

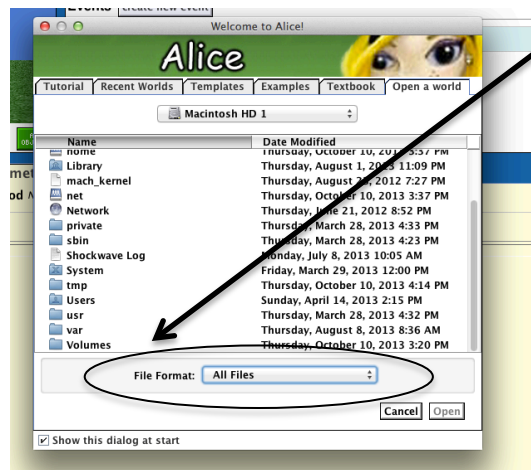
Challenge 1: Tami's World

In this challenge, you will:

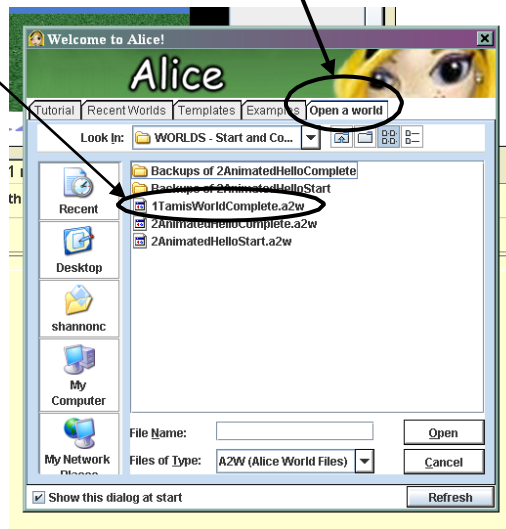
- Practice using the main areas of Alice including the Scenes window, Objects tree, details panel and method editor
- Add objects from the local gallery
- Program an object to talk

A First you will look at what your completed challenge will look like:

1. Open Alice.
2. Navigate to Alice Worlds folder on f: drive. ****Make sure file format is set to All Files****



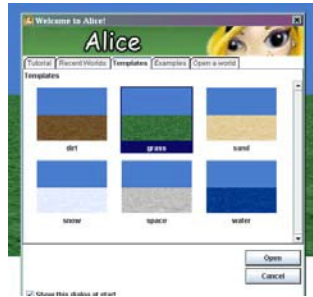
3. In the box that appears, make sure the *Open a world* tab is selected and open **1TamisWorldComplete.a2w**.



4. Click on *Play* button to see what the completed challenge looks like.
5. Click on the X in the upper right OR click on *Stop* button to close the world.

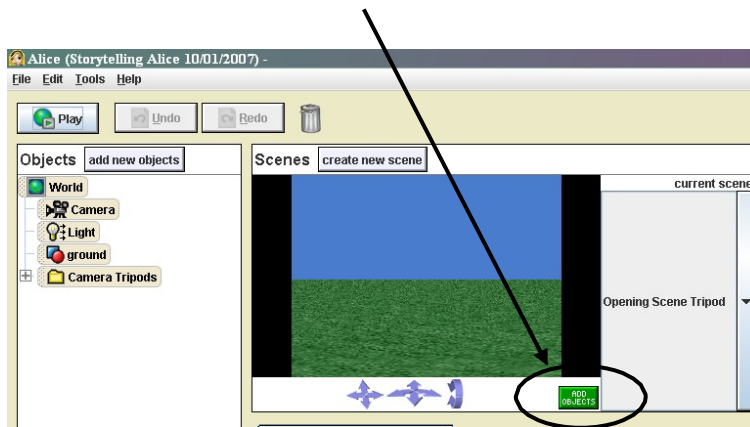
B Now you will make your own Alice world.

1. Go to the *File* menu and chose *New World*
2. In the box that appears, make sure that the *Templates* tab is selected and click on *grass* and then *Open*.

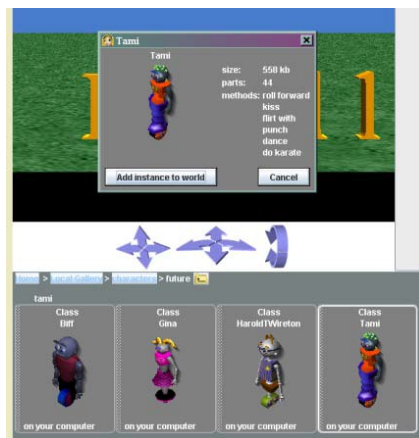




C Next you will add a character to your world.

1. Click on the green *ADD OBJECTS* button.



2. Click on one of the *Characters* folders at bottom of screen.
3. Click one of the *Class* tiles to open the information box.
4. Click on *Add instance to world* button to add a copy of the character to the world.



5. Click on the DONE button  or *done adding objects* button  (upper left in Object window) to return to the Method editor.
6. Go to the *File* menu and choose *Save World As* to save your world in the appropriate Student Alice Projects folder on F: drive.

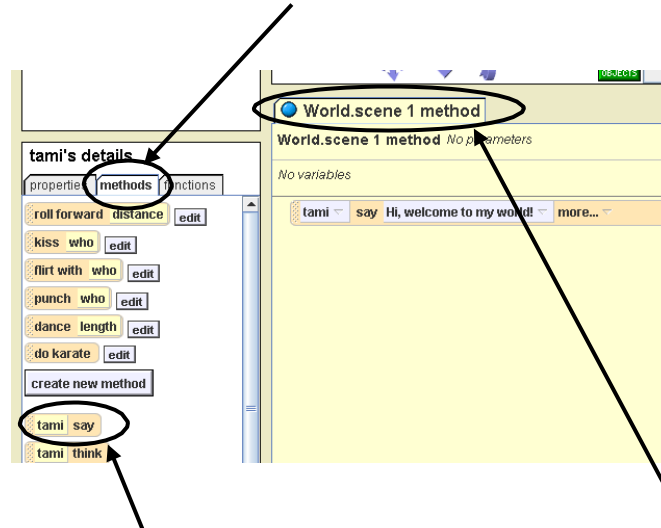
HINT: See your file saving instructions or ask your teacher where to save files.

D Now you will make your character talk:

1. Click the *character* tile in the *Objects* tree.



2. In the *details* panel, select the *methods* tab.

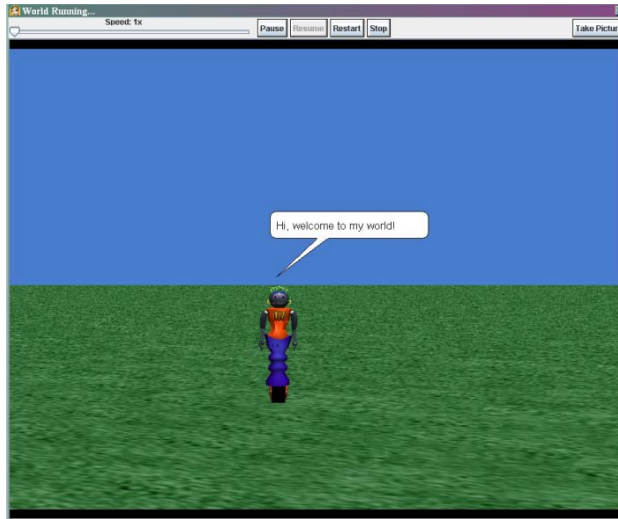


3. Click and hold the character *say* tile and drag and drop into *World.scene1 method*.

HINT: Your version might say *World.myfirst.method*

4. Select *other* on the menu that appears.
5. In the *Enter a string* box that appears, type *Hi, welcome to my world!* (or some other greeting) and click OK.
6. Hold down the triangle next to *more* and point to *duration*.
7. Click *other* on the menu that appears.

8. In the *Custom Number* box, type 4 and click OK
9. Save your world by selecting *Save World* under the *File* menu.
10. Click on the *Play* button (upper left) to make sure your world works!

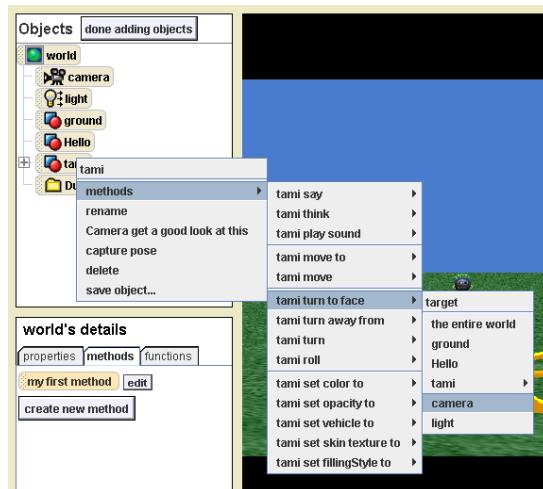


If your world does not work the way it should, ask the teacher for a copy of the code and compare it to what you programmed.

Challenge 2: Animated Hello

In this challenge, you will:

- Practice using the main areas of Alice
 - Learn how to program parts of objects
- A. First you will look at what your completed challenge will look like:
1. Open Alice.
 2. In the box that appears,
 1. Make sure the *Open a world* tab is selected
 2. Make sure file format All Files is clicked
 3. Navigate to the Alice Worlds folder
 4. Open **2AnimatedHelloComplete.a2w**.
 3. Click on *Play* button to see what the completed challenge looks like.
 4. Click on the X in the upper right OR click on *Stop* button to close world.
- B. Now open the Alice world you will use to start the challenge:
1. Go to *File* menu and select *Open World*.
 2. Find the world titled **2AnimatedHelloStart.a2w** and open it.
 3. *Play* the world to see what is already programmed into it (animated hello).
 4. Close to return to Scenes and Method editors.
- C. First you need to put your character into this world.
1. Click on *ADD OBJECTS* > *Characters* folder > Choose a *Character* class > Choose a *Character* > *Add instance to world* > DO NOT CLICK DONE YET
- D. You will now do three things to put your character in the right place before your world first plays.
1. Right click on the *character* tile in the Objects tree.
 2. Point to *methods* on the menu that appears.
 3. Point to *character turn to face* (You will only see a slight movement of the character.)
 4. Click on *camera*



5. Again right click on the character tile in the Objects tree.
6. Point to *methods* on the menu that appears.
7. Point to *character turn*
8. Point to *right*
9. Click on $\frac{1}{4}$ revolution

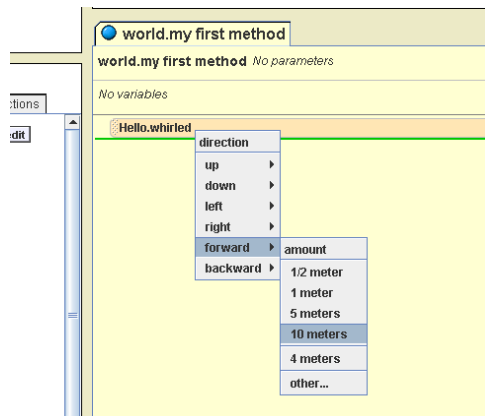
HINT: Whenever you program objects to turn in a certain direction, pretend you are that object or character in the world to figure out which way to turn (opposite of how you see it on the screen).

10. Again, right click on the *character* tile in the Objects tree.
11. Point to *methods* in the menu that appears.
12. Point to *character move*.
13. Point to *backward*.
14. Click *10 meters*. (Character should now off the screen. If not check with your teacher.)
15. Go to the *File* menu and choose *Save World As* to save your world in the appropriate class period folder in Student Alice Projects folder on f: drive.
16. Click DONE button or *done adding objects* button to return to the Method editor.

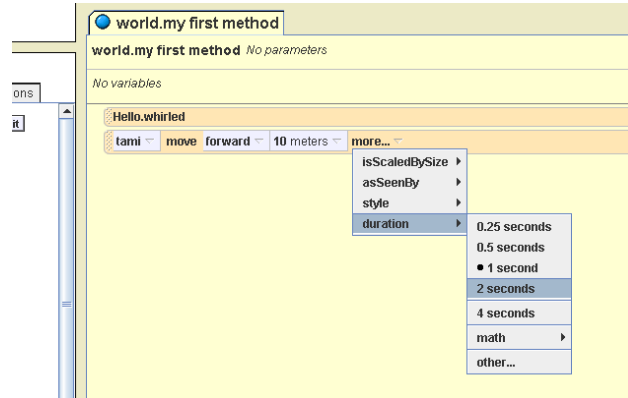
E. Now you will make your character move when your world plays:

1. In the Objects tree, click on the *character* tile.
2. In the Details panel, make sure the *methods* tab is selected, click on *character move* tile and drop into *world.my first method* below the *Hello.whirled* tile.

HINT: A **green line** will appear where your detail will be placed in your code list inside the *world.myfirstmethod* area.



3. Select *forward* on the menu that appears.
4. Click on *10 meters*
5. Click on the triangle next to *more* at the end of the *character move forward* tile and point to *duration*.
6. Select *2 seconds*.



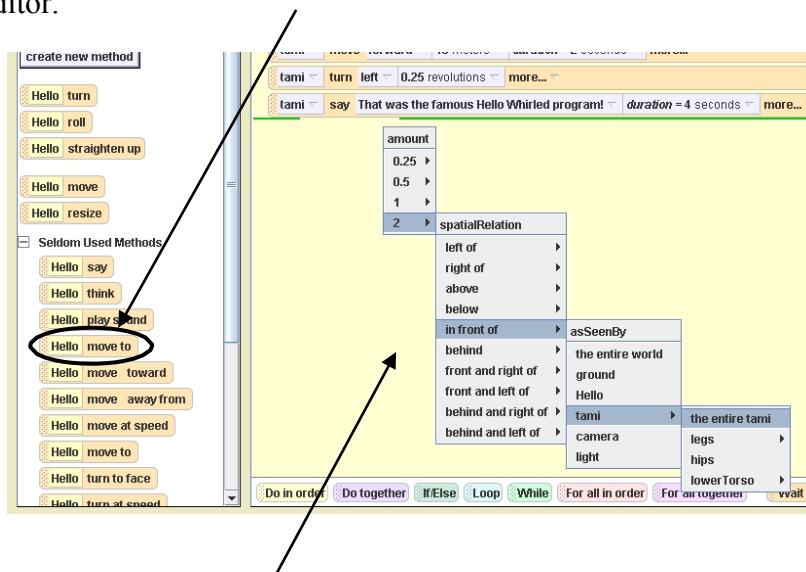
7. In the Details panel, drag a *tami turn* tile and drop it into *world.my first method* below the last tile.
8. Select *left*.
9. Select $\frac{1}{4}$ *revolution*.

F. Make your character say something:

1. From character's Details panel, drag and drop *character say* tile into *world.my first method* **below** the last tile.
2. Select *other* > type *That was the famous "Hello Whirled" program* > click OK.
3. Hold down the triangle next to *more* > point to *duration* > Click *other* > type *4* > click OK.

G. Make the spinning Hello drop on top of your character:

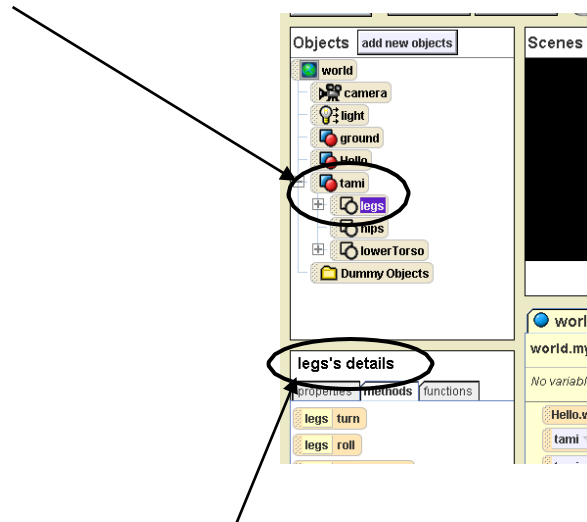
1. In the Objects tree, click on Hello so you can see its Details panel below.
2. Drag and drop the *Hello move to* tile **below** the *character say* tile in the Method editor.



3. Select *character*, then *the entire character*.
4. At the end of that tile, click the *more* triangle > *duration* > *0.25 seconds*.

H. Now make just your character's lower body spin – first to the left and then to the right:

1. In the Objects tree click the plus (+) sign next to the *character* tile.
2. Click the tile for whatever body part listed is the lower body



3. In the Details panel (this is for the body part only), drag and drop the *body part* tile below the last tile in *world.my first method*.
4. Select *left* and then *4 revolutions*.
5. In *world.my first method*, right click the *character body part turn* tile and select *make copy* on the menu that appears. There are now 2 copies of the *body part turn* tile.
6. In the bottom copy, click the triangle next to *left* and select *right*. (Now the *body part* will turn left and then right.)

I. Make character fall over and say one last thing:

1. Click on character in the Objects tree to see character details below.
2. Drag and drop character tile under the last tile in the Method editor.
3. Select *backward* > $\frac{1}{4}$ revolution.
4. Under *more* triangle, select duration > 0.25 seconds.
5. Make *character say I don't write this stuff* > duration > 4 seconds.

J. Save your world!

K. Play your world to see how it works.

If your world does not work the way it should, ask the teacher for a copy of the code and compare it to what you programmed.

Challenge 3: Harry the Lion

In this challenge you will:

- a. Practice skills you have already learned but with fewer directions
 - b. Learn how to change an object's properties like color and visibility
 - c. Have an opportunity to add your own methods
- A. First look at what your completed challenge will look like:
1. Open Alice
 2. File > Open World > Navigate to Alice Worlds Folder on f: drive > **3HarrytheLionComplete.a2w**
 3. Play to see what the completed challenge looks like. Click *Stop* to close the world.
- B. Now set up your own Alice world.
1. *File > New World.*
 2. Select the any template.
 3. *Save World As* in the appropriate folder on F: drive
- C. Add a character to your world:
1. ADD OBJECTS > *Characters* folder > *scary* folder > Harry the Lion > *Add instance to world.*
 2. [**NOTE:** If your character is already in the middle of the screen, skip this step.]
Click on and drag your character to place him in the middle of the screen facing the camera as shown.



3. Click DONE to return to the Method editor

D. Make your character say he is going to make a noise and then make him do it:

1. In the Objects tree, select character
2. *Character's details > character say > other > Now I'm going to _____* (fill in the blank with the noise your character is going to make).
3. *more > duration > 2 seconds*
4. *Character's details > play sound*

NOTE: You can use one of the installed sounds or you can import a sound file. For instance in my Challenge 3, I imported a penguin sound.

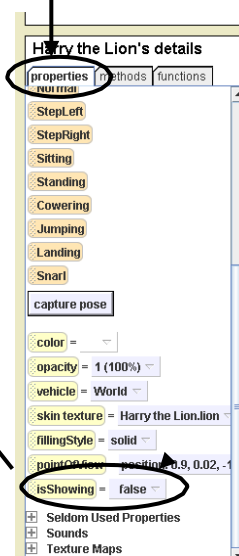
If you want to import a sound file, do the following before step 4 above:

1. Save your world.
2. Minimize (not close) Alice
3. Go to soundboard.com (most of the sound files here are public share, free and in MP3 format.)
4. Find a sound you like. (Don't spend all of the rest of the class looking for a sound. Pick one and move on.)
5. Save the sound to the desktop of the computer
6. Got back to step D. 4. above.

NOTE # 2: Regardless of whether or not you import a sound, be sure to set the duration! You may or may not want the duration set to the entire length of our sound file. 3-5 seconds of the sound should be enough.

E. Next make Character say he is going to disappear and then make him/her do it:

1. *Character's details > Character say > other > Now I'm going to disappear!*
> OK.
2. *more > duration > 2 seconds.*
3. In character's Details, click on the properties tab. Scroll down until you find his *IsShowing* tile.



4. Drag and drop the *isShowing* tile under the last tile in the Method editor.
5. Select *false* (this says that character IS NOT showing – he has disappeared).
6. *More > duration > other > 3 > OK.*

F. Now make character reappear:

1. Right click on *Character IsShowing* tile in *world.my first method*.
2. Select *make copy*.
3. In the bottom copy, change *false* to *true* (this says that character IS showing).
4. *Save your world*.
5. *Play* the world to see your character make sound, disappear and reappear.
6. *Save your world*.

G. Now make your character say and do other things that you choose (two maximum):

1. For ideas, look at the tiles in your character's methods (cower, roll, walk, etc.).
2. *Play* your world after each action you program to make sure it works the way you want it to.

H. Now make your character say *Goodbye* and then turn and walk out of sight.

HINT: It usually takes *10 meters* for an object to move off the screen to where you can't see it.

I. *Save your world* one last time.

J. *Play* it to make sure it works the way you want it to.

If your world does not work the way it should, ask the teacher for a copy of the code and compare it to what you programmed.

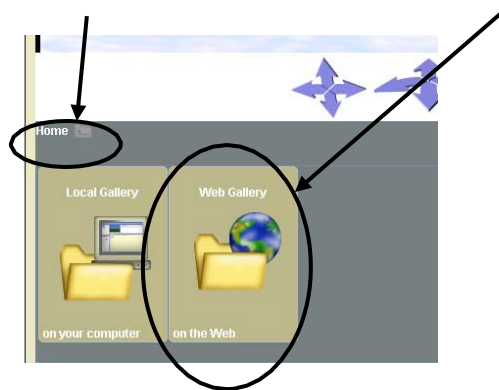
Bonus Challenge 3.1: Add Your Own Methods

In this challenge, you will:

- a. Practice skills you have already learned but with fewer directions.
- b. Learn how to add objects from the Alice gallery on the Web.

This is similar to the Harry the Lion Challenge but without any directions. You will start a new world, add any character you want and add methods that tell the character what to do.

- A. Start Alice and start a new world with any one of the six templates – dirt, grass, snow, sand, space or water.
- B. *Save* your world in the appropriate folder.
- C. Add any character you want from either the *characters* folder in the *Local Gallery* or from the *Web Gallery*. To add a character from the *Web Gallery*:
 1. Click on *ADD OBJECTS*, then *Home* on bottom left and then on *Web Gallery*.



2. Click OK if you get a pop up window that says: “*Web gallery may be slow.*”
3. Look through the folders until you find a character that you like.
4. Click on the character and add to the world the same as you’ve done before.

- D. Make your character say what it will do and then have the character do it.

HINT: Each character has their own set of methods already built in. Remember to look in *Seldom Used Methods* in the Details panel for more methods. *Seldom Used Methods* may not be an option in your version of Alice. If it is not, the list of methods is complete for your character.

- E. After your character does at least two different things, play your world to make sure it works correctly and fix any problems you find.
- F. *Save* your world before closing Alice or starting another challenge.

Bonus Challenge 3.2: Do Together and Do In Order

In this challenge, you will:

- Practice skills with fewer directions.
- Learn how to use the “Do together” command.
- Learn how to use “Do In Order” command
- Learn how to import sounds and make it play when a character does something.

A. First look at a completed version of this challenge:

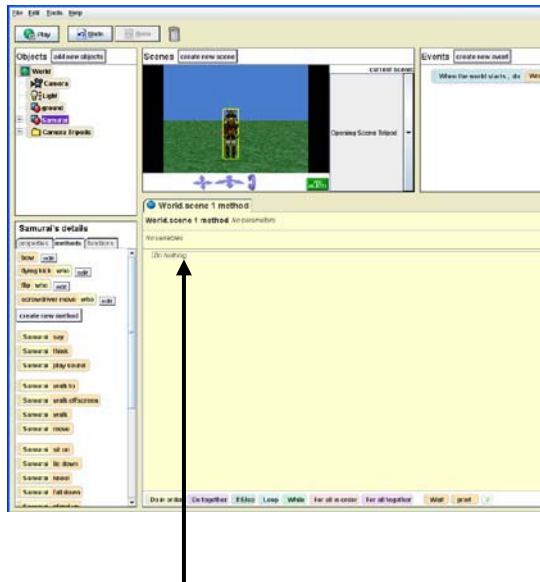
1. In Alice, open and play: **3.2SoundComplete.a2w**.

