

Regis University CC&IS
CS210 Introduction to Programming
Alice Programming Assignment 2: Methods and Variables

This program will create an animation which uses scene/shot methods, class methods, object and camera markers, and local method variables.

The primary purpose of this assignment is to learn how to **define and call methods** (both scene and class methods) and **define and use variables**. Even if your animations perform the specified tasks, if you do not create the required variables and implement the specified procedure, scene, and shot methods, you will lose significant points.

Animation Overview (details will follow later)

- Your animation will contain three objects, a snow person, an animal, an ice formation.
- In scene 1:
 - The camera will point at the snow person only. The snow person will rock back and forth, move forward, say something, and move back.
 - The snow person will calculate the degrees above freezing.
 - The camera will move to point toward both the snow person and animal.
 - The animal will turn and move towards the snow person. The animal will turn its head, think something, and turn its head back to its original position.
- In scene 2:
 - The camera will move to point towards the ice formation.
 - The ice formation will turn, roll, and turn it back to its original position.

Program Requirements

Create an Alice program that *minimally* does the following:

- Start with a **Snow** blank slate.

Setting the Scene

- Adjust the camera view so that the horizon is approximately in the *middle* of the scene.
- Add a snow person from the Biped classes to your scene (from either the SnowMan or SnowWoman class).
 - Position the snow person in the middle of the scene, facing front.
- Add an animal from the Quadraped class to your scene
 - Size the animal and snow person so that they are proportional to each other (as they would be in the real world).
 - Position the animal out of view, off the right side of the scene, facing right.
- Add an Ice object from the Prop class to your scene (choose from an Iceberg, IceBlock, IceBlockHalf, or IceFloe).
 - Size the ice object to be approximately the same height as the snow person.
 - Position the ice object out of view, off the left side of the scene, facing forward.


****Remember that saving your program and re-opening it will generally fix any Alice bugs.**

Adding Markers

- Center the camera so that only the snow person is visible in the middle of the scene.
- Add a camera marker to save the position of the camera (starting camera position).
- Add an object marker to save the starting position of the snow person (centered in the scene).
- Move the snow person forward towards the camera, until you see a close up of the snow person's face (note: you may need to use the Z position under the Properties panel to move far enough forward). Add a second object marker to save the snow person's position (close to the camera).
- Use the snow person starting position object marker that you previously created, and an object marker button, to move the snow person back to its original position.
- Move the camera a bit to the right, so you can see both the snow person and the animal at the same time. Add another camera marker to save the position of the camera (view of snow person and animal).
- Move the camera again, this time to the left, so that it now points at the ice object, with the ice object on the left side of the scene. In this camera position, you should not be able to see the snow person or animal. Add a camera marker to save the camera position (view of ice).

Coding the Actions with Multiple Scene/Shot Methods

NOTE: When using the markers to position either the camera or the objects, you must use the **moveAndOrientTo()** method in order for the positioning to be correct


Click the **Edit Code** button to return to the Code Editor. From the  **Scene** tab:

- Add 2 scene methods (**doScene1** and **doScene2**)
- Add 2 shot methods for use within the first scene (**doShot1a** and **doShot1b**)

Add code to **myFirstMethod** to:

- Call both of the scene methods (**doScene1** and **doScene2**)

Creating the Class Method to use in doShot1a

- From the class navigator drop down 

Within the *Snow person class* that you created your object from, add a **rock** method.

- The method will implement rocking a snow person back and forth once, as follows:
 - Create a variable to hold a **roll left amount** and initialize it to 0.05.
 - Create a variable to hold a **roll right amount**.
 - Use a math equation to store **twice** the value stored in the **roll left amount** variable into this variable.
 - Send a message to roll the object left by the **roll left amount** stored.
 - Send a message to roll the object right by the **roll right amount** stored.
 - Send a message to roll the object left again by the **roll left amount** stored (back to the original position).

