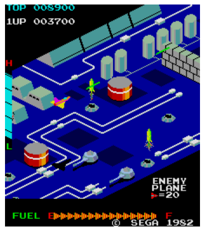
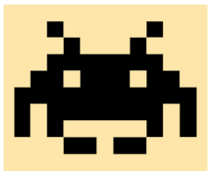
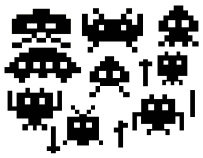
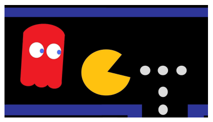


CSCI 1106
Lecture 21



Buttons and other Controls





Announcements

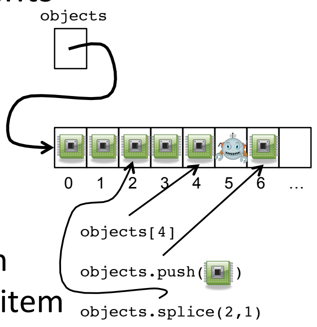
- Today's Topics
  - Projectiles
    - Collisions
    - Elimination
  - Buttons
  - Other controls

Projectile Collisions

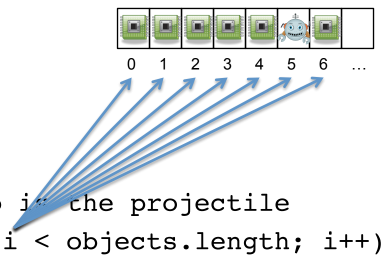
- Idea: Purpose of projectiles is to collide!
- Idea: On each `NEXT_FRAME` event
  - Check if projectile has collided with
    - Avatar (player or enemy)
    - Other game objects (terrain, walls, bricks, etc)
  - How?
    - Keep an array of all game objects
    - Cycle through array testing collisions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

An Array is

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

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);
    ```

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