B. Now set up and save your own world.

1. *File > New World*.
2. Select the *grass* template.
3. *Save World As* in the appropriate folder.

C. First set the scene by adding a character and making him bow.

1. Add the a character. Remember you can search the Gallery for other characters, though the version of Alice we are using is the most recent; therefore, the list of characters in the local Gallery should be a complete listing of available characters.
2. Place your character so it looks like the samurai in the picture and click DONE



3. Drag the *Do Together* tile from the bottom of the methods area and put it in place of *Do Nothing*

4. Drag *Do In Order* tile from the bottom of the methods area and put it in place of *Do Nothing* ***inside*** the *Do Together* tile
5. Click the plus sign next to your character tile in the Objects area (this will show body parts).
6. Click to highlight the upper body tile in Objects area.
7. Drag the upper body turn tile ***inside*** *Do In Order* tile > forward > 0.25 revolution > duration 2 seconds
8. Right click upper body turn forward tile you just programmed.
9. Click make a copy.
10. Change turn forward to turn backward and leave everything else the same.
11. *Play* your world to test that it works correctly and *Save* again.

D. Now add the sound of a gong that will play when the samurai bows.

1. Drag and drop the character play sound tile into the *Do Together* area, under *Do In Order* in the method editor area. (Be sure the green line is ***NOT*** in the yellow tile inside the purple tile.
2. Select a sound to be played.
3. Select *duration* > *Full Length*.
4. Play and save your world.
5. Bonus to the Bonus: If you want to and have time, you can use more sounds, either from the sounds folder or from the Internet. If you have a microphone, you can also record your own sounds by using the *record* button. (You may need to see Senor Compuman Morter for help with this. Also, see Important Notes About Sound below.)

If your world does not work the way it should, ask the teacher for a copy of the solution and compare it with the commands you programmed.

Important Notes About Sound:

- Sounds need to be mp3 or wav files.
- According to copyright laws, only 30 seconds of a song can be used in your world. For more information, see: www.utsystem.edu/ogc/INTELLECTUALProperty/cmcguid.htm
- You can use the free sound-editing program, Audacity, to edit your song or to create your own audio file. See: audacity.sourceforge.net/download/

Challenge 4: Chicken Farmers Advertisement

In this challenge you will:

- Use blue camera controls.
- Practice skills you have already learned – especially methods that affect parts of an object.
- Practice skills with limited instructions.

A. First look at what your completed challenge will look like:

1. Open Alice.
2. Open **4ChickenFarmersAdComplete.a2w**
3. *Play* to see what the completed challenge looks like.

B. Open the Alice world you will use to start the challenge:

1. *File > Open World > 4ChickenFarmersAdStart.a2w*.
2. There should just be a cow facing you. It is not programmed to do anything yet.
3. *Save World* in appropriate folder

C. First turn the cow so that you see it from the side and it's facing to the left as shown below.

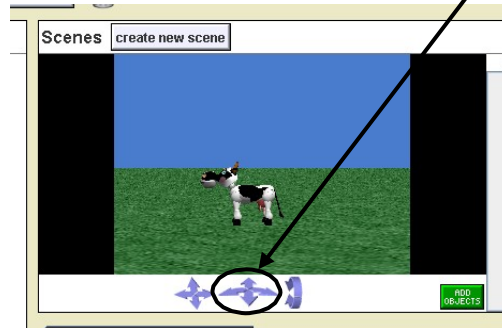
HINT: Start by right clicking on the *cow* in the Objects tree and pointing to *methods* for choices on how to position the cow. Remember when thinking of which direction to turn, put yourself into the scene as the cow.



Adapted from SHELLY/CASHMAN/HERBERT. *Alice 2.0: Introductory Concepts and Techniques*, 1E. ©2007 South-Western, a part of Cengage Learning, Inc. Reproduced by permission. www.cengage.com/permissions.

D. Next, make it so that the cow is off the screen when the world starts:

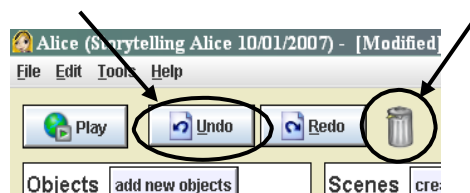
1. Click and hold on the left part of the middle *blue camera control arrow* below the scene to move the camera position to the left so that the cow is off the screen on the right.



E. Now follow the abbreviated instructions to make the cow talk and move.

HINTS:

- For most of the steps select *cow* in the Objects tree and then drag and drop tiles from the cow's Details panel into the Method editor.
- You will have to select the *duration*, *speed*, etc. from the drop down menus that appear after placing the tile in the Method editor. For some you will need to go into the *more* part of the method tiles.
- Some of the instructions are *turn* instructions for the cow's *neck*. For each of these you will need to click on the plus (+) sign next to the *cow* tile in the Objects tree to see that tile and its methods below in the Details panel.
- If you make a mistake you click on the Undo button or drag tiles to the trash can.



1. *Cow.walk times = 10; speed = 2*
2. *Cow.tailSwish times = 2; speed = 2*
3. *Cow.neck turn left 1/4 revolution*
4. *Cow say "Chicken is healthy."; duration = 2 seconds*

5. *Cow say* “Real healthy.” *duration* = 2 seconds
6. *Cow say* “And tasty, too.” *duration* = 2 seconds
7. *Cow.neck turn right* $\frac{1}{4}$ revolution
8. *Cow.tailSwish* *times* = 2; *speed* = 2
9. *Cow.walk* *times* = 2; *speed* = 2
10. *Cow.tailSwish* *times* = 2; *speed* = 2
11. *Cow.neck turn left* $\frac{1}{4}$ revolution
12. *Cow say* “Don’t even think beef.”; *duration* = 2 seconds
13. *Cow.neck turn right* $\frac{1}{4}$ revolution
14. *Cow.tailSwish* *times* = 2; *speed* = 2
15. *Cow.walk* *times* = 11; *speed* = 2

F. *Save* your world.

G. *Play* your world to make sure it works like the completed challenge you looked at in the beginning.

If your world does not work the way it should, ask the teacher for a copy of code and compare it with what you programmed.

Challenge 5: Kayla in Wonderland

In this challenge, you will:

- Create new methods and add them to the world.
- Change the size of objects.
- Use the clipboard to copy and paste methods.

A. First look at what your completed challenge will look like:

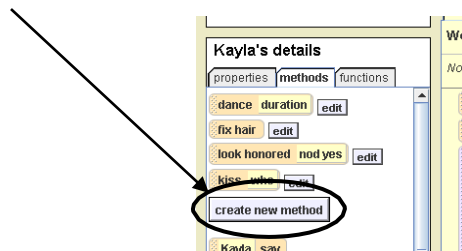
1. In Alice, open and play: **5KaylaWonderlandComplete.a2w**.

B. Next, open and play **5KaylaWonderlandStart.a2w**. The cat says things to Kayla but she does nothing but say something about the cat in the end. You will make Kayla kiss the flower and tree and grow larger and smaller.

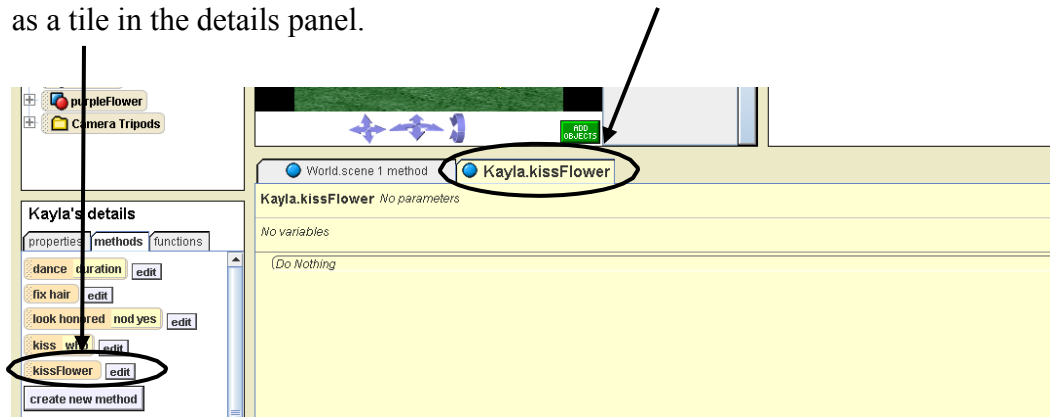
C. First *Save World* in appropriate folder. **DO NOT SIMPLY PRESS THE SAVE BUTTON!!!**

D. Now make a new method that tells Kayla to kiss the flower when the cat tells her to.

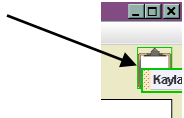
1. Go to Kayla's *methods* in her details panel.
2. Click on *create new method* button and name it *kissFlower* in the box that appears.



Notice that your new method now appears as a tab in the method editor and as a tile in the details panel.



3. Make Kayla turn left .25 revolutions.
4. Make Kayla walk 0.5.
5. In the Objects tree, open Kayla's subparts by clicking on the (+) next to her tile.
6. Click on *torso* to see the methods for that body part listed below in the Details panel.
7. Drag and drop a *torso turn* tile into the *Kayla.kissFlower* method.
8. Select *forward 0.35 revolutions > duration > 2 seconds*.
9. Copy the tile you just created (*Kayla.torso turn forward ...*) by dragging it to the clipboard (in the upper right) and dropping it so the clipboard turns from brown to white.



HINT: To drag a tile that is in the Method editor, you need to click somewhere on the colored part of the tile, not on the white areas.

10. Click and hold the mouse on the clipboard and drag and drop the tile you copied below the last tile in the Method editor. You now have two copies of the *Kayla.torso turn forward* tile in the Method editor.
 11. On the bottom tile, change *turn forward* to *turn backward*.
 12. Add *Kayla turn > right > 0.25 revolutions > more > style > begin gently*
 13. Add *Kayla resize > 2*
- E. Now add the *kissFlower* method you just created to your world.
1. Click on the *World.scene 1 method* tab at the top of the Method editor. This opens the first scene of your world.
 2. Got to Kayla's Details panel and drag a copy of the *Kayla.kissFlower* method tile and drop it between the first tile (*blackCat say Kiss the purple flower*) and the *blackCat say Kiss the tree* tile.
 3. Test your changes by clicking the Play button.
 4. Save your world.

F. Next you will make Kayla kiss the tree when the cat tells her to.

1. Go to Kayla's Details panel, click on the methods tab and click on *create new method* button.
2. Name the new method *kissTree*.
3. Add the following this new method:
 - *Kayla move > forward > 0.5 meters*
 - *Kayla.torso turn > forward > 0.1 revolutions > duration = 2 seconds.*
 - *Kayla.torso turn > backward > 0.1 revolutions > duration = 2 seconds.*
 - Add *Kayla resize > 0.5* (half as big)

HUNT: Remember that you need to go into the subparts of Kayla in the Objects tree to see Kayla's *torso* Details.

4. Place a copy of this new method (*kissTree*) into your world by clicking on the World.scene 1 method tab at the top of the Method editor.
2. Add the *Kayla.kissTree* method tile under the *blackCat say Kiss the tree* tile and before the *Do together* tile.
3. Test your changes by clicking the Play button.
4. Save your world.

If your world does not work the way it should, ask the teacher for a copy of the code and compare it with what you programmed

Challenge 6: Penguin Dance

In this challenge, you will:

- Practice making and using several new methods.
- Learn how to test a method and change the method that starts when the world plays.
- Learn how to make things happen at the same time.

A. First look at what your completed challenge will look like:

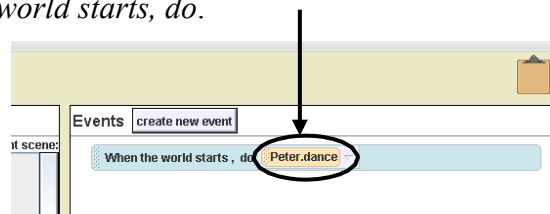
- D 1. Open Alice.
- D 2. Open **6PenguinDanceComplete.a2w**
- D 3. *Play* to see what the completed challenge looks like.

B. Open the Alice world you will use to start the challenge:

- D 1. *File>Open World> 6PenguinDanceStart.a2w*.
- D 2. There should be three penguins facing you, named from left to right Pamela, Peter and Paul.
- D 3. *Save World As 6PenguinDance* followed by your initials.

C. First, you will make Peter Penguin dance.

- D 1. In Peter's Details panel (methods tab), click on *create new method*. Name the method *dance*, and click *OK*.
- D 2. Add the following to this new *Peter.dance* method:
 - *Turn > left > 1 revolution > more > asSeenBy > Paul > the entire Paul > duration = 3 seconds.*
 - *Turn > right > 1 revolution > more > asSeenBy > Pamela > the entire Pamela > duration = 3 seconds.*
- D 3. To test this new method and see Peter dance you need to run the method when the world starts to play. To do this, drag Peter's *dance* tile from his Details panel and drop it in place of the *world.scene 1 method* in the Events editor after where it says *When the world starts, do*.



- D 4. Click on the Play button to see the middle penguin (Peter) circle the penguin on your right and then the penguin on your left.

D. Now you will make Pamela Penguin dance.

- D 1. In Pamela's Details panel (methods tab), click on *create new method*. Name the method *dance*, and click *OK*.

- D 2. Add the following to this new *Pamela.dance* method:

- *Turn >right >3 revolutions >duration = 3 seconds.*
- *Jump >times = 3.*

- D 3. Test this new method. Drag Pamela's *dance* method tile from her Details panel and drop it in place of the *Peter.dance* method tile in the Events editor after *When the world starts, do*.

- D 4. Click on the Play button to see the penguin on your left (Pamela) spin around to her right 3 times, and then jump up and down 3 times.

E. Next you will make Paul Penguin dance.

- D 1. In Paul's Details panel (methods tab), click on *create new method*. Name the method *dance*, and click *OK*.

- D 2. Add the following to this new *Paul.dance* method:

- *Jump >times = 3.*
- *Turn >left >3 revolutions >duration = 3 seconds.*

- D 3. Test this new method by dragging Paul's *dance* method into the Events editor. Like Pamela, this penguin on your right (Paul) should jump 3 times, and then spin around to his left 3 times.

F. Now you will have the penguins say hello when the world starts.

- D 1. In the Objects tree, click the *World* tile, and *create new method* called *opening*.

- D 2. Add the following to this new *world.opening* method:

- *Peter say > Hello.*
- *Pamela say > Hello.*
- *Paul say > Hello.*

- D 3. Test the new method by dragging World's *opening* method into the Events editor. First Peter should say hello, then Pamela and then Paul.

G. Next, you will make an ending in which each penguin glides on its belly out of the picture.

- D 1. In the World's Details panel, *create new method* named *closing*. In this new method you will make the penguins turn at the same time.
- D 2. To do this, drag and drop a *Do together* tile from the bottom of the screen into the new *world.closing* method.



- D 3. In the *Do together* tile drop and program the following in order:
- *Peter >turn >right >1/4 revolution.*
 - *Pamela > turn > right > 1/4 revolution.*
 - *Paul > turn > right > 1/4 revolution.*
- D 4. Then add the following tiles under the set of *Do together* instructions (not inside the same box):
- *Pamela glide*
 - *Peter glide*
 - *Paul glide*
- D 5. Test the new method by dragging the *world.closing* method into the Events editor and clicking on Play. The three penguins should turn to their right together, then one at a time in order they should glide off of the screen on their bellies.

H. Finally, you will complete your world by putting all of the methods together.

- D 1. In World's Details panel click the *edit* button next to *scene 1 method*. Add the following to this method from the methods tab in the Details panel for each penguin and the world.
- *World.opening* (Drag opening tile from World's Details method tab)
 - In a *Do together* tile:
 - Peter.dance*
 - Pamela.dance*
 - Paul.dance*
 - *World.closing*
- D 2. Drag the *scene 1 method* tile and drop it in place of *world.closing* into the Events editor.

- D 3. Test all of the methods by clicking on the Play button. All three penguins should first say hello, then they should perform their dances together, and in the closing method, they should leave the screen on their bellies one at a time.
- D 4. Fix any mistakes you've made and then save the world.

If your world does not work the way it should, ask your teacher for a copy of the code and compare it with what you programmed.

Bonus Challenge 6.1: Multiple Methods

In this challenge, you will:

- Practice skills you have already learned but with fewer directions
- Practice adding beginning, middle, and end methods
- Practice changing and testing the method that plays when the world begins

This is similar to the Penguin Dance Challenge but without any directions. You will start a new world, add two or more characters and create beginning, middle and end methods.

- D A. Start Alice and start a new world with any one of the six templates – dirt, grass, snow, sand, space or water.
- D B. *Save* your world with a name that begins with **6.1** and end with your initials.
- D C. Add two or more characters that you want from either the *characters* folder in the *Local Gallery* or from the *Web Gallery*.
- D D. Create and name *beginning*, *middle* and *end* methods where the characters do something different in each method.
- D E. Test each method as you create it by making it play when the world starts.
- HINT:** Place each method in Events editor after *When the world starts, do*.
- D F. To see it all play in order, you need to go back into *World.scene 1 method* and add each of the methods you created in the order you want them to play.
- D G. Then select *World.scene 1 method* from the drop down menu in the Events editor so that method plays when you play your world.
- D H. *Play* your world to see if it works the way you wanted it to!
- D I. *Save* again before quitting.

Bonus Challenge 6.2: Moving Things Together

In this challenge, you will:

- Learn about and practice using vehicle property to move things together
- Learn how to make an event (user interaction) involving the arrow keys
- Create a new world where a fairy rides the fish when you press the arrow keys

A. First look at what your completed challenge will look like:

- D 1. Open Alice.
- D 2. Open **6.2MovingThingsTogetherComplete.a2w**.
- D 3. *Play* to see what the completed challenge looks like. Use the arrow keys to make the fair and fish move around in the aquarium.

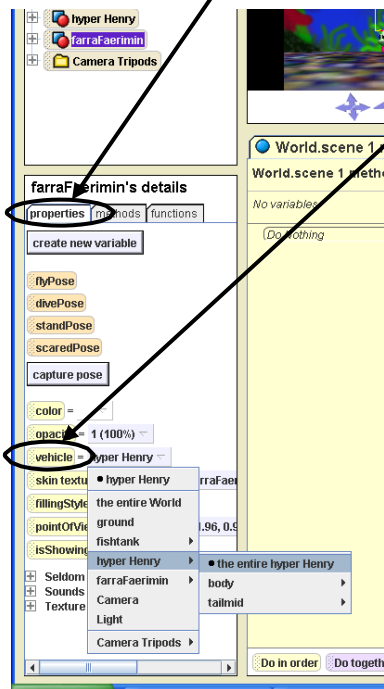
B. Set up your own world.

- D 1. *File > New World*
- D 2. Select the *water* template.
- D 3. *Save* world with any name but start with **6.2** and end with your initials.
- D 4. From the *Local Gallery Scenes & Characters* add:
 - *Aquarium > Fishtank*
 - *Fantasy > FarraFaerimin* (fairy)
 - *Undersea > Hyper Henry* (fish)
- D 5. Move the characters so they look like the picture below and click DONE.

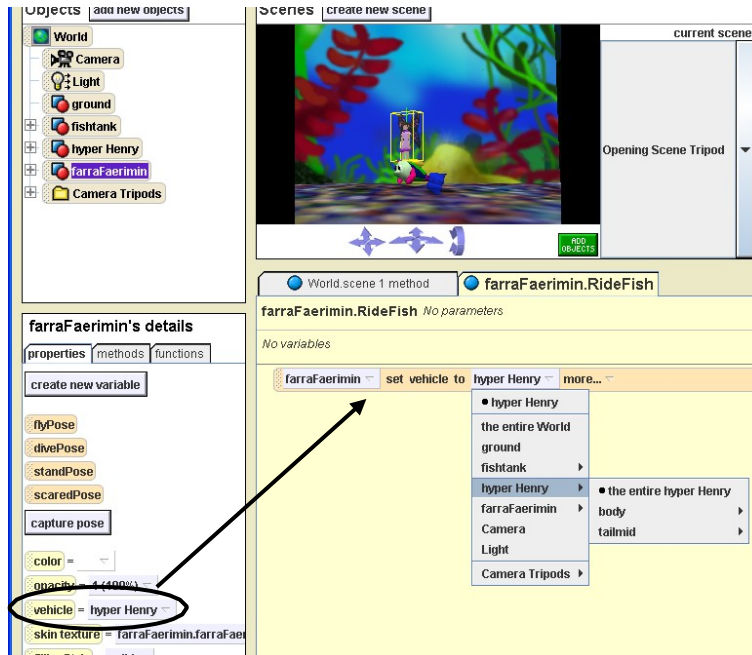
HINT: Use the blue camera controls to move around the characters and see that they are positioned so that the fairy is riding on top of the fish.



- C. Now you will “Glue” the fairy to the fish so they will stay together when they move.
- D 1. Click on *FarraFaerimin* in the Objects tree and her *methods* tab in the Details panel.
 - D 2. Create a new method and call it *RideFish*.
 - D 3. Next, click on her *properties* tab and after the *vehicle* property tile, choose *hyper Henry > the entire hyper Henry*.



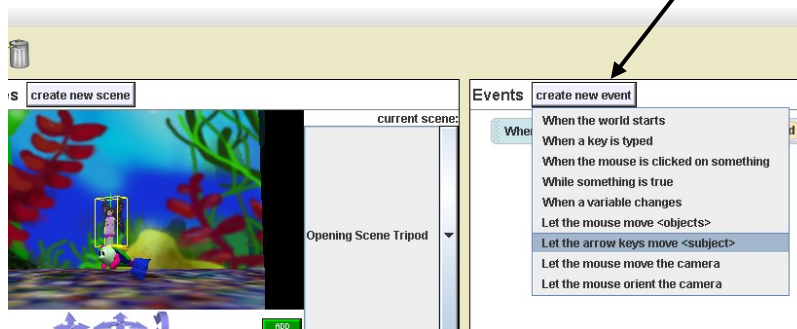
- D 4. Drag the *vehicle* property tile into the *RideFish* method window and set the vehicle as *hyper Henry > the entire hyper Henry*.



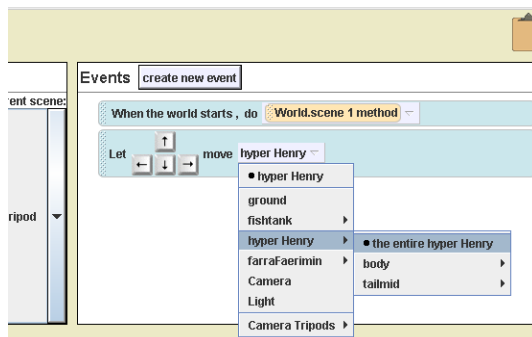
D 5. Save your world.

