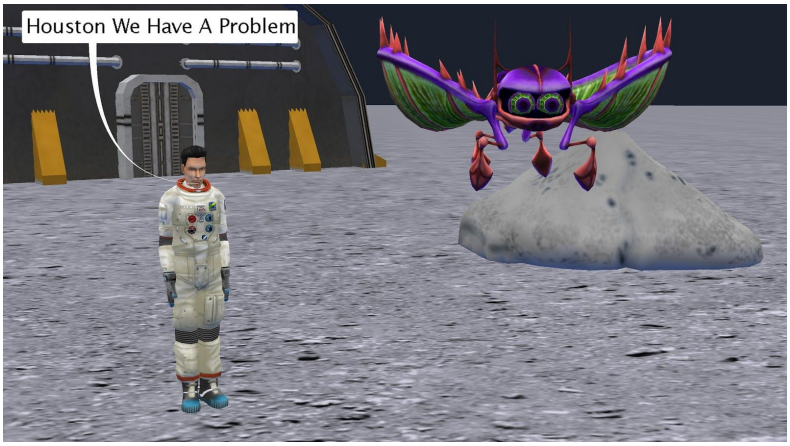
Alice Exercise

Tutorial: Building a Program



Introduction – “Houston we have a problem”

This tutorial exercise provides step-by-step directions for building a simple animation. Here you will explore the different methods that can be used for each skill. This tutorial will provide an overview of the Code Editor in Alice and teach you the basic skills to get you started creating programs.

You will need to access other printed materials or have access to the alice.org website to view the *How To* videos called out within these materials. Materials can be downloaded and printed for offline use from Alice.org

Don’t forget to save your projects frequently.

The Story

In programming it is a good idea to plan out your program before you begin. For this project we are providing you with the output of the design phase to use as reference for the creation of this animation. We have a whole lesson on design that we encourage you to look at.

The story as a script:

EXTERIOR. MOON SURFACE -

An astronaut is out for a space walk in front of a space lab complex.

An alien slowly descends above the astronaut and begins wiggling its many legs

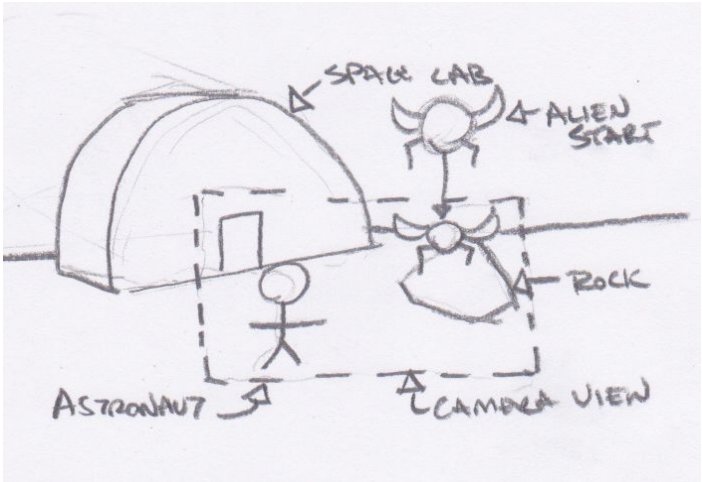
ASTRONAUT
Houston we have a problem

A storyboard of the story:

Setting Up the Scene

To set up your scene you can either open the supplied starter world and skip to the adding procedural methods section or you can create the scene described from scratch.

To determine what needs to be in the scene you can reference the script or storyboard or create a unique set or level design for the world. This set design calls out all of the key scene elements. It is important to note that the Alien begins off the visible scene from the view of the starting camera.

