User Manual for the

Encounter Editor

What use are Encounters, anyway?

After many years of trying, I have learned the hard way that the procedurally intense interactions provided by the Storytron technology lack the color that most people expect from traditional storytelling. There's a repetitive, mechanical feel to those interactions, and while they are dramatically more intense, more significant, they are like the skeleton of the story, the core elements, in need to fleshing out with muscle and skin. That's the purpose of Encounters. They provide a more data-intense form of interaction that is shallower in dramatic significance, but more colorful. I believe that a proper balance between the strong interaction of the core technology and the greater color of the Encounters will provide a powerful dramatic experience.

But to work properly, we need hundreds of Encounters in the storyworld. I could have written them all — indeed, in the time it took me to program the Encounter Editor, I probably could have written a few hundred Encounters. But I believe that the Encounter Editor provides multiple benefits:

1. We'll get a greater range of ideas for Encounters from a larger group of authors.
2. The Encounter Editor is an easy way to learn about the basic technology used in the Storytron technology. It prepares people for tackling the more powerful technology.
3. The Encounter Editor should be easy and fun enough to attract people into the world of interactive storytelling.

What’s in it for you?

The ultimate goal of the Encounter Editor is to have lots of people like you writing Encounters that we will include in the final game. There will be no payment of any kind for your services. This entire project is a non-profit community effort in which NOBODY gets paid. We have been getting some money from a Patreon group, but so far it’s not enough to provide a month’s pay for a decent programmer. But hey — we’re starting a revolution here, and we got no room for no stinkin’ capitalists!

Your authorship of Encounters will be shown in the game, so if this thing really does get the ball rolling for interactive storytelling, you’ll be famous! If you’re a guy, eager crowds of nubile nymphs will follow you everywhere. If you’re a gal, guys will be green with envy and attack you on social media.

Yeah, right…

Backstory

In order to understand the context of the storyworld in which the Encounters are written, you must read the novel. There are two versions of the novel included in this package. The first version is “Old Novel”. This is the version written several years ago; it contains material explaining the background of the storyworld well before the game starts.

The second novel is called “Novel Draft 13” and presents only the parts of the novel necessary for playing the game.

You must read both of these novels. They provide you with the backstory for this storyworld.

The Lectures

I prepared three lectures presenting the background ideas for the Encounter Editor and explaining how to use it. You should watch those lectures before going any further.

Lecture #1: https://youtu.be/yH-g7GLe0Uc
Lecture #2: https://youtu.be/l87BcwMMP6U
Lecture #3: https://youtu.be/JjU7OtMGcdM

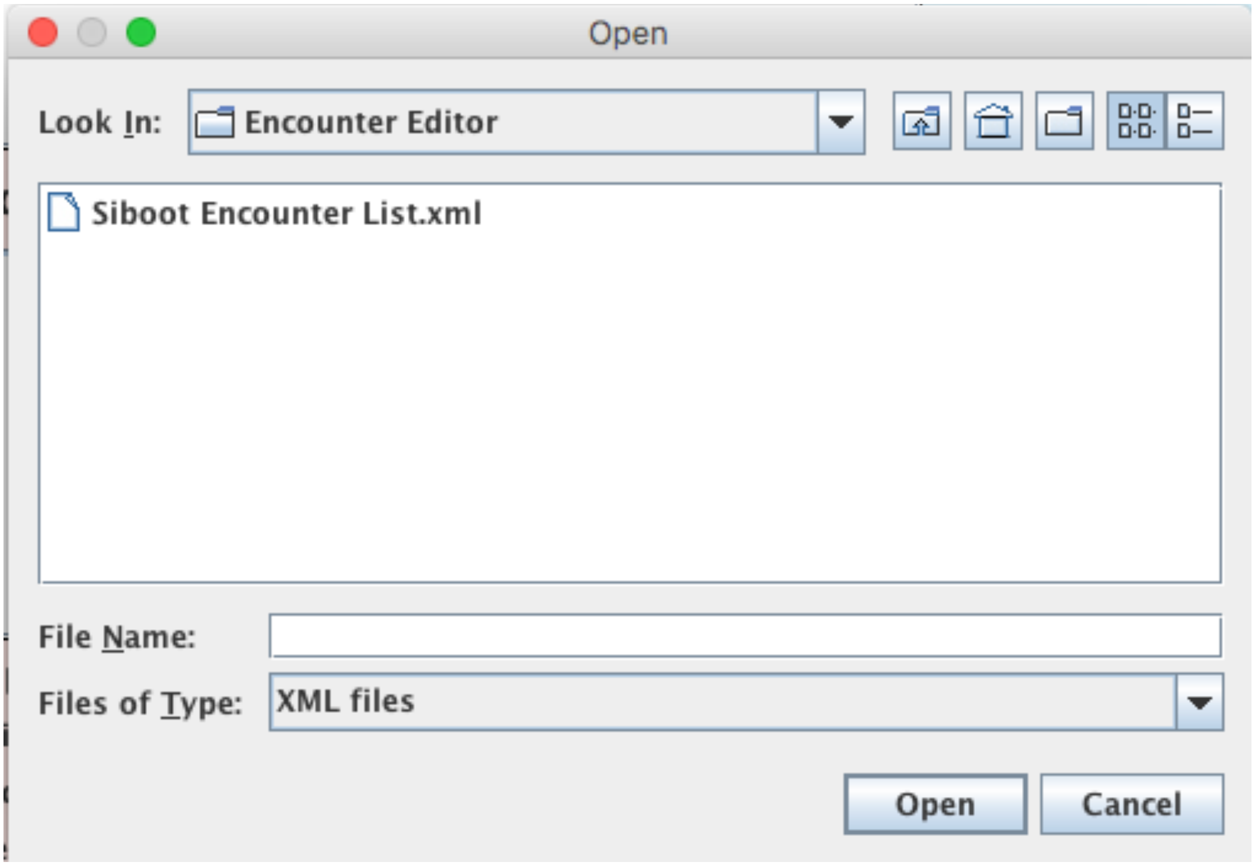
Now let’s turn to the software.

Installing Java

The Encounter Editor is a Java program; you must have Java installed on your computer in order to run the program. Many computers have Java installed at the factory. If you double-click on the Encounter Editor and nothing happens, then you must download and install Java yourself. Start here: https://www.java.com/en/download/ and follow the directions to install Java on your computer.