D. Now make the fairy and fish move together when you press the arrow keys.

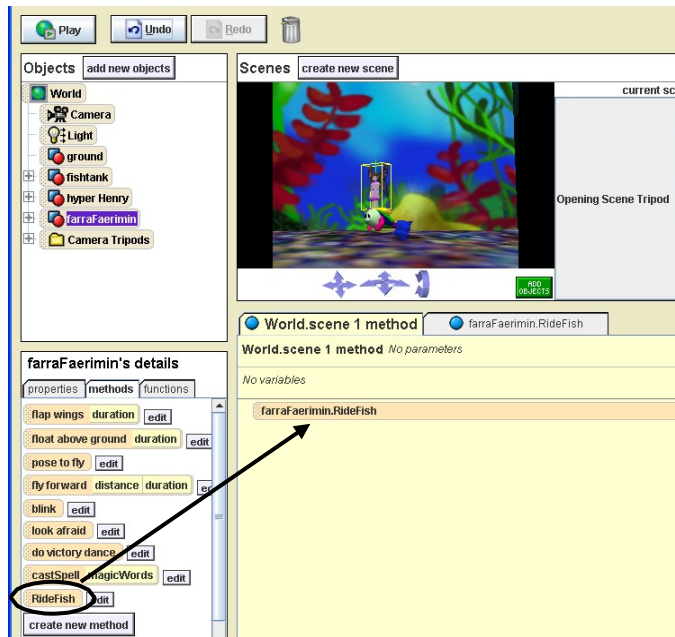
D 1. In the *Events* window in the top right, click the *create new event* button and then select *Let the arrow keys move <subject>*.



D 2. Choose *hyper Henry > the entire hyper Henry* as the subject.



- D 3. Click on the tab for the *World.scene1* method and put the *RideFish* method (one of the fairy's methods) in place of *Do Nothing*.



- D 4. *Play* your world to see if it works correctly. Both the fairy and the fish should move when you press the arrow keys.

HINT: As you press the arrow keys and you see the characters from different angles, you may find that the fairy is not right on top of the fish. If so, use the blue arrows to move the fairy and keep testing until she is correctly placed.

- D 5. *Save* your world again before quitting.

If your world does not work the way it should, ask the teacher for a copy of the code and compare it to what you programmed.

Challenge 7: Make Your Own Story

In this challenge, you will:

- Look at the different parts of a story and draw a simple storyboard.
- Make your own storyboard on paper.
- Program your story in Alice.

A. First look at a simple story:

1. Open Alice.
2. Open **7FishStoryComplete.a2w**.
3. *Play* the world to see a short story about an octopus, two fish and a hermit crab.

B. Now you will make a very simple storyboard of the Fish Story you just looked at.

A **storyboard** is a group of drawings and words on paper that show what you plan to program in the computer.

1. Draw and write about the beginning, middle and end of the Fish Story in the following boxes. Divide the boxes into more sections if you need to.

Fish Story Storyboard

BEGINNING	MIDDLE	END

C. Now that you've had some practice, you will storyboard your *own* story.

- 1. Open a *New World* in Alice.
- 2. Add a scene and at least 2 characters to your world.
- 3. Make a short story about these characters by drawing and writing the story's beginning, middle and end in the areas below.
 - Keep the story short like the Fish Story.
 - Divide the boxes below into more sections if you need to.
 - You don't need to include ALL the details in your story, but it should be enough of a plan to help guide you when you program it in Alice.

BEGINNING	MIDDLE	END

D. Using your storyboard above as a plan, go back to the world you started and program your story in Alice.

E. *Save* your story with any name you want, but start with **7** and end with your initials

Bonus Challenge 7.1: Make Your Story into a Game

In this challenge, you will:

- Brainstorm ideas for turning a story into a game.
- Begin turning a story you made in Alice into a game.

A. First, look at the story you created for Challenge 7.

1. Open Alice.
2. Open and play the story you made for Challenge 7.

A. Next, brainstorm ways to turn that story into a game.

Brainstorming is when you write down as many ideas as you can think of in a short amount of time. You don't think about whether they are "good" or "bad" ideas and you don't decide if you will use them. Just write them down!

1. Using the boxes on the next page, write down as many ways that you can think of to turn your story into a game.

HINT: Think about other games you have played, in iGame or other places.

Now	Later	Ideas to Turn Story into Game

C. Look at your list of ideas to decide which, if any, of your ideas, you can do now and which you need to wait until later when you learn more about Alice.

- 1. Which ones can you do now with what you have learned? Check the **Now** column for those ideas.
- 2. Which ones do you not know how to do, but want to learn so you can add them later? Check the **Later** column for those ideas.

D. Change your story into a game with your Now ideas.

- 1. Go back to your Challenge 7 world. *Save* it with any name (it can be the same as before or a new name) but with **7.1** in the beginning and your initials at the end.
- 2. If you have any ideas checked off for **Now**, program those ideas in your world.
- 3. *Save* your game and go back to it when you learn how to do the other things you want to do!

Challenge 8: Changing Scenes

In this challenge, you will

- Practice skills you have already learned but with fewer directions.
- Learn how to add new scenes and connect those scenes together.
- Learn how to fade in and out when changing scenes.
- Learn how to add invisible markers to control where characters go.

A. First look at what you completed challenge will look like:

- D 1. Open Alice
- D 2. Open **8ChangingScenesComplete.a2w**.
- D 3. *Play* to see what the completed challenge looks like.

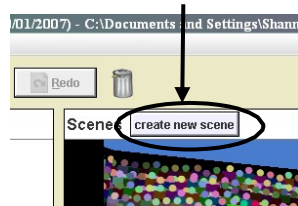
B. Open the Alice world you will use to start the challenge:

- D 1. File>Open World>**8ChangingScenesStart.a2w**.
- D 2. This has the first scene (in the *beginning* method). You will add methods for the middle and end scenes.
- D 3. *Save World As* **8ChangingScenes** followed by your initials.

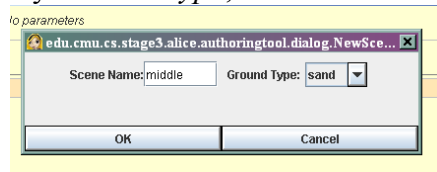


C. First, you will add a middle scene.

- D 1. Click the *create new scene* button underneath the trash can.



- D 2. Type “middle” in the box next to *Scene name*.
- D 3. By *Ground Type*, choose “sand” from the drop down menu.



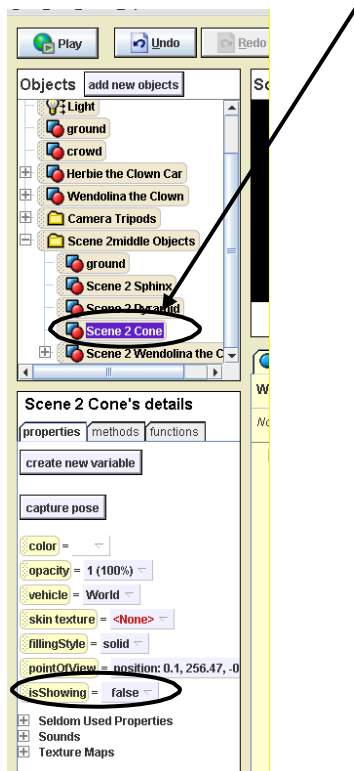
- D 4. Click on OK.

D. Next, add objects to this scene. One of these objects (cone) you will make invisible so Wendolina the Clown will know where to go when she walks into the scene:

- D 1. Add a Pyramid and Sphinx as shown in the picture below. (*Local Gallery > Scenes > Egypt.*)



- D 2. Add a cone to this scene. (*Web Gallery > Shapes*). Put it as shown in the picture above.
- D 3. In the Objects tree, in *Scene2middle Objects* folder, click on the *Scene 2 Cone*.



- D 4. In *Scene 2 Cone's details* panel, click on the *properties* tab.
- D 5. Next to the *isShowing* tile, choose *false* (see above). (The cone should disappear from the scene.)

E. Now you will put another copy of Wendolina the Clown into your scene but move her off the screen.

- D 1. Add *Wendolina the Clown* from the *Local Gallery* and place her so she looks like the picture below:



- D 2. Click DONE!
- D 3. In the Objects tree, under *Scene 2middle Objects*, **right** click on *Scene 2 Wendolina the Clown*.

HINT: Be sure to click on the correct Wendolina. It's the one in the *Scene 2middle Objects* folder NOT the one above it.

- D 4. Choose *methods > Scene 2 Wendolina the clown turn > left > 1/4 revolution*.
- D 5. Right click on her again and choose *methods > Scene 2 Wendolina the clown walk > other > 5 meters*. (Wendolina should walk off the screen.)
- D 6. Click DONE!

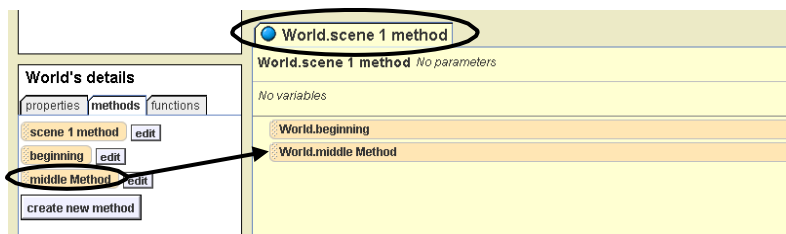
F. Now you will fade out and into scenes and have Wendolina walk into a scene and talk:

- D 1. Make sure the *World.middle Method* tab is showing. If not, go to Objects tree > *World > methods > click on edit button next to middle Method*.
- D 2. Look at the Camera's methods (Objects tree > *Camera > Details > methods*)
- D 3. Drag and drop tiles and choose from menus to program the camera fading out and back in:
 - o *Camera fade to black*.
 - o *Camera move to scene tripod > Camera Tripods > middle Scene Tripod*.
 - o *Camera fade up from black*.

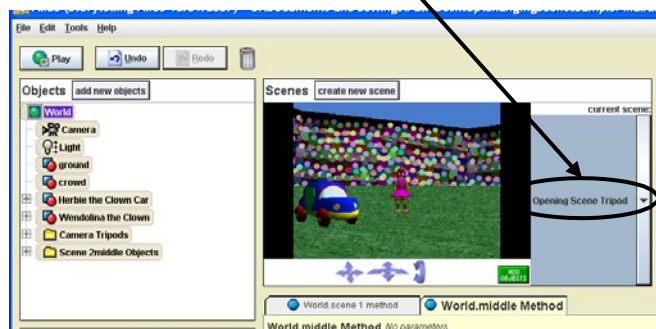
- D 4. Look at Scene 2 Wendolina the Clown's methods (Objects tree > *Scene 2 Wendolina the Clown* > Details > methods).
- D 5. Drag and drop tiles and choose from menus to program Wendolina to walk into the scene and talk:
 - o *Scene 2 Wendolina the Clown walk to* > *Scene 2 Cone*.
 - o *Scene 2 Wendolina the Clown turn to face* > *Camera*.
 - o *look shocked*.(this is a built in method)
 - o *Scene 2 Wendolina the Clown say* > *Oh no, it's too hot here in the desert!* > *duration=5 seconds*.

G. Connect this middle scene to the beginning scene.

- D 1. In the Objects tree, click on *World* and its methods tab in the Details panel to see all the methods.
- D 2. Open the *World.scene 1 method* by clicking on its tab or the *edit* button next to its tile in the Details panel.



- D 3. Drag the *middle Method* tile and put it as the last tile, after *World.beginning* method.
- D 4. Go back to the *Opening Scene Tripod* in the Scene window (as shown in the picture below) and play your world to see if it works correctly.



- D 5. Fix anything that doesn't work and then *Save* your world.

H. Now add your ending scene.

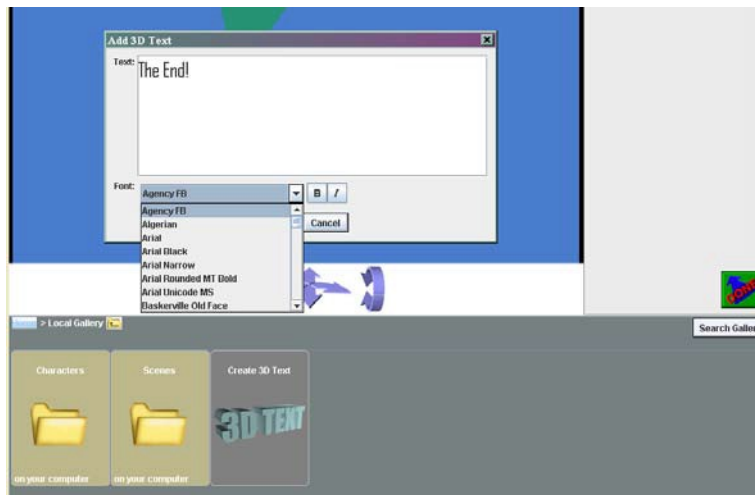
D 1. Create and name a new scene:

- Click on *create new scene* button.
- Type “end” for the *Scene name*.
- Choose “water” as the *Ground Type*.
- Click OK.

D 2. Use the blue camera controls to tilt scene so only the sky shows.

D 3. Add 3D Text:

- *ADD OBJECTS* > *Local Gallery* > *Create 3D Text* > type “The End!” > choose font from drop down menu > OK



D 4. Use the blue camera controls to find the text on the screen (this can be tricky!)

D 5. Use the mouse controls (faces buttons) to make the text look like the picture below:



D 6. Click DONE! to return to method editor.

- D 7. From the Camera's methods, drag and drop tiles and select from drop down menus to program the camera to fade out of the *middle* scene and into the *end* scene:

HUNT: Make sure you are putting these tiles into the *World.end Method*.

- *Camera fade to black.*
 - *Camera move to Scene tripod > Camera tripods > end Scene Tripod.*
 - *Camera fade up from black.*
- I. Finally, connect the *end* scene to the other scenes so it all works together.
- D 1. Click on the *World.scene 1 Method* tab.
- D 2. In the Objects tree, click on *World* and its methods tab in the Details panel.
- D 3. Drag and drop the *end Method* tab into the *World.scene 1 Method* under *World.middle Method*.
- D 4. Choose the *Opening Scene Tripod* (if it's not already selected) and *Play* your world to test it.
- D 5. Fix anything that doesn't work and *Save* your world.

If your world does not work the way it should, ask the teacher for a copy of the code and compare it with what you programmed.

Bonus Challenge 8.1: More Changing Scenes

In this challenge, you will

- Create a world from scratch that includes multiple scenes.
- Practice skills you have already learned but with fewer directions.

- D 1. Open Alice and start a new world with any one of the six templates – dirt, grass, snow, sand, space or water.
- D 2. *Save* your world with a name that begins with **8.1** and ends with your initials.
- D 3. Add any characters and objects that you want from either the *Local Gallery* or the *Web Gallery*.
- D 4. Make at least 1 character do something.
- D 5. Create 2 or more scenes.
- D 6. In one of these new scenes, use a character that you have used in another scene (like you did with Wendolina the Clown in the last challenge).
- D 7. Use 3D text in at least one of the new scenes.
- D 8. Make all the scenes work together.

HINTS:

- Use *Camera move scene to tripod* tiles (methods) in Camera's details to connect the scenes together.
- Place method tiles from *World's* details into *World.scene 1 method* to tell Alice which method to play first, second, third, etc.
- When you *Play* test your movie, remember to return to *Opening Scene Tripod* first.

- D 9. *Save* again before quitting.

Challenge 9: Interactive Penguin Dance

In this challenge, you will:

- Create new events.
- Make objects do something when a key is typed.
- Make an object do something when you click the mouse on that object.
- Use the clipboard to copy and then change events.

A. First, look at what your completed challenge will look like:

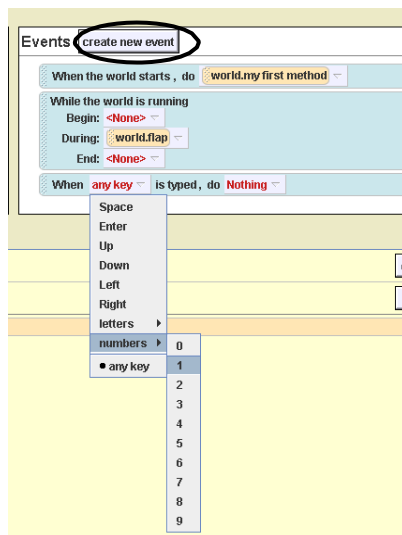
- D 1. Open Alice.
- D 2. Open **9InteractivePenguinDanceComplete.a2w**.
- D 3. *Play* to see what the completed challenge looks like. You will see the three penguins you made dance in a previous challenge and something new – directions.

B. Open the Alice world you will use to start the challenge:

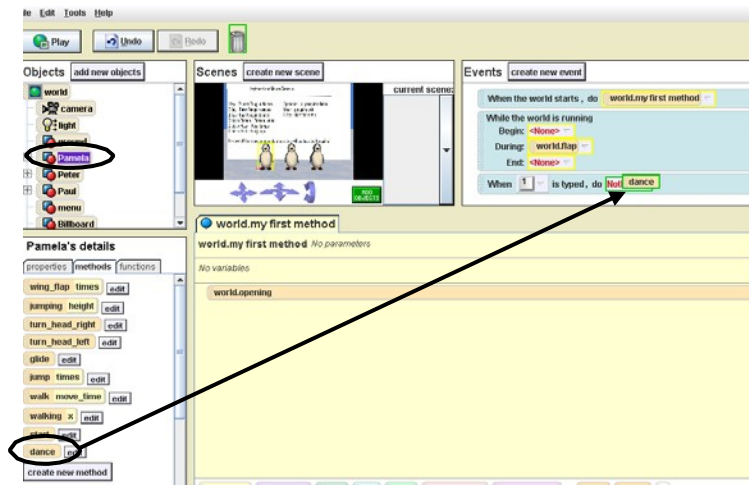
- D 1. *File>Open World>9InteractivePenguinDanceStart.a2w*.
- D 2. *Save World As 9InteractivePenguinDance* followed your initials.

C. Create an event to start Pamela Penguin's dance when the number 1 on the keyboard is pressed.

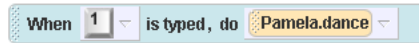
- D 1. Click on the *create new event* button in the Events editor and select *When a key is typed* from the drop down menu.
- D 2. Click *any key*, point to *numbers*, and then *1*.



- D 3. From Pamela's Details panel (methods tab), drag the *dance* method tile and drop it in the new event tile in place of *Nothing*, following the word *do*.



- D 4. The new event should look like this:



- D 5. Save your world again.

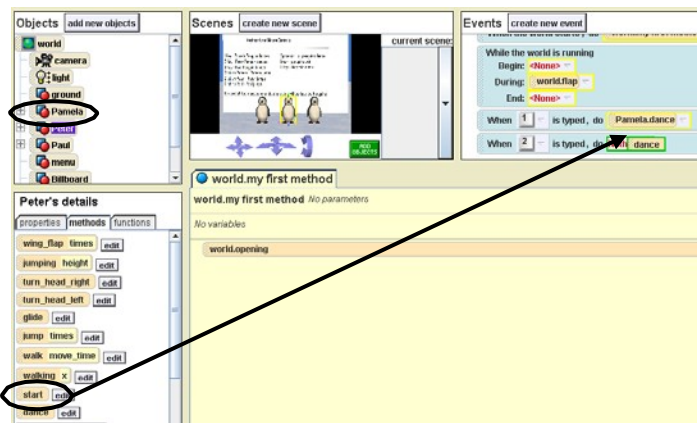
- D 6. To test what you just did, click *Play* and then press *1*. Pamela Penguin should dance.

D. Next, create an event to make Peter Penguin dance when the 2 key is pressed.

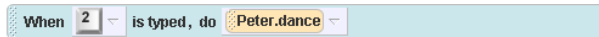
- D 1. Click on the *create new event* button and select *When a key is typed*.

- D 2. Click *any key > numbers > 2*.

- D 3. From Peter's Details panel (methods tab), drag the *dance* method tile and drop it in the new event tile in place of *Nothing*, following the word *do*.



- D 4. The new event should look like this:

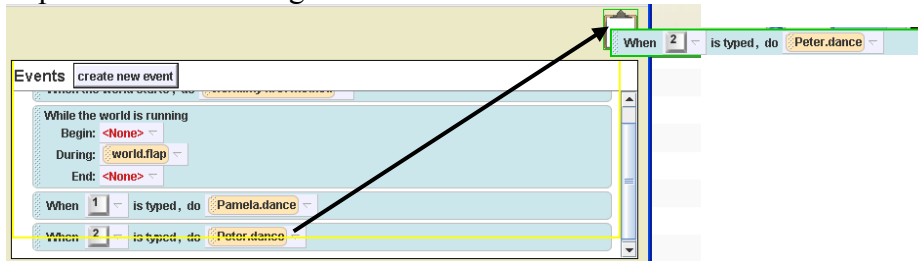


- D 5. Save your world again.

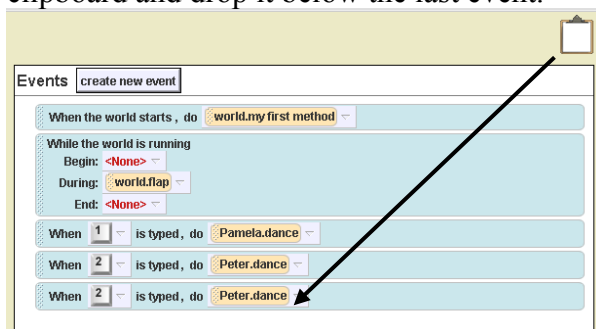
- D 6. Test what you just did. Click *Play* and then press *2*. Peter Penguin should dance.

E. Next, you will make an event to start Paul Penguin's dance, but instead of starting with a new event, you will copy the last event you made.

- D 1. Drag the last event *When 2 is typed, do Peter.dance* and drop it on the clipboard. The clipboard should change color from brown to white.

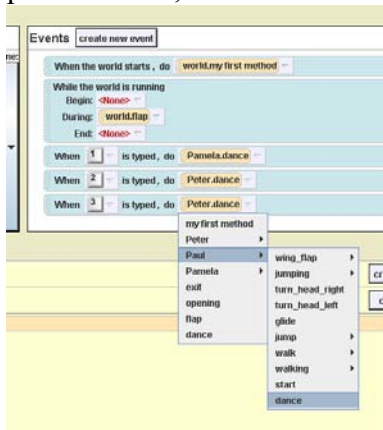


- D 2. Click and hold the mouse down on the clipboard, and then drag the contents of the clipboard and drop it below the last event.



- D 3. Click on the 2 in the bottom copy of the last event, point to *numbers*, and then 3.

- D 4. Click the *Peter.dance* following the word *do* in the bottom copy of the last event, point to *Paul*, and then select *dance*.

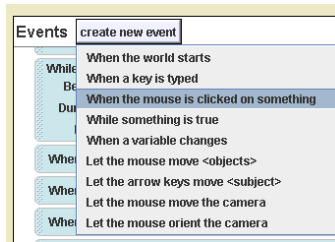


- D 5. *Save* your world again.

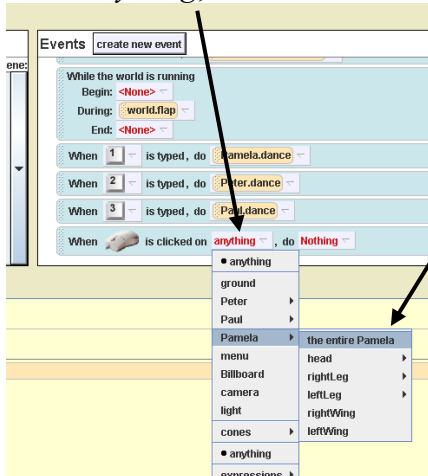
- D 6. *Play* the world and press 3. Paul Penguin should dance.

F. Next, you will make an event to make Pamela jump twice when the mouse is clicked on her.

D 1. Click on *create new event* and choose *When the mouse is clicked on something*.

