CISC 110 Week 4

Functions, Methods, and Events
Today

- Methods
- Programming elements
- Events in ActionScript
  - Mouse Events
  - Programming Animation with Frame Events
- MovieClips
- Event Driven Programming

Reminder: your assignment is due during lecture next week instead of your lab
Programming Elements

Formally, a programming language can be comprised of the following elements, some of which we have already seen:

- **Statements**
  - Variable assignment (=)
  - Branching (if-else)
  - Loops (while, for)

- **Operations**
  - Built-in methods
  - Programmer defined functions

- **Primitive Data Types**
  - int
  - uint
  - char

- **Data Structures**
  - Strings
  - Classes
  - Array
Reference Variables

- A Reference variable is a variable that refers to the location where values of other variables are stored.
- For example, an object is a reference variable. It refers to the location where the values of the object’s properties are stored.

Examples of Single Variables:
- age: 4
- height: 38.5
- cute: true

An object instance:
- age: 50
- height: 75
- cute: true

ship1:
- height: 20
Assignment to Variables

- When we set/change an object’s properties, we are assigning values to variables:
- For example:

  ```
  age = age + 1; // Assigning new value to age
  ship1.x = ship1.x + 5; // Properties are variables
  ```
Using Methods from Flash Library

- We’ve seen two kinds of methods so far:
  - **Class methods** are called on a class
    
    `<class name>..<method name>(<parameters>)`

    **Example:**
    
    ```javascript
    trace( Math.round( x ) );
    ```
  - **Instance methods** are called on an instance of a class

    `<object name>..<method name>(<parameters>)`

    **Example:**
    
    ```javascript
    ship1.gotoAndPlay(15 );
    ```
Class Constants

- Constants are used in many classes to give meaningful names to values
  - To use a class constant, specify:
    
    `<class name>..<constant>`

- For example, the *Keyboard* class defines constants for commonly used Key Codes

**Example:**

```
trace( Keyboard.LEFT );  // 37, the Key Code
  // for the left arrow key
```
Events in ActionScript

Event Class: Base class for the creation of event objects
MouseEvent: Generated by a mouse or trackball
KeyboardEvent: Generated by user pressing a key
TextEvent: Generated when user types text in a text field
Buttons

- Allow user interactivity
- Two Steps to Create Buttons:
  1. Create the *appearance* of the button
     - What it looks like when the mouse rolls over or when its clicked
     - Define its active area that responds to a mouse click or other mouse event
       - Done easiest with menu and panel options
  2. Define what *action* it performs in response to a mouse click or other mouse event
     - With ActionScript
Defining Actions for Buttons

Two Steps to Define a Button’s Action

1. Define an *Event-handler Function* that says what to do when a mouse click or other mouse event occurs

2. Specify which event-handler function to execute when a particular button is clicked with the `addEventListener` method. This is called *Registering a Listener*
Event-Handler Functions

Example event-handler function that displays a message and plays a movie starting from frame 1 when the user presses a button:

```javascript
function hello ( evt: MouseEvent )
{
    OutputBox.text = "Hello tree!";
    gotoAndPlay( 2 );
}
```

It’s called when a button is clicked. Which button??
Registering a Listener

Bind an event-handler to the event of a user clicking the mouse on a button called HelloButton, so the hello function is called when that event occurs:

HelloButton.addEventListener(MouseEvent.CLICK, hello);

Object that will now listen for a mouse click  Event Type in MouseEvent class  Function that will be called when button is clicked

For simple scripts, place this line inside the constructor function so it will be active for the whole script
Registering a Listener on the Whole Stage

Bind an event-handler to the event of a user clicking the mouse on the stage, so the `hello` function is called when the user clicks anywhere on the stage:

```
stage.addEventListener( MouseEvent.CLICK, hello );
```

- `stage` will now listen for a mouse click.
- `MouseEvent.CLICK` is the event type in the `MouseEvent` class.
- `hello` is the function that will be called when the stage is clicked.
Removing Event Listeners

We should remove event listeners once we don’t need them, since they use extra time & space.

For instance, once an object’s alpha property has reached 0, it is invisible, so it is useless to continue subtracting from it.

```javascript
stage.removeEventListener( Event.ENTER_FRAME, fadeSquare );
```
MouseEvent Class

- Some types of events…

**CLICK:** “click” event
**DOUBLE_CLICK:** “doubleClick” event
**MOUSE_DOWN:** “mouseDown” event
**MOUSE_UP:** “mouseUp” event
**MOUSE_OVER:** “mouseOver” event
**MOUSE_WHEEL:** “mouseWheel” event
Event Type Constants

- Constants are defined in each class to specify the numeric values of event types

  - MouseEvent.CLICK
  - MouseEvent.MOUSE_OVER
  - MouseEvent.MOUSE_UP
  - MouseEvent.MOUSE_OVER
  - MouseEvent.MOUSE_WHEEL
Event Class

Properties

target: target object of the event
type: type of event dispatched

Event Types

COMPLETE: “complete” event object
ENTER_FRAME: “enterFrame” event object
Using Frame Loops to Program Animation

We can create a *frame loop* by listening for an enter frame event, which is fired as each frame is played, i.e., at the same frame rate as the movie.

// For simple scripts, put this line inside the constructor function
stage.addEventListener( Event.ENTER_FRAME, moveShip );

// Put this function definition after the constructor function
function moveShip( evt: Event )
{
    ship1.x = ship1.x + ship1.speed;
}
Side Note on MovieClips

- Each MovieClip symbol has its own timeline, so it can be used as a mini-movie that loops over and over when an instance of it is added to the stage.

- A MovieClip can also be built as a composite object composed of other MovieClips that can each be programmed separately, for instance a bird composed of a head, body, wings and feet. Each could change in position or shape.
  - This is called *nesting*
Event-Driven Programming

Simple Programming: Program runs

Event-Driven GUI Loop: Wait for Input Event

Program runs synchronously with user input; waits for input (key press, mouse), then does something
GUI Loop and Event Handlers

- When an event occurs, the GUI loop is interrupted and the appropriate event handler is called.
If a listener is registered for an event, the event dispatcher calls the specified event listener/handler and sends it an event object as its parameter with information about the event.
KeyBoard Event Class

Properties

**charCode:** character code of key pressed

**keyCode:** key code value of key pressed

**ctrlKey:** true if ctrl key is pressed, false if not

Event Types

**KEY_DOWN:** “keyDown” event object

**KEY_UP:** “keyUp” event object
Next Week

- Conditionals!
- … and Loops!