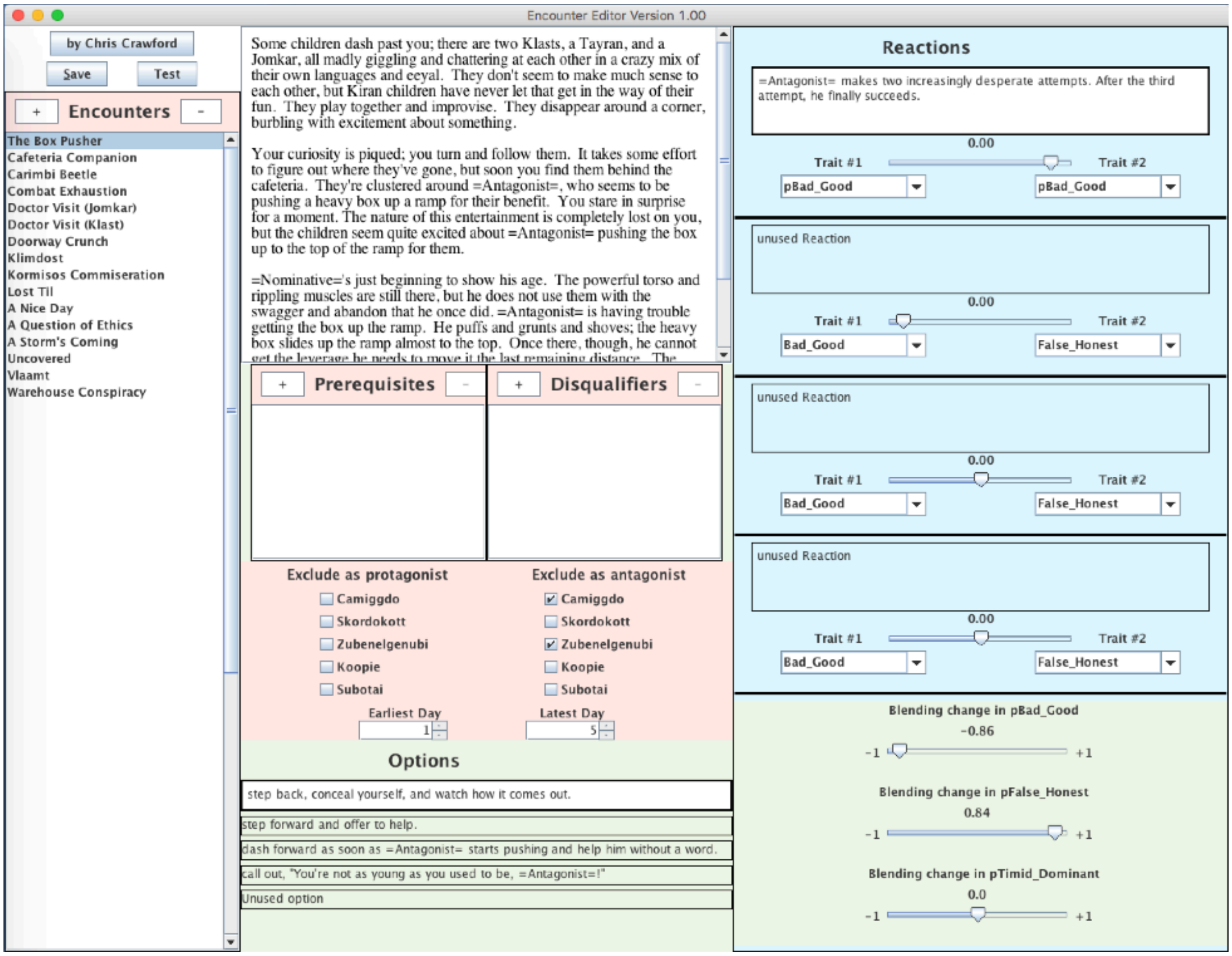
I must advise you that Java is not absolutely, positively secure against evil hackers. Now, I’m not an evil hacker, and the Encounter Editor is safe to use, but if you have Java installed and you visit an evil website, it can use the Java in your computer to do evil things. The vast, vast majority of websites are not evil, so you’re probably safe. I’ve had Java installed on my computer for 15 years now, and I use the web heavily, and I’ve never had a problem. But it would be unconscionable for me NOT to mention this.

OK, so now you’ve gotten Java installed and you can launch the program. Before anything else can happen, it will demand that you provide it with an Encounter List:



You've got just one choice. Take it. Later on, when lots of different people are working on different Encounter Lists, this will be useful. But not yet.

Once you've given it an Encounter List to work with, you'll see this window:



The Box Pusher
Cafeteria Companion
Carimbi Beetle
Combat Exhaustion
Doctor Visit (Jomkar)
Doctor Visit (Klast)
Doorway Crunch
Klimdost
Kormisos Commiseration
Lost Til
A Nice Day
A Question of Ethics
A Storm's Coming
Uncovered
Vlaamt
Warehouse Conspiracy

Some children dash past you; there are two Klasts, a Tayran, and a Jomkar, all madly giggling and chattering at each other in a crazy mix of their own languages and eeyal. They don't seem to make much sense to each other, but Kiran children have never let that get in the way of their fun. They play together and improvise. They disappear around a corner, burbling with excitement about something.

Your curiosity is piqued; you turn and follow them. It takes some effort to figure out where they've gone, but soon you find them behind the cafeteria. They're clustered around =Antagonist=, who seems to be pushing a heavy box up a ramp for their benefit. You stare in surprise for a moment. The nature of this entertainment is completely lost on you, but the children seem quite excited about =Antagonist= pushing the box up to the top of the ramp for them.

=Nominative='s just beginning to show his age. The powerful torso and rippling muscles are still there, but he does not use them with the swagger and abandon that he once did. =Antagonist= is having trouble getting the box up the ramp. He puffs and grunts and shoves; the heavy box slides up the ramp almost to the top. Once there, though, he cannot get the leverage he needs to move it the last remaining distance. The

Exclude as protagonist	Exclude as antagonist
☐ Camiggdo	☑ Camiggdo
☐ Skordokott	☐ Skordokott
☐ Zubenelgenubi	☑ Zubenelgenubi
☐ Koopie	☐ Koopie
☐ Subotai	☐ Subotai

Earliest Day: 1
Latest Day: 5

step back, conceal yourself, and watch how it comes out.
step forward and offer to help.
dash forward as soon as =Antagonist= starts pushing and help him without a word.
call out, "You're not as young as you used to be, =Antagonist=!"
Unused option

=Antagonist= makes two increasingly desperate attempts. After the third attempt, he finally succeeds.
0.00
Trait #1: pBad_Good
Trait #2: pBad_Good

unused Reaction
0.00
Trait #1: Bad_Good
Trait #2: False_Honest

unused Reaction
0.00
Trait #1: Bad_Good
Trait #2: False_Honest

unused Reaction
0.00
Trait #1: Bad_Good
Trait #2: False_Honest

Blending change in pBad_Good: -0.86 (-1 … +1)
Blending change in pFalse_Honest: 0.84 (-1 … +1)
Blending change in pTimid_Dominant: 0.0 (-1 … +1)

(If you expand the PDF window you're looking at, you'll see the window at full size.)

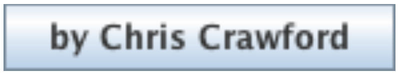
The window is organized into three columns. There are three levels of editing. Every Encounter contains some Options; every Option contains some Reactions. Places where you edit Encounter properties are colored pink. Places where you edit Option properties are colored green. Places where you edit Reaction properties are colored blue.

