Computer Science Lab:

Scratching the Surface of Software Engineering

August 1, 2013

Prof. Paul Sivilotti
Dept. of Computer Science & Engineering
The Ohio State University
paolo@cse.ohio-state.edu

Introduction

Scratch

In this lab, we will be using *Scratch*, a graphical programming language in which you write programs by stacking instruction blocks one on top of the other. A Scratch project consists of a collection of sprites (2-dimensional images or characters in a game or story). Each sprite has behavior controlled by its program(s). Scratch lets you mix together pictures, sound, movement, and animation, to create multimedia expressions.

Scratch is free to use. It runs within any browser that supports Flash (eg Firefox, Internet Explorer, Safari, Chrome, *etc*.). See:

http://scratch.mit.edu

This site also contains many galleries of projects that other people have developed with Scratch. You can browse through to see examples of programs, games, and animations that others have created using this language. You can even download these programs to edit and modify for your own creations.

You can sign up for an account on this site (it is free), which allows you to post your own projects. But signing up for an account is not required to use Scratch.

Lab 1: Save the Turtle!

Background

A turtle has wandered out on the beach and must now return to the safety of the ocean before it becomes dehydrated and dies. The beach is a dangerous place: there are several perilous obstacles between the turtle and the ocean. Touching any of these obstacles is immediately fatal for the turtle. Each obstacle is surrounded by a green warning zone. Crossing into one of these warning zones reduces the turtle's life points, but is not immediately fatal.

Your mission is to write a program that safely guides the turtle back to the ocean. As an extra challenge, try to maximize your score.

Setup

Copy the Scratch program to your own computer:

1. Login to your machine using your assigned login and password.
2. Open a browser (Firefox or Internet Explorer) and go to http://scratch.mit.edu
3. Launch Scratch by selecting Create at the top of the page.
4. Select File, Upload from your computer, then find the file J:, src, stt, TheBeach.sb2
   - Click Save, and when asked to replace the current project, click OK
   - You should see the beach and obstacles in the main pane on the left.
5. Select File, Download to your computer, then find the directory J:, group?? (where ?? is *your* group number)
   - Give your project a name and click Save.
   - Remember to Save your project periodically.

Constraints

To make things more interesting, you are asked to respect the following rules:

- Solve the problem by *adding* blocks *only* to the turtle program that begins "**when green flag clicked**".
  - *Do not* modify any other sprites, scripts, costumes, variables, etc.
- Move the turtle forward (or backward) *only* with the motion block **"crawl [ ] steps"** (found under **More Blocks**)
  - *Do not* use motion blocks such as **move**, or **goto** or **set**
  - Turning motion blocks (i.e., **turn**, and **point**), however, *can* be used

These constraints are in place to prevent trivial solutions in which the turtle just magically warps over to the ocean. Our stranded turtle, like any ordinary turtle, can only crawl slowly around the beach.

Challenges

There are several different variations on the basic problem posed in this lab. These variations differ in what is known about where the turtle begins and where the turtle must arrive. Which variation is being solved is determined by the second block in the turtle script: **"start at [ ] with goal [ ]"**. Solve the challenges in the order given below, by setting the values in this block accordingly.

Specification A) Fixed Start / Reach the Ocean

start at [3] with goal [1]

Input: The initial position *and* direction of the turtle are both known.

Output: The turtle must reach (anywhere in) the ocean.

Specification B) Random-Facing Start / Reach the Ocean

start at [2] with goal [1]

Input: Only the initial position of the turtle is known, *but not the direction it is facing*. Every time you run the program, the turtle could start in a different direction, but your program must always work, and bring the turtle to its goal.

Output: The turtle must reach (anywhere in) the ocean.

Specification C) Fixed Start / Reach the Ship

start at [3] with goal [2]

Input: The initial position *and* direction of the turtle are both known.

Output: The turtle must reach *the ship floating in the ocean*.

Specification D) All-Random Start / Reach the Ocean

start at [1] with goal [1]

Input: *Neither the initial position nor direction of the turtle are known*. Every time you run the program, the turtle starts in a different direction and from a different position! Your program must always work regardless, guiding the turtle to it goal.

Output: The turtle must reach (anywhere in) the ocean.

Lab 2: Dragons and Butterflies

Background

Dragons roam the planet. They fly where ever they wish and breathe fire too. The world is inhabited by butterflies as well, fluttering in the breeze. What else is part of this world? Maybe witches, horses, or princesses? It is up to you.

In this lab you will create a complex virtual world with multiple characters that move and interact. In part I, each group works on a small part of larger program. In part II, you will assemble the individual sprites written by other groups to form your virtual world. You can then modify, extend, and customize the program however you wish. In part I you are asked to respect some very specific requirements, but in part II you are limited only by your imagination!

Part I: Creating a Sprite

You should start by working on your assigned sprite(s) described below. If you don't know your assigned sprite, ask us.

The end product of your efforts should be a *single* sprite implementing your assigned specification. In the course of developing your main sprite, however, you may find it useful to create other sprites too. These other sprites can be used, for example, to test your main sprite's behavior. When you have finished this part of the lab, let us know so we can collect your main (assigned) sprite.

The specifications given below are just basic requirements. Feel free to add whatever you wish by exploring all aspects of Scratch (e.g., sound, color, costumes). Anything is fine, so long as your sprite implementation satisfies its specification, of course!

