edit these characters using the full set of stimuli/response types if you wish.

Create Your Own Preset—find yourself making a lot of similar characters, but none of our presets quite fits? This lets you create up to four of your own presets. (Not available in the Light version.)

Register Player—In order for your NPCs to keep track of their attitudes and feelings toward your player(s), they have to know how to reference him/her/them. You will need to use this menu to register the player(s). Note that any characters you create with the editor or presets will automatically be registered for you, so that they can interact with each other.

About Quantum Tiger Games—gives us the opportunity to talk about ourselves.

Ultimately, of course, the extent to which you consult the personality engine is up to you; if you want a certain NPC to follow a very prescribed set of actions, that's fine; it does not interfere with the Engine's functioning to do this (although it won't be able to develop that NPC's personality, obviously). If an NPC has to give out a piece of important information, the Engine will not forbid it (how could it?); what you might do, however, is have the NPC's phrasing be different depending on how she feels about this player. Even subtle differences like this can make the gameplay even more interesting than it already is, can add a bit of mystery and intrigue and depth (especially depth) to the world and the characters and people in it, and ultimately lead to a more satisfying experience for the (presumably human) players playing your game.

* The full references can be found at the back of this manual, for those interested.

Updates and Registration—gives you the opportunity to register Extreme AI, which gives you quick access to updates and other info.

Personality Info Tool—only available in the full version of Extreme AI. Allows you to see the underlying facet values for each character, rather than just the stimulus/response types.

2.1.5 Creating/editing characters

There are two ways to create a character: fine-tuning a character in detail, or using a preset to make a quick general type. For example, you could tailor a specific NPC to be more kind, intellectual, and imaginative (which will make that character more prone to wittiness and slightly less likely to anger as well, due to the underlying facets). You could also try the Professor preset, which will give you a basically intellectual character, but not one who is necessarily kind or imaginative. Characters created from a preset can be fine-tuned later using the character editor.

Following are detailed instructions for creating characters.

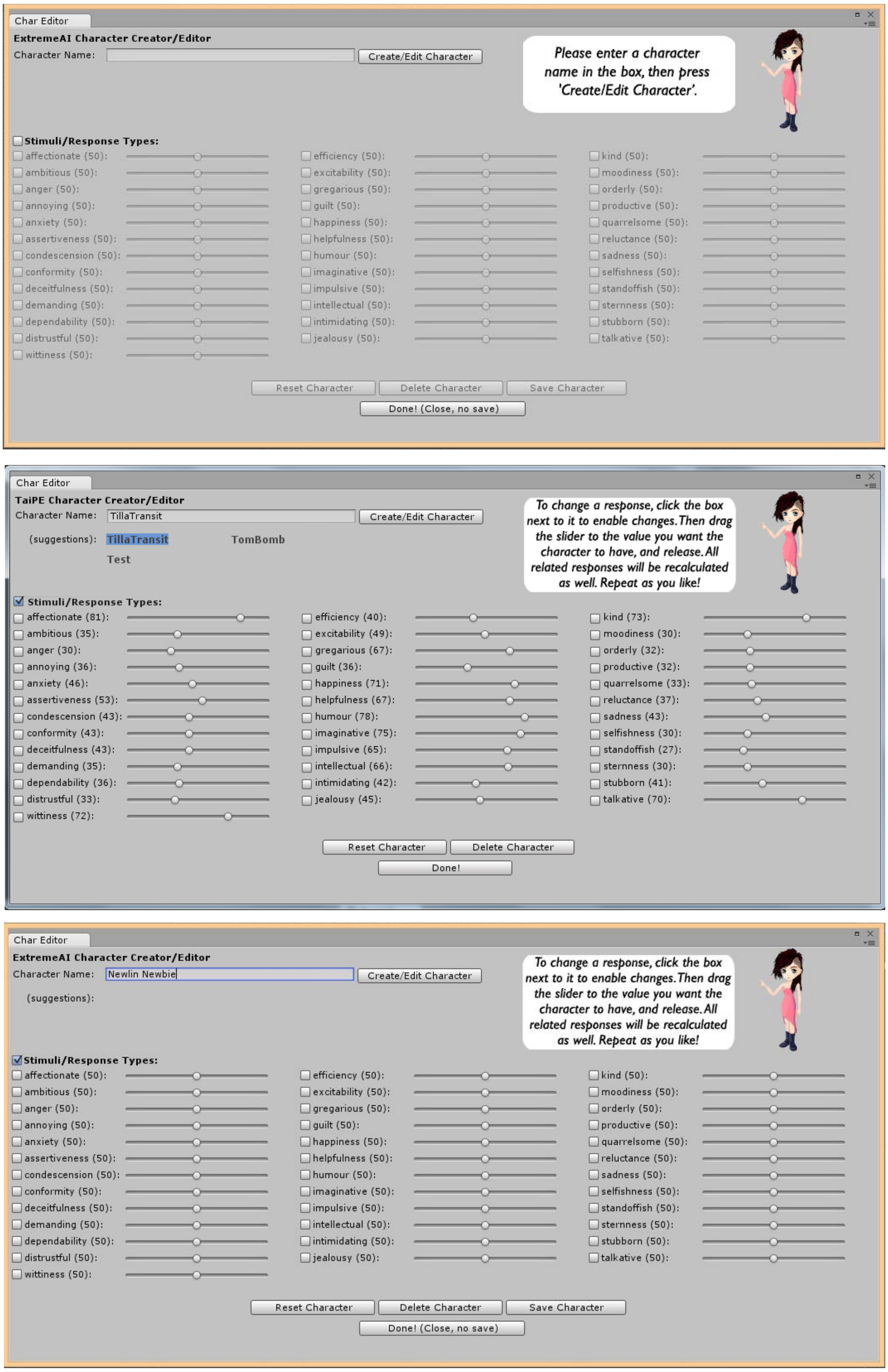
Character Editor: Create a character in detail

If you click the "Character Editor" menu, a window opens containing a text field for name entry and a set of sliders (see Figure 2, top). Typing in a name will provide a sort-of drop-down list of existing characters; you can double-click on one of these to edit him or her, or you can continue typing in the name of a new character (the drop-down also helps prevent creating two characters with the same name).

Once you have typed/double-clicked on a name, press the Create/Edit Character button. This will activate the sliders, each set to a value between 0 (the character has an extremely low propensity for this) and 100 (the character reacts in this way very intensely) (see Figure 2, middle or bottom). Zero kindness would indicate a very unkind person; 100 would be someone who was almost saintly. If the character already exists, its current values from her/his facets will be calculated (Figure 2, middle); a new character will start out with all 50s (Figure 2, bottom).

Who is that character to the side? To the right of the various ExtremeAI screens is Raina (super-deformed version), QTG's spokescharacter (she hopes to be like Sonic or Mario one day—hey, aim high, right?). She offers advice about using the ExtremeAI engine, for those times when you don't want to drag out the user manual.

To adjust the values, click the toggle box next to a response type, then move the slider up or down. When you release the slider, the character's facets will be refigured to match the new response value, and then all his/her responses will be recalculated to match the new facets. For example, changing The New Guy's Affectionate score from 50 to 70 also increases his Kindness (to 57) but decreases his Anger (tendency to become angry) somewhat (to 46), along with changing several other values slightly. This is because the psychological factors underlying the Kindness response (the facets) were changed when you adjusted Kindness, and thus every other response based on any of the same factors


Char Editor

ExtremeAI Character Creator/Editor

Character Name: [Create/Edit Character]

Please enter a character name in the box, then press 'Create/Edit Character'.