Adding code to the doShot1a method

- Send messages (using the **rock** method) to make the snow person rock back and forth *twice*.
- Use a previously created object marker to move the snow person forward, close to the camera.
- Have the snow person say something other than the default string.
- Use a previously created object marker to move the snow person back to its starting position.
- Create a variable to hold the *current temperature*.
- Use a function to have the snow person ask the user what the current temperature is, and store the input value into the *current temperature* variable.
- Based on the user input, calculate the degree difference between the current temperature and a freezing temperature (32 degrees), and store the value into a **difference** variable.
- Have the person give an explanation and state the difference between the current temperature and a freezing temperature.

Adding code to the doScene1 method

- Use a previously created camera marker to position the camera to its starting position for the start of the first scene.
- Use a previously created object marker to position the snow person object to its starting position. (Hint: set durations to 0 to make the movements instantaneous)
- Add the code to call **doShot1a**

Run your program to test the **doShot1a** method. Debug if necessary.

Adding code to the doShot1b method


- Use a previously created camera marker to position the camera to view both the snow person and animal at the same time.
- Create a variable to hold a *distance to move*.
- Use a function to calculate the distance from the snow person to the animal, and store the resulting value into the *distance to move* variable.
- Modify the value stored in the *distance to move* variable by subtracting 2 from it.
- Turn the animal 180 degrees, to face the snow person.
- Move the animal towards the snow person, by the amount stored in the *distance to move* variable (i.e. so the animal stops within a distance of 2 units from the snow person).
- Turn the animal's head slightly to left (towards the camera).
- Have the animal think something.
- Turn the animal's head back to its original position.

Adding code to the doScene1 method

- After calling **doShot1a**, add the code to call **doShot1b**

Run your program to test the **doShot1b** method. Debug if necessary.

Creating the Class Method to use in doScene2

- From the class navigator drop down 
 - *Within* the **ice object class** that you created your object from, add a **roll** method.
 - The method will implement **one** forward roll of an ice object as follows:
 - Create a variable to hold the height of the object.
 - Use a function to get the object's height and store the resulting value into the **object height** variable.
 - Create a variable to store a **forward distance**.
 - Calculate and store 2/3 the **object height** into this variable. Use both multiplication (x2) and division (/3), instead of rounding.
 - Send messages to roll the object once. One roll will be equivalent to:
 - one **move** up (using the **object height** as the amount)
 - followed by a complete **turn** (1 rotation) forward
 - followed by two moves performed at the same time:
a **move** forward (using the **forward distance** as the amount) together with
a **move** downward (using the **object height** as the amount).

Adding code to the doScene2 method

- Use a previously created camera marker to move the camera to view the ice object only.
- Turn the ice object a quarter turn so it is facing towards the right side of the scene.
- Use the **roll** method to make the ice object roll forward once.
- Turns the ice object back a quarter turn, to face the camera.

Run your program to test the **doScene2** method. Debug if necessary.

Final Testing

- Run, test, and debug your Alice program, until it works correctly.
- Make sure the duration of each movement in your animation is long enough to view comfortably.

Extra credit* (5 pts)

**Save a copy of the normal part of the assignment first, and then save it under a new name before starting on the extra credit, in case your efforts on the extra credit are unsuccessful.*

Optionally you may add another action to the animation for extra credit, as follows:

- *Within* one of the object **classes**, create a **new class procedure** method that:
 - Initializes at least one **variable** to some value.
 - Causes an object of that class to perform a new behavior, implemented using the **variable** value(s).

NOTE: The new behavior cannot be a behavior implemented by any of the course materials, or be a behavior already implemented in this program or program 1.
- Add an additional scene, called **doExtraScene**.

- Add code to **doExtraScene** to:
 - Position the camera to point to the object you will animate.
 - Call your new class procedure method, causing the object to perform the new behavior.
 - Do anything else you want.
- Call **doExtraScene** from **myFirstMethod**

Submission

This programming assignment is due by midnight of the date listed on the Assignments page.

Submit your program source code (the **.a3p** file) to the **Alice Prog Assn 2** submission folder (located under the **Assignments/Dropbox** tab in the online course).

Before submitting your program file, you **MUST** re-name it as follows:

LastNameAliceAssn2.a3p

For example: **SmithAliceAssn2.a3p**

Grading

The rubric that will be used to grade your program is linked on the same assignment page that you downloaded this file from.

WARNING:

*Programs submitted more than 5 days past the due date will **not** be accepted,
and will receive a grade of 0.*