**Scratch Lesson Plans (version 1.0)**

**Terence Soule, University of Idaho,** [**tsoule@cs.uidaho.edu**](mailto:tsoule@cs.uidaho.edu)

**Introduction:** These are a series of lessons design for a short, introductory course on programming using the Scratch programming environment developed at MIT. The lessons were originally designed for a one week code camp for middle school students. The participants worked on Scratch roughly 3 hours a day. Thus, the total time to work through all of the lessons is roughly 15 hours.

The lessons are designed to be presented consecutively. Over the course of the lessons the students create at least two complete scripts. The first is an animated scene with multiple characters that move and interact, and includes dialog, sound effects, changing scenes, and potentially visual effects. The second is a simple game based on one of three basic templates: land a vehicle, catch falling objects, or navigate a maze or ‘minefield’.

**General notes on the lessons:**

Each lesson is meant to take 30 minutes of which 10-20 minutes is instruction and the remaining time is for working on the tasks. In addition, students can be encouraged to follow along with the instructor in Scratch during the instruction time. Students can either load and remix the pre-written scripts from the Dign IT portfolio or the instructor can walk them through the creation of a new script similar to the one’s in the portfolio. For the simpler scripts it is better to have the student write their own, using the portfolio scripts as a reference as necessary. For the later, more complex scripts, it may take too long to write them from scratch. Instead have the students load the script from the portfolio; the instructor should then explain the important parts of the script and let the students remix it.

**Scratch Accounts:** Go to scratch.mit.edu and follow the instructions under *join Scratch*. Setting up an account is free. Users must enter their age and an email address. Users under 16 are asked for a guardian’s email address, so in setting a course it is useful to get parent emails for the students to use. Alternatively they could all enter an email address for the institution hosting the lessons.

**Sample Projects:** All of the sample projects can be found by searching for “Dign IT” within scratch. The sample projects are labeled *Script N – Name*, where N is the script number and name is the associated name.

**Timer:** it is very helpful to use an electronic timer to time the students’ work time. The timer should be visible to the students and have an audible alarm. A timer widget works well in Windows. Announce to the students how much time they have and set the timer. With a timer students can see how much time they have left and the audible alarm makes it easier to get everyone’s attention to start the next lesson.

**About Scratch:** Scratch (scratch.mit.edu) was developed at MIT Media Labs as a tool for early introduction of programming. It uses a graphical, drag-and-drop programming interface, with easy to understand graphical icons representing basic program elements. Scratch is free and runs as a web application or as a downloadable, standalone application. It has a large on-line community, with a strong emphasis on sharing and “remixing” projects. This on-line community emphasizes programming as a cooperative, social activity, with lots of opportunities to share, discuss, and modify programs. It is also an excellent environment for exploring programming by encouraging the sharing ideas and methods. Programs can be stored indefinitely, and without cost, on the Scratch web site, allowing participants to continue to develop their projects after the course has ended. Programs can be shared via the Scratch website or embedded in regular webpages, making it easy to present projects to family and friends. Scratch has simple interfaces for drawing, and for incorporating sound and images, allowing users to easily create interesting, multi-media programs.

**Supplies:** Each student or student team should have a computer with access to the internet. Optional supplies includes:

A microphone for recording sound clips.

A digital camera for taking pictures to imbed in Scratch scripts.

A drawing program for creating images for Scratch scripts.

Paper and pens or markers for students who would like to draw pictures to embed in scratch scripts. (This also requires a digital camera or scanner to digitize the pictures.)

A drawing tablet.

**Teams:** The lessons are best done with teams of 2-3 students, although students can work alone or in larger groups. For larger groups the team members should be given more explicit roles as part of each lesson, so each member has a specific job to do. Unless there are serious problems, the groups should remain the same between lessons as the lessons are designed to build upon each other leading to the two final projects. Keeping the groups consistent allows the members to continue to work on the same project.

**Lesson 1: Introduction to Scratch**

**Topics:** Scratch website, accounts, running scripts, *see inside* option, editing scripts.

**Script:** Script 1

**Notes:** Go to the Scratch website (scratch.mit.edu) and have the students set up an account. Younger students will need to enter a parent or guardian’s email. Walk the students through the Scratch site.

In Scratch, show the students:

* Other Scratch projects
* How to search for the Dign IT (or other course) portfolio
* User’s folders
* Sprites: scripts, costumes, and sounds tabs.
* Backgrounds
* Show code menus, drag and drop, play and stop buttons.