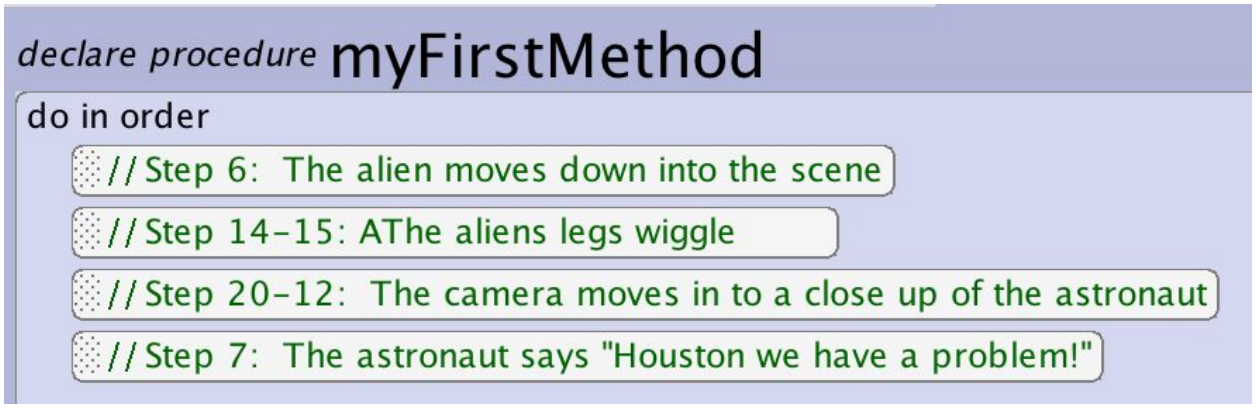


For the following steps, you may wish to check the Scene Editor Overview *How To* video and check the corresponding Quick Reference Guide.

1. Start Alice.
2. Navigate to the *File System* tab and browse to the prepared starter world saved file or
3. Select the ***Moon*** template from the Blank Slates section of the **Select Template Dialog** box.
4. Save your world with an appropriate name like HoustonWeHaveAProblem.
5. Add a startingCameraLocation camera marker. If you need to move the camera around to setup your scene you can always return to this camera marker for the starting camera position and to check your camera framing.
6. Build a moon scene similar to that pictured above:
   a. Select an **Adult** object from the gallery. Choosing Adult from the gallery will bring up the Sims builder allowing you to customize many features of a human character, including the ability to select the astronaut themed outfit.
   b. From the ***Outer Space*** section of the *Browse Gallery by Theme* tab of the Gallery add the **MarsOutpostBunker**, **Boulder,** and **AlienRobot** or build your own simple moon setting but be sure to add the **AlienRobot.**
7. Move objects around the scene so that the **AlienRobot** is placed just out of sight above the camera view of the scene.
   a. You can do this by putting the alien in the scene where you want it to be after it moves, and then using a one shot to move it up and out of the scene.
   b. Keep track of the direction and distance you use to move the alien off the scene so that you can do the opposite when programming the alien's entrance.
8. If needed, navigate back to the code editor.

Planning Your Program (Comments)

If you used the starter world you will see that the basic outline of the animation has been inserted as comments. You can then add the code following each statement in segments. If you built your own scene try adding in comments to plan your program before beginning to add code statements.



9. In the myFirstMethod tab of the code editor add comments outlining the major parts of the animation.

Adding Procedural Methods (Statements)

Let's start by animating the major parts of the story. The Alien enters the scene and the astronaut then says "Houston we have a problem".

For the following steps, you may wish to watch the Using Procedures Overview *How To* video and check the corresponding Quick Reference Guide.

10. Animate the **AlienRobot** entering the scene:
    a. Select the **AlienRobot** object from the object menu.
    b. Drag the **move** tile into the *myFirstMethod* tab of the editor.
    c. From the prompted drop downs select the **direction down** and the **distance .25**.
11. Animate the astronaut saying "Houston we have a problem":
    a. Select **AdultPerson** (or the name of the character you created) from the object menu drop down.
    b. Drag the **say** tile into the *myFirstMethod* tab of the editor after the first statement.
    c. Select the Custom TextString from the pop up drop down.
    d. Input "Houston we have a problem" in the input field.
12. Save the project.

Testing, Editing, and Modifying Procedures

As you build your animation you will want to iterate often by running your program to test it and then make edits and modifications to smooth out the animation. For the following steps you may wish to watch the Using Procedures Overview *How To* video or check the corresponding Quick Reference Guide.

