Scratch/BYOB

James MacGlashan
Scratch and BYOB

- **Scratch (MIT)** is a visual programming language
  - Operates in conjunction with “sprites” on a canvas
  - Language is not typed on a keyboard but has code components visually dragged onto the screen and connected together to make code sequences
  - Clicking on any code sequence executes it

- **BYOB: Bring Your Own Blocks (Berkeley)**
  - Extension to Scratch adding functions, recursion objects, more complex lists, etc.
  - We will use BYOB, but its interface is largely the same as Scratch, but with more stuff
  - Later will become Snap! but that is still in alpha
Sprite-based

- Scratch has a canvas on which “sprites” are displayed
  - 2D images
  - Student can import images or paint them within Scratch

- A sprite has its own set of code it follows
  - similar to object-oriented programming

- Sprites can respond to interactions with other sprites (message passing, intersections on canvas, etc.) or user input (keyboard keys, mouse, text, etc.)

- Sounds/music can also accompany events
Connecting students

- Students can register on MIT’s scratch website to upload their projects and view others
- Projects are playable in the web browser
  - uses Java plugin
- Code from existing projects may also be downloaded
- Example: Moo
The BYOB Interface
The BYOB Interface

Canvas: displays sprites
The BYOB Interface

List of sprites
The BYOB Interface

Editing Pane for the current sprite
The BYOB Interface

Various types of code components
The BYOB Interface

Code components of selected type
Motion Code

- Motion code components affect a sprite’s position and rotation on the canvas
- The canvas is a 640x480 pixel space with (0,0) at the center
- Components can be dragged onto the script pane and are executed by clicking on them
Control Code

• Control structures such as conditionals and loops
• Also contains control for key/mouse events
• Special case is Green Flag
  - effectively the “main” function of Scratch
  - is called whenever user clicks on green flag above the canvas
Operators Code

• Contains binary and unary operators
  - arithmetic
  - boolean comparisons
  - boolean logic
  - math functions (sqrt, sin, etc.)

• Other options too
  - random number generators
  - text manipulators
  - scripts/blocks (BYOB)
Sensing

- List of common variables and tests in the Scratch environment
  - key presses
  - position of sprites
  - whether sprites are touching
  - distance between sprites
  - text prompts
Variables

- Manipulators for variables
- Manipulators for lists
- Allows creation of variables and lists
  - variable may be local to a sprite or global
- Allows creation of blocks (BYOB)
Looks

- Lets sprites “speak”
- Changes colors of sprites
- Adds graphics effects to sprites
- Change size of sprites
- Set Costume
  - costumes are sets of pictures associated with a sprite between which it can alternate
  - costumes for a sprite are defined in the “Costumes” tab of the editing pane
Sound

- Code for playing sounds and music
- Volume control
- Temp Control
- Sound for a specific sprite can be recorded or imported in the “Sounds” tab of the editing pane
Pen

- Code for allowing a sprite to draw wherever it moves
- Change pen color, size, and shade
- Sprite can also “stamp” and image of itself on the canvas
- Also contains code for clearing the screen of drawings
Demonstration

- Drawing patterns
BYOB: ‘for’ and Templates

- BYOB adds a bunch of cool stuff for making your own block structures
  - similar to defining functions
- Also contains many templates
- Example: creating simple for loop block
- Example templates
Exercise 1: Aquarium

• Make an aquarium with fish that swim around randomly
• Make one that follows another
• Make a school of fish that stick together
Exercise 2: Bouncing Ball

• Based on physics equations:
  \[ y = y_0 + v_0 t + \frac{1}{2} a t^2 \]
  \[ v = v_0 + a t \]

  where \( y \) is position, \( t \) is time, and \( a \) is acceleration

• Will require variables for time increment, velocity, and acceleration
Exercise 3: Moon Lander

• Based on similar physics equations as bouncing ball except is vector with user controllable thrust

• Acceleration from thrust in y and x direction:
  \[ a_x = p \cos(\theta) \]
  \[ a_y = p \sin(\theta) \]

• Win loss conditions on speed and location of collision

Shell Program:
http://maple.cs.umbc.edu/~jamesmacglashan/cs4hs/moonLanderShell.ypr
Free Exercise

• Do something new on your own,
• Work on an existing exercise, or
• More Finch, if you’d like too