Making games using sprites and messages
FIT1040 Programming Fundamentals (Week 10)
Lecturer, Campus
Faculty of Information Technology
The value of iteration in development: The marshmallow challenge
Learning objectives

By the end of this week students should:

- Be able to create a game that involves multiple sprites interacting via messages
- Be able to create a game that involves *Scribble* sprites responding to events
- Be able to create a *Scribble* application that uses Sprite clones
Today’s agenda
Today’s agenda

- Some *Scribble* awesomeness
  - Making sprite clones
  - Using the `run` code block
  - The `script` code block
  - Using `run` and `script` with a list of sprites (yes, you read that correctly, a list of sprites)

- Refactoring Tic-Tac-Toe
- Making a BCG Grid with Sprites
- Asteroids– in *Scribble*!
- Summary
- Reminders
Scribble awesomeness
The clone operator (reporter)
The clone operator

- Makes a “clone” of a sprite
- A clone is a special copy of a sprite. Has all the scripts and variables of the “parent” sprite. Gets a default name.
- Becomes “live” on stage on creation
- Can be stored in a variable or a list
(attribute) “of” (object) reporter & object reporter

- Object reporter returns an object
- (attribute) “of” (object) reporter returns some value of an object

- All three scripts on the left do exactly the same thing – say the x-coordinate Sprite1
Run command block

- This command block runs a script
  - What’s the big deal about that?

- We’ll see in a moment …
The *script* command block

- Let’s you create something like a custom code block on the fly.

- Seems just plain weird.
Putting *run* and *script* together

- Let’s one sprite (or the canvas) run a script that “runs” in another sprite.
- This script in sprite1 will cause sprite2 to say it’s name
Putting it all together (cloning a sprite to a variable)
Cloning a sprite into a variable
Cloning a sprite
Putting it all together (making a list of sprites)
Making a list of sprites
Making a list of sprites
Refactoring Tic-Tac-Toe
Tic-Tac-Toe

- One sprite was manually copied 8 times and moved into place
- Each one was given a row and column number to use when updating the “board”
- Scripts they run were identical
- Why not create (a hidden) a single sprite that gets cloned 9 times?
  - Change the row and column numbers of each of the nine.
  - Move them into position (using the row and column numbers).
  - Sprites could be incorporated into “board” list (not going to do that but could).
- No change needed anywhere else!
BCG Grid
BCG Grid

- We looked at this in week 4.
- Products are classified as Stars, Question marks, Cash cows or Dogs according to their market share and the growth of the market they are in.
- Different strategies can apply to different classifications.
- Widely used in company and product portfolio analysis.
- Example:
  - **Star**: high market share in high growth market, make lots of money but at the same time require lots of investment as the market is likely to be volatile and change quickly (think Apple’s iPad).
2x2 BCG Matrix

Taken from: http://www.quickmba.com/images/strategy/matrix/bcg/growthshare.gif
An example BCG Matrix

Taken from http://upload.wikimedia.org/wikipedia/commons/7/76/Folio_Plot_BCG_Matrix_Example.png
A case study to work through …

- A company selling watches, has 3 products

<table>
<thead>
<tr>
<th>Product</th>
<th>Sales ($)</th>
<th>Market Share</th>
<th>Comp. Market Share</th>
<th>Market Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mens Watch</td>
<td>$200,000</td>
<td>5%</td>
<td>15%</td>
<td>2%</td>
</tr>
<tr>
<td>Mens Chronograph</td>
<td>$250,000</td>
<td>7%</td>
<td>8%</td>
<td>7%</td>
</tr>
<tr>
<td>GPS Sports Watch</td>
<td>$100,000</td>
<td>30%</td>
<td>15%</td>
<td>15%</td>
</tr>
</tbody>
</table>
A matrix made with *Scribble*
BCG Matrix application

- Uses a standard “pencil” sprite to draw the grid and the axes
- Uses a “product” sprite as a template to represent each product entered.