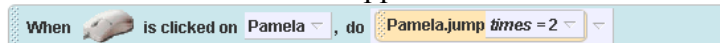


D 2. Click *anything*, and then *Pamela > the entire Pamela*.



D 3. From Pamela's Details panel (*methods* tab), drag the *jump times* method and drop it in the new event in place of *Nothing*, following the word *do*.

D 4. Click on 2 in the menu that appears. The new event should look like this:



D 5. *Save* your world again.

D 6. Test by playing the world and clicking on Pamela. She should jump two times.

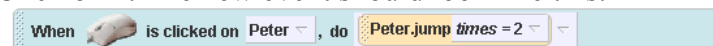
G. Next, you will make an event to make Peter jump twice when the mouse is clicked on him.

D 1. Click on *create new event* and choose *When the mouse is clicked on something*.

D 2. Click *anything*, and then on *Peter > the entire Peter*.

D 3. From Peter's Details panel (*methods* tab), drag the *jump times* method and drop it in this new event in place of *Nothing*, following the word *do*.

D 4. Click on 2. The new event should look like this:



D 5. *Save* your world again.

D 6. Test by playing the world and clicking on Peter. He should jump two times.

H. Next, you will make an event to make Paul jump twice when the mouse is clicked on him.

- D 1. Follow the same steps you just took to make Pamela and Peter jump two times. This time, the new event should say:

When (the mouse) is clicked on Paul, do Paul.jump times=2.

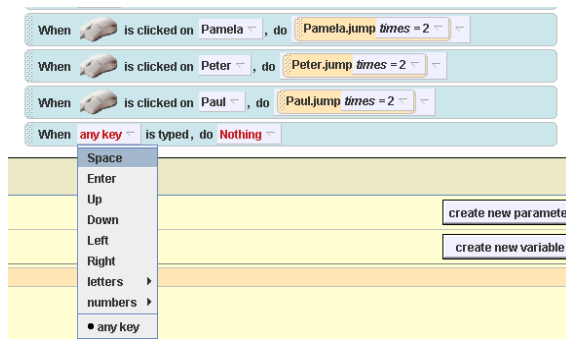
- D 2. Save your world again.

- D 3. Test by playing the world and on Paul. He should also jump two times.

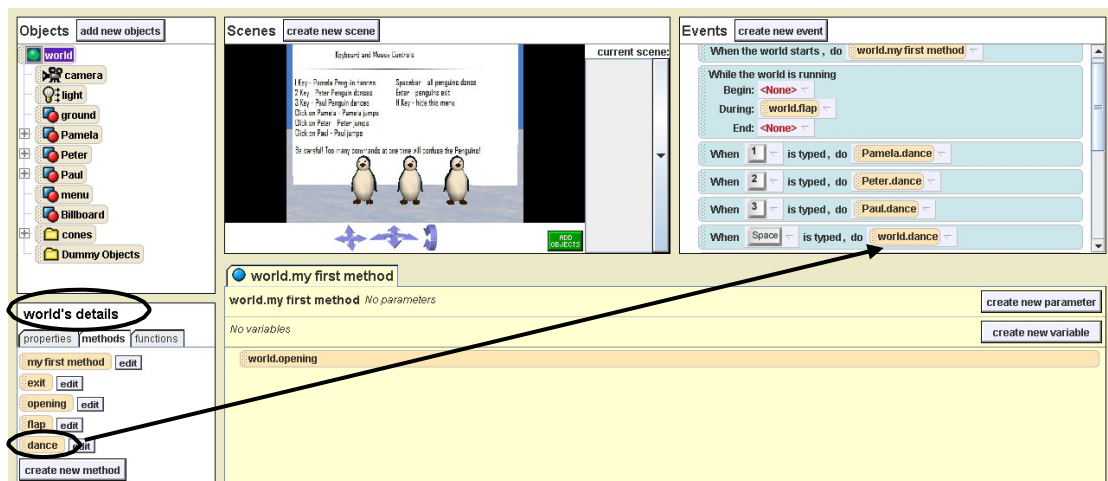
I. Now, add a control to make all three penguins dance when the space bar is pressed:

- D 1. Click on *create new event* and choose *When a key is typed*.

- D 2. Click on *any key*, and choose *Space*.



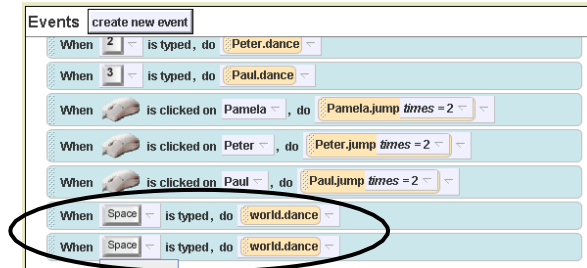
- D 3. From the world's Details, drag the *dance* method and drop it in the new event tile in place of *Nothing*, following the word *do*.



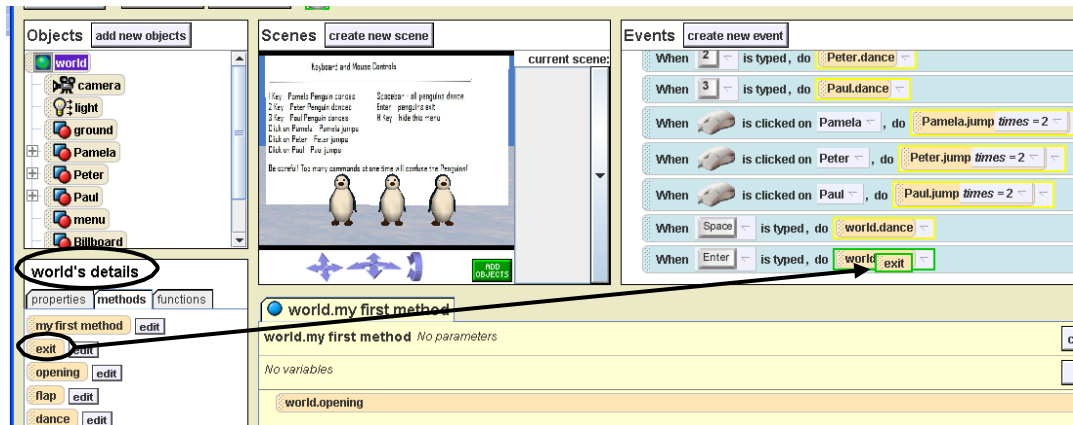
- D 4. Save your world again, and then test the new event by playing the world and pressing the space bar. All three penguins should dance.

J. Program an event to make all the penguins exit by copying the last event and changing it:

- D 1. Drag the last event tile, *When Space is typed, do world.dance*, and drop it on the clipboard.
- D 2. Click and hold on the clipboard, drag and drop underneath the last event tile.
- D 3. You should now have two tiles that look exactly the same (see below):



- D 4. Click *Space* in the bottom copy and then select *Enter*.
- D 5. In world's Details, click on the *exit* method tile and drag and drop it in the new event tile in place of *world.dance*, following the word *do*.

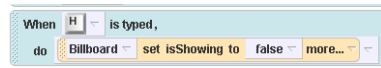


- D 6. Save your world again, and test by pressing *Enter*. All three penguins should slide out of the picture one by one.

K. The last event that you will create is the one to hide the directions (the billboard) when the H (for hide!) key is pressed:

- D 1. Click *create new event* and then select *When a key is typed*.
- D 2. Click *any key > letters > H*.
- D 3. In the Objects tree, click on *Billboard*, select the *properties* tab its Details panel.

- D 4. Drag *isShowing* tile and drop it in the new event in place of *Nothing*, following the word *do*. Select *false* from menu that appears. This event should look like this:



- D 5. *Save* your world again.
- D 6. Do final test where you play the world and try all of the controls on the billboard (directions).
- D 7. Fix any mistakes and then *save* your world.

If your world does not work the way it should, ask your teacher for a copy of the code and compare it to what you programmed.

Bonus Challenge 9.1: Billboard

In this challenge, you will:

- Learn how to create a billboard, which can be used to tell someone how to play your game.
- Learn how to make this billboard appear and disappear.

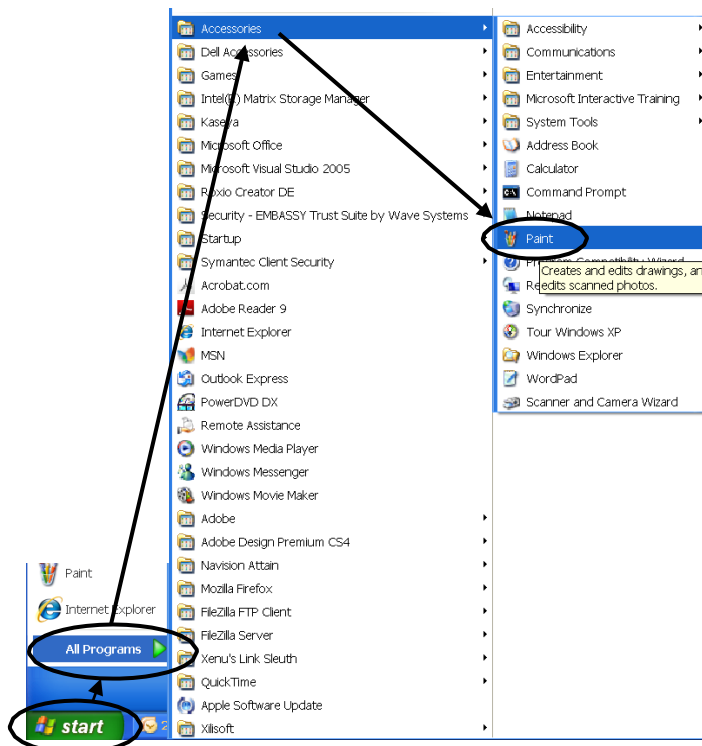
A. First, look at what your completed challenge will look like:

- D 1. Open Alice.
- D 2. Open **9.1BillboardComplete.a2w**.
- D 3. *Play* the world, click on *H* and then on *I* to see the instructions hide and then appear.

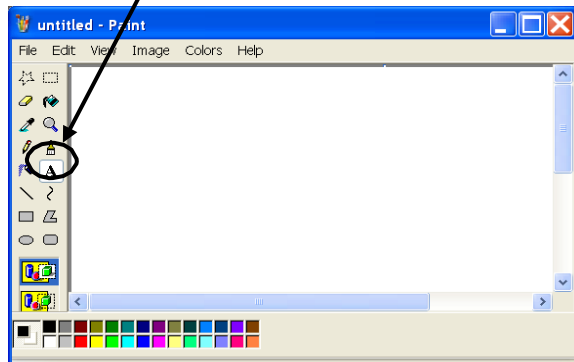
B. Now you will use software called Paint to make your own billboard.

HINT: Paint is included on all PCs running Windows. If you are using a Macintosh computer or some other computer that does not have Paint, you can use any software that lets you draw and save files as jpeg images. These include Photoshop, Illustrator and PowerPoint.

- D 1. To open Paint, go to *start* (on the bottom of your computer) and then to *All Programs* > *Accessories* > *Paint*. You DO NOT need to close Alice.

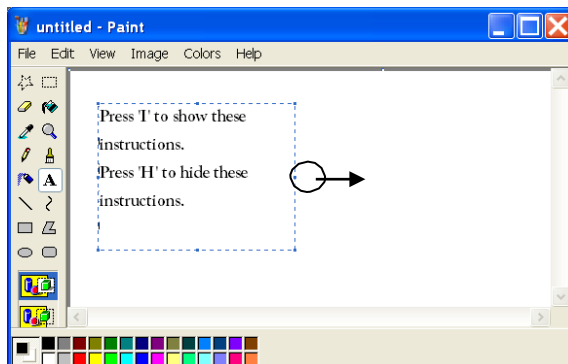


- D 2. Click on the text tool, which looks like letter “A,” and then click in the blank (white) screen.



- D 3. Type the following:
- *Press 'T' to show these instructions*
 - *Press 'H' to hide these instructions*

HINT: If the size of your text box is too small, you can make it bigger by clicking and dragging on the middle handle as shown here:



- D 4. *File > Save as:*
- *File name: Billboard*
 - *Files of Type: JPG*
 - Remember where you saved it so you can get it later!

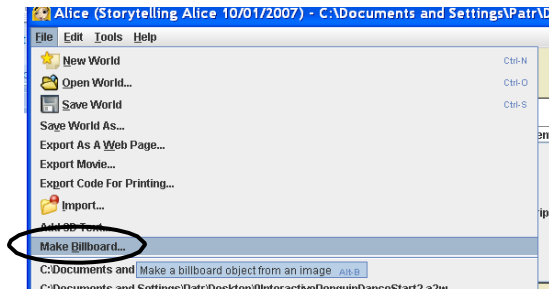
- D 5. Exit *Paint* and return to *Alice*.

C. Next, you will bring your Billboard into Alice.

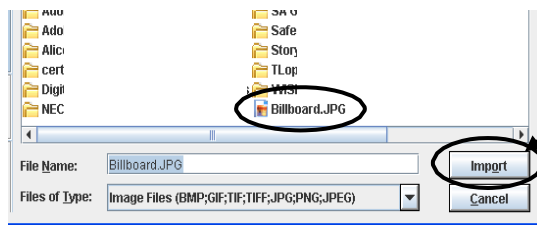
- D 1. Open **9.1BillboardStart.a2w**.

- D 2. Save this world as **9.1Billboard** followed by your initials.

D 3. Go to *File > Make Billboard*.



D 4. Select *Billboard.jpg* (that you just made in *Paint*) and click *Import*. You now have a new object in the Object tree called *Billboard*.



D 5. Click *Add Objects* and use the mouse controls to move and change the size of the Billboard.

D 6. When the Billboard looks the way you want it to, click *DONE!*

D 7. Click *Play* to check that your world starts and your instructions billboard is showing.

D 8. *Save* your world.

D. Now, make events so the Billboard hides and shows when the player hits the H and I keys.

D 1. Click on the *World.showBillboard* tab. If it's not showing in the methods editor, go World's Details panel (methods tab) and click on the *edit* button next to *showBillboard*.

D 2. Click on the *properties* tab in Billboard's Details panel and scroll until you find Billboard's *isShowing* tile.

D 3. Drag and drop *isShowing* in the Method editor.

D 4. Select *true* (this says that Billboard IS showing).

D 5. Click on the *World.hideBillboard* tab. If it's not showing, find it and its *edit* button under the World's Detail panel (methods tab).

D 6. Again drag the *isShowing* tile from the Billboard's Details panel (*properties* tab) into the Method editor. This time choose *false* (this says that Billboard IS NOT showing).

- D 7. To test your world, click *Play* and press *H* (billboard should disappear) and press *I* (billboard should show up again).
- D 8. Fix any mistakes and *save* your world.

If your world does not work the way it should, ask your teacher for a copy of the code and compare it with what you programmed.

Bonus Challenge 9.2: Green Ghost

In this challenge, you will:

- Practice changing the color of the sky.
- Learn how to add fog.
- Learn how to light up parts of the world by adding spotlights and stage lights.

A. First look at what your completed world will look like:

- D 1. Open Alice.
- D 2. Open **9.2GreenGhostComplete.a2w**.
- D 3. *Play* to see what the completed challenge looks like.

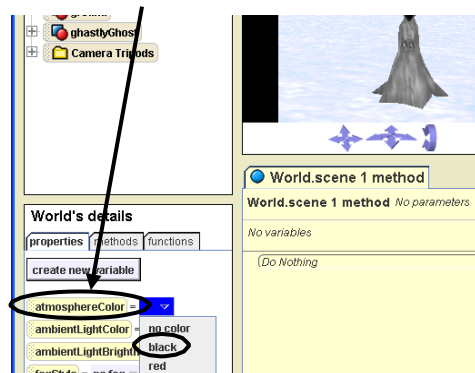
B. Next, start a new world and add a ghost.

- D 1. Start a *New World* and select the *snow* template (because snow reflects light easily).
- D 2. *Save as 9.2GreenGhost* followed by your initials.
- D 3. *ADD OBJECTS > Local Gallery > Characters > scary > ghastlyGhost*
- D 4. Move the ghost or adjust camera so it looks like the picture below and click **DONE!**



C. Now you will change the lighting in the world so looks spookier.

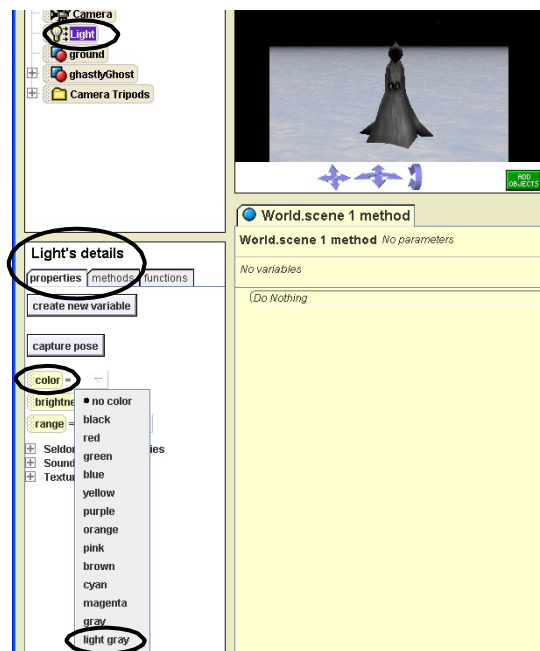
- D 1. In the World's Details panel, click on the *properties* tab.
- D 2. Select *atmosphereColor* and change it to *black*. The sky will turn black.



- D 3. Try changing the sky to other colors and then change it back to black.
- D 4. Right below the atmosphere color is *ambientLightColor*. Set it to *black*. Your world should now look like this:



- D 5. To make the world foggy, go to the *fogStyle* property (also in World's details) and select *distance*.
- D 6. Finally, change the Light to light gray:
Objects tree > *Light* > Light's details > *properties* > *color* > *light gray*

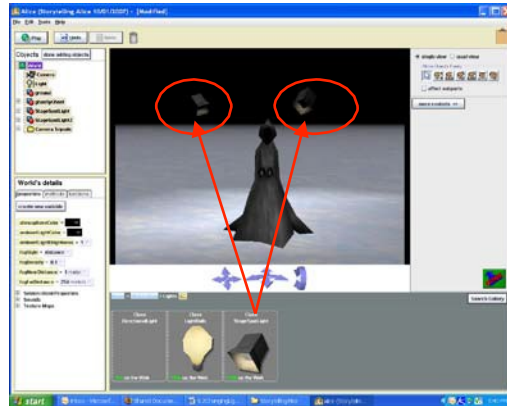


- D 7. *Save* your world again before continuing.

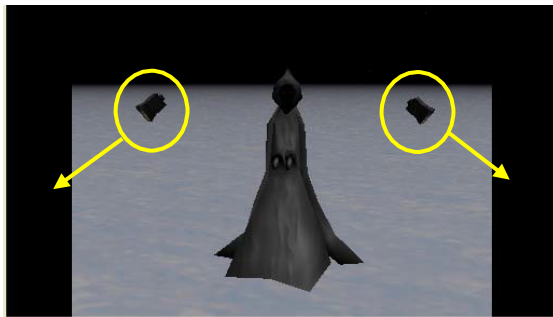
D. Next you will add some lights to control how the light looks in parts of your world and make it change when the world plays.

- D 1. Add one light to each side of the ghost like the picture shows below:
Go to *Web Gallery > Lights > StageSpotLight*.

HINT: If you can't see the light, try moving the ghost. It may be behind it.



- D 2. Move and turn the lights so that they are pointing off stage as shown below:



- D 3. For each spotlight, select them in the Objects tree, go to their *properties* tab and set *isShowing* to *false*. The spotlights should become invisible.

- D 4. Make the lights turn to face the ghost when the world plays:

- Got to *World.scene 1 method*.
- Drag a *Do together* tile (from the bottom of the screen) in place of *Do Nothing*.
- Go to *StageSpotLight's* details (the first light) > *Seldom Used Methods*
- Drag and drop the *StageSpotLight turn to face* tile into *Do together*.
- Select *ghastlyGhost > the entire ghastlyGhost > duration = 1 second*.

- D 5. Repeat Step #4 for the *StageSpotLight2* (the second light) so that both stage spot lights turn together at the same time to face the ghost.

HUNT: Be sure to put the *StageSpotLight2* tile into the *Do together* method under *StageSpotLight*.

- D 6. *Play* your world to see that looks the way it looks below. The light should move from off the screen to onto the ghost.

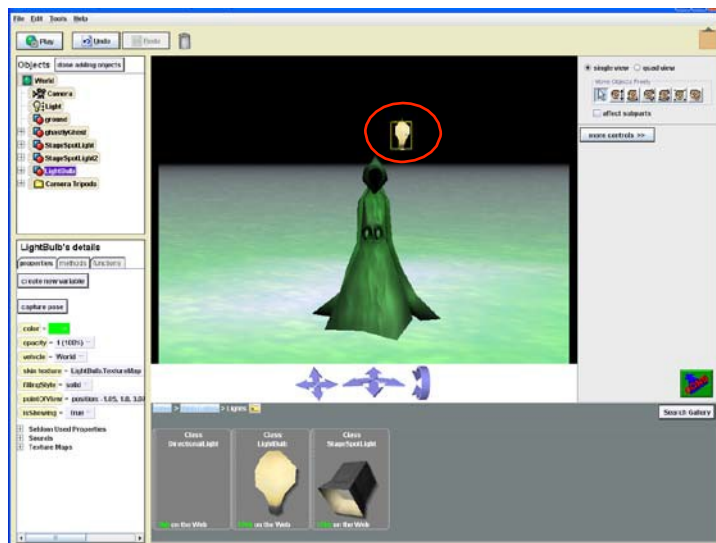


- D 7. *Save* your world.

E. Finally, you will add a light that will shine a green light on the ghost.

- D 1. Follow the same steps you just took to add a *StageSpotLight* (see D1), but this time add a *LightBulb* to your world.

- D 2. Move it to where it shines down on the ghost like in the picture below:



- D 3. Program the light to turn green on the ghost when your world plays.
- o Go to *World.scene 1 method* > Objects tree > *LightBulb* > *LightBulb's details* > *properties* > *color*
 - o Drag and drop the *color* tile into your method under the *Do together* tiles.
 - o Select *green* > *duration = 1 second*.
- D 4. Make the light bulb invisible by turning *isShowing* to *false* (also in *properties* tab)
- D 5. Test and *save* your world.

If your world does not work the way it should, ask your teacher for a copy of the code and compare it with what you programmed.

Challenge 10: Fishy List

In this challenge, you will:

- Learn how to create and use a list of objects.

HINT: Lists are useful because you can program the list (with more than one object in it) to do something (one copy of a method) instead of having to program each object separately (method for each object).

A. First, look at what your completed challenge will look like:

- D 1. Open Alice.
- D 2. Open **10FishyListComplete.a2w**.
- D 3. *Play* to see what the completed challenge looks like.