☐ **Stimuli/Response Types:**

☐ affectionate (50):	☐ efficiency (50):	☐ kind (50):
☐ ambitious (50):	☐ excitability (50):	☐ moodiness (50):
☐ anger (50):	☐ gregarious (50):	☐ orderly (50):
☐ annoying (50):	☐ guilt (50):	☐ productive (50):
☐ anxiety (50):	☐ happiness (50):	☐ quarrelsome (50):
☐ assertiveness (50):	☐ helpfulness (50):	☐ reluctance (50):
☐ condescension (50):	☐ humour (50):	☐ sadness (50):
☐ conformity (50):	☐ imaginative (50):	☐ selfishness (50):
☐ deceitfulness (50):	☐ impulsive (50):	☐ standoffish (50):
☐ demanding (50):	☐ intellectual (50):	☐ sternness (50):
☐ dependability (50):	☐ intimidating (50):	☐ stubborn (50):
☐ distrustful (50):	☐ jealousy (50):	☐ talkative (50):
☐ wittiness (50):		

[Reset Character] [Delete Character] [Save Character]
[Done! (Close, no save)]

Char Editor

TaiPE Character Creator/Editor

Character Name: TillaTransit [Create/Edit Character]

(suggestions): TillaTransit TomBomb
Test

To change a response, click the box next to it to enable changes. Then drag the slider to the value you want the character to have, and release. All related responses will be recalculated as well. Repeat as you like!

✓ **Stimuli/Response Types:**

☐ affectionate (81):	☐ efficiency (40):	☐ kind (73):
☐ ambitious (35):	☐ excitability (49):	☐ moodiness (30):
☐ anger (30):	☐ gregarious (67):	☐ orderly (32):
☐ annoying (36):	☐ guilt (36):	☐ productive (32):
☐ anxiety (46):	☐ happiness (71):	☐ quarrelsome (33):
☐ assertiveness (53):	☐ helpfulness (67):	☐ reluctance (37):
☐ condescension (43):	☐ humour (78):	☐ sadness (43):
☐ conformity (43):	☐ imaginative (75):	☐ selfishness (30):
☐ deceitfulness (43):	☐ impulsive (65):	☐ standoffish (27):
☐ demanding (35):	☐ intellectual (66):	☐ sternness (30):
☐ dependability (36):	☐ intimidating (42):	☐ stubborn (41):
☐ distrustful (33):	☐ jealousy (45):	☐ talkative (70):
☐ wittiness (72):		

[Reset Character] [Delete Character]
[Done!]

Char Editor

ExtremeAI Character Creator/Editor

Character Name: Newlin Newbie [Create/Edit Character]

(suggestions):

To change a response, click the box next to it to enable changes. Then drag the slider to the value you want the character to have, and release. All related responses will be recalculated as well. Repeat as you like!

✓ **Stimuli/Response Types:**

☐ affectionate (50):	☐ efficiency (50):	☐ kind (50):
☐ ambitious (50):	☐ excitability (50):	☐ moodiness (50):
☐ anger (50):	☐ gregarious (50):	☐ orderly (50):
☐ annoying (50):	☐ guilt (50):	☐ productive (50):
☐ anxiety (50):	☐ happiness (50):	☐ quarrelsome (50):
☐ assertiveness (50):	☐ helpfulness (50):	☐ reluctance (50):
☐ condescension (50):	☐ humour (50):	☐ sadness (50):
☐ conformity (50):	☐ imaginative (50):	☐ selfishness (50):
☐ deceitfulness (50):	☐ impulsive (50):	☐ standoffish (50):
☐ demanding (50):	☐ intellectual (50):	☐ sternness (50):
☐ dependability (50):	☐ intimidating (50):	☐ stubborn (50):
☐ distrustful (50):	☐ jealousy (50):	☐ talkative (50):
☐ wittiness (50):		

[Reset Character] [Delete Character] [Save Character]
[Done! (Close, no save)]

Figure 2 Create a detailed character. (top) Enter a character name, then (middle, character exists, or bottom, new character) edit her or his details.

changed as well (see Figure 3).

When you are finished adjusting the values, click the Save Character button to save your edits. And if you're unhappy with your changes and would like to come back to this character later (or never), click Done. This closes the editor window without saving. You can also click Done without ever entering a character name; this closes the window without creating or editing anything.

NOTE: These settings create a base personality, applied initially to every interaction the character has with anyone and anything in the game world. As the character interacts with other individuals, her reactions toward those individuals will vary depending on what those interactions have been; in other words, she may grow to despise a player who has been continually nasty to her, yet love someone who has been nothing but kind and giving. Additionally, an NPC's base responses (to strangers, say) will be affected by all of her interactions with others and the world; an NPC who has been treated badly by almost everyone will grow to be cynical and distrustful, even if he started out a kind and genial soul.

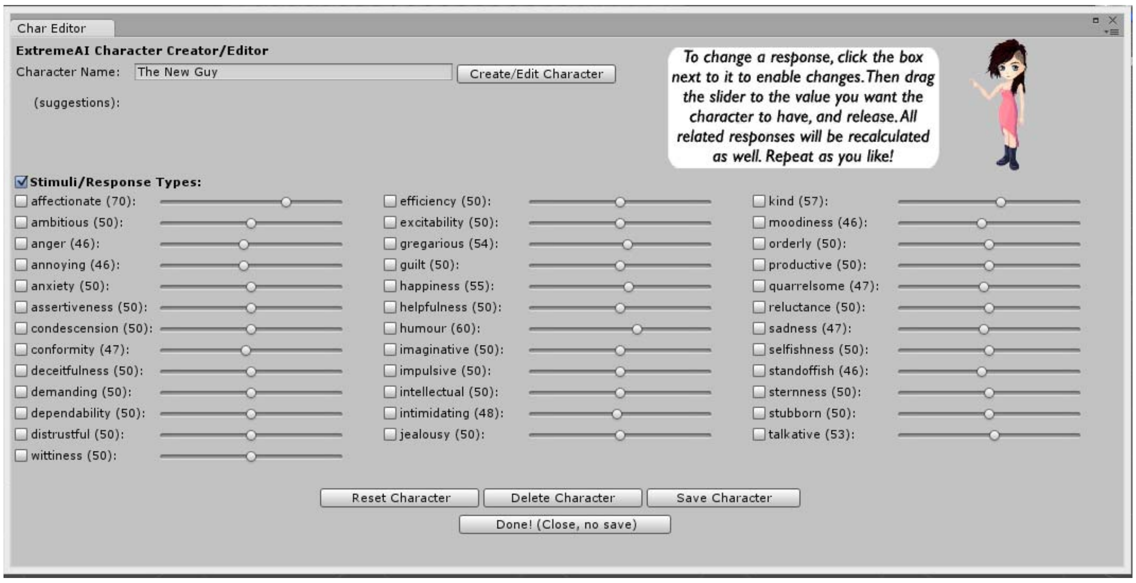
And what do the other buttons on this screen do?

