Alice Exercise

Tutorial: Making Procedural Methods

(http://www.alice.org/resources/exercise-and-project/tutorial-creatin…ocedural-methods/)

Introduction – Wonderland Tea Party Arrival

This tutorial exercise provides step-by-step directions for guiding you through creating your own custom procedures. Here you will explore the different ways that a custom procedure can be created and how that affects where it can be used. This exercise will provide an overview of the process and give you the basic skills to get you started. This exercise has extended content for those comfortable using parameters that is optional.

You will need to access other printed materials or have access to the alice.org website to view the how to content called out within these materials. These materials can be downloaded and printed for offline use from Alice.org

Don't forget to save your projects frequently

Planning – Designing the Program

For this exercise we are going to start with the general scene and algorithm as follows:

Scene Description:

The Madhatter stands waiting to greet guests to a tea party.

Script Algorithm

The guests start to arrive one by one.

The CheshireCat walks up to the MadHatter
The MadHatter says “Welcome to the Tea Party!”

The PlayingCard walks up to the MadHatter
The MadHatter says “Welcome to the Tea Party!”

The White Rabbit walks up to the MadHatter
The MadHatter says “Welcome to the Tea Party!”

… This continues for the remainder of the guests.

Setting Up the Scene

For this exercise you can either use the provided world or create your own.

1. Start Alice
2. Open the MadHatterGreeting.A3P world or/
3. Create your own scene with the following:
   a. the host of the party positioned on one side of the screen
   b. several party guests positioned just off the screen - to do this we set up the scene we desired and used one-shots to move the character backwards 10 allowing us to use a move forward 10 to put them in the final position
   c. For this exercise use characters all of the same top level class hierarchy ie Bipeds, Quadrupeds, Swimmers etc

Creating a Custom MadHatter Procedure

Create a custom procedure to replace the repititious:

The MadHatter says “Welcome to the Tea Party!”

For the following steps, you may wish to watch the Videos or check the Quick Reference Guides for the How To: *Creating a Procedure*

4. From the class select drop down menu found at the top of the code editor select the **MadHatter** class opening the sub menu.
5. Select the *add MadHatter procedure* option from the drop down or/ select the **MadHatter** class folder from the top of the list.
6. If you selected the **MadHatter** class folder click the *Add MadHatter Procedure* button in the procedures section of the summary page that opened.
7. Name your procedure following Alice programing naming conventions:
   a. Name must be all one word use camelCase if multiple words are desired

b. No special characters (only letters and digits no ?)
c. Name it something that someone looking at your program could easily understand what the functionality will be in this case **Greeting**

The code editor should now be opened to the **Greeting** custom procedure tab

Program Your Greeting

You can now build your greeting however you would like or following along and have the MadHatters mouth open and close in time with "Welcome to the Tea Party" being displayed. You may wish to watch the Videos or check the Quick Reference Guides for the How To: *Using Procedures Overview* and other How To materials for the code editor.

8. Add a **say** procedure and use *Custom TextString* to input "Welcome to the Tea Party!"
9. Select **getMouth** joint from the object menu drop down. Use the object drop down showing "this" and use the right arrow to navigate to the joint list.
10. Drag in a **turn** statement and select *forward* and a value of .125
11. Drag in another **turn** statement or copy the first and set the direction to *backward* and the value to .125
12. Create a *do together* code block and add all of the above statements into it
13. Create a *do in order* code block inside of your *do together* code block and add the mouth turn commands into that.
14. Select add details and change the *duration* of the say statement to be 2 so that it will match the duration of the two turn statements with default durations of 1 each.

Using a Custom Procedure

Let's go ahead and make use of your new custom procedure to program the scene:

15. Add a procedure statement **move** forward with *distance* 10 for CheshireCat - the distance we moved the character out of the scene using a one shot when setting up the scene
16. Add the new **Greeting** procedure for MadHatter
17. Repeat the above steps for all of your party guests.
18. Run your program

Iterating a Design

It seems rude that the guests don't respond to the MadHatter. What if we wanted them all to say "Hello MadHatter" or something similar in response? For the next steps let's edit our script algorithm to add in a response

The CheshireCat walks up to the MadHatter
The MadHatter says "Welcome to the Tea Party!"