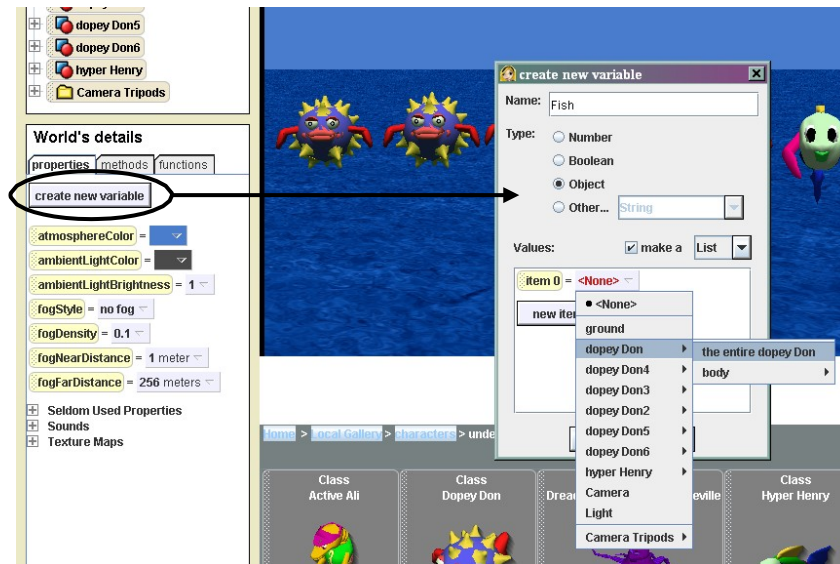
B. Now set up your own Alice world:

- D 1. *File > New World*
- D 2. Select the *water* template.
- D 3. *Save World As 10FishyList* followed your initials.

C. Add and position characters in the scene:

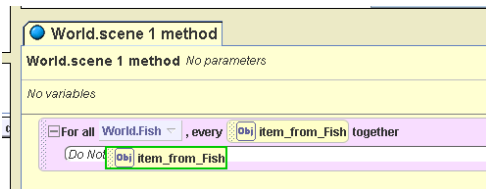
- D 1. Add 6 copies of *Dopey Don*. You can add him multiple times or copy the first one.
- D 2. Add one *Hyper Henry*.
- D 3. Move and resize all the fish so you can see all of them in a row in the water.

- D. Next, you will create a list of fish objects that has all the copies of Dopey Don and Hyper Henry.
- D 1. Go to the World's properties: Objects tree > *World* > *World's details* > *properties*.
 - D 2. Click on *create new variable* button.
 - D 3. Name it "Fish." (See picture below to help with the next steps.)
 - D 4. Select the *Object* radio button.
 - D 5. Check the box next to *make a list*.
 - D 6. Click on the *new item* button.
 - D 7. Next to *item 0*, click on the red <None> > select *dopey Don* > *the entire dopey Don*.

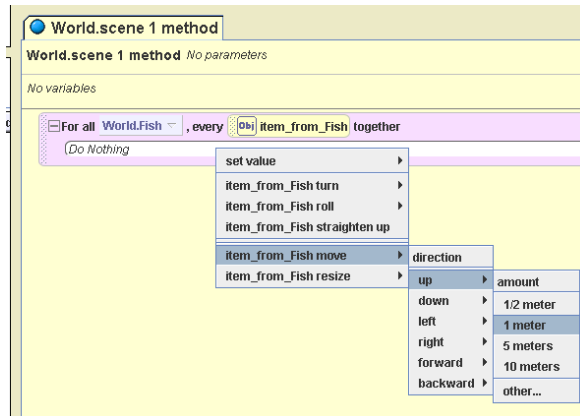


- D 8. Now click on *new item* button.
 - D 9. For *item 1*, click on <None> > select *dopey Don2* > *the entire dopey Don2*.
 - D 10. Continue doing the same until you have added all the copies of *dopey Don* and *hyper Henry* to the *Fish* list. You should have *item 0* through *item 6* in your list.
 - D 11. Click *OK* to return to your scene.
- E. Now, you will use the list you just created to make the fish all move up together and them move down one at a time.
1. Drag and drop a *For all together* tile (from the bottom of the screen) into *World.scene 1 method*.
 2. Select *expressions* > *World.Fish*.

3. Click and hold on *item_from_Fish* and drag it on top of the *Do Nothing*.



4. In the drop down menu, select *item_from_Fish move > up > 1 meter*.



5. Drag and drop a *For all in order* tile into the method editor under the last tile.
6. Select *expressions > World.fish*.
7. Click and hold on *item_from_Fish* and drag it on top of the *Do Nothing*.
8. In the drop down menu, select *item_from_Fish move > down > 1 meter*.

NOTE: Like other built in methods for objects in Alice, only what is on the drop down menu is available for lists. The options are methods that all the objects in the list have in common.

9. *Play* to test your world. All of the fish should move up together and then they all should move down separately, one at a time.
10. Fix any mistakes and *save* your world.

If your world does not work the way it should, ask your teacher for a copy of the code and compare it to what you programmed.

Bonus Challenge 10.1: Loopy Fairy

In this challenge, you will:

- Learn how to use loops in your Alice worlds.

HINT: Loops are used to make groups of actions that you want to repeat a number of times.

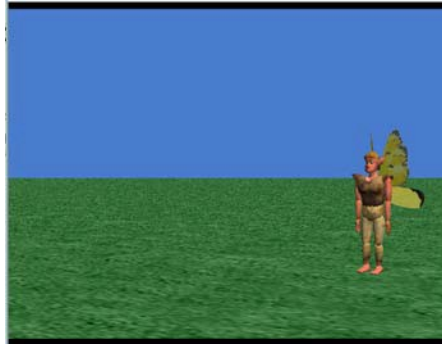
A. First, look at what your completed challenge will look like:

- D 1. Open Alice > **10FishyListComplete.a2w** > *Play* > Stop or X

B. Now set up your own Alice world with a grass template and a fairy:

- D 1. *File* > *New World* > *grass* template.

- D 2. Add the character *CordFlamewand* from *Local Gallery* > *Characters* > *Fantasy* and place him as shown in the picture below.



- D 3. *File* > *Save World As* **10.1LoopyFairy** followed **your initials**.

C. Now you will program the fairy to flap his wings while he is walking across the grass.

- D 1. From the bottom of the method editor, drag and drop a *Loop* tile into the *World.scene 1 method*.

- D 2. Select *5 times* in the menu that appears.

- D 3. Drag and drop a *Do together* tile into the *Loop* method in place of *Do Nothing*.

- D 4. In the Objects tree, select *CordFlamewand*.

- D 5. From *CordFlamewand*'s details, drag and drop the *flapWings* method into the *Do Nothing* of the *Do together* tile.

- D 6. Choose *duration = 0.25* from the menu.

- D 7. Drag and drop *CordFlamewand*'s *walk* tile under the last tile in *Do Together* (under *CordFlamewand.flapWings*).

- D 8. Select *I* in the pop-up menu that appears.
- D 9. Click *Play* to test your world. CordFlamewand should flap his wings at the same time as he is walking across the grass.

If your world does not work the way it should, ask your teacher for a copy of the code and compare it to what you programmed.

Challenge 11: Amusement Park

In this challenge, you will:

- Create keyboard controls that allow the user to move through the amusement park.
- Program six camera control events that work with the above keyboard controls.
- Create events to start each ride when the user clicks on the ride.

A. First look at what your completed challenge will look like:

- D 1. *Open and Play* in Alice: **11AmusementParkComplete.a2w**. Try moving around with the arrows, F and B keys and clicking on rides.

B. Open the Alice world you will use to start the challenge:

- D 1. *File > Open > 11AmusementParkStart.a2w*.

- D 2. This world has no events (keyboard controls or camera angles) programmed yet. It does have all the ride methods programmed.

- D 3. Save world as **11AmusementPark** followed by **your initials**.

C. First you will create 6 events that will control the camera in the following ways:

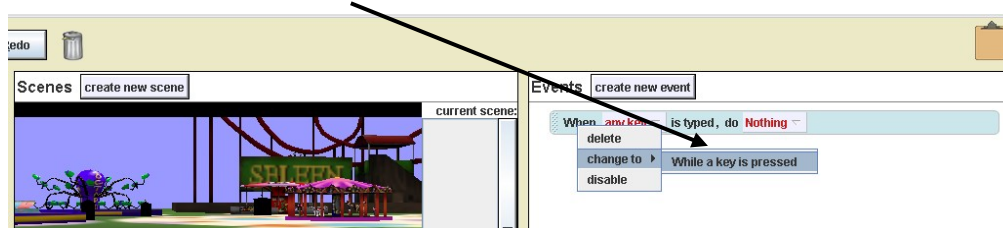
- *Left arrow* = moves the camera left at .1 revolutions per second.
- *Right arrow* = moves the camera right at .1 revolutions per second.
- *Up arrow* = moves the camera forward at 5 meters per second.
- *Down arrow* = moves the camera backward at 5 meters per second.
- *F* = tilts the camera forward at .1 revolutions per second.
- *B* = tilts the camera backward at .1 revolutions per second.

D. Start by creating a control for the *left arrow* key.

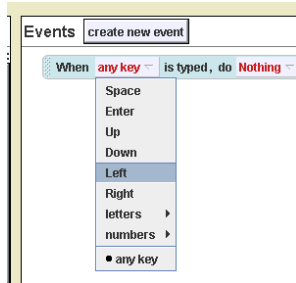
- D 1. Click on *create new event* button in the Events editor.

- D 2. Select *When a key is typed* from the drop down menu.

- D 3. Change the *When* to *While* by right-clicking on the blue area of this new event and choosing *While a key is pressed* in the menu that appears.



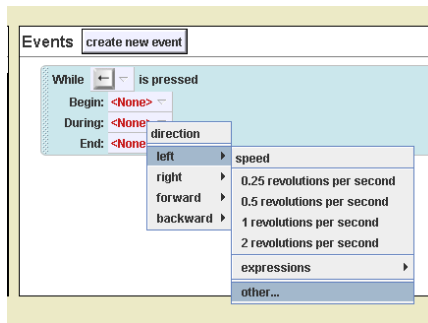
D 4. Now hold down the triangle next to *any key* and choose *left*.



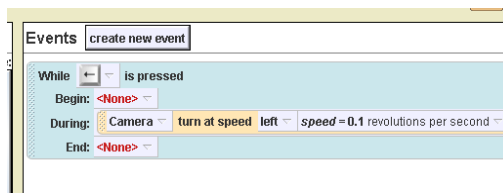
D 5. In the Objects tree, click on *Camera*.

D 6. Under Camera's details > *methods* > *Seldom used methods*, drag and drop *Camera turn at speed* tile in place of <None> after *During* in the Events editor.

D 7. Choose *left* > *other* > and type *0.1 revolutions per second*.



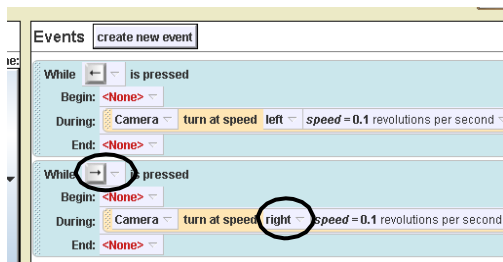
D 8. When you are done the event should look like this:



E. Now create a control for the *right arrow* key.

D 1. Copy the *While* event you just created for left arrow by dragging it onto the clipboard and then out of the clipboard to below the first event.

D 2. Change the controls from drop down menus so when the *right arrow* is pressed, the camera turns to the *right*. When you are done, the second event should look like this:



F. Repeat the above steps to create the events that control the other camera moves.

HINTS:

- Look at **Step C** above for the direction and speed of the events.
- The camera method for the *up* and *down* arrow keys is *move at speed*.
- The camera method for the F and B keys is *turn at speed*.

G. Save your world. Play to test it and fix anything that doesn't work.

H. Now that you have finished the camera control events, you will create the events that let the user start the rides by clicking on them with their mouse. These are the *World* method tiles that go with the following named rides:

- *Octopus* = *octoAnimationLoop*
- *Skyride* = *skyrideAnimation*
- *Teacups* = *teacupBaseAnimationLoop*
- *FerrisWheel* = *ferrisAnimation*
- *Carousel* = *carouselAnimationLoop*
- *Swings* = *swingsAnimation*

1. Start programming these events by clicking on *create new event* button in the Events editor and selecting *When the mouse is clicked on something*. This event should stay as *When* (not *While* like with the camera controls).
2. In this event, click the triangle next to *Anything* and select *Octopus > the entire Octopus*.
3. In the Objects tree, click on *World*. Notice all the methods for the amusement park rides in the World's details panel.
4. Drag and drop the *octoAnimationLoop* tile into this new event in place of *Nothing* in the Events editor. The event should now read like this:



5. To create the other events that start rides, you can use the clipboard to copy and paste the event you just made or you can create new events.

To copy:

- Drag the Octopus ride event you just created onto and off of the clipboard and place below the last event.
- Select the next ride in the list in **Step H** above (*Skyride > the entire Skyride*) from the drop down menu following *When [mouse] is clicked on*.
- Repeat this for the next four rides listed in Step H.

To create new events:

- Click on the *create new event* button and change the parameters listed above in Steps H1-H4 to program controls for all the rides listed in Step H.

6. Save your world. Play and test your world. Fix anything that doesn't work.

If your world does not work the way it should, ask the teacher for a copy of the code and compare it with what you programmed.

Challenge 12: Counting Fish

In this challenge you will:

- Create a counter (world variable that is used to count).
- Create code to increment (add 1 to) that counter.
- Create code print (show the user) the value of the counter.

A. First look at what your completed challenge will look like:

- D 1. Open and Play in Alice: **12CountingFishComplete.a2w**.
- D 2. See how it works by clicking on any fish as many times as you want. When you are done clicking on the fish, click on the snail. See at the bottom of the screen how it tells you how many times you clicked on the fish.

B. Open the Alice world you will use to start the challenge:

- D 1. *File > Open > 12CountingFishStart.a2w*.
- D 2. *Play* this world by clicking on some fish and then the snail. Notice how the message that prints out at the bottom says “You clicked on zero fish.” This tells you that the counter is not programmed yet.
- D 3. *Save* world as **12CountingFish** followed by **your initials**.

C. Before you start, first look at what is already programmed into the world:

- D 1. In the events window notice how it says that if the mouse is clicked on any of the fish, a method called *World.countFish* is run.
- D 2. Open and look at the *World.countFish* method: Objects tree > *World* > *World's details* > *methods* > *edit* button next to *World.countFish*.
- D 3. Notice there is nothing programmed. You will add the code so that the *World.countFish* method actually counts fish that are clicked on!

D. Start by creating a counter variable to count the number of times any fish is clicked on.

- D 1. Go to the Worlds properties: Objects tree > *World* > *World's details* > *properties*.
- D 2. Click on *create new variable* button.
- D 3. Name it “fishCounter.”
- D 4. Make sure the *Number* radio button is selected.

D 5. Change the *Value:* to 0 (zero) and click OK.



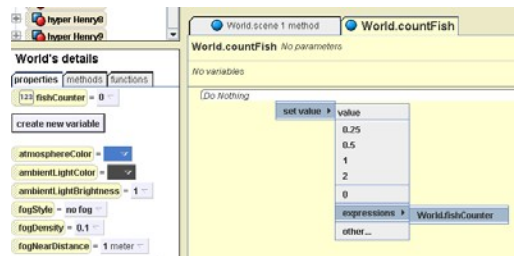
E. Next, add code to add 1 to (increment) the *fishCounter* variable you just created whenever the mouse clicks on a fish.

D 1. Select the *World.countFish* tab in the method editor.

D 2. Go to the World's *properties* again.

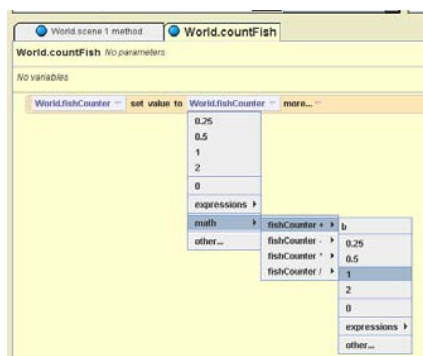
D 3. Drag and drop the *fishCounter* variable (at the top of World's details panel) into the *World.countFish* method.

D 4. Select *set value > expressions > World.fishCounter* from the drop down menu.

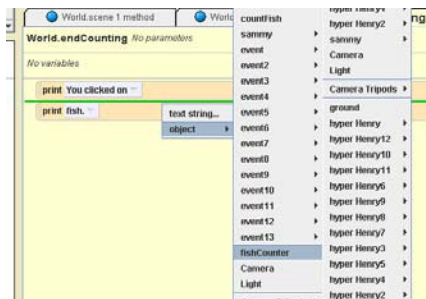


D 5. Click on the drop down menu for the second *World.fishCounter* tile listed.

D 6. Choose *math > fishCounter + > 1* from the drop down menu.



- F. Now program your world so that the counter value shows at the bottom of the world when the snail is clicked on.
- D 1. Open the *world.endCounting* method: Objects tree > *World* > *World's details* > *methods* > *edit* button next to *endCounting*.
 - D 2. Delete the *print zero* tile from the method editor (right click > delete OR drag to trash can).
 - D 3. Drag a *print* tile from below the method editor and drop it between the two print tiles in the method editor.
 - D 4. Choose object > *fishCounter* from the drop down menu.



- G. *Save, Play* and test your world. Check that the program counts the number of fish that are clicked on.

If your world does not work the way it should, ask a teacher for a copy of the code and compare it with what you programmed.

Bonus Challenge 12.1: Count Park Rides

In this challenge you will:

- Create and add a counter to a starting world (similar to Challenge 11).
- Create code to increment (add 1 to) that counter.
- Create code print (show the user) the value of the counter.

A. First look at what your completed challenge will look like:

- D 1. Open and *Play* in Alice: **12.1CountParkRidesComplete.a2w**.
- D 2. Click on a few of the rides and see what gets printed below the amusement park display. Notice how it tells you that you have been on a ride and then adds how many you have ridden as you keep clicking on rides.

B. Open the Alice world you will use to start the challenge:

- D 1. *File > Open > 12.1CountParkRidesStart.a2w*.
- D 2. *Play* the world by clicking on some of the rides. Each time you click on a ride, notice how the message printed at the bottom is “You rode on a ride” but no counting is done.
- D 3. *Save* world as **12.1CountParkRides** followed by **your initials**.

C. Before you start, first look at what is already programmed into the world:

- D 1. Go to the Events window and scroll down to see how, when the mouse is clicked on any of the rides (Octopus, Skyride, etc), three methods are run in order (*World.addRide*, *World.printNumberOfRides*, and the animation for the ride itself).
- D 2. You will be adding code to the *World.addRide* and *World.printNumberOfRides* methods so that they include a counter variable that increases by 1 and prints that information for the user when they click on the rides.

D. Start by creating a counter variable to count the number of times a ride is clicked on.

- D 1. Go to the Worlds properties: Objects tree > *World* > *World's details* > *properties*.
- D 2. Click on *create new variable* button.
- D 3. Name it “number of rides.”
- D 4. Make sure the *Number* button is selected.
- D 5. Change the *Value:* to 0 (zero) and click *OK*.

- E. Next, edit in *World.addRide* method to add 1 to the number (increment) of the *number of rides* variable whenever the mouse clicks on a ride.
- D 1. Open the *World.addRide* method: Objects tree > *World* > *World's details* > *methods* > *edit* button next to *addRide*.
 - D 2. Go to the World's properties (Objects tree > *World* > *World's details* > *properties*).
 - D 3. Drag and drop the *number of rides* variable (at the top of World's details panel) into the *World.addRide* method.
 - D 4. Select *set value* > *expressions* > *World.number of rides* from the drop down menu.
 - D 5. Click on the drop down menu for the second *World.number of rides* tile listed.
 - D 6. Choose *math* > *number of rides* + > *1* from the drop down menu.
- F. Now program your world so that the counter value shows at the bottom of the world after clicking on each ride.
- D 1. Open the *world.printNumberorRides* method: Objects tree > *World* > *World's details* > *methods* > *edit* button next to *printNumberofRides*.
 - D 2. Drag a *print* tile from below the method editor and drop it below the other *print* tile already in the method.
 - D 3. Choose *object* > *number of rides* from the drop down menu. Look for *number of rides* in the middle of the large list of choices (after *event10*, *event11*, etc.).
- G. *Save, Play* and test your world. Check that the program counts the number of rides that are clicked on.

If your world does not work the way it should, ask a teacher for a copy of the code and compare it with what you programmed.
--

Challenge 13: Exploding Fish

In this challenge you will:

- Create a timer (world variable to be used as a timer).
- Create code to decrement (subtract 1 from) that timer.
- Learn how to use the timer in a while loop so the timer stops when it reaches 0.
- Learn how to use the wait instruction.

A. First look at what your completed challenge will look like:

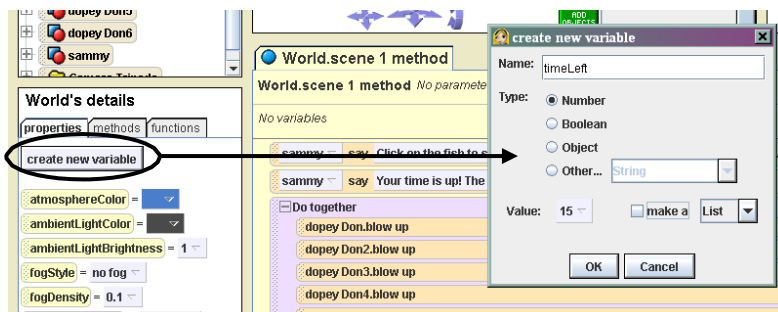
- D 1. Open in Alice: **13ExplodingFishComplete.a2w**.
- D 2. Before playing this world, click on the properties in the World's details panel so you can see the variables change while the world is running.
- D 3. Now click on *Play* to see how it works. Notice how the `timeLeft` variable is counting down.

B. Now open the Alice world you will use to start the challenge:

- D 1. *File > Open > 13ExplodingFishStart.a2w*.
- D 2. Notice how the fish will immediately blow up because there is no time left.
- D 3. *Save world as 13ExplodingFish* followed by **your initials**.

C. The first thing you will do is create a timer by making a variable (named `timeLeft`) with a starting value of 15 (for 15 seconds).

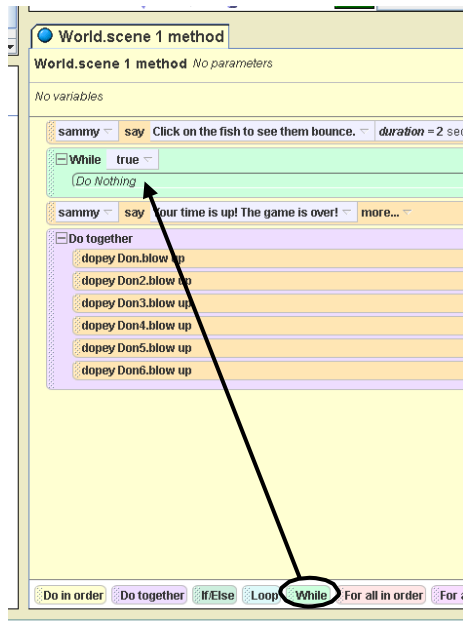
- D 1. Go to the Worlds properties: Objects tree > *World > World's details > properties*.
- D 2. Click on *create new variable* button.
- D 3. Name it "timeLeft."
- D 4. Make sure the *Number* button is selected.
- D 5. Change the *Value:* to 15 and click *OK*.



D. Next you will add code to make the timer count down from 15 seconds and when it reaches 0, all the fish will blow up.

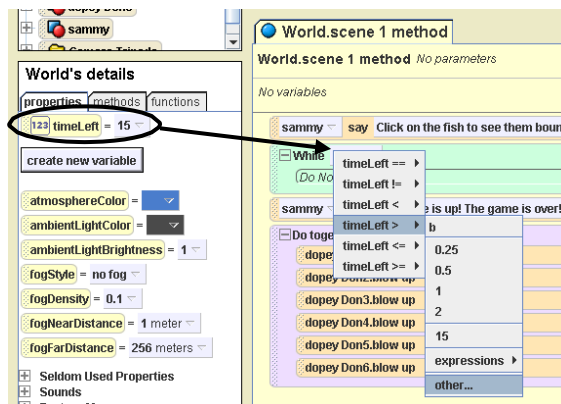
D 1. Drag and drop a *While* tile (from below the method editor) in between the two Sammy say instruction tiles in the *World.scene 1 method*.