Reset Character—The Reset button resets a character's slider values to those he/she had when imported (or created, if new). New characters will revert to all average values.

Delete Character—Deletes the current character. Note that this is not reversible!

Choose Preset: Create a character using presets

Clicking ExtremeAI -> Choose Preset allows you to create a character quickly, without having to adjust slider values. There are 28 different presets (14 in the Light version), some of which are subtypes of the same general personality.



Char Editor

ExtremeAI Character Creator/Editor

Character Name: The New Guy

(suggestions):

To change a response, click the box next to it to enable changes. Then drag the slider to the value you want the character to have, and release. All related responses will be recalculated as well. Repeat as you like!

✓ **Stimuli/Response Types:**

affectionate (70):	efficiency (50):	kind (57):
ambitious (50):	excitability (50):	moodiness (46):
anger (46):	gregarious (54):	orderly (50):
annoying (46):	guilt (50):	productive (50):
anxiety (50):	happiness (55):	quarrelsome (47):
assertiveness (50):	helpfulness (50):	reluctance (50):
condescension (50):	humour (60):	sadness (47):
conformity (47):	imaginative (50):	selfishness (50):
deceitfulness (50):	impulsive (50):	standoffish (46):
demanding (50):	intellectual (50):	sternness (50):
dependability (50):	intimidating (48):	stubborn (50):
distrustful (50):	jealousy (50):	talkative (53):
wittiness (50):		

Figure 3 The New Guy with adjusted stimuli/response values

First, type a name for the character in the Character Name box (see Figure 4); this is the same as in the Create/Edit Character window: Typing in a name will provide a sort-of drop-down list of existing characters; you can double-click on one of these to edit him or her, or you can continue typing in the name of a new character. Click “Begin Preset Character” when ready.

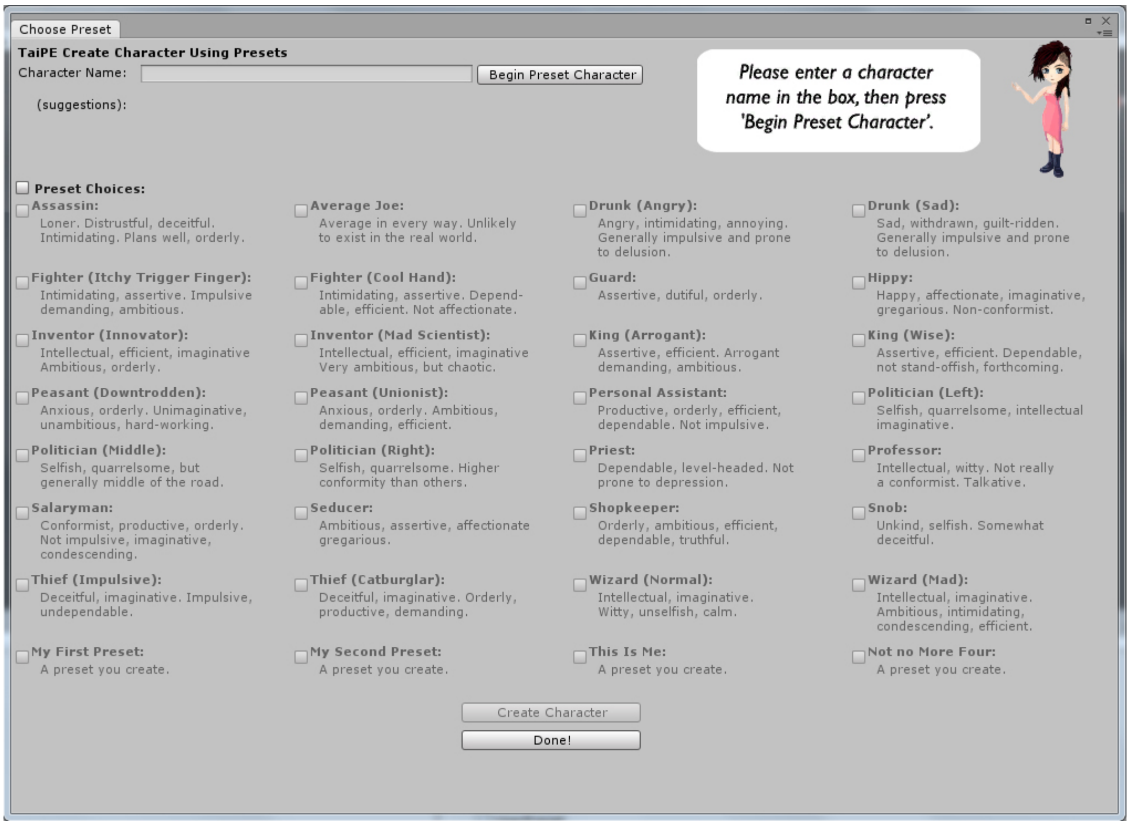
After this the Preset Character options will become available (see Figure 5). Click the check box next to the desired preset, then click “Create Character”. This creates the character.

At this point you can type in another name to create another preset character, or you can press “Done” to close the window. Pressing “Done” before pressing “Create Character” closes the window and does NOT create the preset character.

The last row of preset choices is all presets you can create and edit yourself, as covered next. (These are not available in the Light version.)

Create Your Own Preset: Your personal army of preset personalities
If none of the existing presets works for you, you can create up to four of your own (full version only). You do this by selecting Create Your Own Preset from the ExtremeAI menu.

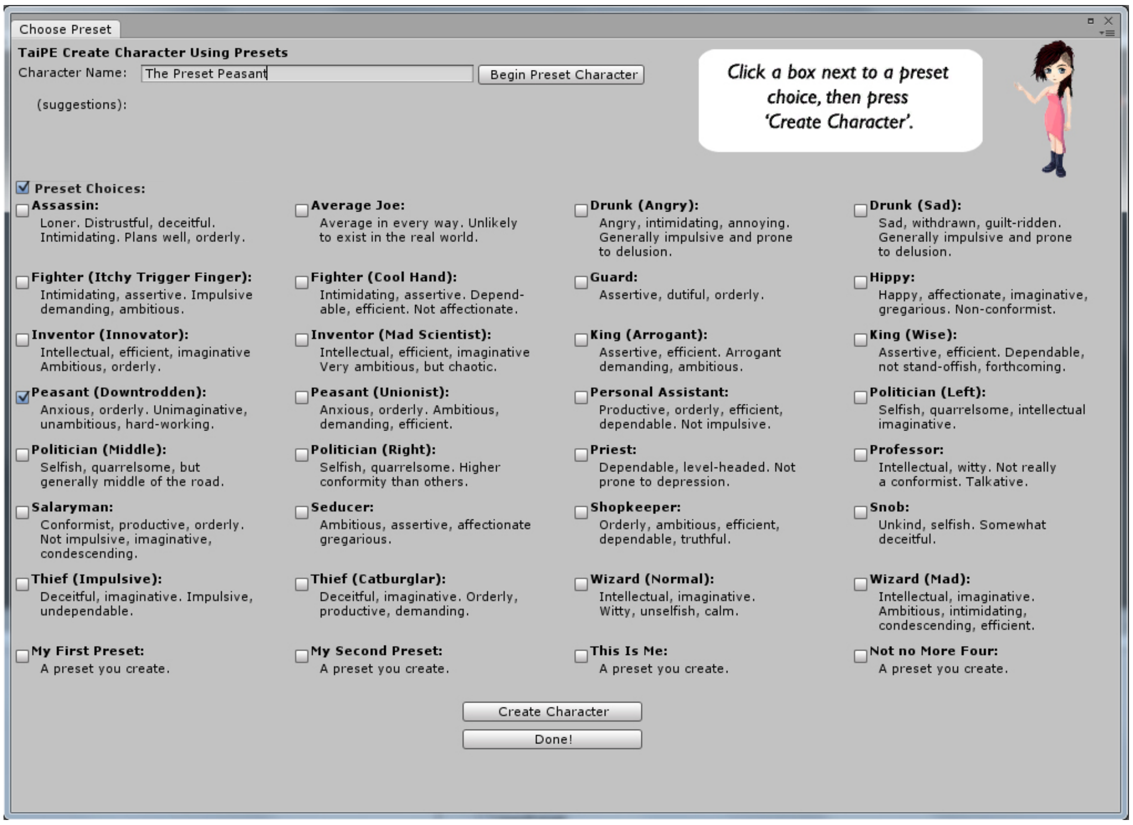
In the window that comes up, you will be given four options from which to choose (see Figure 6); these are the current user-created presets (or the defaults if you haven’t creat-