User-animosity

Most programs these days are laden with all sorts of user-friendly features to make life easier for users. I went to a great deal of effort to make this program fast and easy to use if you know what you're doing. But I didn't include anything to help you correct mistakes. There are no guard rails on this highway. There are no undo capabilities. If you accidentally delete something, it's gone forever. If you screw up badly, your best bet is to quit the program without saving. Of course, you'll lose all your work since you last saved. Enjoy!

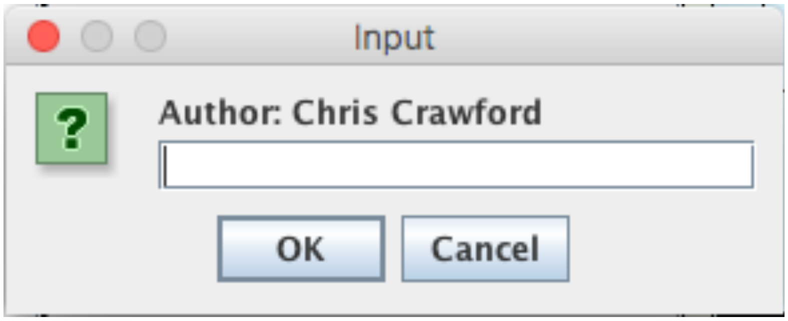
Left column

The left column provides general features and allows you to create new Encounters. From the top down, they are:



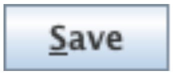
Authorship button

Click on this and you'll get a dialog box:



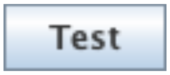
Author: Chris Crawford

Type in your name, or your pen name, or whatever name you wish to be known by. Then press the OK button. You are now the official author of this Encounter. Please don't change authorship of other people's Encounters; that's a serious no-no.

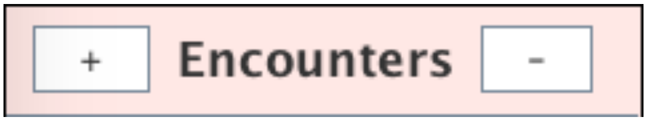


Save this file

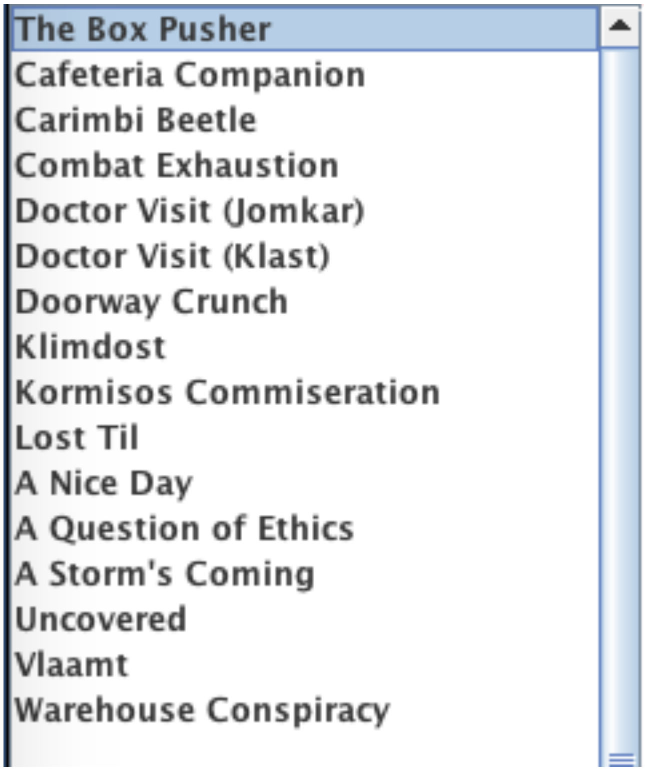
This should be obvious. Save your work early and often. Remember, this is experimental software; it could explode in your face at any time, or perhaps pee bytes on your foot.



This is a useful tool, but I'll put off explaining it until we've gone through some other stuff.



This is where you add or delete Encounters. I think you can figure it out from here. Click on the plus button and you get a dialog box requesting the name of the new Encounter. Click on the minus button and the currently selected Encounter will be deleted. There's not even an "Are you sure?" dialog.

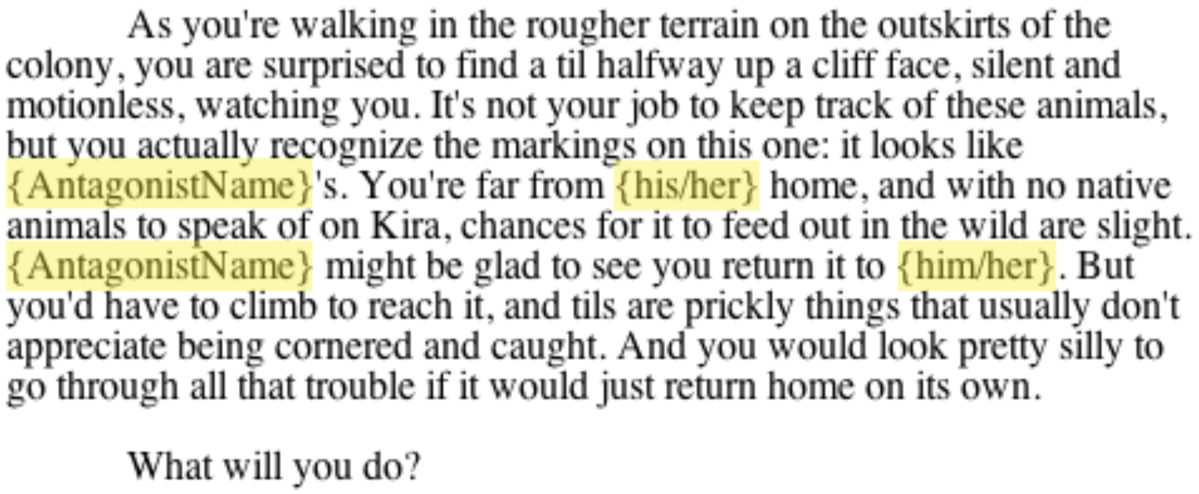


The Box Pusher
Cafeteria Companion
Carimbi Beetle
Combat Exhaustion
Doctor Visit (Jomkar)
Doctor Visit (Klast)
Doorway Crunch
Klimdost
Kormisos Commiseration
Lost Til
A Nice Day
A Question of Ethics
A Storm's Coming
Uncovered
Vlaamt
Warehouse Conspiracy

This is the list of all the Encounters in the file. It's simple enough — just click on an Encounter to make it editable. Double-clicking does nothing. Neither does quintuple-clicking.