**Tasks:** Create a team account. Use an anonymous name, explain how sharing will work.

**Lesson 2: First Script**

**Topics:** How to use Scratch

**Script:** Script 2

**Notes:** Walk the students through the creation of a basic script or have them load Script 2 and walk them through it.

Code topics:

* Sprites and sprite library
* Start flag and stop sign
* Script menu for Sprites
  + Control menu: on flag, wait (for timing)
  + Movement menu: goto, glide,
  + Looks menu: say, think
* Movement and coordinates (point out how to use the mouse to find coordinates in the stage, it may also be useful to show the coordinates background)
* Timing
* Backgrounds and background library

**Tasks:** Add additional lines (say), movement, and a background to their script.

**Lesson 3: Timing versus Broadcasting**

**Topics:** Coordinating sprites

**Script:** Script 2

**Notes:** Discuss two ways to coordinate sprites. One, by using the wait operation to have the sprites alternatively wait for the other sprite to finish. Two, by using broadcast and receive to have sprites wait to receive a message before continuing. To help illustrate this idea it’s useful to sketch two sprites and their alternating parts in a conversation, use arrows to show when the ‘action’ switches from one sprite to the other; discuss how waiting or broadcast/receive can be used to govern the switching.

With the broadcast/receive approach each receive broadcast statement should be at the head of its own mini-script. This can be used to introduce the idea of parallelism/concurrency – each of the scripts is running in parallel, they are all waiting for their own broadcast message to begin.

**Tasks:** Add more dialog exchange to the scene using waiting or broadcast/receive.

**Lesson 4: Animation and Loops**

**Topics:** Animating sprites, loops

**Script:** Script 3

**Notes:** Have the students load script 3 and walk them through it. Show the different costumes on the costumes menu and point out how they are slight variations of each other. Show the animation loop.

Discuss how animation works in general: frames per second and how the brain interprets the images.

Discuss parallel scripts: having an animation script and a separate movement script.

Show how the mouth animation worked in script 1.

Show sprites in the library with multiple costumes for animation.

If there is time load an animated sprite and walk the students through writing a basic animation script.

**Tasks:** Play with the animation in Script 3. Try different wait durations and movement speeds.

**Lesson 5: Effects**

**Topics:** Effects

**Script:** Script 2

**Notes:** Load script 2 and discuss the final effects. Start a new script and walk through the different effect options. Explain how a loop is used to iterate through the effect values. Use this discussion to reinforce the previous lesson on loops.

This is a good lesson to introduce the idea of a setting up a script and, more generally, a program. Explain the need to set the initial conditions for the script/program. In this case script has to reset the effects at the beginning of the script.

**Tasks:** Add some effects to their on-going script, from lesson 2.

**Lesson 6: Drawing sprites**

**Topics: Costumes tab, the drawing programs**

**Script:** New script

**Notes:** Start a new script. Load a sprite from the library, copy it in the costumes tab. Modify the copy to create an animation. Add a loop to show the animation. This doesn’t have to be a great animation – the goal is to show how to draw and/or modify a sprite.

Create a new sprite from scratch using the paint program to draw the sprite. Explain the difference between bitmap and vector drawing. Show the different tools that are available. Make sure to point out the zoom buttons, undue buttons, and fill option. Point out the + symbol marking the middle of a sprite. It may be helpful to explain how, if a sprite is drawn off center, rotating it can also cause it to seem to move – because it rotates around the center of the image (the +) and not the center of the draw portion of the sprite.

**Tasks:** Draw a sprite, copy it, modify the copy to create a simple animation.

**Lesson 6: Loading Images**

**Topics: Loading images, editing images,**

**Script:** New script

**Notes:** This lesson should start with a new script. Load a background from an image file. It’s helpful to use photo editing software to show how an image can be edited, then included as a background.

Create a new sprite by loading an image. Show how erasing part of the image makes it transparent. Show how the paint option can be used to further modify the image. A good example is to load a picture with a face, erase everything except the face, then add a stick body. Time permitting duplicate the sprite and modify the stick body to create another animation.

**Tasks:** Incorporate an image into the scene, either as a sprite or a background.

**Planning a Scene**

**Topics:** Storyboarding

**Script:** None

**Notes:** Short instruction, most of the time of for the students to plan and start programming their scene.

Discuss storyboarding – one quick sketch per major scene. Choose main media forms: pre-programmed sprites and backgrounds, drawn, or photographed. Storyboards should also include notes on media.

**Tasks:** Create a storyboard.

**Programming Languages**

**Topics:** Non-graphical programming languages

**Script:** Script 2

**Notes:** Compare how Scratch’s graphical language icons compare to programming languages.

Use Processing or a similar language for contrast. Explain variables and loops.

**Tasks:** Add a sprite and an effect to script 2. Discuss team strengths and interests.

**Sound**

**Topics:** Sounds

**Script:** Script 2 - reuse

**Notes:** Load script 2, add sound.

Show sound options: pre-recorded, user recorded, loaded from file, or programmed.

**Tasks:** Begin programming scene.

**Add Media**

**Topics:** Media

**Script:** Can use any script

**Notes:** Short instruction, most of the time is programming or collecting/adding media.

Remind students of options for adding media.

**Tasks:** Collect and add media.

**HTML**

**Topics:** HTML

**Script:** No script, use Code Camp webpage

**Notes:** Brief intro to HTML

Discuss webpages in general – files stored on computers, displayed with a browser.

Show view source.

Discuss matched commands <> </>

Commands: paragraph, link, image.

**Tasks:** More programming

**Finish Scene**

**Topics:** -

**Script:** -

**Notes:** Students should be finishing their scene.

**Tasks:** Finish scene.