Please enter a character name in the box, then press 'Begin Preset Character'.

Assassin: Loner. Distrustful, deceitful. Intimidating. Plans well, orderly.	**Average Joe:** Average in every way. Unlikely to exist in the real world.	**Drunk (Angry):** Angry, intimidating, annoying. Generally impulsive and prone to delusion.	**Drunk (Sad):** Sad, withdrawn, guilt-ridden. Generally impulsive and prone to delusion.
Fighter (Itchy Trigger Finger): Intimidating, assertive. Impulsive demanding, ambitious.	**Fighter (Cool Hand):** Intimidating, assertive. Dependable, efficient. Not affectionate.	**Guard:** Assertive, dutiful, orderly.	**Hippy:** Happy, affectionate, imaginative, gregarious. Non-conformist.
Inventor (Innovator): Intellectual, efficient, imaginative Ambitious, orderly.	**Inventor (Mad Scientist):** Intellectual, efficient, imaginative Very ambitious, but chaotic.	**King (Arrogant):** Assertive, efficient. Arrogant demanding, ambitious.	**King (Wise):** Assertive, efficient. Dependable, not stand-offish, forthcoming.
Peasant (Downtrodden): Anxious, orderly. Unimaginative, unambitious, hard-working.	**Peasant (Unionist):** Anxious, orderly. Ambitious, demanding, efficient.	**Personal Assistant:** Productive, orderly, efficient, dependable. Not impulsive.	**Politician (Left):** Selfish, quarrelsome, intellectual imaginative.
Politician (Middle): Selfish, quarrelsome, but generally middle of the road.	**Politician (Right):** Selfish, quarrelsome. Higher conformity than others.	**Priest:** Dependable, level-headed. Not prone to depression.	**Professor:** Intellectual, witty. Not really a conformist. Talkative.
Salaryman: Conformist, productive, orderly. Not impulsive, imaginative, condescending.	**Seducer:** Ambitious, assertive, affectionate gregarious.	**Shopkeeper:** Orderly, ambitious, efficient, dependable, truthful.	**Snob:** Unkind, selfish. Somewhat deceitful.
Thief (Impulsive): Deceitful, imaginative. Impulsive, undependable.	**Thief (Catburglar):** Deceitful, imaginative. Orderly, productive, demanding.	**Wizard (Normal):** Intellectual, imaginative. Witty, unselfish, calm.	**Wizard (Mad):** Intellectual, imaginative. Ambitious, intimidating, condescending, efficient.
My First Preset: A preset you create.	**My Second Preset:** A preset you create.	**This Is Me:** A preset you create.	**Not no More Four:** A preset you create.

Figure 4 Create a character using presets.


Choose Preset
Begin Preset Character
Click a box next to a preset choice, then press 'Create Character'.
Create Character
Done!

TaiPE Create Character Using Presets

Character Name: The Preset Peasant

(suggestions):

[✓ **Preset Choices:**]

Assassin: Loner. Distrustful, deceitful. Intimidating. Plans well, orderly.	**Average Joe:** Average in every way. Unlikely to exist in the real world.	**Drunk (Angry):** Angry, intimidating, annoying. Generally impulsive and prone to delusion.	**Drunk (Sad):** Sad, withdrawn, guilt-ridden. Generally impulsive and prone to delusion.
Fighter (Itchy Trigger Finger): Intimidating, assertive. Impulsive demanding, ambitious.	**Fighter (Cool Hand):** Intimidating, assertive. Depend-able, efficient. Not affectionate.	**Guard:** Assertive, dutiful, orderly.	**Hippy:** Happy, affectionate, imaginative, gregarious. Non-conformist.
Inventor (Innovator): Intellectual, efficient, imaginative Ambitious, orderly.	**Inventor (Mad Scientist):** Intellectual, efficient, imaginative Very ambitious, but chaotic.	**King (Arrogant):** Assertive, efficient. Arrogant demanding, ambitious.	**King (Wise):** Assertive, efficient. Dependable, not stand-offish, forthcoming.
[✓ **Peasant (Downtrodden):**] Anxious, orderly. Unimaginative, unambitious, hard-working.	**Peasant (Unionist):** Anxious, orderly. Ambitious, demanding, efficient.	**Personal Assistant:** Productive, orderly, efficient, dependable. Not impulsive.	**Politician (Left):** Selfish, quarrelsome, intellectual imaginative.
Politician (Middle): Selfish, quarrelsome, but generally middle of the road.	**Politician (Right):** Selfish, quarrelsome. Higher conformity than others.	**Priest:** Dependable, level-headed. Not prone to depression.	**Professor:** Intellectual, witty. Not really a conformist. Talkative.
Salaryman: Conformist, productive, orderly. Not impulsive, imaginative, condescending.	**Seducer:** Ambitious, assertive, affectionate gregarious.	**Shopkeeper:** Orderly, ambitious, efficient, dependable, truthful.	**Snob:** Unkind, selfish. Somewhat deceitful.
Thief (Impulsive): Deceitful, imaginative. Impulsive, undependable.	**Thief (Catburglar):** Deceitful, imaginative. Orderly, productive, demanding.	**Wizard (Normal):** Intellectual, imaginative. Witty, unselfish, calm.	**Wizard (Mad):** Intellectual, imaginative. Ambitious, intimidating, condescending, efficient.
My First Preset: A preset you create.	**My Second Preset:** A preset you create.	**This Is Me:** A preset you create.	**Not no More Four:** A preset you create.

Figure 5 The Preset Choices

ed any). Double-click on one of these to select it for editing.

Once selected, the name of the preset will appear in the “New Name (if desired)” box and the sliders will become available. To rename the preset, type a new name in the box and click “Change Preset Name”. To change the preset’s stimulus/response values, click the toggle box next to a response type, then move the slider left or right. When you release the slider, the preset’s facets will be refigured to match the new response value, and then all its responses will be recalculated to match the new facets (just as when using the Character Create/Edit screen). The preset’s values are changed in the database live as you change them. (Note, however, that the preset’s name is changed only if you click the “Change Preset Name” button.) See Figure 7.