Center column

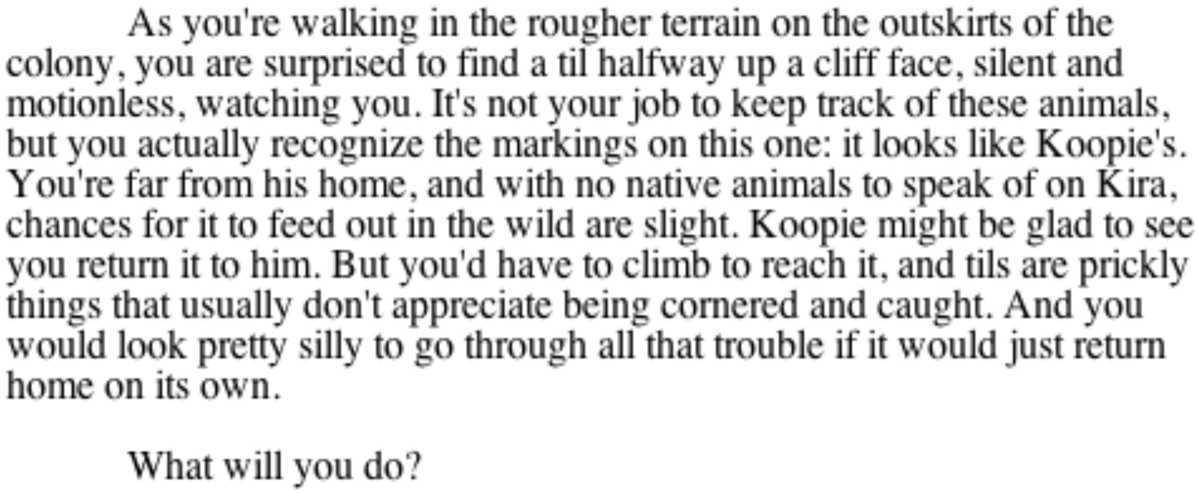
At the top we have the space for the introductory text. This is the story that will be presented to the players prior to challenging them to select an Option. It’s a text-editing box: just click somewhere and start typing your story. You are probably nonplussed by the odd items in this text. Here’s a screen shot that I have highlighted to show the odd items.



As you're walking in the rougher terrain on the outskirts of the colony, you are surprised to find a til halfway up a cliff face, silent and motionless, watching you. It's not your job to keep track of these animals, but you actually recognize the markings on this one: it looks like {AntagonistName}'s. You're far from {his/her} home, and with no native animals to speak of on Kira, chances for it to feed out in the wild are slight. {AntagonistName} might be glad to see you return it to {him/her}. But you'd have to climb to reach it, and tils are prickly things that usually don't appreciate being cornered and caught. And you would look pretty silly to go through all that trouble if it would just return home on its own.

What will you do?

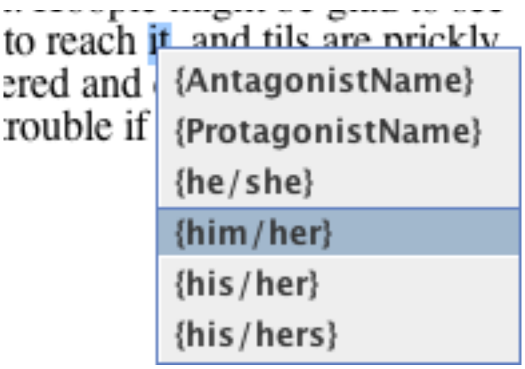
These are called *text variables.* They allow the text to apply to any character who plays a role in the Encounter. Just before the text is displayed to the player, these text variables will be replaced with their actual values. For example, if a player saw this Encounter with Koopie (a male) as the Antagonist, the final text would look like this:



As you're walking in the rougher terrain on the outskirts of the colony, you are surprised to find a til halfway up a cliff face, silent and motionless, watching you. It's not your job to keep track of these animals, but you actually recognize the markings on this one: it looks like Koopie's. You're far from his home, and with no native animals to speak of on Kira, chances for it to feed out in the wild are slight. Koopie might be glad to see you return it to him. But you'd have to climb to reach it, and tils are prickly things that usually don't appreciate being cornered and caught. And you would look pretty silly to go through all that trouble if it would just return home on its own.

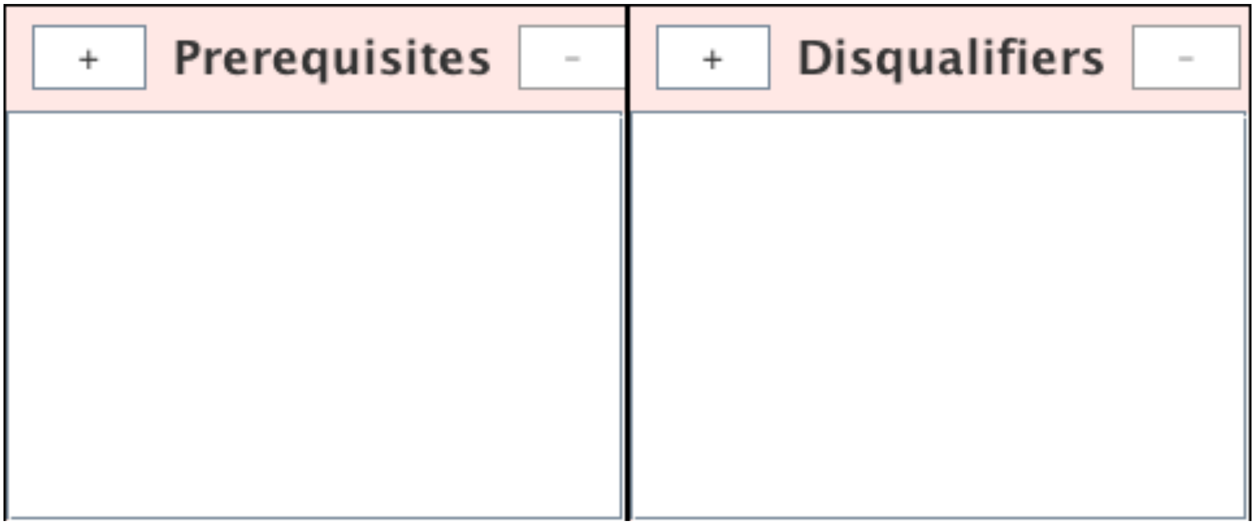
What will you do?

Where do text variables come from? Well, you could type them in by hand, but I’ve struggled across deserts, climbed mighty mountains, and survived endless dangers to bring you a special feature. Just select a place (or even a word) in the Introductory text and click and hold the RIGHT mouse button (the one that you don’t use much). A little popup menu will appear:



Select the text variable you desire from the menu and poof! It’s there! (Please hold the applause until the end of the show.)

