CSCI 1106
Lecture 3

Sprites
Announcements

• Today’s Topics
  – Sprites
  – Costumes
  – Stage
  – Properties
  – Variables
  – Scripts
  – Cloning
  – Communication among Sprites
Recall: Sprites

• A sprite is a graphical object that is placed on the stage
• A sprite has associated with it
  – costumes
  – properties
  – variables
  – scripts
• A sprite represents game artifacts
  – Characters
  – Obstacles
  – Projectiles
  – Etc
Naming Sprites

• Key Idea: Each sprite has a name, e.g., Ball
  – The name should describe what the sprite is
  – Different sprites may have the same name
  – The name identifies the type of sprite, rather than a specific sprite
    • e.g., You can have several different car shaped sprites, all of them call Car
  – Most sprites will be unique

• Key Idea: Sprites are referred to by their name
  – There is no other way to refer to a sprite
Costumes

• Idea: A sprite can change its look by putting on a different costume
• A costume is a graphical representation of the sprite
• Each sprite has at least one costume
• Each costume has a name
• A sprite can change its look by switching costumes
• Most sprites have only one costume
The Stage

- Idea: The *Stage* is a special sprite on which all other sprites are displayed.
- The stage does have *backdrops* rather than costumes, but they serve the same purpose.
- All sprites will always be in front of the stage.
- Like other sprites, the stage has
  - properties, sounds, and scripts associated with it.
Properties

• Key Idea: All sprites have intrinsic *properties *
• A *property* is a characteristic of the sprite, e.g.,
  – *position* on the stage  
  – *direction* of sprite (in degrees)  
  – *costume* currently worn  
  – *size* of the sprite  
  – *visibility* (showing or hidden)  
  – also: *colour*, *depth*, etc...

• Key Idea: Sprites are manipulated my modifying their properties

• But ... what if want to associate additional information with the sprite?
Extrinsic Properties

• Problem: We may wish to associate additional (*extrinsic*) information with a sprite, e.g.,
  – Lives or health of a character
  – Difficulty of destroying an obstacle
  – The amount of power in a power-up

• Observation:
  – Properties are typically represented as numbers, e.g.,
    • x position, y position, direction, etc...
  – Most extrinsic information is also represented as numbers, e.g.,
    • Health, Lives, Score, ...

• Solution: Use variables to associate extrinsic properties with a sprite
Variables

• Idea: A variable is a location in the program or a sprite that stores a value
• A variable has a name by which it is referenced
• A variable can be
  – accessed (read) to retrieve the value it stores
  – mutated (written) to modify the value it stores
• Idea: The scripts associated with a sprite can access and mutate the sprite's variables
Summary So Far

Sprite Name: Invader

Properties
- 10
- 42
- 90
- 100
- x position
- y position
- direction
- size

Variables
- Score: 123
- Level: 4
- speed: 5
- Lives: 2

Costume 1

Costume 2
A Sprite's Script

- Is a sequence of blocks
- Starts on a \textit{when} block
- Contains
  - \textit{motion} blocks
  - \textit{control} blocks
  - \textit{sensing} blocks
  - \textit{operator} blocks
  - \textit{data} blocks
  - \textit{event} blocks
- Is executed when an event occurs
A Script for the Stage Sprite

• Idea: Your game will need a FRAME event
  – 30 times per second
  – Allows sprites to update themselves
  – Generated by a script associated with the Stage
  – Generated when game is running

• Use the following script
  – when game starts
  – repeat forever
    • wait 1/30th of a second
    • generate FRAME event
Manufacturing Sprites
Cloning Sprites

• Idea: We can make multiple copies of a sprite by cloning it.

• When a sprite is cloned, everything is copied
e.g., properties, variables, costumes, scripts, etc

• Key Idea: Manipulation of the clone or the original does not affect the other
e.g., changing the clone's position will not move the original

• Both the clone and the original have the same name

• Two differences between clones and originals
  – clones are notified when they are created
  – clones can be destroyed
Cloning Example

Sprite Name: Invader

Properties

- 10
- 42
- 90
- 100

Variables

- Score: 123
- Level: 4
- Speed: 5
- Lives: 2

Sprite Name: Invader

Properties

- 10
- 42
- 90
- 100

Variables

- Score: 123
- Level: 4
- Speed: 5
- Lives: 2

(create clone of myself)

when I start as a clone
Communication Between Sprites

- Key Idea: Sprites communicate by broadcasting messages (events)
  - A broadcast means every sprite receives the message
    e.g., Stage broadcasts FRAME 30 times per second
  - A sprite can respond to a specific message (event) by having a script that receives it

- Messages cannot be directed at a specific sprite unless only that sprite has a script to receive that message
Broadcast Example