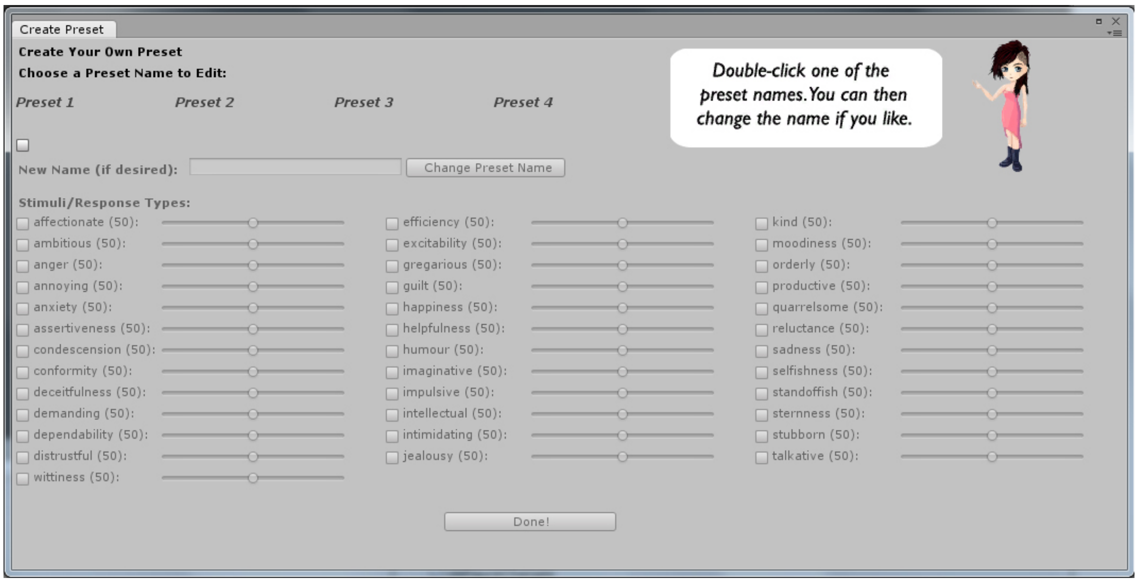
When you’re done editing the sliders, press Done! to exit the screen, or double-click another preset to edit.

You can now use your new preset to create NPCs!

2.1.6 Other ExtremeAI menu functions

Register Player

As mentioned before, in order for your NPCs to keep track of their attitudes and feelings toward your player(s), they have to know how to reference him/her/them. in this window you can type in the name of a player and press “Register Player”. That’s it! Note that you’ll receive a message telling you whether the player was registered successfully or already exists.



Create Preset

Create Your Own Preset

Choose a Preset Name to Edit:

Preset 1 *Preset 2* *Preset 3* *Preset 4*

Double-click one of the preset names. You can then change the name if you like.

☐

New Name (if desired): [Change Preset Name]

Stimuli/Response Types:

☐ affectionate (50):	☐ efficiency (50):	☐ kind (50):
☐ ambitious (50):	☐ excitability (50):	☐ moodiness (50):
☐ anger (50):	☐ gregarious (50):	☐ orderly (50):
☐ annoying (50):	☐ guilt (50):	☐ productive (50):
☐ anxiety (50):	☐ happiness (50):	☐ quarrelsome (50):
☐ assertiveness (50):	☐ helpfulness (50):	☐ reluctance (50):
☐ condescension (50):	☐ humour (50):	☐ sadness (50):
☐ conformity (50):	☐ imaginative (50):	☐ selfishness (50):
☐ deceitfulness (50):	☐ impulsive (50):	☐ standoffish (50):
☐ demanding (50):	☐ intellectual (50):	☐ sternness (50):
☐ dependability (50):	☐ intimidating (50):	☐ stubborn (50):
☐ distrustful (50):	☐ jealousy (50):	☐ talkative (50):
☐ wittiness (50):		

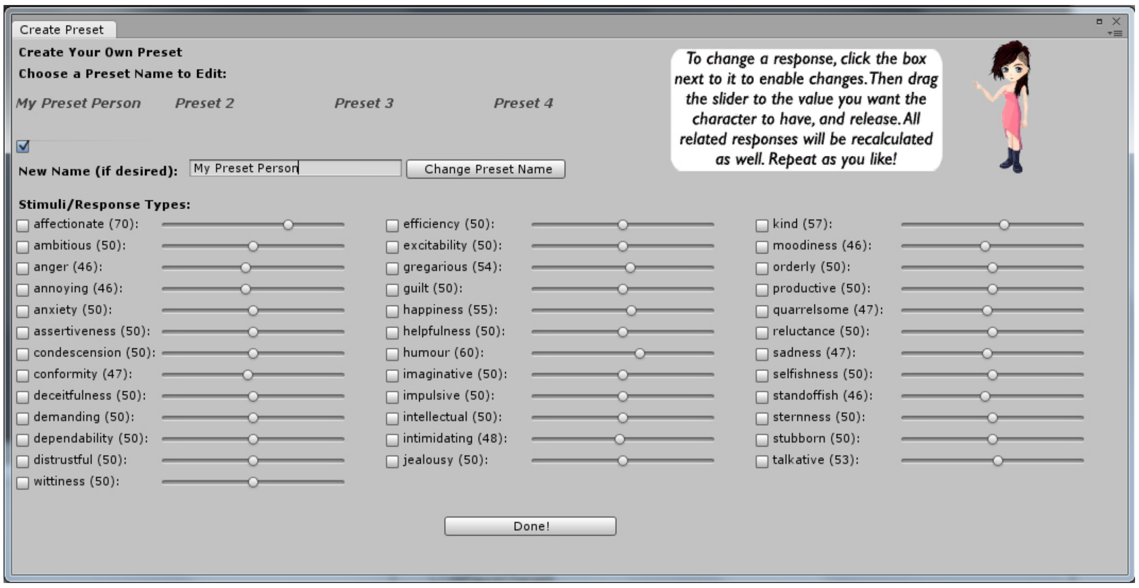
[Done!]

Figure 6 Double-click a preset name to edit

Remember that the player name you register is the one that must be referenced when checking/changing NPCs' atttudes toward that player, even if you have options in game for the human playing the player character to change names. For instance, if you've registered "Player1" as the player name, and in-game that name is changed to Joe the Barbarian, references in the personality engine would still be to "Player1".

Note that you can also set up non-player "players" with which to interact. For instance, you can create a way to have the NPC react to world events by creating a "world_events" player.

The generic name "Player" has been pre-registered for you. Also, any characters you create using the Character Editor or presets will automatically be registered for you, so that they can interact with each other as well.



Create Preset

Create Your Own Preset

Choose a Preset Name to Edit:

My Preset Person *Preset 2* *Preset 3* *Preset 4*

To change a response, click the box next to it to enable changes. Then drag the slider to the value you want the character to have, and release. All related responses will be recalculated as well. Repeat as you like!