The introductory text is the easiest part of creating an Encounter. Next come these two little brats:

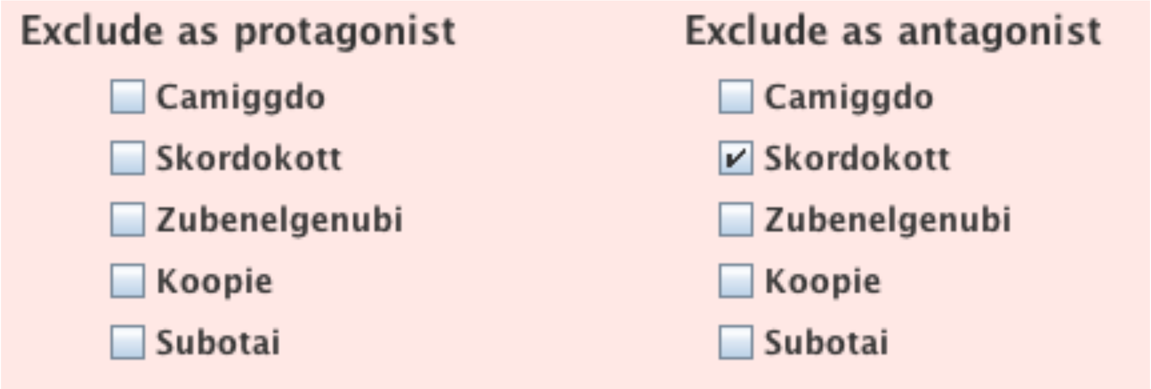


As a beginner, you won't need to use them. Later on, when you come back to this manual as an experienced Encounter-creator, you'll want to read this:

Prerequisites are Encounters that must take place before this Encounter can take place. Disqualifiers are Encounters that must NOT take place before this Encounter can take place. Together, you can use these two features to set up a tree of sequential Encounters presenting a much deeper dramatic sequence than you can get out of a single Encounter.

To add to either list, simply click on the + button. A window will pop up showing all of the Encounters in the storyworld. Select the desired Encounter by clicking on it. Poof! Now it's listed in one of these boxes. To remove an encounter from either list, simply click on the - button. It will not delete the Encounter; it will only remove that Encounter from that particular box.

Now for a somewhat more useful set of features:

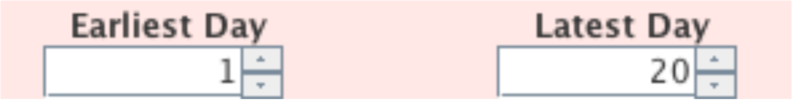


Exclude as protagonist
☐ Camiggdo
☐ Skordokott
☐ Zubenelgenubi
☐ Koopie
☐ Subotai

Exclude as antagonist
☐ Camiggdo
☑ Skordokott
☐ Zubenelgenubi
☐ Koopie
☐ Subotai

These allow you to customize an Encounter to a particular set of characters. For example, if your Encounter assumes that the Protagonist must be male and the Antagonist must be female, then you would exclude the females (Camiggdo, Zubenelgenubi, and Subotai) from the first column and the males (Skordokott and Koopie) from the second column. You could even restrict an Encounter to a single character.

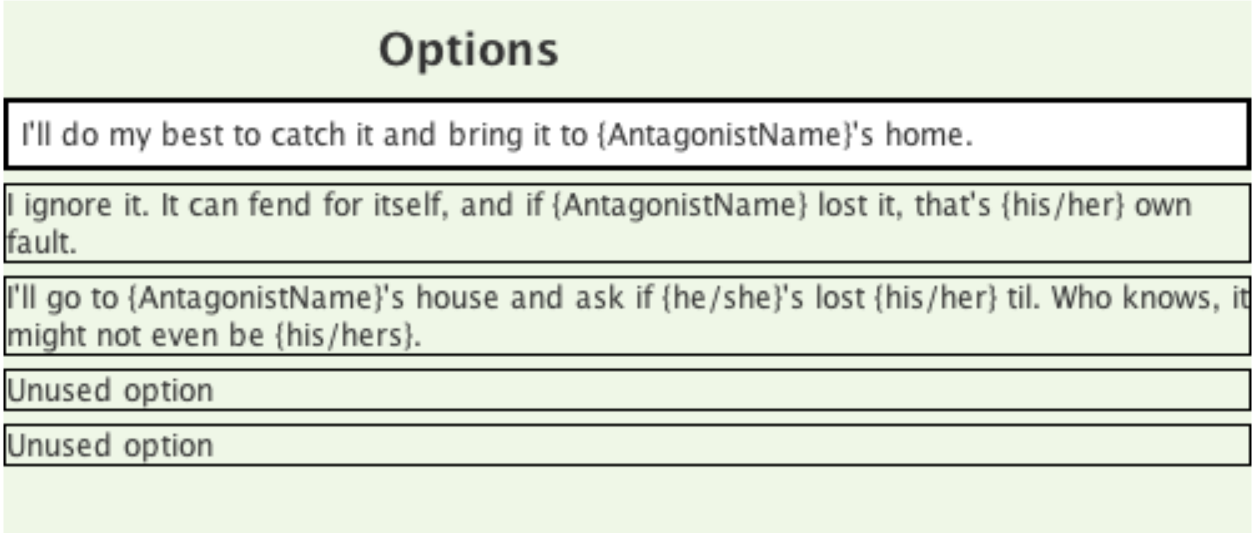
Here's another minor feature:



Earliest Day: 1
Latest Day: 20

This allows you to specify that your Encounter can take place only on the days specified in the two number boxes. Not many Encounters will need this feature.

Now let's get more complicated:



I'll do my best to catch it and bring it to {AntagonistName}'s home.

I ignore it. It can fend for itself, and if {AntagonistName} lost it, that's {his/her} own fault.

I'll go to {AntagonistName}'s house and ask if {he/she}'s lost {his/her} til. Who knows, it might not even be {his/hers}.

Unused option

Unused option

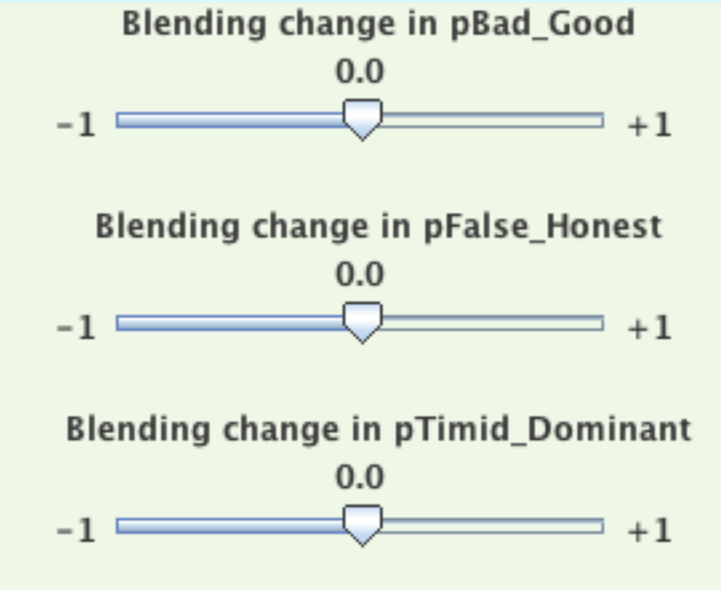
This is where you create the Options that you want to give to the player. Don't worry if the text boxes look too short to type in a long string of text; the text boxes automatically expand as you type in more text. However, you really should keep these options short. You are giving the player options to choose from a menu; you don't want to boggle your player's mind with an encyclopedic option.