D 2. Select *true*.



D 3. Drag and drop the *timeLeft* from the World's *properties* over the *true* of the *While* instruction tile.

D 4. Select *timeLeft > > other > 0*.

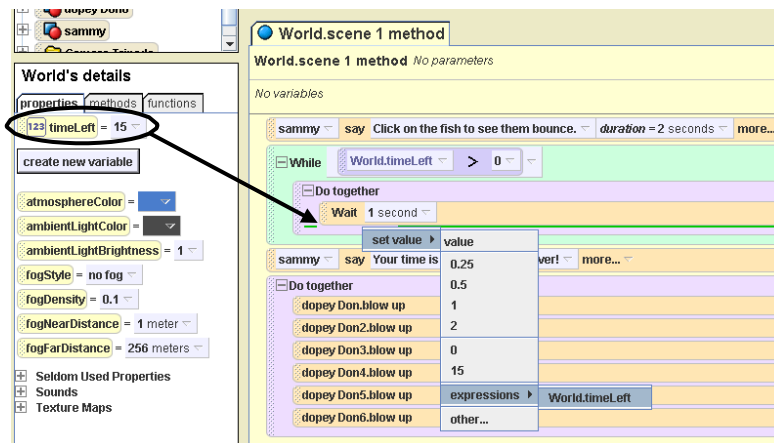


D 5. Drag and drop a *Do together* tile onto the *Do Nothing* of the *While* instruction tile.

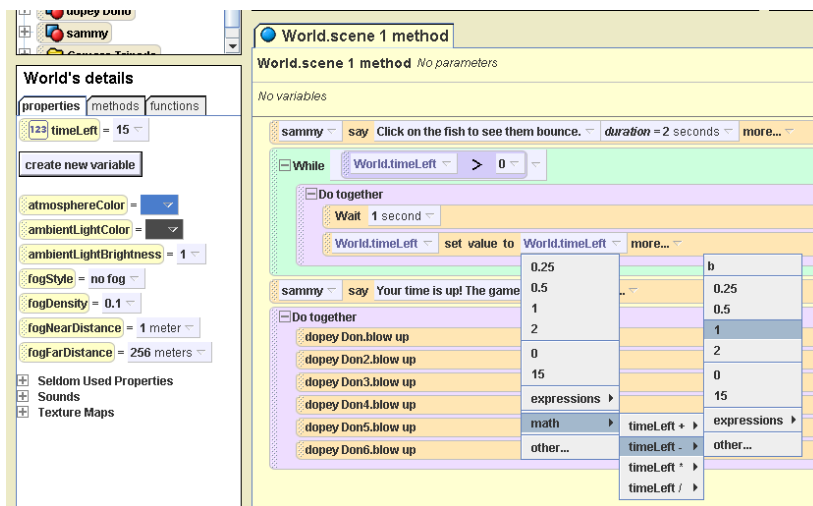
D 6. Drag and drop a *Wait* tile (from below the method editor) onto the *Do Nothing* of the *Do together* instruction.

D 7. Select *duration > 1 second*.

- D 8. Drag another *timeLeft* variable from the World's *properties* and drop it after the *Wait* instruction.
- D 9. Select *set value* > *expressions* > *World.timeLeft* from drop down menus.



- D 10. Click on the SECOND *World.timeLeft* in the *Do together* instruction.
- D 11. Select *math* > *timeLeft - > 1*.



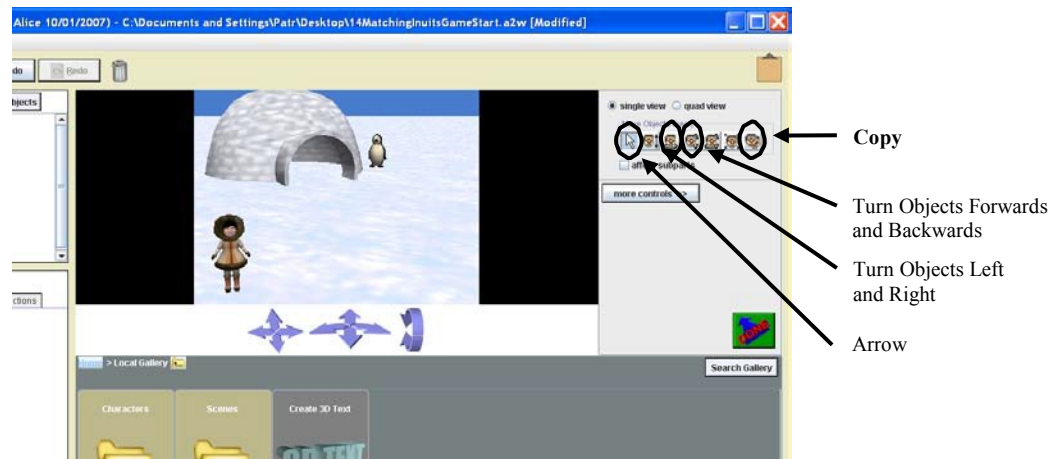
- D 12. *Save* and test your world by clicking on *Play*, watching the timer variable countdown and checking to see that the fish blow up when the timer reaches 0.

If your world does not work the way it should, ask a teacher for a copy of the code and compare it with what you programmed.

Challenge 14: Matching Inuits Game

In this challenge, you will:

- Build a simple game. In this game, the player is asked to find the object that is slightly different from the other objects.
- A. First, look at what your completed challenge will look like:
- D 1. Open Alice.
 - D2. Open **14MatchingInuitsGameComplete.a2w**.
 - D 3. *Play* to see what the completed challenge looks like.
- B. Now create your own Inuit matching game:
- D 1. *File > Open > 14MatchingInuitsGameStart.a2w*.
 - D 2. *Save World As 14MatchingInuitsGame* followed **your initials**.
- C. Add and place 5 more Inuit girls to this scene:
- D 1. Click on *add new objects* button.
 - D 2. Click the *copy mouse control* button (last face) on the far right of the Scene Editor.



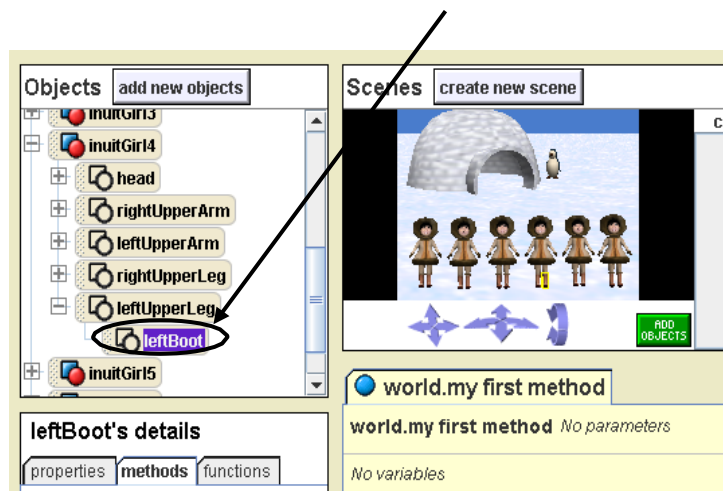
- D 3. Click one time on *inuitGirl1* to make a copy of her.
- D 4. Click on her 4 more times so there are 6 copies of the Inuit girl in the scene.
- D 5. Check in the Object tree for *inuitGirl1* through 6.
- D 6. Click on the *arrow* mouse control (first blank button – not a face) to stop from making too many copies.

HINT: They will look like they are stacked on top of each other. If you make too many, delete them by clicking on them in the Object tree and dragging them to the trash can.

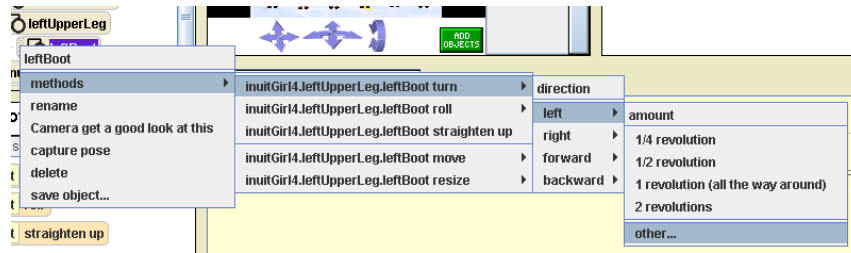
- D 7. If it is not already selected, click on the *arrow* and move each Inuit girl so they are evenly spaced as shown below.



- D. Make all the Inuit girls face the same direction:
- D 1. Click on the *turn objects left and right* mouse control button (second face button from the left) and select and turn each Inuit girl so she is facing the camera as shown. Make sure they are all facing exactly the same way.
 - D 2. Click on the *turn objects forwards and backwards* button (button in the middle) and select and turn each Inuit girl so she is perfectly straight and all copies look exactly the same.
- E. Change ONE of the Inuit girls by making her left boot turn left.
- D 1. In the Objects tree, click on *inuitGirl4*.
 - D 2. Click on the plus (+) sign to see all of the subparts of *inuitGirl4*.
 - D 3. Click on plus (+) sign by *leftUpperLeg*, then on *leftBoot*.



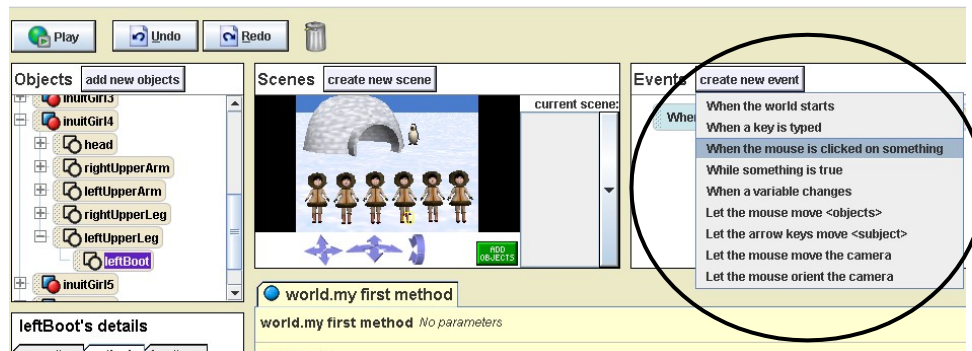
- D 4. Right click on the *leftBoot* tile and point to *methods* > *inuitGirl4.leftUpperLeg.leftBoot turn* > *left* > *other* > type *.1* > click *OK*.



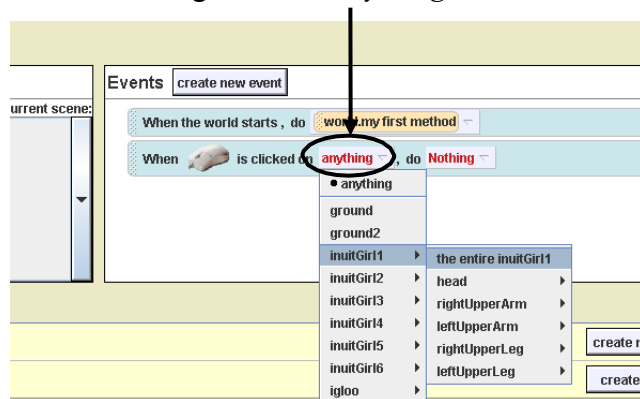
- D 5. Click *DONE*.
D 6. *Save* your world again.

- F. Now you will program events (the player interactions) like you did for Challenge 9: Interactive Penguin Dance. First you will control the penguin so that when the mouse is clicked on the Inuit girls who are the same, the penguin will say, “Sorry, that’s wrong!”

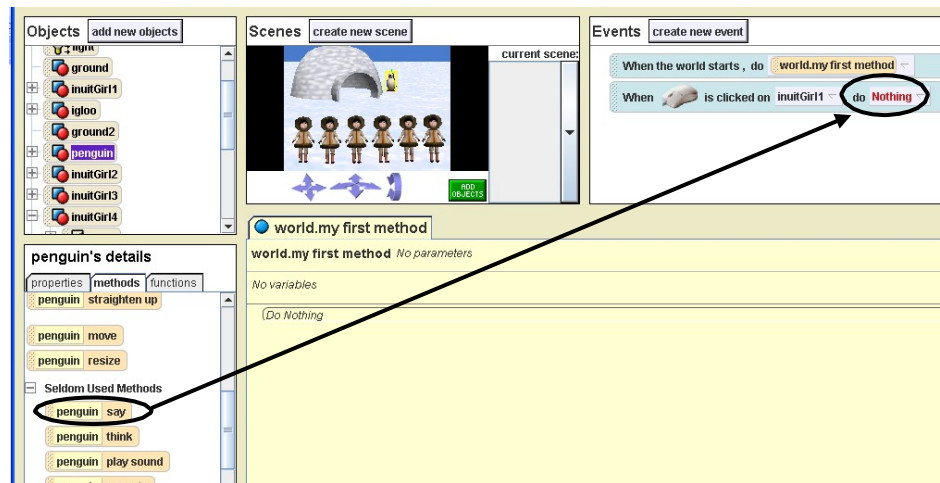
- D 1. Click *create new event* button.
D 2. Select *When the mouse is clicked on something*.



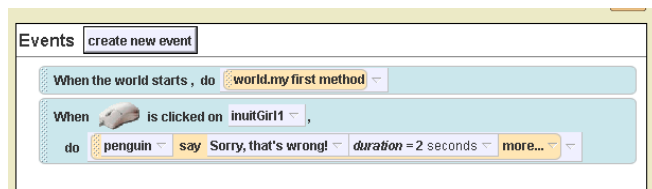
- D 3. Hold down the triangle next to *anything* > choose *inuitGirl1* > *the entire inuitGirl1*.



- D 4. Go to penguin's details > methods > *Seldom Used Methods*
- D 5. Drag and drop the *penguin say* tile into the blue Event editor (NOT the Scene editor) where it says *Nothing* in red.



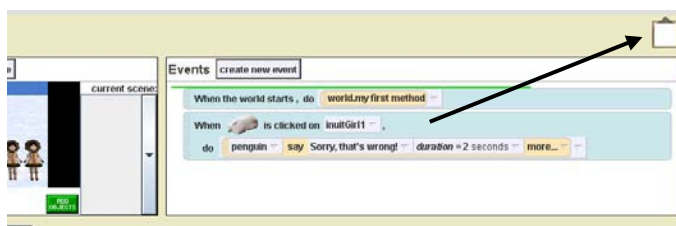
- D 6. Choose *other* > type *Sorry, that's wrong!* > OK.
- D 7. Click on *more* > *duration* > *2 seconds*.
- D 8. Check to see that the final event looks like this:



- D 9. Save the world again and *Play*. Try clicking on the *inuitGirl1* to test if the penguin says "Sorry, that's wrong!"

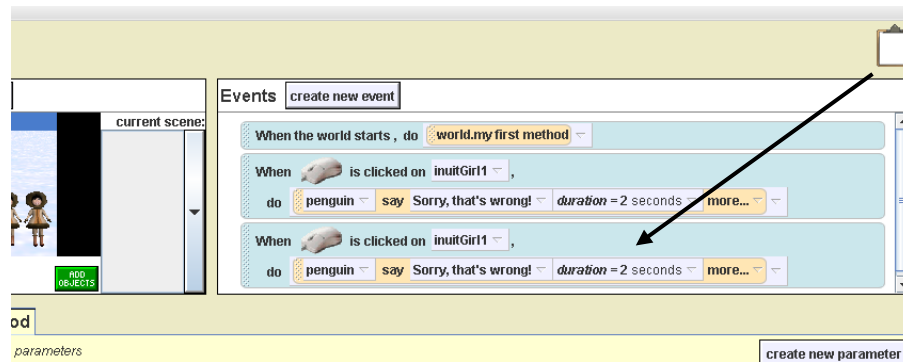
HINT: If you don't know which Inuit Girl is which, click on one in the scene and see the title of object that gets highlighted in the Objects tree.

- G. Copy the event for the four other Inuit girls who are exactly the same.
 - D 1. Click anywhere on the blue background of the event. Hold down the mouse, drag the event to the clipboard and release the mouse when the clipboard turns white.

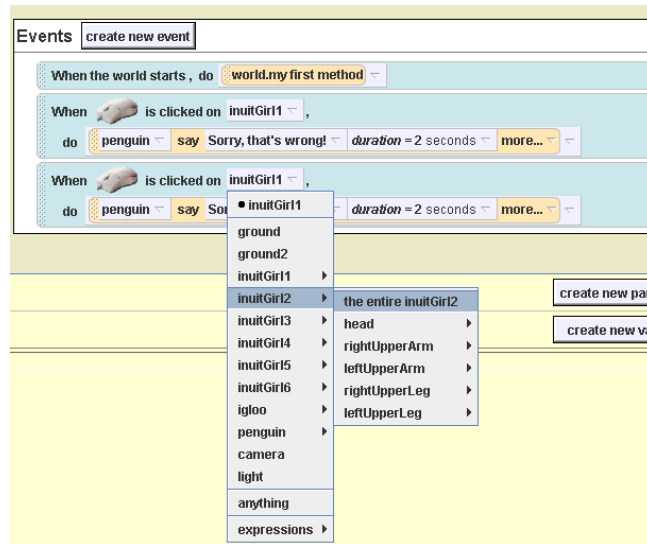


HINT: Be sure to select the Event box you want, NOT the *When the world starts, do world.myfirst method* box.

- D 2. Click on the clipboard and drag a copy of the event and put it underneath the last event.

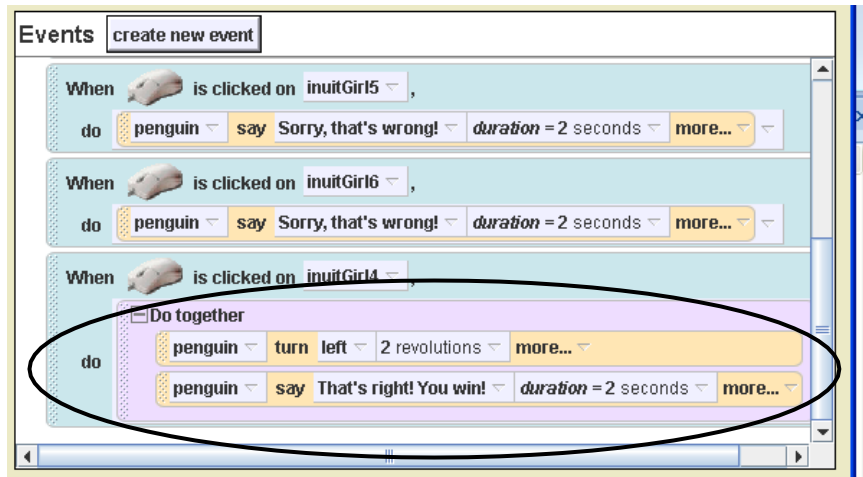


- D 3. In the bottom event copy, hold down the triangle next to *inuitGirl1* > choose *inuitGirl2* > *the entire inuitGirl2*.



- D 4. Repeat steps 1-3 until you have the same event for Inuit girls 1, 2, 3, 5 and 6 (all but *inuitGirl4*).
- D 5. *Save* and *Play* test your world to make sure the penguin says, “Sorry, that wrong!” every time you click all but *inuitGirl4*.
- H. Create the event (telling the player they have won) for the Inuit girl who is different:
- D 1. Click *create new event* button > choose *When the mouse is clicked on something* > click on *anything* > select *inuitGirl4* > *the entire inuitGirl4*.
- D 2. Drag and drop a *Do together* tile in place of *Nothing* in the new event tile in the Events editor.
- D 3. Drag and drop a *penguin turn* tile in the *Do Nothing* area below *Do together*.
- D 4. Select *left* > *2 revolutions*.
- D 5. Open the penguin’s *Seldom Used Methods* and drag and drop *penguin say* tile below the *penguin turn* tile in the *Do together* area.

- D 6. Choose *other* > type *That's right! You win!* > click OK.
- D 7. Click on *more* > select *duration* > *2 seconds*. Your event for *inuitGirl4* should look like the following:



- D 8. *Save* and play test your game!

If your world does not work the way it should, ask your teacher for a copy of the code and compare it to what you programmed.

Bonus Challenge 14.1: My Own Matching Game

In this challenge, you will:

- Build your own matching game with the objects you choose.
- A. First, make a new world:
- D 1. Open Alice.
 - D 2. Open a new world and choose whatever template you want.
 - D 3. Save the world as **14MyOwnMatchingGame** followed by your initials.
- B. Now make your own matching game:
- D 1. Add at least two character objects, one for the matching task and the other to tell the player when he or she is right or wrong. Add any other objects that you want, but it is best to keep it simple.
 - D 2. Copy and evenly space your matching character objects and then change one of the copies slightly.
 - D 3. Make the events that tell the user when he is right and wrong whenever the mouse is clicked on one of the matching objects.
 - D 4. Save and test your game often. If it does not work the way it is supposed to, compare it with the Inuit Matching Game you made.

Challenge 15: Princess Adventure Game

In this challenge, you will:

- Program an adventure game from scratch.
- Practice most of the things you have learned in previous challenges.
- Learn how to save and return to different camera views.

A. First look at what your completed challenge will look like:

- D 1. Open in Alice: **15PrincessAdventureGameComplete.a2w**.
- D 2. Play to see how it works and what your completed challenge will look like.

B. Now set up your own Alice world:

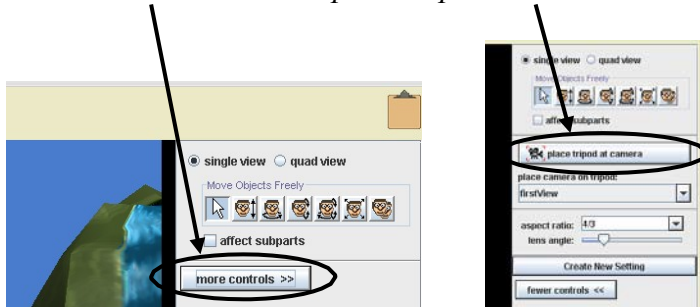
- D 1. *File > New World > grass* template.
- D 2. Add the following objects from the Local Gallery:
 - *Scenes > Waterfall > Waterfall*
 - *Characters > Kids > Jenni*
- D 3. From Web Gallery > *Animals*, add a *frog*.
- D 4. Position them in your world like the picture below. Make sure the frog is not too close to the left or bottom of the scene.



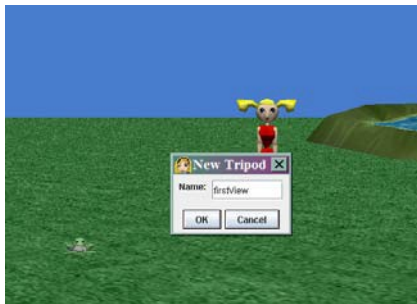
- D 5. Make sure Jenni and the frog are looking straight at you. There are two different ways to do this:
 - Use the *Turn Objects Left and Right* mouse control *OR*
 - In the Objects tree, select each object (*Jenni, frog*) > right click > select *methods* > *turn to face* > *Camera*.
- D 6. *Save world as 15PrincessAdventureGame* followed by **your initials**.

C. Now you will save this camera view so you can return to it later.

- D 1. Under the buttons to the right of the scene window, click *more controls*, then *place tripod at camera*.



- D 2. Name the New Tripod *firstView*, click *OK* and then *DONE*.



- D 3. In the Scene window, click on the down arrow next to *firstView* and select *Opening Scene Tripod* to return to that view.