☑

New Name (if desired): My Preset Person [Change Preset Name]

Stimuli/Response Types:

☐ affectionate (70):	☐ efficiency (50):	☐ kind (57):
☐ ambitious (50):	☐ excitability (50):	☐ moodiness (46):
☐ anger (46):	☐ gregarious (54):	☐ orderly (50):
☐ annoying (46):	☐ guilt (50):	☐ productive (50):
☐ anxiety (50):	☐ happiness (55):	☐ quarrelsome (47):
☐ assertiveness (50):	☐ helpfulness (50):	☐ reluctance (50):
☐ condescension (50):	☐ humour (60):	☐ sadness (47):
☐ conformity (47):	☐ imaginative (50):	☐ selfishness (50):
☐ deceitfulness (50):	☐ impulsive (50):	☐ standoffish (46):
☐ demanding (50):	☐ intellectual (50):	☐ sternness (50):
☐ dependability (50):	☐ intimidating (48):	☐ stubborn (50):
☐ distrustful (50):	☐ jealousy (50):	☐ talkative (53):
☐ wittiness (50):		

[Done!]

Figure 7 Change the name and adjust the sliders

About Quantum Tiger Games
Self-explanatory, really. Clicking this on the ExtremeAI menu gives you more info about the personality engine and Quantum Tiger Games (and shows Raina in her non-SD glory).

Updates and Registration
Also self-explanatory. You can register your copy of Extreme AI, giving you quick access to updates and other information. Also talks about us advertising any games you create using Extreme AI and (hopefully!) giving us a credit in your game.

Personality Info Tool
(Full version only.) Gives you the ability to see the underlying 30 facet values for a character, rather than just the stimulus/response values shown in the character editor. You type a character's name in the box, click the Check Character button, and the character's facet values will print in the Debug Log. Also displays any specific player adjustments for this character (typically none for newly created characters). Click Done to exit the window.

2.2 Using the character personalities in a game

Now that you have the base personalities for your NPCs, it's time to make them game-accessible. Make sure the TaiPE_Lib or TaiPE_Lib_Light dll has ben added to your Scripts folder, as per the installation instructions (see Section 1.2).

2.2.1 For each NPC …

For each NPC using the personality engine, attach a script creating its own instance of AICharInfo (the interface between the character and the engine). It also must know how to use the TaiPE_Lib.dll. For example, in C# you would:

1) place a "using TaiPE_Lib;" statement before the class definition in your code ("using TaiPE_Lib_Light;" in the Light version)
2) declare a variable of type AICharInfo (e.g., "AICharInfo myAICI;")
3) instantiate a copy of the module (e.g., "myAICI = new AICharInfo();")

Please see the example Unity code in part 3.

2.2.2 Naming the NPC

You should also set a variable equal to the NPC's name (matching a character whose personality you created when creating characters using the ExtremeAI menus). The NPC's name will be needed when you query the engine on his/her behalf. Note that this connection is all you need to associate personalities with NPCs.

For example, if you've created a personality for Edgar the Excellent in the database, then in scripts accessing Edgar's personality you'll need a string variable set to "Edgar the Excellent". The engine is case-sensitive.

2.2.3 What you can do

TaiPE_lib (or TaiPE_Lib_Light) contains the following user-accessible functions:

Init(string myTag, string myNameIs) [returns nothing]
ChangeBaseChange(float amount) [returns nothing]
ChangeBkgrndChange(float amount) [returns nothing]
AIReturnResult(string playerName, string stimulus, bool toChange = true, bool posChange = true) [returns int]
AINoResult(string playerName, string stimulus, bool posChange = true) [returns nothing]
SavePersonality() [returns nothing]

Advanced functions (available only in the full version of Extreme AI) include:
CharAttValueTool(ref int[] atrArray, ref string[] nmArray) [returns int]
CharAttSingleValueTool(string attributeName) [returns int]
CharFacetsTool(ref string[] charFacetNames, ref float[] charFacets) [returns nothing]
CharSingleFacetTool(string facetName) [returns float]
CharVsPlayerValuesTool(ref string[] playerNames, ref string[] facetNames, ref float[] facetValues) [returns int]
CharVsOnePlayerValTool(string playerName, string facetName) [returns float]

These are described in detail below (and again, see the examples in Part 3).

Init(string myTag, string myNameIs) [returns nothing]
This function initialises the AI module, and must be called before you try to use it. “myTag” is the Unity Tag value of this character in the Inspector (so that the GameObject that is the NPC could be located by, for example, GameObject.FindWithTag(myTag). “myNameIs” is the name that matches the NPC’s entry in the personality engine.

This function doesn’t return anything.

ChangeBaseChange(float amount) [returns nothing]
This function changes the base rate at which the NPC’s personality changes in relation to an individual player (or event type). The default is 5.0. This is a bit more quickly than people’s personalities change in real life, but to notice the changes in a game, wherein the player may only speak to an NPC a few times, this is the value chosen after testing various rates. Remember that changing this rate for one NPC does not change it for any other (so you can have characters whose personalities differ in terms of rate of change, just like humans).

ChangeBkgrndChange(float amount) [returns nothing]
This function changes the base rate at which an NPC’s personality changes in relation to EVERYTHING; i.e., if one player is particularly nice to the NPC, this rate determines the amount the NPC is more kindly disposed to everyone in the world as a result. The default for this rate is 0.1, which again is higher than the rate for humans in the real world, but is desirable for noticing the changes in-game. Changing this rate by even a fraction has a large effect; raising it as high as 1 would create a wildly changing personality.

0	Extremely low; could indicate a reaction opposite to that queried, or just an absolutely neutral response (e.g., a kindness intensity of 0 could mean the NPC just ignores the player, or perhaps reacts negatively—the actual reaction is that which suits the situation. Or, if another check indicates the NPC has a high anger intensity, perhaps the NPC reacts very negatively to the player’s kindness … it’s up to you how many checks you want to make for any given interaction)
1	Low; slightly negative response, or relatively neutral
2	Average; reacts as the “average” person would in this situation
3	High; reacts positively (note that having a high, positive reaction for one stimulus (e.g., kindness) may be very different from a high, positive response to another (e.g., anger)
4	Extremely high; intensely positive response

Again, changing this rate for one NPC does not change it for any other.

AIReturnResult(string playerName, string stimulus, bool toChange = true, bool posChange = true) [returns int]
This is one of the primary functions for the Personlaity Engine; it takes a stimulus/response type (e.g., kind) and returns an integer from 0 to 4 indicating the intensity of the response:

Why only five possible intensities? Because we had to choose between further shades of intensity and speedier response times from the engine. Five seemed the best compromise.

AIReturnResult can be used in different ways. The default takes the character name, player name, and stimulus/response type, and changes the personality of the NPC in regard to both this player and to everything else in the world (see also the discussion of the base change functions, above). Alternatively, you can call the function with an added boolean indicating that no changes are to be made to the personality (done by setting the toChange bool to false). You may want to do this when you are simply checking on an NPC’s reaction intensity but aren’t in a situation where the personality would be changed, such as checking whether an NPC would be the first to start a conversation (checking gregariousness); no real interaction with a player has yet occurred, so there isn’t a reason to change the NPC’s reactions. There also may be times when you don’t know at the outset in what way the NPC’s personality would be changed by an event; say you check her impulsiveness to see whether she’ll follow the player off the edge of a cliff. It would depend on the outcome of this action whether she later thought of it as a negative or positive reinforcer of her impusiveness. Which brings us to …

AINoResult(string playerName, string stimulus, bool posChange = true) [returns nothing]
So your NPC has jumped over a cliff and had an absolutely smashing time … in the best sense of the word. Her impulsiveness should be increased, but you don’t need any sort of value returned for a new response—the response and event already happened.