When you click inside a text box, that particular Option becomes THE selected Option, and you can edit its components. It's just like THE Encounter that you select from the list on the left.

You can use text variables inside the Option text by right-clicking at the location where you want the text variable.

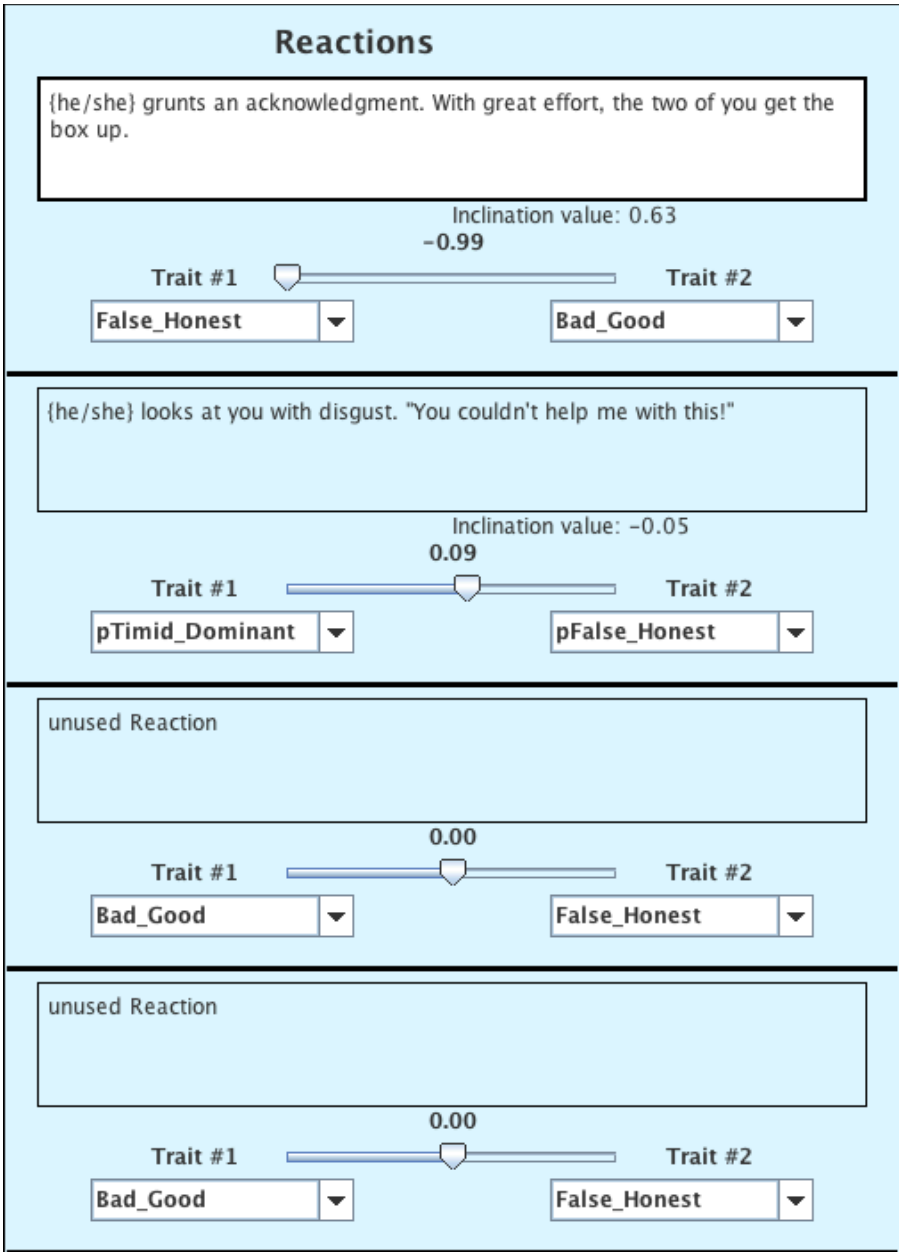
Right Column

This has two parts: one for editing reactions and another for editing the emotional responses to each Option. It's kinda screwed up: I wanted to fit all the green Option stuff into one column, but I couldn't make everything fit nicely, so I ended up using some extra space in the lower right part of the window. So let's begin with the green part that applies directly to the Option:


Blending change in pBad_Good
0.0
-1
+1
Blending change in pFalse_Honest
0.0
-1
+1
Blending change in pTimid_Dominant
0.0
-1
+1

These sliders specify how the Antagonist's feelings toward the Protagonist change in response to the Protagonist's choice of option. Think about it: a situation arose (the Introductory text) and the Protagonist responded by selecting an Option. Perhaps that was a nice Option; perhaps it was a nasty Option. This is where you specify how the Antagonist feels about that choice. The numbers on these sliders represent the change in the Antagonist's feeling towards the Protagonist.

Now we come to the final part of the window: the Reaction editing slots. There are only four of these because you are permitted a maximum of four Reactions for each Option. If you think that’s not enough, then go write your own damn Encounter Editor! Harrumph!



Reactions

{he/she} grunts an acknowledgment. With great effort, the two of you get the box up.

Inclination value: 0.63

-0.99

Trait #1: False_Honest

Trait #2: Bad_Good

{he/she} looks at you with disgust. "You couldn't help me with this!"

Inclination value: -0.05

0.09

Trait #1: pTimid_Dominant

Trait #2: pFalse_Honest

unused Reaction

0.00

Trait #1: Bad_Good

Trait #2: False_Honest

unused Reaction

0.00

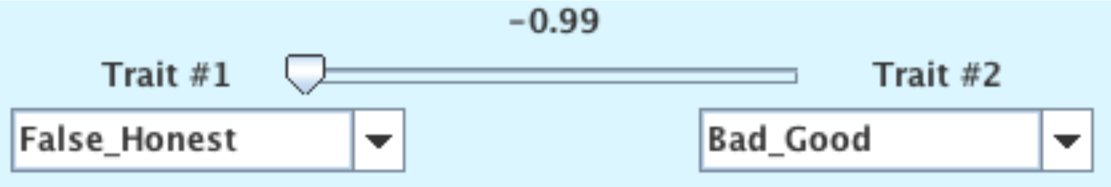
Trait #1: Bad_Good

Trait #2: False_Honest

I have created only two Reactions for this Option, so the remainder are listed as “unused Reaction”. The Encounter Engine is smart enough to know that any Reaction that has the text “unused Reaction” is not an actual

Reaction and it will skip over it. However, if you change the text in ANY manner, it will treat it as a real Reaction and include it in its calculations, no matter how silly it seems.

Editing the text for the Reactions is exactly the same as with the Options and the Introductory text. But now let's zero in on this strange stuff:



This is the most important — and most difficult — part of the Encounter Editor. This is where you create the algorithm used to determine which Reaction the Antagonist will choose. Remember, the Antagonist is always a computer-controlled character, so you must provide it with a way of making decisions.

