CSCI 1106
Lecture 13

Introduction to Game Architecture

Announcements

• Today’s Topics
  – Introduction to the Game Design Module
  – Components of a game
  – Introduction to event driven programming
  – Introduction to Flash and ActionScript

• Robotics module technical reports due on Thursday
The Game Design Module

Topics
- Structure of a game
- Game mechanics
  - Collision Detection
  - Player movement
  - Autonomous Game Elements
  - Randomness
  - Controls
- Playability and play testing

To Do List
- Five tutorials:
  - Implement a game
  - Learn about game design
- Game Design Project
  - Design your own game
  - Implement the game
  - Write a technical manual
  - Write a user manual

Components of a Game

- Stage: Displays (renders) the game
- Graphical Game Objects
  - Objects that interact on the stage
  - Represent various artifacts in the game
    - Characters
    - Projectiles
    - Goals
    - Dangers
- Game Code
  - Governs interactions between player and game
  - Implements the rules of the game
  - Governs interaction between game objects
  - Contains event handlers that respond to events in the game
Flash Game Files

- The *stage* is located in a *.fla* file
- All graphical objects are located in the same *.fla* file
- Game code is located in *.as* files
- Once the game is completed, a “compiled” version is found in a *.swf* file

Flash Game Components

- Stage (*.fla* file)
- Game objects (*.fla* file)
  - Graphics
  - Movie clips
- Game Code (*.as* files)
  - ActionScript 3
  - One class per file
- How are these components linked?
Key Ideas

- Objects are designed and used on the stage
- Objects are controlled by code in .as files
- User actions on stage (actions on objects) invoke code in .as files
- Computer generated actions invoke code in .as files

Observation

- Observation: A game performs “some action” when “something” happens
- Examples:
  - Character moves *when* the mouse is moved
  - An object explodes *when* it is hit by a laser
  - The screen is updated *after* 1/60th of a second
  - The stage is populated *when* the game starts up
- The “something” are called *events*
The Event-Driven Paradigm

- Idea: Game code simply responds to events
- Possible events:
  - External events
    - Player movement (mouse, keyboard, kinect, etc)
  - Internal events
    - Start of game
    - New Frame: stage update (every 1/60th of a second)
    - Timer expired
    - Objects appear/disappear
- Each event is handled by an *event handler*
- The game code simply consists of event handlers that handle all aspects (behaviours) of the game!

The Main Loop

- Idea: The main loop is implemented for you
- Main Loop:
  - Event (action) occurs
  - Handle (respond to) event
  - Update (modify) object(s)
- All you need to do is write the event handlers!
The Movie Metaphor

• Key Idea: Flash updates the stage every 1/60th of a second
• The update consists of:
  – ENTER_FRAME event
  – Redraw all objects on the stage
  – LEAVE_FRAME event
• Key Idea: Our game handles the ENTER_FRAME event
  – Updates the positions and properties of all objects
  – Adds/removes objects as needed
  – Updates graphics as needed
• Idea: The Flash system redraws all objects that are on stage
  – All we need to do is update the objects!

Events in Flash

• ADDED_TO_STAGE: object appears
• REMOVED_FROM_STAGE: object disappears
• ENTER_FRAME: objects are about to be redrawn
  – Occurs every 1/60th of a second
• LEAVE_FRAME: objects have been drawn
• ON_KEY: keyboard key pressed
• ON_MOUSE: mouse movement or click
• ON_TIMER: timer went off
ActionScript

- **Package**
- **Imports**
- **Class**
  - Associates code with an object
  - Has a name
  - Main is where the code starts running.
  - Contains variables and functions
- **Functions**
  - Each function has a specific purpose
  - Contains code
  - Statements, end in ;
  - control structures (if, for, etc)
  - Defined public function name(params) { ... }
- Code is delimited by { and }

Variables

- Each variable has a type
  - Such as uint, String, etc
- Variables store data of that type
- Declared as
  - var name:type;
  or
  - var name:type = value;
- Declared in a class or function
- Only visible (accessible) in block where they are declared.
- Assigned using the = operator
  - e.g.,
  - Number = 42;
Event Listeners

- **Idea:** To handle an event, your program has to *listen* for it.

- **How?**
  - Create functions that will *handle the event*
    - These are called *listeners*
  - When program starts up, register (add) listeners

- When an event occurs, Flash will call the listener