That’s when you use AINoResult. This function changes the NPC’s personality without returning a result. The default, taking the character name, player name, and stimulus, changes the values in a positive way, indicating a step towards more intenisty the next time the stimulus is checked.

What if our NPC had had a smashing time in the worst possible sense of the word? You could then change her impulsiveness to be less likely to rule her thoughts by calling AINoResult with an additional boolean value (posChange) set to false.

SavePersonality() [returns nothing]
While NPCs’ personalities change in-game with no need to save them, in order to keep changes after a player quits or closes you will need to call SavePersonality for each character using the engine. Because it can take a noticeable amount of time, we’d suggest only doing this at save points or other instances when the gameplay isn’t impacted.

Note that to be able to return to earlier instances of NPCs’ personality states, you will need to save a copy of all tables in the Application.persistentDataPath + “/ExtremeAI/Resources/Tables” folder, which is where files are saved during play (save your copies somewhere else, obviously). When reverting to a saved instance, you’ll need to move the copied tables back into the Application.persistentDataPath + “/ExtremeAI/Resources/Tables” folder, overwriting the tables that are already there.

Advanced methods available in full version only:
CharAttValueTool(ref int[] atrArray, ref string[] nmArray) [returns int]
This tool allows you to get an array (by reference) of all 37 raw stimulus/response attributes for a character. Actually, you get two arrays: one is the attribute values, one is the attribute names. Probably most useful in testing a game, so you can get a snapshot of how a character is changing. Takes a noticeable amount of time to return, so not a good idea to run this in the middle of a lot of game action. Note that these are raw values, not calculated personality-based responses (for which you’d use **AIReturnResult** or **AINoResult**).

CharAttSingleValueTool(string attributeName) [returns int]
This tool returns a single raw stimulus/reponse attribute value (as opposed to calculating a character’s actual personality-based response; for that you’d use **AIReturnResult** or **AINoResult**). You provide the attribute name, and the method returns the value of that attribute.

CharFacetsTool(ref string[] charFacetNames, ref float[] charFacets) [returns nothing]
This tool allows you to get an array (by reference) of all 30 raw facet values for a character (you also get a matching array of the facet names). This shows you the character’s personality at its most basic level. Again, this doesn’t provide a calculated personality-based response (for which you’d use **AIReturnResult** or **AINoResult**).

CharSingleFacetTool(string facetName) [returns float]
This tool returns a single facet value (any of the 30 underlying facets in the Five-Factor

model), and thus a snapshot of the most basic level of a character’s personality. Again, this doesn’t provide a calculated personality-based response (for which you’d use **AIReturnResult** or **AINoResult**).

CharVsPlayerValuesTool(ref string[] playerNames, ref string[] facetNames, ref float[] facetValues) [returns int]
Returns (by reference) all player/other NPC/world-based adjustments to a character’s underying facets, thus showing you how this character feels about specific individuals. Most useful in testing a game; it can take a noticeable amount of time to return (especially as the game goes on and an NPC has met a lot of other people and/or had a number of interactions with a few people), so not a good idea to run this in the middle of a lot of game action. These are obviously not calculated personality-based responses.

CharVsOnePlayerValTool(string playerName, string facetName) [returns float]
Returns a character’s adjustment value for a single facet toward a specific player/other NPC/etc. Let’s you know how much more/less a character feels, for instance, trusting of the player than he does a total stranger (or affectionate, or any of the other facets).

See Part 3 for examples of the basic functions in use, and take a look at the sample Unity scene for a demonstration. You can also visit ***quantumtigergames.com*** for another example (the voters demo).

3.0 Examples

Of course, just the cut-and-dry enumeration of function calls doesn't really demonstrate the possibilities, the range of character responses and development when using the ExtremeAI engine. Following are a few examples of using the engine in a Unity environment. (Note that the examples assume you are using C#.)

3.1 Remember to set the NPC and player names!

As mentioned before, it's a good idea to set a string variable to the name of the NPC whose personality you are accessing, as this name will be necessary every time you query the database. However, you can change an NPC's personality from any script, anywhere, as long as you use his/her name in the function call.

Also, the Engine must know which player (or thing) is interacting with the NPC in order to return values that make sense. Setting the player name is done by registering it in the ExtremeAI menu, as described in section 2.1.6.

3.2 Getting it all started: using Init

To use the engine, you have to create an instance of it for a character and then initialize it. This is easy to do: in a script you attach to a character, do the following (and see the boldface items in Figure 3.1):

1) make sure to include a "using TaiPE_Lib;" (or "using TaiPE_Lib_Light;" in the Light version) statement above the class declaration
2) when declaring variables for use in the class, instantiate a copy of AICharInfo
3) in the Start function provided by Unity (or some other function called when the script is first called), continue the instantiation (C# requires not only declaring it in the variables, but then setting it equal to a "new AICharInfo()")
4) also in the Start function, call the Init(myTag, myNameIs) function of AICharInfo to initialize the Engine for this character

That's it! Now you're ready to access the character's personality, which you created using the character setup menus described previously (see Section 2.1).

3.3 Changing speed of change

If you want to change the speed at which this character's personality changes (not recommended until after you've used the PE for a while to get a sense of how it works), now would be the time to do it—just after initializing the Engine, still within the Start function. (Not that this is required. The speed at which different facets change in real people's lives is already factored into our Engine, and so continually tinkering with the overall speed of change will make the NPC more unrealistic and possibility quite wacky.)

To do this (see Figure 3.2):
1) call the **ChangeBaseChange** function of your instance of the AIClientInterface (in the example we've called it "myAICI"), with a float representing the new value. This will change per interaction the attitude the NPC has toward specific individuals, rather

```
using UnityEngine;
using System.Collections;
using TaiPE_Lib;

public class NPCClientSide : MonoBehaviour {

	/*
		Author: Jeffrey Georgeson, build 13 April 2012
		Copyright Quantum Tiger Games, LLC 2012
	*/

	//who is the NPC we're dealing with in this script?
	string NPCName = "Edgar the Excellent";

	//instantiates copy of AICharInfo
	AICharInfo myAICI;
	//alternatively, could use AICharInfo myAICI = new AICharInfo();
	//which would take care of instantiation all at once

	...

// Use this for initialization
	void Start () {

		//continue instantiation of AI interface
		myAICI = new AICharInfo(); //unless already done through alternate
		//code above

		myAICI.Init(this.tag, NPCName);

		...
```

Figure 3.1 Initializing a copy of AICharInfo for a character

```
//use the following function to change the rate of change per interaction with each player.
//this represents the attitude the NPC has toward specific individuals, rather than the overall
//change (i.e., towards Mike the First Player, but not towards everyone and everything
//generally).
//note that the default is 5f

float someAmount = 5f;
myAICI.ChangeBaseChange(someAmount);


//use the following function to change the rate of change for every interaction the NPC has.
//this is like changing the rate an overall personality changes. In the real world, such changes
//usually take years; the default rate for gameplay is 0.1f (which is still faster than real-world
//changes, but necessary for most gameplay situations).

float someOtherAmount = 0.1f;
myAICI.ChangeBkgrndChange (someOtherAmount);
```

Figure 3.2 Altering the speed at which a character's personality changes

than change the overall personality. The default is 5. Note that whole number changes in this value will have a very significant effect.
2) call the **ChangeBkgrndChange** function of myAICI to change the rate of change for every interaction the NPC has. This is like changing the rate an overall personality changes. In the real world, such changes usually take years; the default rate for game-play is 0.1 (which is still faster than real-world changes, but necessary for most game-play situations).

