Asteroids!

Michael Marion
under Prof. Susan Rodger
Duke University
July 2012
What Does The Game Do?

• We’re going to re-create the 1979 Atari classic video game “Asteroids” in Alice.
• In the game, you pilot a ship around in any direction and use the spacebar to shoot a laser at incoming asteroids.
• The incoming asteroids vary in size and speed and fly in from off the screen.
• If you hit all of the asteroids, you win! If you crash into one, you lose.
Topics Covered

• Building a game like this covers many topics in both Alice and computer science in general

• Main Topics:
  – Collision Detection
  – Game Loops
  – Conditional “If/Else” Statements

• Other Topics:
  – Texture Mapping
  – Random Numbers
  – Lists
First Steps

• Open Alice and select the “space” default.
• The first thing we want to do is remove the ground. Go to “ground” in the object tree and click on it. Navigate to ‘properties’ in the lower left corner of the screen and set ‘isShowing’ to ‘false’.
First Steps

- Now we want to add the fighter that we’ll fly around in space. Click on the “add objects” button.
- Scroll over to “SciFi” and add a “fighter” to the world.
First Steps

• Use the camera controls to get an overhead view of the fighter.

• Go to world → properties and make a new boolean variable called “currentlyPlaying”. Set it to “false”.
Making the Asteroids

• Now we want to make the asteroids that we’ll have to shoot in the game.
• For this demonstration, we’re going to use thirty asteroids. However, you can use as many as you like to make the game harder or easier.
• Click on “add objects” and go to “shapes”. Add thirty “icosahedron” objects to the world.
Texturing the Asteroids

• We want to make the asteroids look like real asteroids. To do this, we’re going to steal the moon texture from the ground and apply it to each asteroid.

• For each icosahedron, click on it in the object tree. Click on the “properties” tab.

• Change “skin texture” to “ground.moonTexture”.
Managing the Asteroids

• We’ll want to put the asteroids in a list so that we can manage them later.

• Go to “world” in the object tree and click on “properties”. Click “create new variable”.

• A box should pop up.
Managing the Asteroids

• Name the variable “asteroidsList”.
• Click on “object” under type.
• Make sure to check the box in the bottom-right corner that says “make a List”.
• For each asteroid, click “newItem” and set each new item to its respective icosahedron object.
Resizing the Asteroids

• Move all of the asteroids off-screen out of the camera view.

• Now we want to make the asteroids bigger or smaller and make them all different sizes.

• The good news is that because we made our list, the tedious step is over.

• If we want to do something to all the asteroids, we can now tell the list to modify all of its objects.
Resizing the Asteroids

• Click on “world” in the object tree and create a new method called “resizeAsteroids”.
• Drag in a loop and have it run thirty times (or as many asteroids as you have in your world)
• Pick any icosahedron and drag its resize method into the loop. Set the duration to 0.
Resizing the Asteroids

• Go to world → properties and drag the list of asteroids on to the icosahedron in the method.

• Click “ith item from array” and then expressions → index.
Resizing the Asteroids

• We don’t want every asteroid to be the same size, though. We want some to be smaller and some bigger.
• We’re going to use the random number function to do this.
• Go to world → functions and drag in “random number” to the number in your resize method.
Resizing the Asteroids

- Choose “5” as the minimum and “10” as the maximum. Set the duration to 0.
- Your code should now look like this:
Creating the Laser

• We’re now going to make the laser that we shoot at the asteroids.
• To do this, we’re going to modify a flag object to use just the pole.
• Go to “add objects” and click on the “Medieval” folder. Add a “banner” to the world.
Creating the Laser

- Find the banner in the object tree.
- Click on the plus next to “banner”. Right-click on “flag” and “finial” and delete them both.
- Only “pole” should be remaining.
- Rename the “banner” to “Laser” by right-clicking on “banner” and hitting “rename”.
Creating the Laser

• Position the laser so it’s right underneath the fighter. You may have to resize it, turn it, or roll it. This step requires some tinkering.

