

Logical Reasoning Exercise: NIM

Arrange 15 M&Ms in 3 rows with 3 in the first row, 4 in the second row, and 5 in the third row.

This is a game for two players. You win by picking up the last M&M. When it's your turn, you play by taking as many M&Ms as you like from any row (you may take the whole row if you like) but from one row only. This is the game of NIM and is actually a logical puzzle, for the first player can always win once s/he knows the winning strategy. The puzzle is to figure out that strategy.

Divide into groups of 2 and try playing the game a few times, alternating which person goes first. Think about what strategy to use. If you already know the answer, *don't tell!* You can play to win when you go first; see if your partner can figure out your strategy.

Alice Programming with Lists

At this point you should have seen lists in class. In this activity you will take a program that has lists and modify it to use these lists further. By the end you will hopefully understand lists more, appreciate their use, and feel more comfortable looking at code done by others.

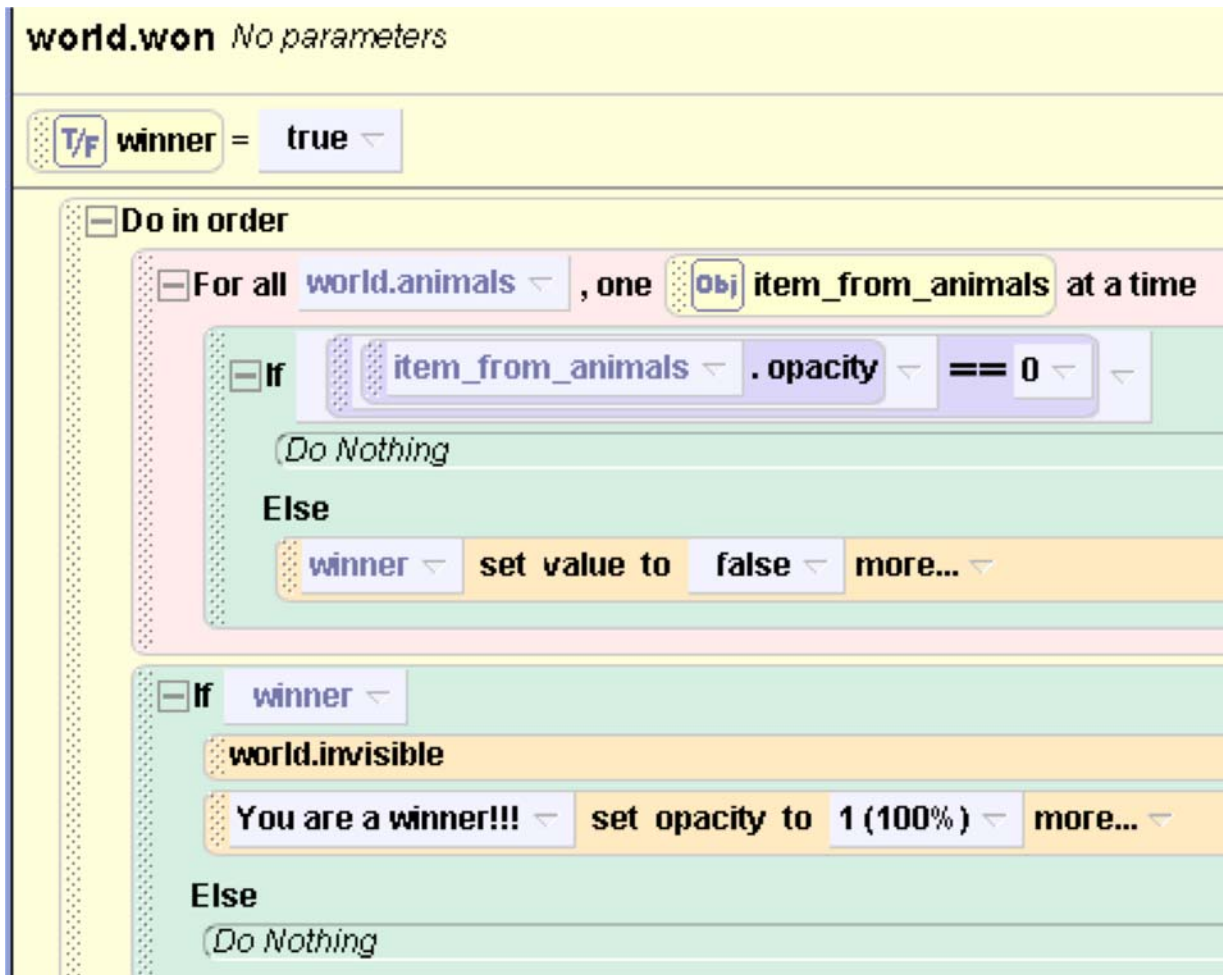
Get the Alice program “5 game students.a2w”. Run it to see what it does. Then look at each method and figure out what it does. You can ask your peer leader for help. Here is an overview:

1. The event determines whenever anything is clicked and calls the clicked method to deal with it.
2. my first method first makes everything invisible, makes the directions appear (3D text), waits, then makes it disappear, and finally makes the game pieces appear. Note the name of the last method is animals.Visible but it makes both animals and non-animals appear (not a great name!).
3. The invisible method uses the animals, nonanimals, and other lists to make everything invisible. Note it uses three lists so that later it can do different things with different lists. Also note that the code that is commented out is what was necessary before lists were used. It is longer and it is much harder to add new items to the game.
4. The animalsVisible method uses the animals and nonAnimals lists to make them visible. It is similar to invisible method.
5. The whatClicked method is only called by the event when something is clicked. First it sees if there are too many clicks. Note it can reuse the invisible method which is a good example of code reuse. If it isn't too many clicks it makes the click animal invisible or does not change any objects if a non-animal is clicked. Note this version does not use lists.

Once you understand the code you can begin to improve it. The first improvement is to change the clicked method to use the lists. Comment out all the checks for each animal. Replace them with a single “For all” loop that checks the animals list. The “whatClicked” parameter has what object was clicked so you can compare that to the current item being considered on the list. If this isn't clear then your peer leader can help. When you are done, the code should work the same but use the lists.

Using lists is good since any change to a list would change the entire game. To see this, add a new animal to the world. Then add it to the animals list. The game should work fine with the new animal.

To make the game a little nicer, we will add a won method that makes the message appear about winning and hides all the other objects. You can add a call to the won method in clicked (in the else clause after you make the object invisible if it was an animal). You could use the lists to check by using:



The loop checks if all the animals are invisible. The winner variable starts true to indicate you won the game. Then loop checks each animal. If it is invisible then it does nothing since you can still win. If it is visible then it sets winner to false since you have lost. The final if statement tells you if you did win and makes objects invisible. This technique uses lists. As an alternative, you will:

1. Create a world variable to keep track of number of animals clicked. This is just like the worldNumberClicked variable to keep track of the total number of objects (not just animals) clicked.
2. Increment your new variable each time animal clicked. You already have the basic logic in clicked. Add something similar to the logic for worldNumberClicked inside your loop where you check if it is an animal. If it is, add a line to increment your new world variable to keep track of how many animals clicked.
3. Have won check if you clicked enough animals and display the winner message if you have. All you need is an if statement that sees if the number of animals clicked is the number in the game.

Congratulations! You have created a fun game.