13. **Run** your world to test to see if the **AlienRobot** enters the scene and stops at the desired height:
    a. Press the **Run** button found on the *camera view* window.
    b. Use the **restart** button to view it as many times as needed to analyze any changes.
    c. Close the runtime window to return to the Code Editor.
14. Edit and Test the statements until the **AlienRobot** moves to your desired height in the scene:
    a. Adjust your **AlienRobot**'s **move distance** by selecting a new value in the drop down or by selecting the **custom value** input from the drop down and typing in a value.
15. Add optional parameters to your existing statements to change the timing of the animation:
    a. On the **AlienRobot**'s **move** statement select **add detail** and add **duration** from the drop down and select a time value from the list to set how long the move will take.

b. On the **AdultPerson say** statement select **add detail** and add **duration** from the drop down and select a time value from the list to set how long the text will be visible on the scene.

16. **Run** your world to test the amount of time needed for the **RobotAlien** to descend into the scene and the length of time needed to read the text that appears on the screen:
    a. Continue to change if needed and **Run** to test until you are happy.
17. Save the project.

Programming SubJoints

To make the AlienRobot more energetic for the next step you will animate parts of the alien to give it more character. For the following steps, you may to refer to the image above, or use your own ideas. You may wish to watch the Manipulating SubJoints *How To* video or check the corresponding Quick Reference Guide.

18. Animate a leg of the **AlienRobot**:
    a. Select the **BackLeftKnee** using the object select menu first selecting the **AlienRobot** and then using the side arrow to navigate to the joint.
    b. Select the **turn** tile and drag it to the editor from the procedure list and set it to turn **backward 0.25** revolutions.
    c. Add another statement that has the same joint **turn forward 0.25** to return it to the original position. Instead of creating a new statement from scratch you can hold down option and click and drag the created statement to duplicate it. You can then edit just the direction to quickly create the new statement.
19. Animate the other legs of the **AlienRobot**:
    a. Add move procedures to make the BackRightKnee turn backward and forward 0.25. You can build them from scratch or you can copy the existing procedures and edit the object of the statement.
    b. Add more procedures for the remaining legs or experiment with other subpart animations for the alien.
20. **Run** the world to test how it works and make any modifications.
21. Save the project.

Using a Do Together

To make the animation of the legs feel more natural and not happen in a robotic sequence you will mix do in orders and do togethers. For the following steps, you may wish to watch the Videos or check the Quick Reference Guides for the How To: *Using a Do Together*

22. Have a couple of the legs move at the same time by putting their movement into a *do together*:
    a. Drag a *do together* block in the *myFirstMethod* editor.
    b. Drag the **AlienRobot** subjoint movements to be timed together into the *do together* block.

c. Experiment with nesting *do together* and *do in orders* to get a set of movements that you like. **Hint – putting two opposite directions inside a do together will cause them to negate each other and nothing will happen.**
d. **Run** and test your program to iterate till you are happy with the animation.

23. Save the project.

Programming a Camera Move

For the following steps, you may wish to use the video associated with this exercise as a reference or come up with your own camera moves. You will use a camera zoom to make the animation more dramatic by zooming in on the astronaut when he/she delivers the punchline. You may wish to watch the Setting Up and Using Camera Markers video or check the corresponding Quick Reference Guides.

24. Navigate to the Scene Editor to set up the required camera markers. (If you used the provided starter world this step has been completed and you should see a second camera marker called closeUpCameraLocation):
    a. Add a **camera marker** at the camera's current position for the starting scene camera. Give the camera an appropriate name, such as wideScene.
    b. Move the **camera** to a close up view of the **astronaut** and add another camera maker. Give the camera an appropriate name, such as **astronautClose.**
    c. Move the camera back to the wideScene **camera marker** to insure the camera is in the right place for the start of your animation. To do this select wideScene from the **camerMarker** list and click the button that indicates it will move the camera to the camera marker (left button).
25. Return to the Code Editor to program the camera move:
    a. Select the **camera** object from the object drop down menu.
    b. Select the **moveAndOrientTo** tile and drag it into *myFirstMethod* and place it in the program in the right location. **Hint – using just the move tile may leave you wondering why the camera is not pointed where you want it. This might be because you used the turn or roll controls when moving the camera. You will need to orient to the new turn and roll not just the new x, y, z**.
    c. In the drop down, select the **astronautClose** camera marker for the target.
    d. **Run** your program to test the camera move.
    e. Use **add details** to adjust the **duration** and **animation style** to achieve the camera move you like. **Hint – the default duration is always 1 second. For a clean jump cut, you will need to custom set the duration to 0.**
26. Save the project.