• In our example, we used the following in-place methods:
  – Move to Fighter
  – Orient to Fighter
  – Quarter Revolution Left
  – Quarter Revolution Backward

• Set the laser’s vehicle to “fighter” and set isShowing to false.
Shooting the Laser

• Create a new world method and call it “laserShot”.

• The basic idea is to...
  – Detach the laser from the ship
  – Make it visible
  – Move it one hundred meters
  – Make it invisible
  – Move it back to the fighter and orient it to the fighter
  – Reattach the laser to the ship.
Your Method Should Look Like This

```
world.my first method  world.laserShot

world.laserShot  No parameters

No variables

Laser  set vehicle to  world  duration = 0 seconds  more...

Laser  set isShowing to  true  duration = 0 seconds  more...

Laser  move down  100 meters  duration = 1 second  style = abruptly  more...

Laser  set isShowing to  false  duration = 0 seconds  more...

Laser  move to  fighter  duration = 0 seconds  more...

Laser  orient to  fighter  duration = 0 seconds  more...

- Do together
  Laser  turn left  0.25 revolutions  duration = 0 seconds  more...

  Laser  roll left  0.25 revolutions  duration = 0 seconds  more...

  Laser  move down  10 meters  duration = 0 seconds  more...

  Laser  set vehicle to  fighter  duration = 0 seconds  more...
```
Starting the Game

• Remember the variable we made called “currentlyPlaying? We can set “currentlyPlaying” to “true” whenever we want the game to be going on, and we can set it to “false” whenever we want the game to stop. More on this later.

• Now we’re going to write two simple methods that will occur when the game is won or lost.
Game Won

• We’re going to make a method called “GameWon” that checks to see if all the asteroids have been shot (are invisible).

• To do this, we’re going to make a true or false “switch”. We’ll first set the switch to “true”, meaning that all the asteroids are gone.

• Then we’ll loop through the list of asteroids. If at any time an asteroid is visible, we flip the switch to “false”.

• At the end of the loop we can check to see what position the switch is in and then react accordingly.
Game Won

• Go to world→methods and create a new method named “GameWon”.
• Drag in a “loop” and set the loop to loop thirty times.
• Drag in an “if/else” into the loop, and another “if/else” below the loop.
• Click “create new variable”. Make it a boolean called allGone. Set the value to “true”.
allGone = true

Loop 30 times

If true
   (Do Nothing
Else
   (Do Nothing

If true
   (Do Nothing
Else
   (Do Nothing

show complicated version
Game Won

• Pick a random icosahedron and go to its properties. Drag “isShowing” into the if/else within the loop.

• Go to world→properties and click on “asteroidsList”. Drag it in to the name of the icosahedron.

• Pick “ith item from list”, “expressions” and “index”.
allGone = true

Loop index from 0 up to (but not including) 30 times incrementing by 1

If

world.asteroidsList
first item from list
last item from list
random item from list
ith item from list

Else

(Do Nothing)

Else

(Do Nothing)

Index

0 1 2 3 4 5 6 7 8 9

expressions

index

3D Text.currentScore

other...
GameWon

• Drag in “allGone” into the first “if/else” statement.
• Set its value to “false”.
GameWon

• Drag in “gameWon” onto the “true” in the second “if/else” statement.
• Go to world ➔ properties and drag in the “currentlyPlaying” variable into the second “if/else”.
• Set it to “false”.
allGone = true

Loop 30 times times

If item index from world.asteroidsList .isShowing

allGone set value to false more...

Else
(Do Nothing)

If allGone

world.currentlyPlaying set value to false more...