First, a bit of technical background. The Encounter Engine selects a Reaction based on a simple criterion. Each Reaction has a little formula that we call the "inclination algorithm". That algorithm is a little calculation that produces a number between -1 and +1. The Encounter Engine does each of the calculations, and the Reaction that produces the highest Inclination is chosen.

The blue chunk above shows how you set up this algorithm. We'll start at the bottom and work up. There are two elements on the bottom, labeled "Trait #1" and "Trait #2". These are the personality traits that you will use to make the decision. As I explained in Lecture #3, you need to determine which traits bear most closely on making the decision. I suppose that I should put that process down in writing but it will have to wait while I finish this user manual.

There are twelve factors from which to choose, but only three basic concepts: virtue, integrity, and dominance. We add the perceived values of these (pBad_Good, pFalse_Honest, and pTimid_Dominant); that doubles the number of factors to six. But there's one other trick that drives some people (are you reading this, Dave?) crazy: we also have the negative values of each of the above six as additional factors. Here's the set of factors you can use:

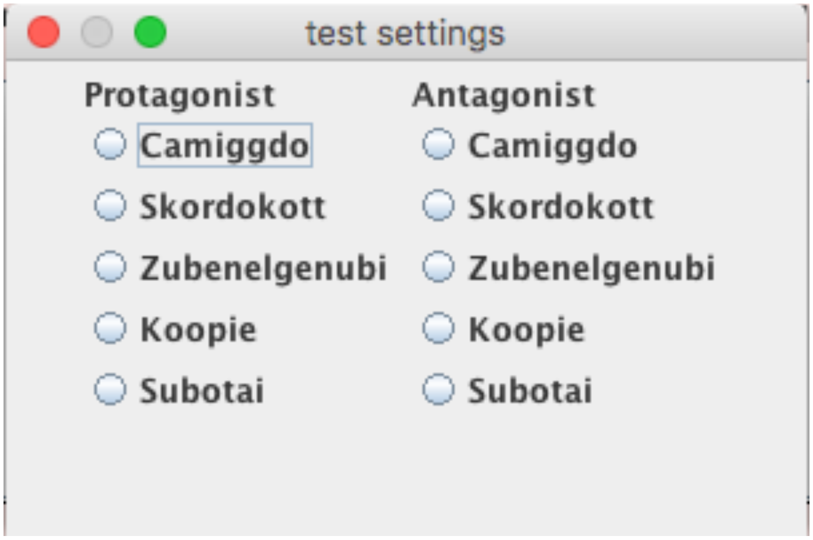
Bad_Good	-Bad_Good
False_Honest	-False_Honest
Timid_Dominant	-Timid_Dominant
pBad_Good	-pBad_Good
pFalse_Honest	-pFalse_Honest
pTimid_Dominant	-pTimid_Dominant

The reason we include the negatives of each factor is to permit you use the opposite of a trait for an inclination. For example, suppose you have a nasty Reaction in which the Antagonist does something mean to the player. What would increase the inclination for that Reaction? It certainly wouldn't be the value of Bad_Good or pBad_Good — you don't do nasty things if you are a good person or you like the other person. You do nasty things if you are a nasty person or you hate the other person. So we need to use the OPPOSITE of the Bad_Good value or the pBad_Good value. And the opposite of a number is its negative value. That's why we include the negative values as potential factors in the algorithms.

Having chosen the two most relevant factors to use, you need only establish the weighting between them. If you want to give them both equal weight, then set the slider in the middle to a value of 0.0. If you want to give one of the factors more weight, push the slider towards it. If you push the slider all the way to 1.00, then the other factor will be completely ignored in the calculation.

Now at last you are ready to go back to this button:

This is a useful tool that you use to adjust your algorithms. When you click on this button, a dialog box appears:

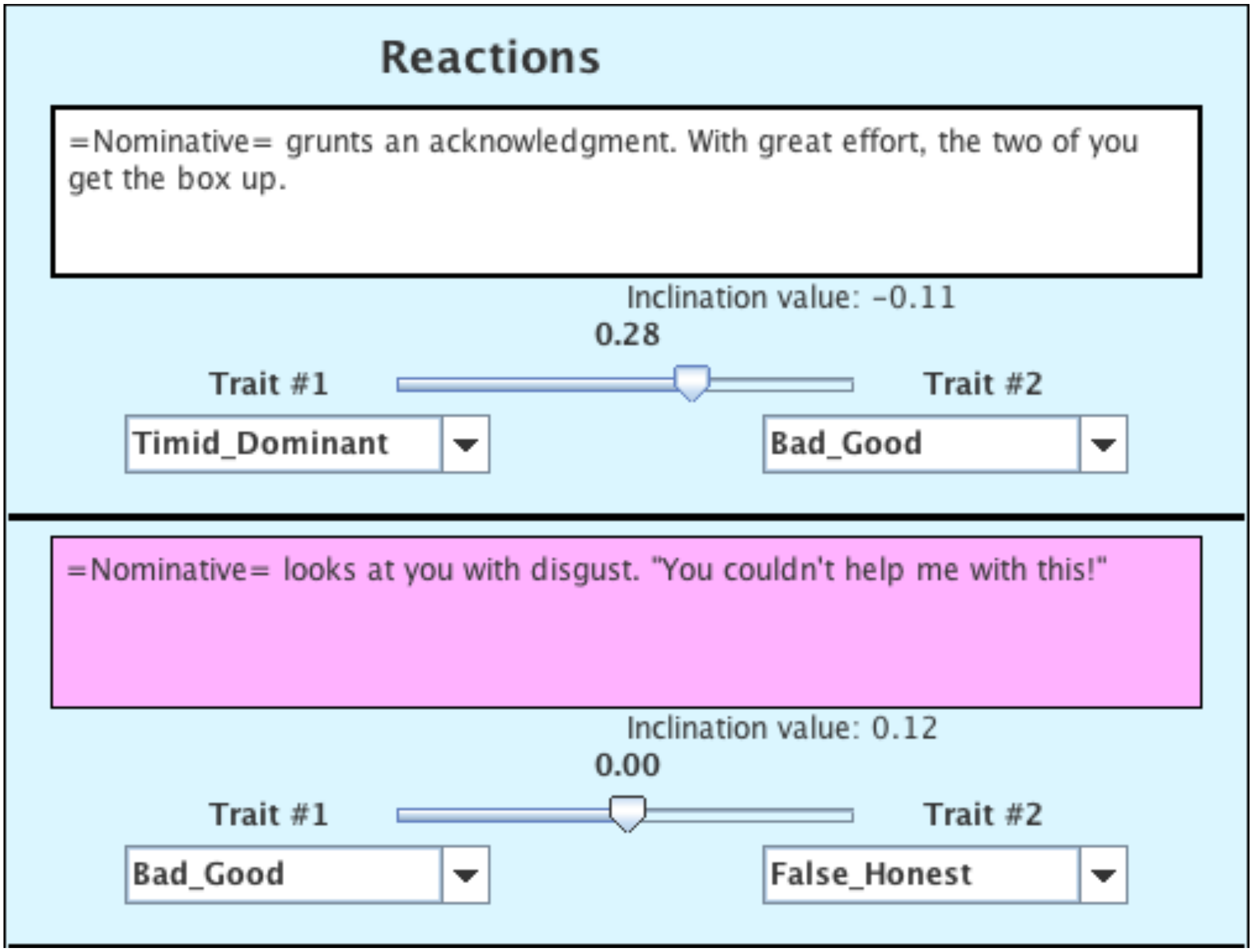


test settings

Protagonist	Antagonist
○ Camiggdo	○ Camiggdo
○ Skordokott	○ Skordokott
○ Zubenelgenubi	○ Zubenelgenubi
○ Koopie	○ Koopie
○ Subotai	○ Subotai