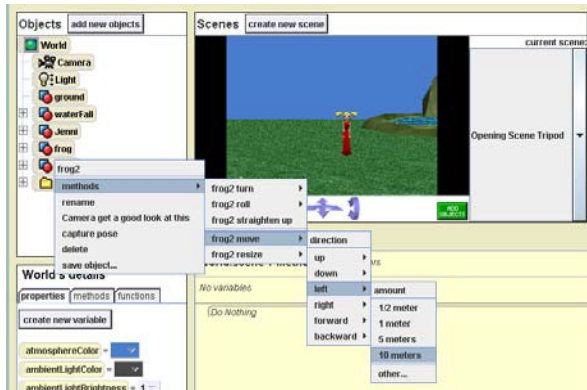


D. You will now make another frog and move it out of the picture for the Opening Scene.

- D 1. Make a copy of the frog: *Add objects > copy* mouse control (last face button on the right) > click on frog.

HINT: Remember to click back on the *arrow* control or you will keep making copies every time you click the mouse.

- D 2. Move the frog: Objects tree > right click on *frog2* > *methods* > *frog2 move* > *left* > *10 meters*. You should see the frog move off the screen to your right.



- D 3. Save your world again before continuing.

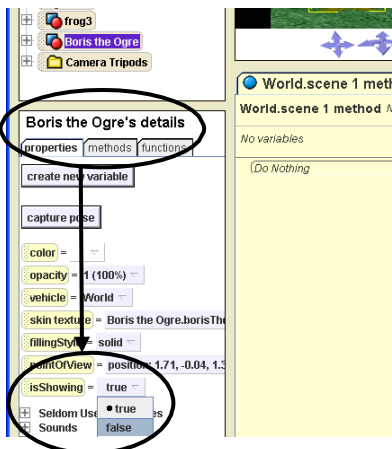
- E. Now you will add *Boris the Ogre* to your world and make him disappear.

- D 1. Add *Boris the Ogre* (*Local Gallery* > *Characters* > *scary*).

- D 2. Move Boris so he's in front of the first frog and is facing straight ahead as shown.



- D 3. Make Boris disappear: Objects tree > *Boris the Ogre* > Details panel > *properties* > *isShowing* > change *true* to *false*. Boris should disappear.



- D 4. Save your world again before continuing.

F. Now you will add a *Knight*, move him in front of *frog2*, and make him disappear.

- D 1. Repeat Steps D1-D3 above, like you did for Boris the Ogre (add, position, make invisible).
- D 2. Move the invisible *Knight* so he is in front of *frog2*: Objects tree > right click on *Knight* > *methods* > *Knight move* > *left* > *10 meters*. Now the *Knight* should be in front of the first frog facing you and invisible.

HINT: It's important that you trust that the *Knight* moved to be in front of *frog2* (that you can't see). If you use the camera angles to check and see it will throw off the rest of these directions.

- D 3. Click *DONE* and *Save* your world before continuing.

G. Now you will create an event to control the princess Jenni's meeting with the first frog. Start by doing the following:

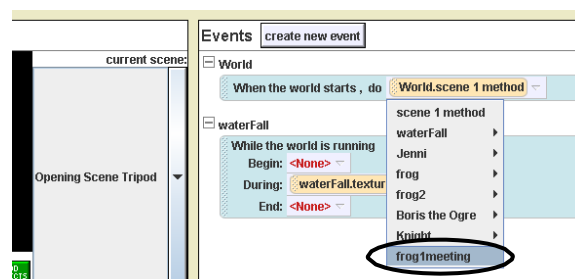
- D 1. Create and name a new method: Objects tree > *World* > Detail's panel > *methods* > *create new method* > name it *frog1Meeting* > *OK*.
- D 2. Drag a *Do in order* tile (from bottom of the Method editor) and drop it into the *Do Nothing* box of this new method.
- D 3. Into the *Do in order* instruction, add instructions to make *Jenni* turn to face the entire head of the *frog*.

HINT: Objects tree > *Jenni* > Details panel > *methods* > *turn to face* > *frog* > *head* > *entire head*).

- D 4. Also in the *Do in order* instruction, make the *frog* turn to face the entire Jenni.

HINT: The frog's *turn to face* tile is under its *Seldom Used Methods*.

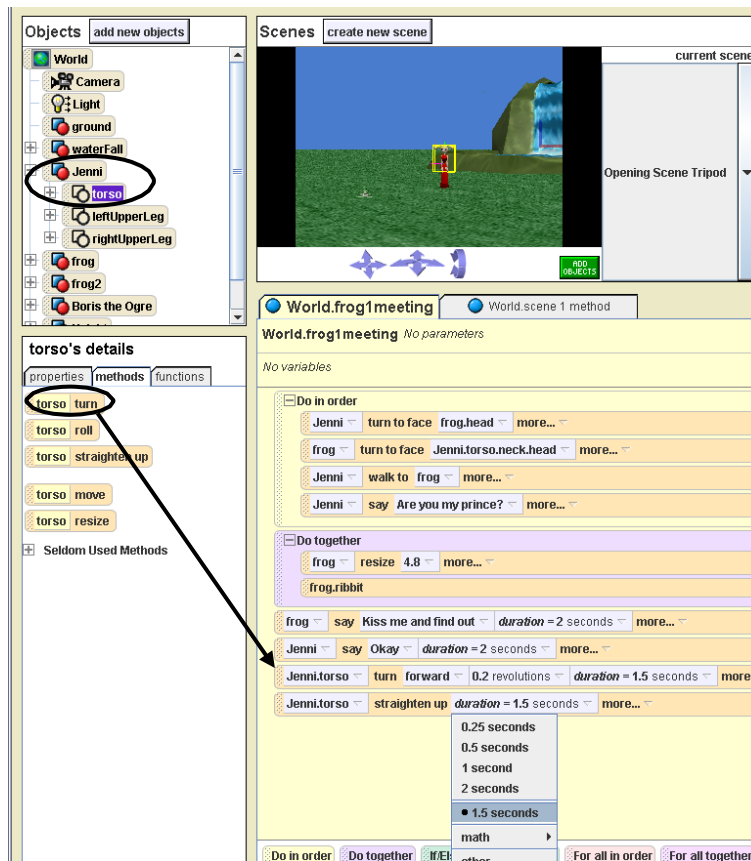
- D 5. Go back to Jenni and add instruction tiles into the *Do in order* instruction so that Jenni *walks to the entire frog* and then says *Are you my prince?*
- D 6. Test this method by selecting *frog1Meeting* in the Events window next to *When the world starts, do*. Then press *Play*.



H. Continue with the following steps:

- D 1. Drag and drop a *Do together* under the *Do in Order* set of instructions.
- D 2. In the *Do together* instruction, add the following method tiles in order:

- *frog resize > other > 4.8.*
 - *frog.ribbit* method (built-in method under the frog's methods tab).
- D 3. Next, outside and below the *Do together* set of instructions, make the frog say “Kiss me and find out” for 2 seconds.
- D 4. *Play* world to test the method.
- I. Finish the meeting by adding and programming the following instructions into *frog1Meeting* method:
- D 1. *Jenni says > Okay > 2 seconds*
- D 2. *Jenni.torso turn > forward > .2 revolutions > 1.5 seconds*
- D 3. *Jenni.torso straighten up > 1.5 seconds*



- D 4. Into another *Do together* instruction add the following:
- *frog > properties > isShowing > false > duration > 2 seconds > style > begin gently.*



- *Boris the Ogre > properties > isShowing > true > style > end gently > duration > 2 seconds.*
- D 5. Into another *Do together* instruction add the following:
- *Boris the Ogre move > backward > 1 meter.*
 - *Boris the Ogre turn to face > Jenni > the entire Jenni.*
 - *Boris the Ogre Roar (built-in method).*
- D 6. Into a last *Do together* instruction for *frog1Meeting* add the following:
- *Jenni turn to face > Camera*
 - *Jenni say > Oh no! Use the arrow keys to help me get out of here and move the mouse to follow me. Click on any frogs you find to make something happen!! > duration > 8 seconds*
- D 7. Outside the *Do together* instruction, as the last instruction, add *Boris the Ogre properties > isShowing > false > duration > 50 seconds.*
- D 8. Test this method and *Save your world* again before continuing.

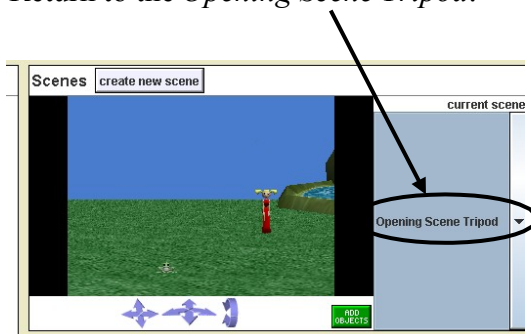
J. Next, you will program the meeting between *Jenni* and *frog2*.

- D 1. First, create a new camera view so you can see what you're working on.
- D 2. Use *blue camera controls* to move around the scene and find *frog2*. Stop when your scene window looks like the one below. Use the positioning buttons to turn the frog so it is facing the same way as the frog in the picture.

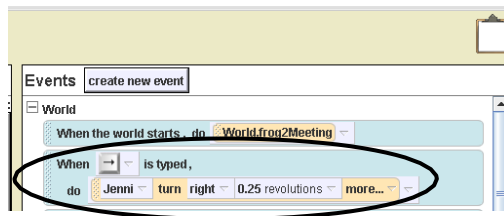


- D 3. Set and name a new camera view by doing what you did in step C1 and C2 above: *Add Objects > more controls > place tripod at camera > New Tripod > frog2view > OK.*
- D 4. Create and name a new method: *Objects tree > World > Detail's panel > methods > create new method > name it frog2Meeting > OK.*
- D 5. Add an instruction for *Jenni* to *walk to frog2*.
- D 6. Drag and drop a *Do in order* tile (from bottom of the Method editor) under *Jenni walk to frog2* tile.
- D 7. Into the *Do in order* instruction, add the following instructions:
- *Jenni turn to face > frog2 > head > the entire head.*
 - *Jenni say > Are you my prince? > duration > 2 seconds.*

- *frog2 > turn to face > Jenni > the entire Jenni.*
- D 8. Into the *Do in order* tile, drag and drop a *Do together* tile and add the following:
 - *frog2 resize > 4.8.*
 - *frog. ribbit* (built-in method).
- D 9. Below the *Do together* tile, but still inside the *Do in order* instruction add the following:
 - *frog2 say > Kiss me and find out. > duration > 2 seconds.*
 - *Jenni say > Okay! > duration > 2 seconds.*
 - *Jenni.torso turn > forward > .2 revolutions > duration > 1.5 seconds.*
 - *Jenni.torso straighten up > duration > 1.5 seconds.*
- D 10. Still inside the *Do in order* instruction, add another *Do together* tile and add the following:
 - *frog2 > properties > isShowing > false > duration > 2 seconds > style > begin gently.*
 - *Knight > properties > isShowing > true > duration > 2 seconds > style > end gently.*
- D 11. Add another *Do together* tile into the *Do in order* instruction and add the following:
 - *Knight kneel before > Jenni > the entire Jenni.*
 - *Knight say > Yes, I'm you're prince! I've been waiting for you!! > duration > 5 seconds.*
- D 12. Add a last *Do together* tile and add the following:
 - *Jenni turn to face > Knight > the entire Knight.*
 - *Jenni walk > 0.75.*
 - *Jenni touch > Knight > torso > neck > head > helmet > the entire helmet.*
- D 13. Under the last *Do together* tile add *Jenni say > At last! > duration > 8 seconds.*
- D 14. Test and *save* your world before continuing.
- K. Now you will make add events so that the player can use keyboard keys to make Jenni move and control what happens in the game.
 - D 1. Return to the *Opening Scene Tripod*.



- D 2. Click *create new event* button in the Events editor.
- D 3. Choose *When a key is typed*.
- D 4. Click on the triangle next to *any key* and choose *Right*.
- D 5. Go to *Jenni's methods* and drag and drop a *turn* tile in place of *Nothing* in the event you just created.
- D 6. Choose *Right > 1/4 revolution*.
- D 7. This event tile should look like this:



- D 8. Copy this event by dragging it into and out of the clipboard and dropping it below the event you just created.
 - D 9. In the copy, change the two places it says *Right* to *Left*. It should show that when the *left arrow key* is typed, Jenni turns *left* 0.25 revolutions.
 - D 10. Copy and paste one of those events again.
 - D 11. On the new bottom event:
 - o Change the key to *Up*.
 - o Delete the *turn* tile.
 - o Drag and drop a *Jenni walk* tile into the *Nothing*.
 - o Select 1.
 - D 12. *Save* and *Play* to test your events. Use the arrow keys to move Jenni around. Fix anything that doesn't work.
- L. Next, you will make events so the mouse will move the camera and make something happen when the player clicks on the frogs.
- D 1. In the Events editor, click on *create new event* button and choose *Let the mouse move the Camera*.
 - D 2. Click on *create new event* button again and choose *When the mouse is clicked on something*.
 - D 3. From the drop down menu next to *anything* choose *frog > the entire frog*.
 - D 4. From the drop down menu next to *Nothing* choose *frog1Meeting*.

- D 5. Use the clipboard to copy that event.
- D 6. In the copy, change *frog* to *frog2* and *frog1Meeting* to *frog2Meeting*.

M. The last thing to do is tell the player how to start the game.

- D 1. Open *scene 1 method* by clicking on its tab in the method editor OR *World > World's details > methods > scene 1 method edit* button.
- D 2. Add the first frog saying "*Click on me!*" with a *duration* of 5 seconds.

N. *Save* your world. *Play* test it to see that everything is working the way you want it to. Fix anything that is not working correctly.

If your world does not work the way it should, ask the teacher for a copy of the code and compare it with what you programmed.

Challenge 16: Jumping in Wonderland

In this challenge, you will:

- Learn how to program if/else structures.
- Learn how to ask and get answers to questions.

A. First look at what your completed challenge will look like:

D 1. Open in Alice: **16JumpinginWonderlandComplete.a2w**.

D 2. Click on *Play* to see how it works. Answer the questions and watch what the characters do.

B. Now open the Alice world you will use to start the challenge:

D 1. *File > Open > 16JumpinginWonderlandStart.a2w*.

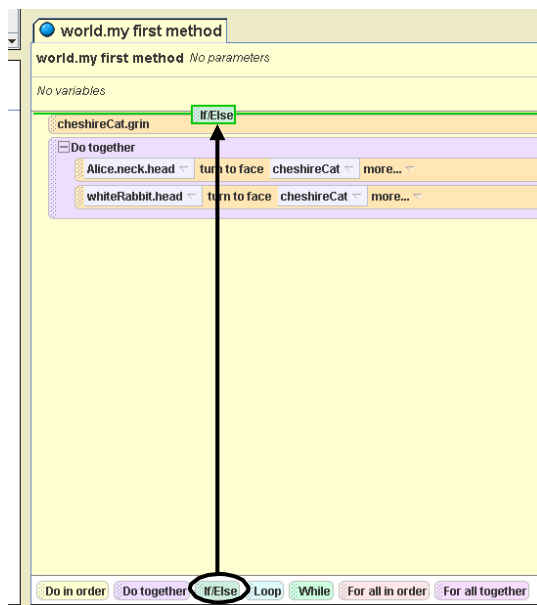
D 2. Notice how there is no question box.

D 3. *Save world as 16JumpinginWonderland* followed by **your initials**.

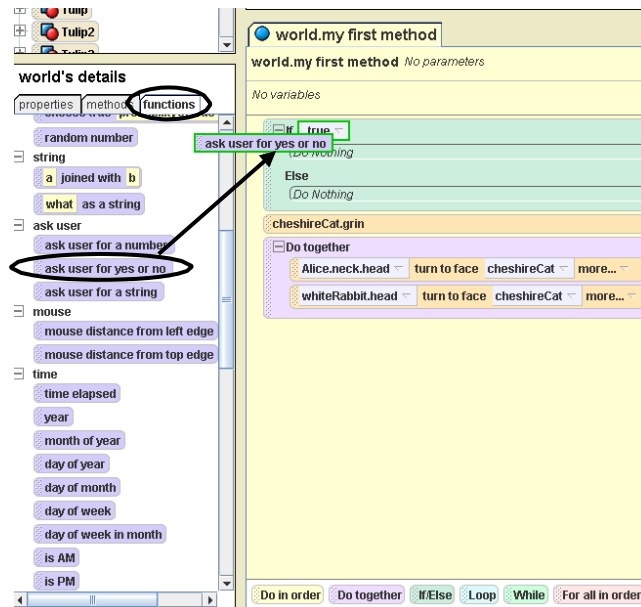
C. First add the instruction that asks the player if he/she wants one of the characters to jump.

D 1. Drag and drop an *If/Else* tile (from the bottom of method editor) into *world.my first method* above the *cheshireCat.grin* instruction tile.

D 2. Select *true* on the menu that appears.



D 3. Open the World's *functions* tab (Objects tree > *World* > *World details* > *functions*).



D 4. Scroll down the list of functions until you see the *ask user for yes or no* function.

D 5. Drag and drop this function into the *If/Else* tile in place of *true* following the word *If*.

D 6. On the menu that appears, click *other* > type *Do you want Alice to jump?* > *OK*.

D. Now you will program things to happen based on the way the player answers the questions. First make Alice jump if the player answers “yes” to the first question.

D 1. Click on the *methods* tab of the World's details.

D 2. Drag and drop a *jump who* tile into the *If/Else* tile in *Do Nothing*, below *If* (above the word *Else*).

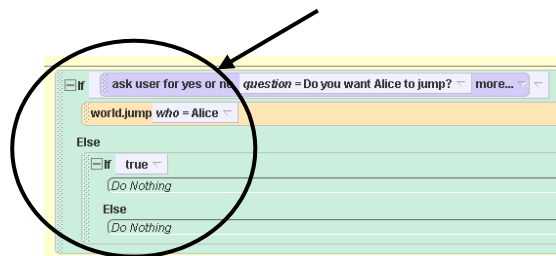
D 3. Choose *Alice > the entire Alice* from the menus that appear. This says that if the player answers “yes” to the question, Alice will jump.

D 4. *Play test* and *save your world*. Answer “yes” to the question and Alice should jump.

E. Next, you will make it so the White Rabbit will jump if the player answers “yes” to the second question.

D 1. In the *If/Else* tile, in place of *Do Nothing* under the word *Else*, drag and drop **another** *If/Else* tile.

D 2. Click *true* when the menu appears. You now should have nested (one inside the other) *If/Else* instruction tiles.



- D 3. Go back to the World's functions tab.
 - D 4. Drag and drop another *ask user for yes or no* function tile into the second (nested) *If/Else* tile in place of *true* following the word *If*.
 - D 5. On the menu that appears, click *other* > type *Do you want the White Rabbit to jump? > OK*.
 - D 6. Click on the *methods* tab of the World's details.
 - D 7. In the second (nested) *If/Else* instruction, drag and drop a *jump who* tile into the *If/Else* tile in *Do Nothing*, below *If* (above the word *Else*).
 - D 8. Choose *whiteRabbit* > *the entire whiteRabbit* from the menus that appear. This says that if the player answers "yes" to the question, the White Rabbit will jump.
 - D 9. *Play* test and *save* your world. Answer "no" to the first question and then "yes" to the next question. The White Rabbit should jump.
- F. Last, make the Cheshire Cat jump if the player answers "no" to having the White Rabbit jump.
- D 1. From the World's *methods*, drag and drop another *jump who* tile in place of *Do Nothing* below the last *Else* (nested *If/Else* tile).
 - D 2. Choose *cheshireCat* > *the entire cheshireCat* from the menus that appear.
 - D 3. *Play* test and *save* your world. Go through all the possible answers to the questions (*using Restart as you need to*) to make sure that it works like the complete file you looked at in the beginning of the challenge.
 - D 4. If you want to, add more questions and make other things happen.

If your world does not work the way it should, ask a teacher for a copy of the code and compare it with what you programmed.
--

Bonus Challenge 16.1: Teacher Helper

In this challenge, you will:

- Create a function to accept user input (what a player types in).
- Use that user input in another method by passing it as a parameter.
- Use that user input as output to the user (what shows up for the player).

A. First look at what your completed challenge will look like:

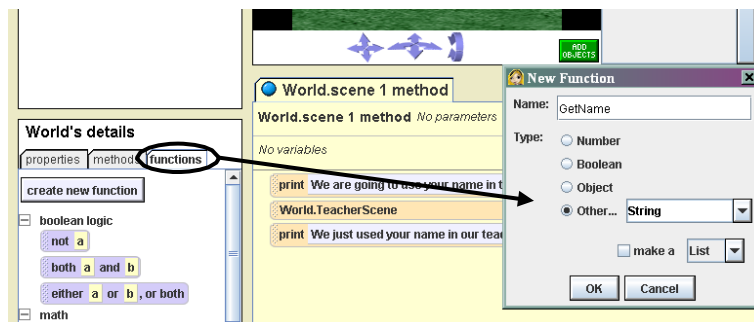
- D 1. Open in Alice: **16.1TeacherHelperComplete.a2w**.
- D 2. *Play* the world to see what happens.

B. Now open the Alice world you will use to start the challenge:

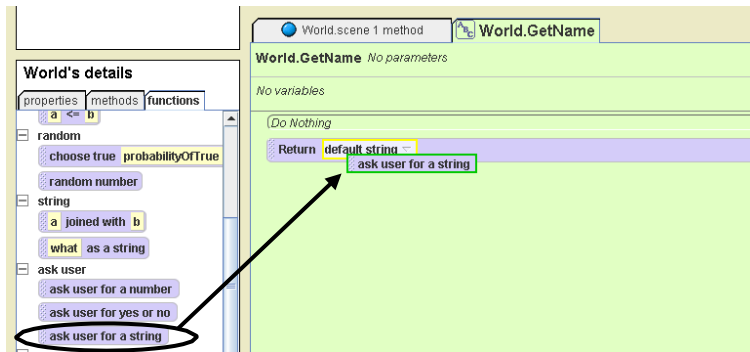
- D 1. *File* > *Open* > **16.1TeacherHelperStart.a2w**.
- D 2. *Play* this world. Notice how the teacher is there but there is no place to type in your name and she says “default string” instead of your name.
- D 3. *Save world as* **16.1TeacherHelper** followed by **your initials**.

C. First, you will create a function to get the name the user types in.

- D 1. Open the World’s *functions* tab (Objects tree > *World* > *World’s details* > *functions*).
- D 2. Click on *create a new function* button.
- D 3. Name the *New Function* “GetName.”
- D 4. Select *Type: Other... String* and click on *OK*.

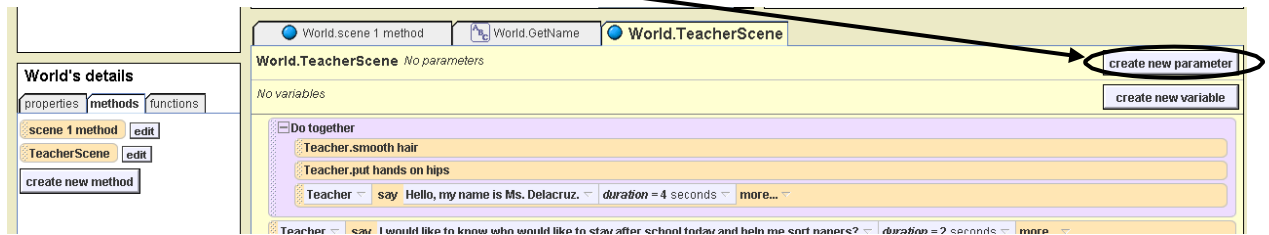


- D 5. From the World's *functions*, drag and drop a *ask user for a string* function into where it says *default string* after *Return*.