Else
(Do Nothing)
Asteroid Movement

• Now we’re going to make a method that moves our asteroids.
• The basic idea is to have them all turn to face the fighter and then rotate a slightly additional amount to the left or right.
• Then, they fire at a random speed and distance until they stop, where they turn to face the fighter again and the process restarts.
Asteroid Movement

• Pick any random icosahedron and drag in three methods:
  – A “turn to face” the fighter. Set duration to 0.
  – A “turn left” with any distance. Set duration to 0.
  – A “move forward” with any distance and any duration. Set style to abruptly.
Randomizing Asteroid Movement

• Go to world→functions.
• Drag a “random number” onto both distances for the “turn left” and “move forward”. Also drag one on to the duration of “move forward”.
Randomizing Asteroid Movement

• Set the random number’s maximums and minimums by clicking on the purple arrow to the right of “random number”.

• Set the values to the following:

<table>
<thead>
<tr>
<th>Method</th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn Left Distance</td>
<td>-0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Move Forward Distance</td>
<td>150</td>
<td>200</td>
</tr>
<tr>
<td>Move Forward Duration</td>
<td>30</td>
<td>70</td>
</tr>
</tbody>
</table>
Parameterizing Asteroid Movement

• We want to make this method more flexible – not just for one asteroid, but for all asteroids.
• We’re going to use a parameter so we can tell this method to work on any asteroid.
• Make sure you have “asteroidMovement” pulled up in your code editor and click on “create new parameter” in the top-right corner.
<table>
<thead>
<tr>
<th>Method: world.asteroidMovement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event: asteroid</td>
</tr>
<tr>
<td>Event Parameters:</td>
</tr>
<tr>
<td>fighter: duration = 0 seconds</td>
</tr>
<tr>
<td>target: random number: minimum = -0.1, maximum = 0.1</td>
</tr>
<tr>
<td>random number: minimum = 150, maximum = 200</td>
</tr>
<tr>
<td>style = gently, duration = random number: minimum = 30, maximum = 70</td>
</tr>
<tr>
<td>forward: random number: minimum = 30, maximum = 70</td>
</tr>
<tr>
<td>create new parameter</td>
</tr>
<tr>
<td>create new variable</td>
</tr>
</tbody>
</table>
Parameterizing Asteroid Movement

- A box will pop up. Name the parameter “asteroid”.
- Click on “object”.
- Click “OK”.
Parameterizing Asteroid Movement

• You should notice `Obj` asteroid appear next to the name of the method.

• In your method, wherever “icosahedron” appears, you should drag in the `Obj` asteroid instead.
Putting It All Together

• Now we’re going to write a method that makes all of the asteroids move at once.
• Pretty simple: go to world ➔ methods and create a new method. Name it “gamePlay”.
• Drag in a do together.
• Inside the do together, drag in thirty copies of the “asteroidMovement” and pick a different icosahedron for each copy.
Your Method Should Look Like This:

```python
world.asteroidMovement asteroid = icosahedron20
world.asteroidMovement asteroid = icosahedron2
world.asteroidMovement asteroid = icosahedron25
world.asteroidMovement asteroid = icosahedron13
world.asteroidMovement asteroid = icosahedron26
world.asteroidMovement asteroid = icosahedron12
world.asteroidMovement asteroid = icosahedron3
```
Check Collisions

• This is going to be the biggest method we write. It’s going to check to see whether all the objects in the world are colliding with the fighter or the laser.

• To start, go to world ➔ methods and create a new method named “check Collisions”.

• Drag in a “for all together” and select “asteroidsList”. Drag in 2 “if” statements into the “for all together”, one after the other.
world.checkCollisions2  No parameters

No variables

For all world.asteroidsList, every item_from_asteroidsList together

If true

Do Nothing

Else

Do Nothing

If true

Do Nothing

Else

Do Nothing
Check Collisions

• In our first “if” statement, we want to check three things:
  – Is the laser not attached to the fighter? (Has it been shot?)
  – Is the laser touching the asteroid?
  – Is the asteroid visible (has it not already been hit)?