You must choose one character to be the protagonist and another to be the antagonist. (The protagonist is the character referred to in the second person in the introductory text; also known as the "player character".)

As soon as you have selected both characters, the display in the right column changes to look something like this:



=Nominative= grunts an acknowledgment. With great effort, the two of you get the box up.

Inclination value: -0.11

0.28

Trait #1: Timid_Dominant

Trait #2: Bad_Good

=Nominative= looks at you with disgust. "You couldn't help me with this!"

Inclination value: 0.12

0.00

Trait #1: Bad_Good

Trait #2: False_Honest

Since there are only two Reactions provided for this Option, they are the only ones with any data, and therefore I show only those two.

The pink highlighting shows you which of the two Reactions will be chosen based on the personality traits of the two selected characters and the existing algorithms. It shows you the results of the calculations with the item labeled “Inclination value:”. The upper Reaction has an Inclination value of -0.11, while the lower Reaction has an Inclination value of 0.12. Since the lower Reaction has the larger Inclination value, it will be selected by the Encounter Engine.

But the best part of the test system is your ability to change the algorithm and see the effects instantly. Grab either of the sliders and try different values; as you do, the Inclination value for that Reaction will change, which just might change the choice of Reactions by the Engine.

You can also change the factors used and see the resulting changes in the Inclination values. Of course, these factors use the existing trait values of the selected characters.

Appendix 1

Personality Traits and Perceptions

There are three basic personality traits for each character:

Bad_Good

This is the easiest trait to understand. It specifies just how good or bad the character is. We could also call it Nasty_Nice or Evil_Virtuous. A value of -0.99 applies to a profound evil person; +0.99 means that the character is a saint.

False_Honest

This specifies a character's integrity or lack thereof. It represents the degree to which a character tells the truth and honors their promises. Among historical characters, I would put Donald Trump at about -0.95 and George Washington ("I cannot tell a lie") at +0.90.

Timid_Dominant

This specifies the "force of character" or self-confidence or determination of a character. It also reflects the degree to which a character is inclined to respect others and pay heed to their desires. A cowering, slobbering sycophant would merit a value of -0.99, while a headstrong, uncompromising character like (again) Donald Trump would merit a value of +0.90.

A table of examples of personality traits

Name	Bad_Good	False_Honest	Timid_Dominant
Darth Vader	-0.80	0.0	+0.8
Obiwan Kenobi	+0.8	+0.5	+0.4
Princess Leia	+0.9	+0.6	+0.4
The Emperor	-0.9	-0.7	+0.9

Name	Bad_Good	False_Honest	Timid_Dominant
Frodo Baggins	+0.8	+0.7	0.0
Samwise Gangee	+0.5	+0.7	-0.5
Gollum	-0.6	-0.7	-0.4
Gandalf	+0.9	+0.7	+0.8
Donald Trump	-0.4	-0.9	+0.9
Barak Obama	+0.5	+0.4	+0.5

Here are the trait values for the five characters in Siboot:

Name	Bad_Good	False_Honest	Timid_Dominant
Camiggdo	-0.4	+0.6	+0.4
Skordokott	0.0	+0.4	+0.6
Zubenelgenubi	+0.8	+0.3	-0.7
Koopie	+0.5	+0.2	-0.2
Subotai	-0.3	-0.1	+0.4

You will note that these values do not correspond with what I said about these characters in the lecture; I realized that the traits needed to be revised to provide better interpersonal conflict.

Perceived Traits are Relationships

Fundamental to this personality model is the observation that a relationship can adequately be defined as a perceived value of a personality trait. This is easiest to understand in the case of trust: trust is clearly the perceived integrity (False_Honest) by one character of another.

The idea extends fairly well to Bad_Good: if you perceive somebody to be good, you like them, and if you perceive somebody to be bad, you dislike them. We can ignore the direction of causality — it doesn't matter whether you like them because you perceive them to be good, or you perceive them to be good because you like them. The correlation is the important point: if you like someone, then you perceive them to be good.

The concept is least effective when applied to Timid_Dominant, because it's difficult to differentiate the perception from the intrinsic reality. Moreover, the perceived value is heavily affected by one's own intrinsic value. Again, let's take Donald Trump, because he is such a dominant character. Because has such high intrinsic dominance, he'll perceive everybody around him to be relatively timid. By contrast, an extremely timid person will perceive others to be dominating. Nevertheless, it is still possible to use the difference between the intrinsic value and the perceived value.

We designate the perceived value with a "p" in front of the intrinsic trait. Thus, pFalse_Honest corresponds to one character's trust in another; pBad_Good corresponds to how much that character likes the other; and pTimid_Dominant corresponds to how intimidated that character is to the other one.

Remember, perceived values are two-dimensional; therefore, we need to use more complex tables to represent the values in Siboot:

pBad_Good values

	Cam	Skor	Zub	Koo	Sub
Camiggdo for	—	-0.4	0.8	0.5	-0.2
Skordokott for	-0.6	—	0.8	0.4	0.2
Zubenelgenubi for	-0.1	0.3	—	0.5	0.1
Koopie for	0.1	-0.5	0.8	—	-0.4
Subotai for	-0.3	-0.2	0.7	0.2	—

pFalse_Honest values

	Cam	Skor	Zub	Koo	Sub
Camiggdo for	—	0.0	0.5	0.0	-0.4
Skordokott for	0.6	—	-0.2	0.2	-0.1
Zubenelgenubi for	0.6	-0.1	—	0.3	0.3
Koopie for	0.0	0.5	-0.1	—	-0.4
Subotai for	-0.5	0.2	-0.3	0.4	—

pTimid_Dominant values

	Cam	Skor	Zub	Koo	Sub
Camiggdo for	—	0.6	-0.7	-0.4	0.2
Skordokott for	0.2	—	-0.7	0.3	-0.1
Zubenelgenubi for	0.3	0.4	—	0.4	0.5
Koopie for	0.2	0.3	-0.7	—	-0.3
Subotai for	-0.2	0.3	-0.5	0.0	—