Ready, SET, go!

Sets: 6

By Melissa Dalis
Professor Susan Rodger
Duke University
July 2011

Overview

- This tutorial will teach you how to build SET, a card game whose objective is to identify as many groupings (sets) of three cards as possible.
- SET can be used as an early introduction to set theory, combinatorics, randomness, probability, and other areas of higher level math.
- Programming concepts reinforced are events, loops, dealing with several methods, and parallel arrays (a intro to a higher-level concept called maps).

Understanding the game

- In our game of SET, we will have 27 cards, each with 3 features: color (red, blue or green), number (1, 2 or 3), and shape (oval, squiggle or rectangle).
- The goal of the game is to find as many sets as possible.
- A set is a collection of 3 cards, where, in each category, all of the cards are the same or all of the cards are different.

<table>
<thead>
<tr>
<th>SET</th>
<th>Not a SET</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="card1" alt="SET Card Examples" /> <img src="card2" alt="SET Card Examples" /> <img src="card3" alt="SET Card Examples" /></td>
<td><img src="card4" alt="Not a SET Card Examples" /> <img src="card5" alt="Not a SET Card Examples" /> <img src="card6" alt="Not a SET Card Examples" /></td>
</tr>
<tr>
<td>All different colors, same shape, and different numbers.</td>
<td>Not all same or different colors.</td>
</tr>
<tr>
<td>All same color, different shapes, and same number.</td>
<td>Not all same or different shapes.</td>
</tr>
<tr>
<td>All different colors, different shapes, and different numbers.</td>
<td>Not all same or different colors.</td>
</tr>
</tbody>
</table>
Understanding the starting world

- Click on the + sign next to cards in the object tree, and you’ll see in that folder all 27 of the cards used in the game.
- Click on the + sign next to squares in the object tree, and you’ll see in that folder the 12 spots on the table where we will put the dealt cards.

Understanding the events

- In the Events section in the top right corner, there are 27 events already set up for you, one for each card.
- When a player clicks on a card, we need to be able to recognize which card, color, number, shape, it is, and if this is the first, second, or third card of the set they are trying to create.
- Each of these events calls a helper method, which we will write to respond to the card they click.

The arrays

- Click on world in the object tree, and go to its properties tab. There are several lists already created.
- Notice that allCards contains all 27 of the cards, and allColors, allShapes, and allNumbers contain the 27 colors, shapes, and numbers, respectively, that correspond to the object in the allCards array.
  
  – For example, item2 in allCards is redSquig1, item2 in allColors is red, item2 in allShapes is squig, and item2 in allNumbers is 1.

Responding to the click

- Click on world in the object tree, and go to its methods tab.
- Click on edit next to the method called helper.
- Notice that helper has 2 parameters: card and index. Since the allColors, allShapes, and allNumbers array correspond to one another, index refers to the correct item in each of the arrays that describes card.
- When a player clicks on a card, we want it to turn yellow, and if he has already clicked on it, we want it to turn back to white.

```
world.my first method
world.helper
```
• Drag in an If/Else statement, and select true.
• Click on one of the cards in the object tree (I chose greenOval1, but it doesn’t matter), and go to its properties tab.
• Drag color onto the true, and select != (not equal to), yellow.
• Drag card from the top of the method onto greenOval1.
• Drag card into the first Do Nothing, and select set color to, yellow.

Keeping track of the cards
• Since a SET contains 3 cards, we want arrays to keep track of the 3 chosen cards.
• Click on world in the object tree, and go to the properties tab.
• Click create new variable, and name it 3Cards, make it type Object, check make a List and change List to Array.
• Click new item three times so that you have items 0-2, and click OK.
• We’ll add values for these later.

Back to helper
• Go back to your helper method.
• First we want to set the numChosen\textsuperscript{th} item in 3Cards to card, the actual card object we passed into the method.
• Drag 3Cards into the If, below the card set color line, and select set item <index> to <item>, expressions, world.numChosen, expressions, card.
• Now drag 3Indices below that line, and select set item <index> to <item>, expressions, world.numChosen, expressions, index.
• Drag **3Numbers** below that line, and select set item <index> to <item>, expressions, numChosen, and 0.

• Drag **allNumbers** (also in world’s properties) onto the 0, and select ith item from the array, expressions, index.

• Now drag **3Colors** below that line, and select set item <index> to <item>, expressions, numChosen, default string, for now.

• Drag **allColors** onto default string, and select ith item from the array, expressions, index.

• Do **3Shapes** the same way.