1. When the green flag is clicked, the grid redraws, cloned sprites delete themselves, a list of sprites is deleted.
2. The user is asked how many products they wish to analyse
3. For each product, data is entered, a clone of “product” is created.
4. At the end all the “product” clones draw themselves on the grid
5. They then can react to “clicks” and display information that is stored within them (sprite scope variables).
Clones being created and added to a list

- Clone is added to a list variable.
- Script is then run (custom code block) to store data in the sprite
Clones drawing themselves

- After all the clones have been created (they start hidden), they draw themselves on the stage.
- Access from the list of sprites, one at a time.
- Note use of code block to make script easier to read.
Clones deleting themselves?

- When the green flag is clicked … make sure the “product” sprite isn’t deleted (check the object name).
  - “Product” refreshes the list of sprites

- Note clones get the variables and the scripts of their “parent”
A taste of OO-programming

- In industrial OO languages … classes are templates used to make objects
- In this app … “product” is like a class
- Classes have data and “processes”
- Our sprites have their own data and their own scripts
- Creating a clone of “product” in this app is just like object instantiation in OO programming

- You’ll get more of this in later units … but it’s nothing to be scared of
Asteroids
Plan the game (reminder)

1. Create a storyboard
   1. Like flowchart where the “nodes” are screen displays
2. What kind of decisions will the players make?
3. What resources are required? (Sounds, video, graphics)
4. What is the goal of the game (what are the players trying to do)?
5. Think about how to keep players coming back (a hook).
Storyboard

Game starts

Bullets shoot

A win if not dead after 90 seconds

Asteroids appear

Asteroids disappear when hit by bullets

A loss if touched by an asteroid.
What kind of decisions will the players make?

- Where to fly the ship:
  - Avoiding collisions
  - Getting in position to fire

- When to fire:
  - Hit incoming asteroids to destroy them
What resources are required? (Sounds, video, graphics)

- Sounds for bullets and collisions
- Sounds for ship movement
- Graphics for the ship
- Graphics for the asteroids
- Graphics for bullets
- Backgrounds for wins and losses
**Game play process**

A. Game starts, timer starts
B. Asteroids randomly appear at edges moving towards middle
C. Ship moves and fires in response to user commands
D. If the ship is hit game over
E. If asteroids are hit by bullets they are deleted
F. If timer gets to 90 seconds, game is won
Variables and events needed

Events:
- Ship moves forward
- Ship moves backward
- Ship turns left
- Ship turns right
- Ship fires a bullet (bullet will be cloned)
- Asteroid enters game
- Asteroid hits ship
  - End of the game
- Bullet hits asteroid
  - Asteroid is destroyed
  - Bullet goes off screen
The ship sprite

- Once script for each event
- “fires” bullet by creating a clone

When up arrow key pressed
move 10 steps

When right arrow key pressed
turn 5 degrees

When left arrow key pressed
turn 5 degrees

When down arrow key pressed
move 5 steps
The bullet sprite

- Non-template sprites deleted when green flag is clicked.
- Respond to “fire” event
- Move in direction set by “ship”
- Delete themselves when at the edge.
The asteroid sprite

- Non-template sprites deleted when green flag is clicked.
- Coupled to canvas (not great design).
- Canvas clones asteroid and sets it in motion.
- Direction randomly chosen
- Location determined by direction.
The asteroid sprite (cont).

- Note use of `<touching (object)>` and `<touching (color)>` operators
- Speed, Started? and AstDirection are local (sprite) variables.
Reflection

- Improvements?
  - Sound
  - Asteroids split when hit (clone to new scale, slightly different directions)
  - Score of hits?
  - Better starting locations for asteroids
  - Levels?
  - Who is up for the challenge?

- Asteroid and canvas a little too coupled
  - Have to be careful with `run` code block
Summary
Summary

- The Marshmallow challenge
  - Experiment, iterate,

- Cloning sprites
  - Creates a which can be stored in a variable, or a list
  - Can be deleted

- Run and Script allow objects to perform operations (scripts) on other objects
  - Can access and alter variables in sprites (rather like object-oriented programming you will meet in later units)

- Scribble has plenty of mind-bending awesomeness
Reminders
Reminders

- Work hard but sensibly on assignment 2
  - Ask for help if you need it.
- Do keep making lecture summaries, and keep up with tutorial and laboratory work
  - Studying for the exam will be so much easier
  - Information about the exam in the next lectures
- Keep **asking** questions
  - If you get stuck … don’t spend hours staring at the screen, get help, try different strategies, don’t forget