Setup for Part I

Create a new project and save it on your computer.

1. Launch Scratch as in the previous lab, and create a new project (File, New)
2. Select File, Download to your computer, then find the directory J:, group?? (where ?? is *your* group number)
   - Give your project a name and click Save.
   - Remember to Save your project periodically.

The Scratch cards are useful for hints. For more help, feel free to ask us, or explore the help information available by clicking "Tips" at the top of the page.

When you have completed Part I, save your *sprite* to the J: drive.

1. Right click on the sprite, save to local file, then find the directory J:, group?? (where ?? is *your* group number)
   - Name your sprite something unique (not Sprite1) before exporting it.

Sprite Specifications

After completing your assigned sprite, feel free to work on other sprites listed below.

Sprite A) Flying Dragon

Design a dragon sprite that moves around the stage in response to the arrow keys (up, down, right left).

Your sprite should have exactly one costume.

Sprite B) Fire-Breathing Dragon

Add to Sprite A by making the flying dragon sprite breathe fire when the space bar is pressed. The dragon's fire should contain the color red. In particular, you should use *exactly* the same shade of red as the sprite "OSU-Scarlet" contained in J:/src/db. (There is an example of such a sprite, called dragon1-b, in the standard Scratch Fantasy library. You do not need to use this particular costume, but whatever you choose should have exactly the same red as OSU-Scarlet in the flame.)

Your sprite should have at least two costumes (i.e., one for when the dragon is breathing fire).

Sprite C) Fluttering Butterfly

Design a butterfly sprite that moves around the screen at random, bouncing off the edges. Animate the butterfly using multiple costumes so it appears to flutter its wings in flight.

Sprite D) Scorched Butterfly

Add to sprite C by making the fluttering butterfly sprite react to touching the color red. In particular, it should detect exactly the same shade of red as the sprite OSU-Scarlet contained in J:/src/db. If and when it touches this color, the butterfly should:

1. Broadcast the message "**scorched**"
2. Appear to burn, or be consumed by flames, or die in some way (changing costumes is probably the easiest way to accomplish this)
3. Reappear at some random place and repeat moving around the screen until the next time it touches red (at which point it should again be burned, broadcast "**scorched**" etc).

Sprite E) Timer and Score Keeper

Design a timer sprite that keeps track of how long has elapsed since the game was started (i.e., since the green flag was clicked). If too much time goes by, the sprite should appear, notifying the user that they have lost.

Design a score-keeper sprite that keeps track of how many times the message "**scorched**" has been broadcast. When this count has reached a certain threshold, the sprite should appear, notifying the user that they have won.

Hints: (i) a variable can be used to keep a count, (ii) make sure a variable is initialized, say to 0, (iii) develop your sprite by creating a project with multiple sprites, one of which is responsible for broadcasting the message.

Sprite F) Other

After completing your assigned sprite, feel free to add whatever effects and functionality you want to that sprite. Alternatively, you could design one or more *new* sprites that might belong in a world with dragons and butterflies. There are no specific requirements for such sprites.

Part II: Creating a Complete Program

In this part of the lab, you will create a complete Scratch project by *importing various sprites that other groups have written and modifying the result*. There should be several versions of each kind of sprite (A through E) available, so you can feel free to mix and match as you like.

Begin by importing one of each kind of sprite (A through E). Then combine the two dragon sprites into one dragon that moves in response to the arrow keys and breathes fire in response to the space bar. Next combine the two butterfly sprites into one butterfly sprite that flutters around the screen, but dies when the dragon burns it with fire. The timer and score keeper sprites should work without modification.

The result should be an interactive game. You may notice some odd or unexpected behaviors, however. Try to fix these problems and add whatever new elements you wish.

Completing Part II

Create a new project and save it on your computer.

3. Launch Scratch as in the previous lab, and create a new project (File, New)
4. Select File, Download to your computer, then find the directory J:, group?? (where ?? is *your* group number)
   - Give your project a name and click Save.
   - Remember to Save your project periodically.

Import a sprite by clicking the folder icon above the sprite pane ("upload sprite from file"). The sprite's costumes and programs (as well as any associated messages and variables) become part of the current program.

Debriefing

Accomplishments

During this lab, you successfully:

- Wrote several computer programs.
  Translation: Played with Scratch! You designed sprites and controlled how they move, look, sound, and even interacted with one another.
- Compared different specifications and evaluated their relative complexity.
  Translation: Problem-solved with Scratch! You used your programming skill to have your sprites accomplish particular tasks.
- Assembled a large program from a collection of smaller components.
  Translation: Created with Scratch! You combined sprites written by different groups to create a complex game in a short amount of time.
- Used the pair-programming method for code development.
  Translation: Worked in teams of 2 at one machine. This is actually a common approach used in the software industry. Programming together is not only more fun, it also results in better software.

Learning Outcomes

In the course of working with Scratch, you encountered some concepts that are actually deep and fundamental ideas in software engineering. These concepts include:

- A computer program is a sequence of *instructions*.
- A computer program describes *how* to transform *inputs* into *outputs*.
- A specification defines *what* transformation should be done (but *not how* to do it).
- Some specifications are easier to implement than others. The more information the specification gives about inputs, the *easier* it is to implement. The more information the specification gives about the outputs, the *harder* it is to implement.
- Smaller programs can be put together to make one larger program.
- Computer programming is fun!