Components of a Game

- **Stage**: Displays (renders) the game
- **Sprites**:
  - Graphical objects that interact on the stage
  - Represent various artifacts in the game
    - Characters
    - Projectiles
    - Power-ups, obstacles, etc.
- **Game Code**:
  - Governs interactions between sprites
  - Governs interactions between player and sprites
  - Implements the rules of the game
  - Contains *event handlers* that respond to events in the game
  - Updates the sprites on the stage
Scratch in a Nutshell

- A Scratch program consists of
  - A *stage* on which sprites are displayed
  - One or more *sprites*
    - graphical objects that interact on the stage
  - Zero or more *scripts associated with the sprites*
- A *sprite* has
  - Properties such as position, direction, size, etc.
  - Zero or more *variables* used to store values
  - One or more *costumes*, describing how it looks
  - Zero or more *sounds* that it can emit
  - Zero or more *scripts* that respond to events
- A *script* responds to an event
  - These scripts are also called event handlers
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
    • Frame (stage update every 1/30th of a second)
    • Timer expired
    • Sprites cloned
• Each event is handled by an event handler
• The game code simply consists of event handlers that handle all aspects (behaviours) of the game!
CSCI 1106
Lecture 3
Sprites
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 must have different names
  – 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 has *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 hiding)
  – also: *colour, depth, etc...*
• Key Idea: Sprites are manipulated by 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 method in the program to store 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
  – Local and Global variables
Summary So Far

Sprite Name: Invader

Properties
- x position
- y position
- direction
- size

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

Costume1

Costume2
A Scratch Script

- Is a sequence of blocks
- Starts on a *when* block
- Contains
  - *motion* blocks
  - *control* blocks
  - *sensing* blocks
  - *operator* blocks
  - *data* blocks
  - *looks* blocks
  - *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 x position
- 42 y position
- 90 direction
- 100 size

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

Sprite Name: Invader

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

Variables
- Score: 123
- Level: 4
- Speed: 5
- Lives: 2
Communication among Sprites

- Key Idea: Sprites communicate by broadcasting messages (events)
  - If you want to tell other sprites when to do something.
  - 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

• Broadcast
  – Sends a message to all the sprites (and the background) to tell them to do something

• Broadcast and wait
  – Sends a message to all sprites to tell them to do something, and wait until they all finish before continuing.

• What do you want sprites to do when they receive the message?
  – When I receive
Broadcast Example
2nd Tutorial

• Game State and Progress
  – Variables
  – Conditionals
  – Cloning
  – Communication among sprites
  – Keyboard Inputs