• We need to use a logic function to check three things at once. Go to world→functions and find “both a and b”. Drag it on to “true” in the first “if” statement. Pick “true”.
Check Collisions

• Now we want to add in another “both a and b” so we can check three things.

• Drag another “both a and b” onto the first true in the statement.
Check Collisions

• How do we know if two objects are touching?
  – Answer: If the distance between the two is less than the width of one of the objects.
  – Think about that for a second. Do you understand why this is?

• We’re going to use our statements to check three things:
  – Is the laser not attached to the vehicle?
  – Is the laser touching the asteroid?
  – Is the asteroid visible (has it not already been hit)?
Check Collisions

- **First Condition: Is the laser not attached to the ship?**
  - Go to the “Laser” object you made out of the flag and go to “properties”.
  - Drag in the laser’s “vehicle” on to the first value.
  - A menu should pop up. Select “laser.vehicle !=“ and then select “fighter”.

Check Collisions

• Second Condition: Is the asteroid invisible? (Has it already been hit)?
  – Pick any icosahedron in your object tree. Go to its properties and drag in “isShowing” on the second “true”.
  – Instead of the individual icosahedron, we want it to check all of the items in the list. Drag in “item_from_asteroidsList” at the top of the “for all together” to the icosahedron.
world.checkCollisions2 No parameters

No variables

1. For all world.asteroidsList, every item_from_asteroidsList together

2. If both both Laser .vehicle != fighter and item_from_asteroidsList .isShowing and

   Do Nothing

   Else

   Do Nothing

3. If true

   Do Nothing

   Else

   Do Nothing
Check Collisions

• Third Condition: is the laser touching an asteroid?
  – Go to Laser’s functions and drag in an “is within [threshold] of” to the last “true”.
  – In the drop-down menu that will pop up, pick any icosahedron and any distance.
  – Drag in item_from_asteroidsList to the icosahedron.
  – Go to the icosahedron’s functions and drag in “width” where the distance is.
  – Drag in item_from_asteroidsList to subject = icosahedron
Check Collisions

• Now we’re going to move into the second “if” statement.
• We’re going to make a copy of our first “if” block because this one will be very similar.
• Drag the first if block to the clipboard to make a copy. Drag in a copy just underneath the first “if” block within the “for all together”.
Your Code Should Look Like This
Check Collisions

• In “[Laser] is within [subject = item_from_asteroid]’s width...” change “Laser” to “fighter”

• Remove the condition that says “Laser.vehicle == fighter” by dragging it to the trash. There should be a “true” value where it was. This is OK.
Your Code Should Look Like This

Continued, in the same line...
myFirstMethod

• Go to world.myFirstMethod.
• Drag in “world.resizeAsteroids”
• Go to world→properties and drag in currentlyPlaying. Set its value to “true”.

![Image of world.my first method panel]
Events

• Go to Events in the top-right corner of the screen.
• There should already be one that says “when world starts, do world.myFirstMethod”.
• Create a new event. Pick “let arrow keys move subject”. Select “fighter”.
Events

- Create two events “while something is true”. Pick “world.currentlyPlaying” for the ‘something’.
- In the first “while” event, drag in “gamePlay” to the “during”.
- In the second “while event, drag in “checkCollisions” to the “during”.
- Create a last event – “when a key is typed.”
- Select space and then drag in “world.laserShot”.


When the world starts, do world.my first method

Let move fighter

While world.currentlyPlaying is true
   Begin: Nothing
   During: world.gamePlay
   End: Nothing

While world.currentlyPlaying is true
   Begin: Nothing
   During: world.checkCollisions
   End: Nothing

When Space is typed, do world.laserShot
Challenges

• Can you create a score object that counts the number of asteroids that have been hit?
• Can you create a billboard with instructions for the game? Can you make it disappear when the game starts?
• Can you create 3D text that appears when you win or lose that tells you whether you’ve won or lost?