<table>
<thead>
<tr>
<th>Set Item</th>
<th>World.numChosen</th>
<th>To</th>
<th>Index</th>
<th>In</th>
<th>World.3Indices</th>
<th>More...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Item</td>
<td>World.numChosen</td>
<td>To</td>
<td>Item index</td>
<td>From</td>
<td>World.allNumbers</td>
<td>In</td>
</tr>
<tr>
<td>Set Item</td>
<td>World.numChosen</td>
<td>To</td>
<td>Item index</td>
<td>From</td>
<td>World.allColors</td>
<td>In</td>
</tr>
<tr>
<td>Set Item</td>
<td>World.numChosen</td>
<td>To</td>
<td>Item index</td>
<td>From</td>
<td>World.allShapes</td>
<td>In</td>
</tr>
</tbody>
</table>

**Finishing helper**

• Since the player has unclicked the card, now we want to decrement numChosen.

• Drag **numChosen** below the no color line, and select decrement by 1.

• *This is important:* make sure the duration of every single line in this method is 0.

• To do this, click more at the end of each line, duration, 0 seconds.

<table>
<thead>
<tr>
<th>Else</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card</td>
</tr>
<tr>
<td>Decrement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Finished helper</th>
</tr>
</thead>
<tbody>
<tr>
<td>If</td>
</tr>
<tr>
<td>Card</td>
</tr>
<tr>
<td>Set Item</td>
</tr>
<tr>
<td>Set Item</td>
</tr>
<tr>
<td>Set Item</td>
</tr>
<tr>
<td>Set Item</td>
</tr>
<tr>
<td>Set Item</td>
</tr>
<tr>
<td>Increment</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>Card</td>
</tr>
<tr>
<td>Decrement</td>
</tr>
</tbody>
</table>
Dealing the cards

• Click on world (object tree), and go to its methods.
• Click create new method, and name it deal.
• This method will deal the 12 cards at the beginning of the hand, and also refill the cards with 3 more after a player has found a SET.
• Create 3 parameters for the method: index (Number) which will refer to the index of the card being dealt, refill (Boolean) which will refer to whether or not we are dealing a new hand or just refilling, and replacing (Object) which will refer to the card we’re replacing in the refill case.

• If the random card has already been dealt, we need to deal another one instead.
• Drag a While into the method, and select true.
• Drag greenOval1 onto the true, and select greenOval1.isShowing.
• From world’s properties, drag allCards onto greenOval1, and select item item from the array, expressions, rand.
• Drag the rand set value line onto the clipboard to make a copy, and paste it into the While.

Choosing a random card

• Create a new Number variable in the method named rand.
• Drag rand into the method, and select set value, 0, for now.
• From world’s functions tab, drag random number onto the 0.
• Click on the purple more 3 times to change the minimum to 0, maximum to 27, and integerOnly to true.

Moving the card to the correct spot

• Create a new Object variable in the method named card.
• Drag card to the bottom of the method, and select set value, camera, for now.
• Drag allCards onto camera, and select ith item from the array, expressions, rand.
• Now, if the card is replacing another card, we want it to move to that card, but otherwise we want to move it to the next open square.
• So drag in an If/Else, and select true.
• Click true, and select expressions, refill.
• Drag card into the first Do Nothing, and select move to, expressions, replacing.
• Drag card into the second Do Nothing, and select move to, camera, for now.
• From world’s properties, drag allSquares onto camera, and select ith item from the array, expressions, index.
• This is important: make sure the duration of every single line in this method is 0 (or false).
• From greenOval1’s properties tab, drag isShowing into the very end of the method, select true, and change greenOval1 to card.

Fixing the cards

• The cards begin in a weird position.
• Go to world.my first method, add a Loop, and select 27.
• Drag a Do together into the Loop.
• Drag greenOval1 into the Do together, and select 3 times so that it: (1) rolls right ¼ revolution, (2) turns to face the camera, and (3) resizes by a factor of 0.7, and change the durations to 0.
• Drag allCards onto each of the greenOvals, and select ith item from array, expressions, index.

Dealing the first 12 cards

• Drag a Loop into the method, and select 12.
• From world’s methods, drag deal into the Loop, and select expressions, index, false, <None> (since we’re not refilling).
• Click play to see your random 12 cards dealt (yours will be different than mine)!
Judging the SET

- In world’s methods, click create new method, and name it judgeSet.
- Remember that a SET needs to have all the same or all different numbers, all the same or all different colors, and all the same or all different shapes.
- Create 6 new Boolean variables in the method: numbersSame, numbersDiff, colorsSame, colorsDiff, shapesSame, and shapesDiff.

world.judgeSet No parameters

- From world’s properties, drag 3Numbers onto the first 1, and select ith item from the array, 0.
- Drag item 0 from world.3Numbers onto the clipboard to make a copy, and paste it onto the other three 1’s.
- Change the second 0 to 1, and the last 0 to 2 so that it basically says item 0==item 1 or item 0==item 2.
- For all the numbers to be different we need to make sure of 3 things: item 0 doesn’t equal item 1 or 2, and item 1 does not equal item 2.
- Now drag numbersDiff into the Do together, and select set value, true.
- Click on the true, and select logic, and, true.
- Click on the second true, and select logic, and, true.

