CSCI 1106
Lecture 19

Randomness, Arrays and Button

Announcements

- Today's Topics
  - Using chance (randomness) in games
  - Other math
  - Arrays
  - Buttons

The Need for Randomness

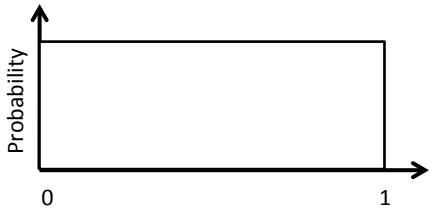
- Most games have some degree of randomness
  - Dice
  - Cards
  - Appearance of various objects
    - Projectiles
    - Power-ups
    - Purple people eaters
- Idea: Randomness (or chance) introduces a degree of "fun"
  - No two games are the same
  - Players don't know what to expect
  - The game appears more intelligent
- Question: How do we use randomness in Flash?

Use the (Random) Source … Luke

- Idea: Most systems have a pseudorandom source of values
  - The source is an infinite sequence of values
  - The values look random
  - Are sufficiently random for our purposes
- Each system is a little different, but all work similarly
  - Each system provides a `Random()` function
  - The function returns a value chosen randomly from a fixed range

Random() in Flash

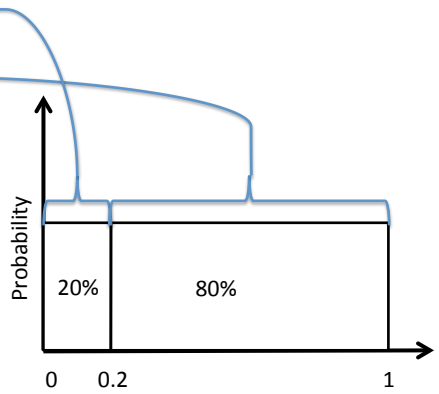
- Flash has a `Math.random()` function
- Returns a value in the range 0.0 ≤ n < 1.0
- Value is selected at random from a *uniform distribution*
- What does a uniform distribution mean?



A Random Code Example

```
if (Math.random() < 0.2) {
   Execute 20% of the time
} else {
   Execute 80% of the time
}
```

- If you wanted to implement a coin toss, how would you do it?

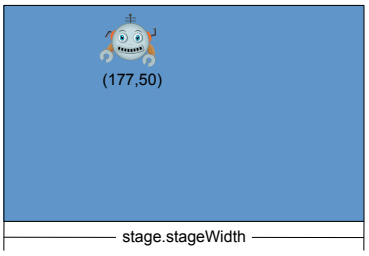


Another Random Example

- How do we place an object at a random horizontal position on the stage?

  y = 50;

  x = ???




```
x = Math.round(stage.stageWidth * Math.random());
```

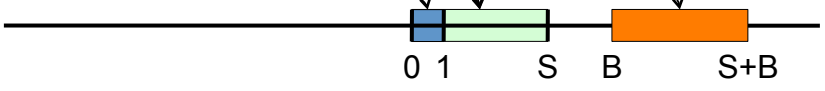
177 — 500 — 176.5 — 0.353

Round up to nearest integer. Not an Integer!

General Procedure

- Generate
- Scale
- Round
- Bias

```
r = Math.random();      0 ≤ r < 1
r = r * S;              0 ≤ r < S
r = Math.round(r);      r = integer
r = r + B;              B ≤ r < S+B
```



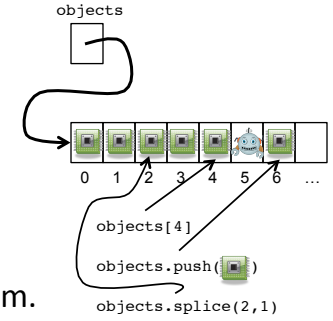
Other Useful *Math* Functions

- Rounding: ceil(), floor(), round()
- Trigonometric: sin(), cos(), tan(), asin(), acos(), atan(), atan2()
- Exponentiation: pow(), exp(), log()
- Other: abs(), max(), min(), sqrt()

- Now that we have randomness, let's use it!

An Array is

- A contiguous sequence of elements.
- Declared using the syntax:
  `var name:Array;`
- Instantiated using the syntax:
  name = new Array();
- Manipulated using:
  `name[i]` to access the ith item.
  `name[i] = …;` to set the ith item.
  `name.push(item);` to append items
  `name.splice(i, count);` to remove items



Looping over an Array


```
// assume p is the projectile
for(i = 0; i < objects.length; i++) {
  if( <collision test between p and objects[i]> ) {
    explode(p);
    remove(p);
    adjustState( objects[i] );
  }
}
```

Don't Push the **Big** Red Button…

- Buttons are screen objects that identify an action and how to perform it
- Buttons identify an area for a user to click on
- Buttons generate an event that the application can respond to by running a listener

Button State

- A button has three (3) states
  - **Up** is the normal state of the button
  - **Over** is when the mouse is hovering on the button
  - **Down** is when the button is pressed
- Idea: For each of the three states the button can have a different look
- Idea: When the button changes state, it generates an event

Rolling Your Own Buttons

- Create a *MovieClip* object to represent the button
- Place the object on the stage
  - The object represents the button's **Up** state
- In a `NEXT_FRAME` event listener
  - If the mouse is over the object (**Over** state)
    - Change the appearance of the object
- In a `MOUSE_DOWN` event listener
  - If the mouse is over the object (**Down** state)
    - Change the appearance of the object
    - Perform action associated with the button
- Is there an easier way?

The Easy Button

- Use the provided library of buttons:
  - Window -> Common Libraries -> Buttons
  - List of the available buttons
    - Any of these can be dragged and dropped into our .fla file
    - E.G., The red button from Classic Buttons / Push Buttons
- Hint: The little play button above its image in the library allows us to see what the button will look like when it is pressed
- Once added, the button appears in the Library
- Use instances of it, like any other object

The Creative Button

- Create a new symbol
  - Insert → New Symbol...
- Choose Button on the form
  - Be sure to export it for ActionScript
- The timeline panel for the button has 4 frames:
  - **Up:** how the button looks normally
  - **Over:** how the button looks when the mouse is over it
  - **Down:** how the button looks when pressed
  - **Hit:** the button area that responds to the mouse
- All you need to do is draw each of these!
- Note: Buttons generate events

Button Events

- `MouseEvent.CLICK` occurs when the button is clicked
- `MouseEvent.MOUSE_OVER` occurs when the mouse hovers over the button
- Other events are documented online
  - Search for *MouseEvent*
- *Note Buttons can be enabled/disabled*
  - To disable button B: `B.enabled = false;`
  - To enable button B: `B.enabled = true;`