CSCI 1106
Lecture 20

More controls and projectiles

Announcements

- Quiz this Wednesday (Friday is holiday)
- Today's Topics
  - Controls
  - Projectiles

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

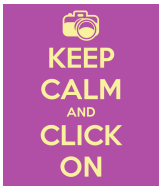
- `MouseEvent.CLICK` occurs when the button is clicked
- `MouseEvent.MOUSE_OVER` occurs when the mouse hovers over the button
- Other events are documented online
  - Search for *MouseEvent*
- *Note Buttons can be enabled/disabled*
  - To disable button B: `B.enabled = false;`
  - To enable button B: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

Other Controls

- Idea: Other standard controls are available
  - Check Box
  - Combo Box
  - List Box
  - Text Area
  - Slider
  - Radio buttons and more...
- Use Google to find the documentation
  - http://help.adobe.com/en_US/ActionScript/3.0_UsingComponentsAS3/

General Principles of Controls

- To use a control, select it from the standard library and add it to your library
- Place an instance of the control where you wish to use it
- Controls generate events when user interacts with them
  - Clicks
  - Edits
  - Selects
  - Scrolls
  - Slides
- To change the state of a control, modify one of its properties (variables)
  - See online documentation for list of properties

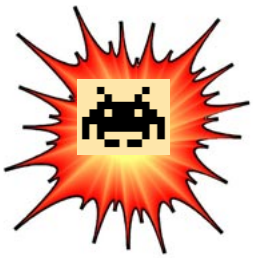


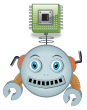
Projectiles

- One of the most common interaction mechanisms in games are projectiles
  - Bullets, lasers, asteroids, ships, boomerangs, etc
- Both the players (good guys) and the game opponents (bad guys) may use projectiles that are launched at the other side
- How do we implement projectiles?

What is a Projectile?

- Appears on the stage when the player/opponent does someth
- Appears initially at the player/ opponent's location
- Moves away from the player/ opponent in a set direction
- Disappears when it hits something
- Causes opponent/player to react in some way





The Projectile Life-Cycle

- Design (during game development)

- Initiation
- Creation
- Motion
- Collision
- Elimination

Projectile Design

- Design projectiles to support the game's unifying theme
- Use the Flash tools to draw projectile objects
- Add the projectile objects to the library
  - Similar to paddle, bricks, ball, etc
- Instantiate the projectiles on the stage when projectiles are needed

Projectile Initiation

- Idea: A projectile is initiated as a result of an event
- Player events:
  - Mouse click or key press — MOUSE_CLICKED or KEY_DOWN
  - Collision with another object — ENTER_FRAME
- Game (opponent) events:
  - Random or regular time intervals — ENTER_FRAME or TIMER
  - Collision of objects within the game — ENTER_FRAME
  - Start of game or level (e.g., the ball in BrickBreaker)
- Obs: All events are handled by event listeners
- Cor: Projectiles are initiated by event listeners

Frequency of Projectiles

Player Options

- Unlimited load and speed
  - As fast as possible
- Limited load
  - As fast as possible for a fixed number of projectiles
  - Require a recharge period to continue firing
- Limited speed
  - Allow player to fire one projectile per time period
  - Many players find this annoying
- Limited load and speed

Opponent (Game) Options

- Regular frequency
  - Create new projectiles on a regular basis
  - Not too fast or too slow
- Random frequency
  - Randomly decide in each time interval
  - Total number of projectiles per unit time should be limited
- Frequency increases as levels increase

Projectile Creation

- Idea: Projectiles are created by an event listener
- To create a projectile, the listener
  - Instantiates the projectile
  - Sets the projectile's position
  - Sets the projectile's velocity
  - Adds an `ENTER_FRAME` event listener
    `p.addEventListener(ENTER_FRAME, moveProj);`
  - Adds the projectile to the stage
    `addChild(p);`

```
p = new Projectile();
p.x = …;
p.y = …;
p.vx = …;
p.vy = …;
```

Projectile Position and Velocity

Player's Projectiles

- Position
  - In front of the player's avatar
- Direction
  - Same as the player's avatar
- Speed
  - Depends on game itself
    - Cannon ball vs laser beam

Opponent's (Game) Projectiles

- Position
  - Front of the opponent's avatar or
  - Random position from edge of stage
- Direction
  - Away from the opponent
  - Towards the player's avatar
  - Parallel to the stage
- Speed
  - Sufficient to give the player a challenge

Projectile Movement

- Idea: Projectiles move just like all other objects
  - Add velocity to position on each `NEXT_FRAME`
- Idea: `NEXT_FRAME` listener must also
  - Check for collisions with other objects
  - Check if projectile leaves the stage
- In either of these cases, the projectile is removed from the stage

Projectile Collisions

- Idea: Purpose of projectiles is to collide!
- Idea: On each NEXT_FRAME event
  - Check if projectile has collided with
    - Avatar (player or enemy)
    - Other game objects (terrain, walls, bricks, etc)
  - How?
    - Keep an **array** of all game objects
    - Cycle through array testing collisions with each of the objects
  - If collision occurs
    - Create some special effects (optional)
    - Adjust state of hit object (health, etc)
    - Remove projectile from stage

Projectiles Moving Off-Stage

- Idea: Projectiles moving off the stage must also be removed
- Idea: On each NEXT_FRAME event
  - Check if projectile has moved off-stage
  - If projectile is off-stage, remove from stage

Projectile Elimination

- Idea: Once a projectile moves off-stage or has collided, remove it!
- You will have run-time errors if you do not!
- To remove a projectile
  - Remove listener

```
p.removeEventListener(ENTER_FRAME,moveProj);
```

  - Remove from stage

```
removeChild(p);
```

Fire away!