Evaluating the Booleans

- Drag in a Do together.
- For the numbers to be the same, the 0th number needs to equal the 1st number and the 2nd number (and, by the transitive property, that would mean that the (1st=2nd).
- Drag numbersSame into the Do together, and select set value, true.
- Click on true, and select logic, and, true.
- From world’s functions, drag a==b onto the first true, and select 1, 1, and do the same for the second, for now.

- Drag a!=b onto each of the true’s, and select 1, 1.
- Drag 3Numbers onto the first 1, and select ith item from the array, 0.
- Drag item 0 from world.3Numbers onto the clipboard to make a copy, and paste it onto the other five 1’s.
- Change the second 0 to 1, the fourth 0 to 2, the fifth 0 to 1, and the sixth 0 to 2, so it basically says item 0 != item 1 and item 0 != item 2 and item 2 != item 1 != item 2.
• Drag `colorsSame` into the Do together, select true, then click on the true and select logic, and, true.
• Create a new String variable in the method called `temp`, so we can use this as a placeholder (instead of the 1) since we’re working with Strings now.
• Drag `temp` onto the first true, and select ==, expressions, temp, and do the same for the second.
• Drag `3Colors` onto the first `temp`, and select ith item from the array, 0.
• Drag that to the clipboard and paste it on the other 3 temps.
• Change the second 0 to 1 and the fourth 0 to 2.

If it is a set...

• That was the hardest part, good job!
• Now drag an If/Else into the very bottom of the method, and select true.
• Click on the true, and select logic, and, true, and do this again for the second true.
• Drag `numbersSame` onto the first true, and select logic, or, expressions, `numbersDiff`.
• Change the second true to `colorsSame` or `colorsDiff`, and the third true to `shapesSame` or `shapesDiff`.

Counting the number of SETs

• Click Add Objects, and create a 3D Text object that says “Sets: 0”.
• In the object tree, rename 3D Text “setsText”.
• Make it smaller, and move it to the top left corner.
• Change its color to black if you want to in its properties tab.
• In the properties tab, click create new variable, name it `numSets`, and set its type to Number and initial value to 0.
• Click Done.
Back to judging

- Go back to your `judgeSet` method.
- Drag the `numSets` variable you just made into the `If/Else`’s first `Do Nothing`, and select increment by 1.
- From `setText`’s `properties` tab, drag `text` below the increment line, select `other`, and type “Sets: “.
- From `world`’s functions, drag a joined with b onto `Sets;`, and select default string.
- Also from `world`’s functions, drag what as a string onto default string, and select camera.
- Also from `world`’s functions, drag int a as a string onto camera, and select expressions, `numSets`.
- Change the `duration` to 0.

If not a set...

- Drag a `Loop` into the `Else`, and select 3.
- From the `object tree`, drag `setsText` into the `Loop`, and select `set color to, no color`.
- Drag `3Cards` onto `setsText`, and select `ith item from array, index`, and change the `duration` to 0.
- From `world`’s `properties`, drag `numChosen` into the very bottom of the method, and select `set value, 0`.

- If it’s a set, we want the 3 cards to disappear and to replace them with 3 new ones.
- Drag a `Loop` below the `setsText` line, and select 3.
- From `setText`’s `properties` tab, drag `isShowing` into the `Loop`, and select false.
- Drag `setText`’s `color` property into the `Loop`, and select no color.
- From `world`’s `properties`, drag `3Cards` onto both `setText` lines, and select `ith item from array, expressions, index`, and change duration to 0.
- From `world`’s `methods`, drag `deal` into the `Loop`, and select `expressions, index, true, <None>`.
- Drag `3Cards` onto the `<None>`, and select `ith item from array, expressions, index`.

- Create a 3D text object, and type “Not a SET!”
- Change the `color` to black, and move and resize it to the middle of the screen.
- Move it forward so it will definitely be in front of the cards.
- In the `object tree`, rename the 3D text to `noSetText`.
- We only want this to show up when the 3 cards aren’t a `SET`, so in `noSetText`’s `properties`, change `isShowing` to false.

- Now drag `isShowing` into `judgeSet`, right before and right after the `Loop` you just made.
- The first `isShowing` should be `true` and the second should be `false`. 
When to judge the set

• When numChosen is 3, we want to run judgeSet to evaluate whether or not the 3 selected cards form a set.
• In the Events section in the top right corner, click create new event and select While something is true (your event will show up at the bottom).
• From world’s properties, drag numChosen onto the <None>, and select ==, 3.
• From world’s methods, drag judgeSet into Begin.

Extensions

• Create a billboard with instructions to display at the beginning of the game.
• Create a timer that counts down from 100 (and pauses for the instructions if you do add instructions).
• Create different levels where the player has less and less time to find a set.

You’re done, good job! Play your world!