3.4 Using AIReturnResult and AINoResult

Sometimes you will want to get a result from the NPC's personality, but not have the mere fact of getting a result alter that personality. In the example in Figure 3.3, we are checking whether an NPC will start a conversation with a player who has just entered the NPC's shop. We don't want to change the NPC's personality (maybe we'd change it after the encounter, though, if for instance things go well and reinforce the NPC's desire to start conversations), so we check using **AIReturnResult**, but with the toChange parameter set to false.

To change the values later without needing another result, we could use **AINoResult**.

```
private bool StartConversation()
{

    //determine whether NPC wishes to start the conversation, or wait for the player to
    //start. Use the AI to determine by checking against NPC's gregariousness

    //pass NPC name, player name, tendency to check, and whether to change
    //personality based on this check (in this case, we don't; just getting a result)

    int startConversation = myAICI.AIReturnResult(plName, "gregarious", false);

    //this returns a value between 0 and 4, 0 being low in this tendency, 4 being very high

    //we don't need a separate response for every return value, so ...
    if (startConversation > 2)
    {
        //will definitely start conversation
        return true;
    } else if (startConversation < 2) {
        //will definitely not
        return false;
    } else {
        //in between; in this case, decided randomly
        System.Random random = new System.Random();
        int randomNumber = random.Next(0, 100);
        if (randomNumber >= 50)
        {
            return true;
        } else {
            return false;
        }
    }
}
```

Figure 3.3 Getting a result without changing the NPC personality

```
private void AfterConversation(bool thingsWentWell)
{

    //after the conversation, we decide to adjust the NPC's gregariousness based on
    //whether there was a positive outcome or a negative one

    //pass NPC name, player name, tendency to check, and whether to change
    //personality in a positive direction or a negative one

    if(thingsWentWell)
    {
        AINoResult(plName, “gregarious”);
        //note that the default is for positive change
    } else {
        AINoResult(plName, “gregarious”, false);
    }

}
```

Figure 3.4 Changing the personality without returning a result

For example, assume that after the conversation, we want to change the personality depending on whether we’ve evaluated things to have gone well (and thus reinforcing a stimulus, in this case the desire to start another conversation) or badly (thus making it less desirable to start conversations in the future, especially with this player). We would write something like the code in Figure 3.4, adjusting in a positive direction (the default) or a negative one.

The default action for any personality check requiring a result is to adjust the NPC’s personality at the same time (as this seems the more likely pattern). Thus a player being intimidating to an NPC will immediately change the NPC’s personality; there’s no need

```
//player has been kind
        whichResponse = myAICI.AIReturnResult(plName, "kind", true, false);
        switch (whichResponse)
        {
                case 0:
                        NPCResponse = "<replies something negative>";
                        break;
                case 1:
                        NPCResponse = "<replies something neutral>";
                        break;
                case 2:
                        NPCResponse = "<replies something kind>";
                        break;
                case 3:
                        NPCResponse = "<replies something really kind>";
                        break;
                case 4:
                        NPCResponse = "<offers help to player>";
        break;
        }

        ...
```

Figure 3.5 Changing the personality (negatively) and returning a result

to wait until later to evaluate things. In this case, call AIReturnResult with toChange and posChange set to True (the default, so really you need only have the playerName and stimulus parameters) or with toChange set to True and posChange set to False (see Figure 3.5).

In the above example, the player is kind to the NPC, and so she checks her response to kindness and changes slightly for the experience. There are more complex ways to check (and change) the NPC's personality, however. In Figure 3.6, we see an example of an NPC being intimidated by a player. In this case, we decide to check not just one but two response types—one for the NPC's intimidation response, and one for her anxiety response. Why? Because her desire to be intimidating right back might be outweighed by her anxiety over the situation. So in this case we check both, evaluate them against each other, and then figure out the NPC's response.

```
//so we're checking how the NPC responds to intimidation (is she more
//anxious or intimidating right back?) using a more complex combination
//of responses

        int isAnxious = myAICI.AIReturnResult(NPCname, plName, "anxiety");
        int isIntimidating = myAICI.AIReturnResult(NPCname, plName, "intimidating");

        whichResponse = isIntimidating - isAnxious;

        switch (whichResponse)
        {
                case -4:
                        NPCResponse = "<calls for help!>";
                        break;
                case -3:
                        NPCResponse = "<replies something frightened>";
                        break;
                case -2:
                        NPCResponse = "<replies something nervous>";
                        break;
                case -1:
                        NPCResponse = "<replies something neutral>";
                        break;
                case 0:
                        NPCResponse = "<replies something neutral>";
                        break;
                case 1:
                        NPCResponse = "<replies something neutral>";
                        break;
                case 2:
                        NPCResponse = "<replies something annoyed>";
                        break;
                case 3:
                        NPCResponse = "<tells player to get lost>";
                        break;
                case 4:
                        NPCResponse = "<ATTACKS!>";
                        break;
        }
```

Figure 3.6 More complex use of AIReturnResult

4.0 Support and Credits

4.1 Support

We're here to help! Contact us at support@quantumtigergames.com with any issues. We'll also post a FAQ on our website (once we have enough questions to have a FAQ!).

4.2 Credits

Design, coding: Jeffrey Georgeson
Raina character created by Mark Foley. Copyright © and TM 2012 Quantum Tiger Games, LLC

Persistence of Memory: In the near future, Quantum Tiger Games will release an even more specific memory package for NPCs, in which they will be able to remember actual events and the people attached to them, rather than just reaction types. We'll notify you when it's ready! (And you'll get a special upgrade deal if you already own the personality engine!)

5.0 Citations

What!? References in a user manual? Yes, indeed; because our game AI is based on research dealing with real humans and personality development. The following are a few of the many works dealing with the Five Factor Model of personality:

Costa, PT, Jr and McCrae, RR 1995, ‘Domains and Facets: Hierarchical Personality Assessment Using the Revised NEO Personality Inventory’, *Journal of Personality Assessment*, vol. 64, no. 1, pp. 21-50.

DeYoung, CG 2010, ‘Toward a Theory of the Big Five’, *Psychological Inquiry*, vol. 21, no. 1, pp. 26-33.

Goldberg, L 1993, ‘The structure of phenotypic personality traits’, *American Psychologist*, vol. 48, no. 1, pp. 26-34.

John, OP et al 2010, ‘Paradigm Shift to the Integrative Big Five Trait Taxonomy: History, Measurement, and Conceptual Issues’, in John et al (ed.), Handbook of personality: Theory and research, 3d ed, Guilford, New York. (US edition.)

McCrae, RR & Costa, Jr PT 2010, ‘The Five-Factor Theory of Personality’, in John et al (ed.), *Handbook of personality: Theory and research*, 3d ed, Guilford, New York. (US edition.)