- D 6. Choose *other* > type in “*Please type in your name*” > click on *OK*.
- D 7. A *Question* box will appear that shows what it will look like to the user/player. Click on the *X* in the upper right corner to close the box and see your completed instruction.
- D 8. *Play test* and *save* your world. Notice that it doesn't ask the question you just programmed yet. This is because you have to place this function into the *World.scene 1 method*.
- D. Next, you will create a variable to hold this name and put it into the *World.scene 1 method*.
- D 1. Click on *World.scene 1 method*.
- D 2. Go into the World's properties (Objects tree > *World* > *World's details* > *properties*).
- D 3. Click on *create new variable* button.
- D 4. Name the variable “*Name*.”
- D 5. Select *Type: Other...String* and click *OK*.
- D 6. Drag and drop this new *Name* variable tile into the method after the first *print* instruction.
- D 7. Select *set value* > *expressions* > *World.GetName*.
- D 8. *Play test* and *save* your name. Notice how it now asks for your name but the teacher still doesn't use it.
- E. Now, you will create a parameter and change methods to see that parameter so that the teacher uses whatever name is typed in when she talks.
- D 1. Open up the *World.TeacherScene method* (World's details > *methods* > *edit* button next to *TeacherScene*).

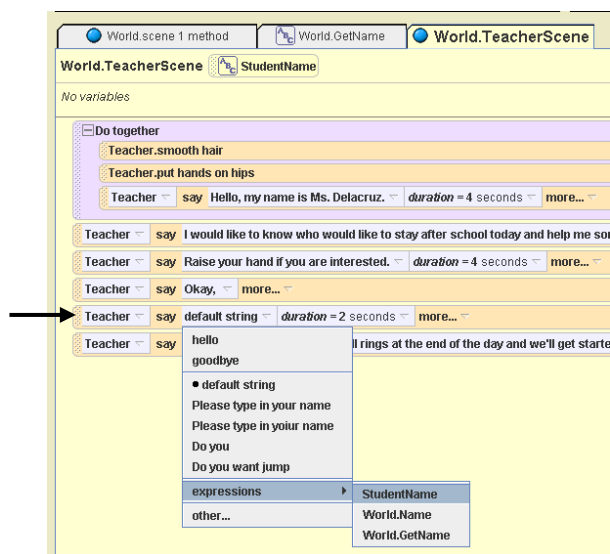
D 2. Click on *create new parameter* in the top right corner of the method editor.



D 3. Name the parameter “StudentName.”

D 4. Select *Type: Other...String* and click *OK*.

D 5. In the instruction under *Teacher say Okay more*, click on the triangle next to *default string* > choose *expressions* > *StudentName*.



D 6. Click back on the *World.scene 1 method* tab.

D 7. Click on the triangle after *default string* after the instruction tile *World.TeacherScene StudentName* => choose *expressions* > *World.Name*.

D 8. *Play test* and *save* your world.

If your world does not work the way it should, ask a teacher for a copy of the code and compare it with what you programmed.

Challenge 17: Cowboy Knowledge Game

In this challenge, you will:

- Take the sample knowledge game and make new questions for the player to answer.
- Add a variable to track the number of right answers.
- Make it so there is more than one right answer to a question.

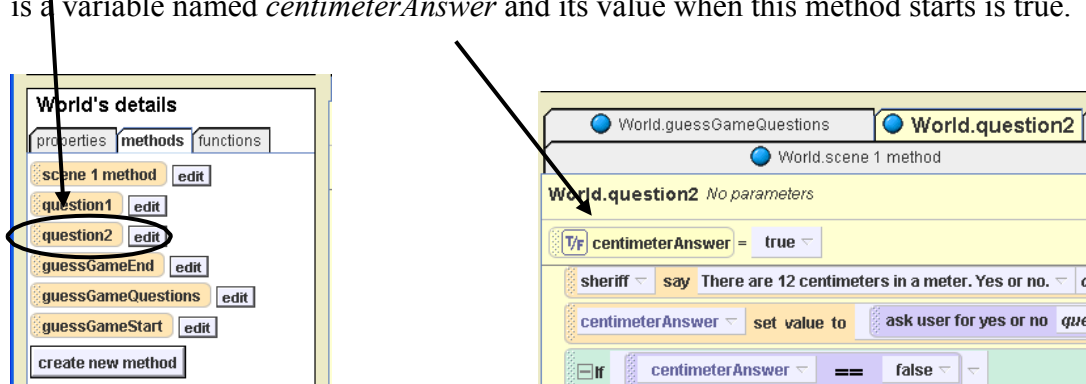


A. First look at the starter world:

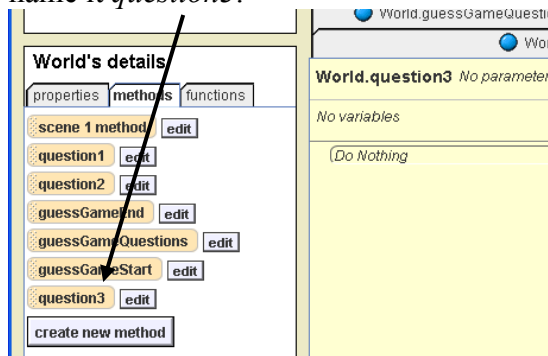
- D1. In Alice, *open 17CowboyKnowledgeGameStart.a2w* and play the world to see what happens.
- D2. Read the words said by the two characters, answer the questions, and see what happens when your answer is right or wrong.
- D3. Save the world as **17CowboyKnowledgeGame** followed by your initials.

B. You will now add another question to the two questions that are already there.

- D1. First look at the instruction tiles for *question 2* (*World > World's details > methods > question2*). The first instruction says *T/F centimeterAnswer = true*. This means there is a variable named *centimeterAnswer* and its value when this method starts is true.

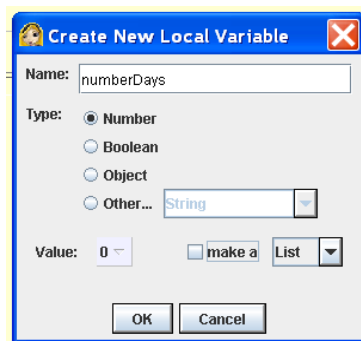


D2. Create a new method called *question3*. Click on the *create new method* button and name it *question3*.



D3. Make a new variable for this method that will store the player's answer to the question.

- Click the *create new variable* button.
- Type *numberDays* for *Name*: (make sure the *Number* radio button is clicked)
- Click the triangle after the *Value: 1*, click on *other* and type *0* in the custom number box.

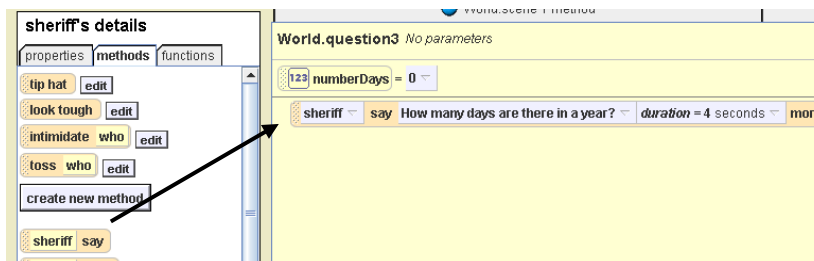


- Click *OK* to close the custom number box and *OK* to return to the method editor. Your new variable should look like the one below:

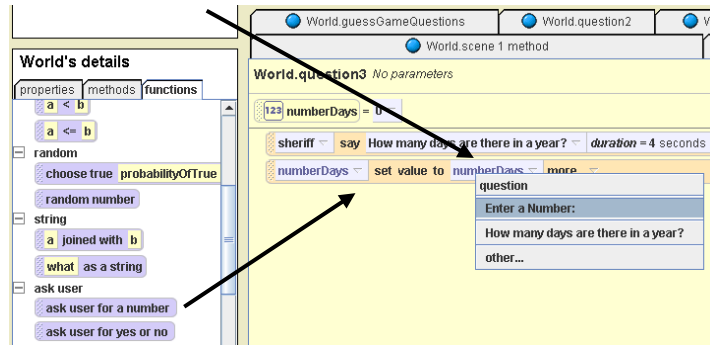


D4. Now add the words that the sheriff will say.

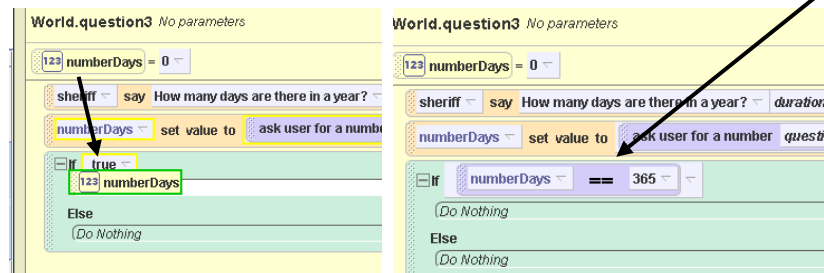
- Choose *sheriff > methods > sheriff say* and put that tile in place of *Do Nothing*.
- Enter *How many days are there in a year?* and click *OK*.
- Change the duration to *4 seconds*.



- D5. Add the code to ask the player for the answer to this question.
- Put the 123 numberDays variable tile under *sheriff say*.
 - Point to *set value*, point to *expressions*, and click on *numberDays*.
 - Click on the *World > functions* and drag the *ask user for a number* tile to replace the SECOND *numberDays*. Click on *Enter a number*.



- D6. Now you will add the code that says that 365 is the right answer to this question.
- Drag an *If/Else* tile from the bottom and put it as the last tile and click *true*.
 - Drag the 123 *numberDays* tile from the top of the method editor to replace the *true* box in the last tile. Point to *numberDays ==* and click on *other*. Enter 365 and click *OK*.

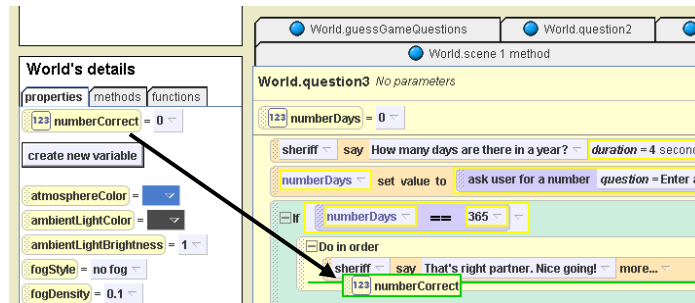


- D7. Save your world again before continuing.

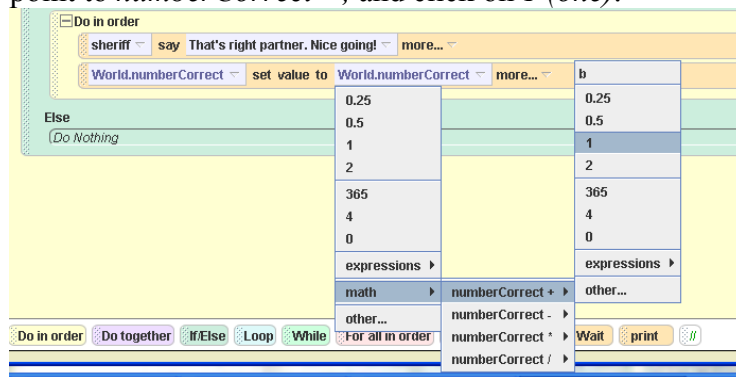
- C. When the player types in the right answer, you want the sheriff to say, “That’s right partner. Nice going!” You also want the counter to add 1 to the number of right answers, and Lana to get bigger, look victorious, and say, “You saved me. Thanks!” Here’s how you do it:

- D1. Put *Do in order* tile (at bottom) in place of the *Do Nothing* in the *If* tile.
- D2. Add *sheriff say* (*sheriff > methods*) to the *Do in order* tile. Type in *That’s right partner. Nice going!*

- D2. Add a statement that adds 1 to the counter when the player gives the right answer.
- Drag the *123 numberCorrect* tile (*World > properties*) into the *Do in order* tile below *sheriff say*. Point to *set value > expressions* and then click on *World.numberCorrect*.

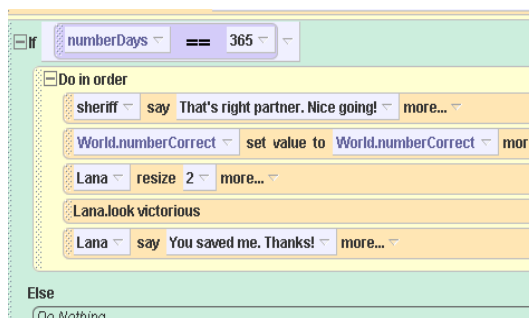


- Click on the triangle after the SECOND *World.numberCorrect*. Point to *math*, point to *numberCorrect +*, and click on *1 (one)*.



- D3. Now make Lana react to the right answer by putting the following *Lana* tiles (*Lana > methods*) below the last tile:

Lana resize 2 (under *Seldom Used Methods*)
Lana.look victorious
Lana say *You saved me. Thanks!*



D. Now you will tell the world what to do if the player gives the wrong answer.

D1. Under *Else* in place of *Do Nothing*, add the following:

- *Do in order*
 - *sheriff turn to face Camera style=abruptly*
 - *Do together*
 - *sheriff turn right 0.25 revolutions*
 - *Lana say No! I'm shrinking! Duration = 2 seconds*
 - *Lana resize 0.35*

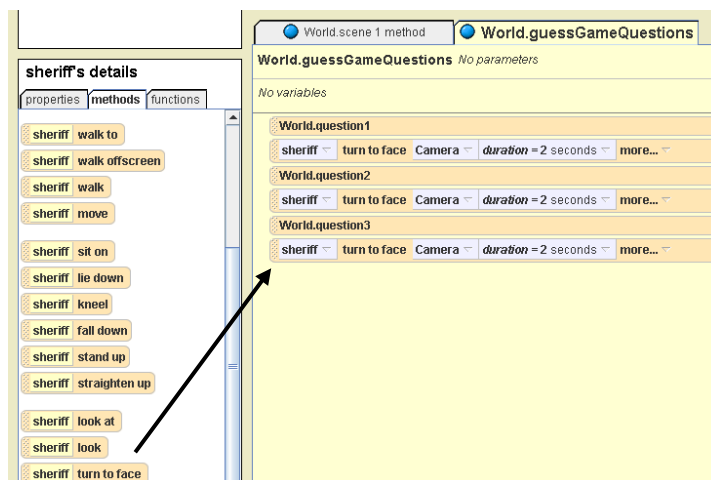
D2. Save and then test your game to make sure everything is working right. Fix any problems and save again before continuing.

E. Now you have to put this new method inside the *guessGameQuestions* method so it will be asked after Question 2.

D1. Click the *edit* button following the *guessGameQuestions* method (*World > methods*).

D2. Drag a copy of the *question3* tile into the space following the SECOND *sheriff turn to face Camera duration = 2 seconds* tile.

D3. Put a copy of the *sheriff turn to face Camera duration = 2 seconds* tile as the last tile in the method.



F. You are almost done. Now the sheriff needs to tell the player how many questions he or she got right.

D1. Open the *guessGameEnd* method (*World > methods > guessGameEnd* method > *edit*).

D2. Drag the *IfElse* tile from below the method editor into the place of the last *Else* above the *sheriff say You got no answers correct duration 3 seconds* tile.

D3. Drag the *sheriff say You got no answers correct duration 3 seconds* tile into the Else part of this last If tile and choose *true*.

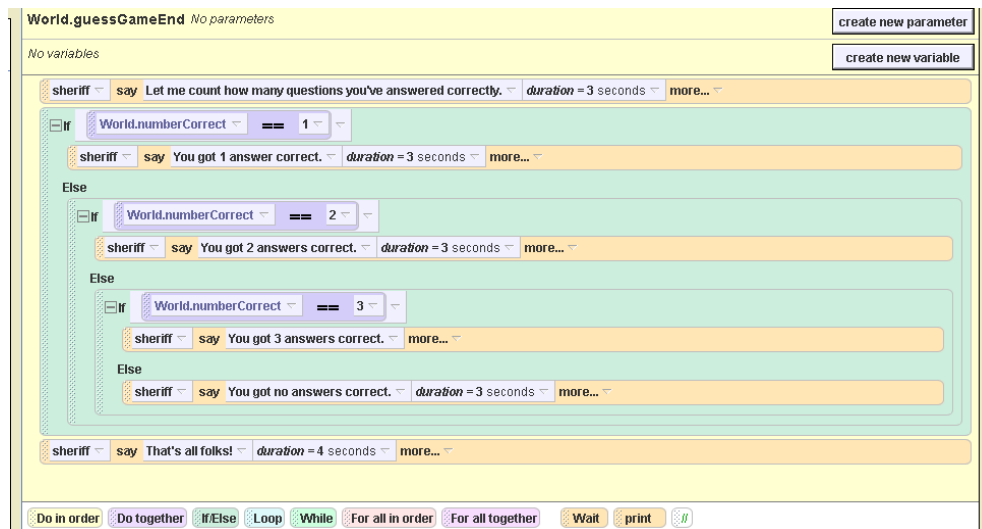
D4. Drag the *123 numberCorrect* tile (*World > methods*) into the *true* part of this last *If* instruction.

HINT: Remember to make sure the *true* turns green so that you are replacing the *true* and not the *Do Nothing* part of the instruction.

D5. Point to *numberCorrect ==*, click on *other* and enter *3 (three)*, and click *OK*.

D6. Add a *sheriff say* tile (*sheriff > methods*) to replace the *Do Nothing* in the *If* part of this last *Else* instruction tile.

D7. Have the sheriff say: “You got 3 answers right.” The last set of instructions should look like this:

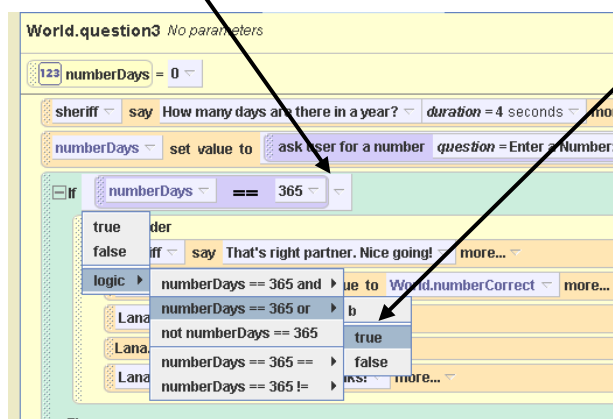


D8. Save and test your world, fix any problems, and then save again.

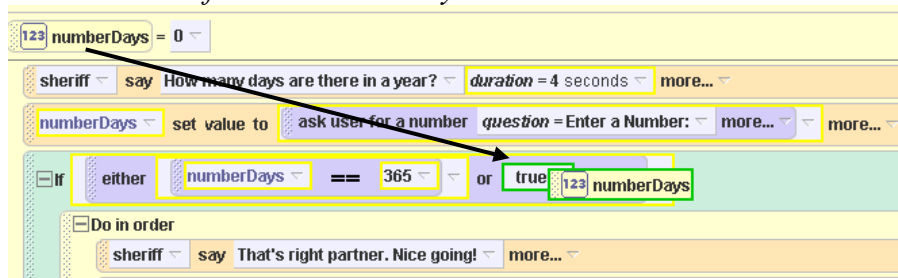
G. Finally, you need to go back and edit *World.question3* so that either 365 or 366 is the answer to the question.

D1. Open the *question3* method (*World > methods > question3 > edit*).

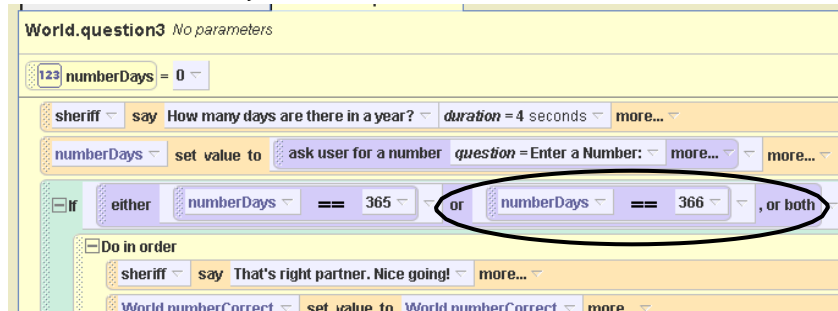
D2. Click the **SECOND** arrow after the 365 in the *If* tile in the method editor. Point to *logic*, point to *numberDays == 365 or*, and click on *true*.



- D3. Drag the `123 numberDays` variable tile from the top of the method editor to replace the `true` of the `If either numberDays == 365 or true` instruction tile.



- D4. Point to `numberDays ==`, click on `other`, and enter `366` and click `OK`.



- H. Save and test your world again. See what happens when you give both right and wrong answers and you give both 365 and 366 as the answer to the last question.
- I. Try adding additional questions. Remember to go through all of the steps that you just followed for each additional question

If your world does not work the way it should, ask a teacher for a copy of the code and compare it with what you programmed.

Challenge 18: Your Game!

In this challenge, you will:

- Look at a game and draw a simple storyboard of it.
- Look at the different games or parts of games.
- Describe what your game will look like.
- Program your game in Alice.

NOTE: If you have already done Challenge 7: Make Your Own Story, **skip to step C** on the next page (playing challenge game examples).

A. First look at an example:

- 1. Open Alice > **FishGameComplete.a2w** > *Play* > Stop or X.

B. Now you will make a very simple storyboard of the Fish Game you just played. This is to give you practice at how to think about the different parts of your game.

- 1. Draw and write about the Fish Game in the following boxes. Divide the boxes into more sections if you need to.

Fish Game Storyboard

OPENING	WHAT HAPPENS?	CLOSING

C. Now look at some examples of games or parts of games:

□1. Open and play each of the following in Alice:

- 8ChangingScenesComplete.a2w
- 9InteractivePenguinDanceComplete.a2w
- 11AmusementParkComplete.a2w
- 12CountingFishComplete.a2w
- 13ExplodingFishComplete.a2w
- 14MatchingInuitsGameComplete.a2w
- 15PrincessAdventureGameComplete.a2w
- 16.1TeacherHelperComplete.a2w
- 17CowboyKnowledgeGameComplete.a2w

D. Now you will write down your ideas for your own game.

- 1. Using the boxes below, write down as many things as you can think of that you would like to include in your game for the Opening, Closing, and What Happens in between.

Ideas for Game Opening	NOW	LATER
Ideas for What Happens	NOW	LATER
Ideas for Game Closing	NOW	LATER

2. Look at your list of ideas to decide which ones you can do **NOW** and which ones you would do **LATER** after learning the skill in a challenge. Use the Challenge Poster to help remind you of what you already know how to do and what more you can learn in other challenges.
- Which ones can you do now with what you have already learned? Put a ✓ or X in the **NOW** column for those ideas.
 - Which ones do you want to learn so you can add them later? Put a ✓ or X in the **LATER** column for those ideas.

E. Using your list of ideas, start to program your game!

1. Open a *New World* in Alice.
2. *Save* this world with any title you want and **GAME** and **your initials**.
3. Add what you want to the world from your list of **NOW** ideas.
4. If you have ideas and you don't know how to program them into your game (from **LATER** ideas):
- Look at the Challenge Poster to help you figure out which challenge would teach you how to do it.
 - Do that challenge and then try to do it in your own game.
5. *Save* your game often.
6. **Good luck and have fun!**