The CheshireCat says “Hello MadHatter!”

The PlayingCard walks up to the MadHatter
The MadHatter says “Welcome to the Tea Party!”
The PlayingCard says “Hello MadHatter!”

Again repeat the change for all the characters

Creating a Custom Biped Procedure

Again you could go ahead and program this for each guest but that would get very repetitive and long. Creating a custom method for each character also wouldn’t save us any time so let’s create the custom class for all BiPeds using the following steps:

19. Following the same method as before (Steps 4-7) create a new Biped procedure named **Salutations**.

The code editor should now be opened to the **Salutations** custom procedure tab

Program the Salutation

You can now build your salutation however you would like or follow along and have the characters mouth open and close in time with “Hello Madhatter” being displayed. You may wish to watch the Videos or check the Quick Reference Guides for the How To: *Using Procedures Overview* and other How To materials for the code editor.

20. Add a **say** procedure and use *Custom TextString* to input “Hello Madhatter!”
21. Select **getMouth** joint from the object menu drop down. Use the object drop down showing “this” and use the right arrow to navigate to the joint list.
22. Drag in a **turn** statement and select *forward* and a value of .125
23. Drag in another **turn** statement or copy the first and set the direction to *backward* and the value to .125
24. Create a *do together* code block and all of the above statements into it
25. Create a *do in order* code block inside of your *do together* code block and add the mouth turn commands into that.
26. Select add details and change the *duration* of the say statement to be 2 so that it will match the duration of the two turn statements with default durations of 1 each.

Using a Custom Biped Procedure

Use the following steps for each of the guests to the party.

27. Navigate to myFirstMethod

28. Add the new **Salutation** procedure before or after the Madhatter welcomes the guest for each guest.
29. Run your program

Creating a Custom Scene Procedure

myFirstMethod is starting to get pretty long even with the use of these custom methods. If we go on to animate a full party scene we would have to scroll past this whole section of code every time to continue to develop. Let's create a scene procedure that is PartyGreetings and put all of this code into it to make our program easier to navigate, edit, and debug! To create a new custom procedure for the Scene use the following steps:

30. Following the same method as before (Steps 4-7) to create a new Scene procedure named **PartyGreetings**

The code editor should now be opened to the **PartyGreetings** custom procedure tab

Using the Clipboard

Use the clipboard to move the existing code into the new procedure. Follow these steps to help move larger chunks of code at one time:

31. Drag each statement separately into the clipboard or/ create a *do together* block at the very top of the myFirstMethod program
32. Drag all of your existing code into this code block
33. Drag the whole *do together* code block up to the clipboard till you see the code disappear and a white page on the clipboard
34. Navigate to the newly created PartyGreetings procedure
35. Drag the code out of the clipboard into the procedure
36. If the nested code block moving method was used right click on the *do together* code block and select dissolve
37. Navigate back to myFirstMethod and select the scene object from the object list
38. Add the newly created method for PartyGreetings
39. Run your program

Iterating a Design - Planning for Parameters

It seems odd that everyone greets the MadHatter the same way. Use a parameter to allow you to display a different response from each character.

The CheshireCat walks up to the MadHatter

The MadHatter says “Welcome to the Tea Party!”
The CheshireCat says “**Hey** MadHatter!”

The PlayingCard walks up to the MadHatter
The MadHatter says “Welcome to the Tea Party!”
The PlayingCard says “**Howdy** MadHatter!”

Again repeat the change for all the characters

Using a Parameter

Add a parameter and use it to customize the Salutation of each character.

40. Add a parameter with a TextString value type named WhatToSay to the Salutation procedure. You will get a warning that you have existing instances of this method and will need to accept the error to move on.
41. Add the WhatToSay parameter to the **say** statement in place of the “Hello Madhatter!” in the Salutation procedure
42. Navigate to PartyGreetings custom procedure and update all null parameters with what you want the characters to say
43. Run your program