Animated Activities

Philadelphia Area Math Teachers Circle
Steve Garland
May 17, 2016

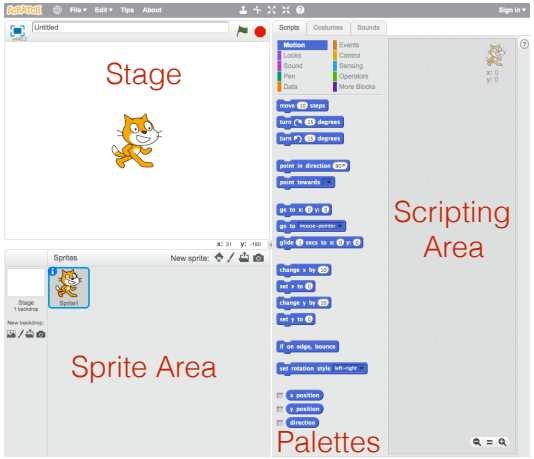
5:15 - 5:20 Slides 1-2

The Problem

- Teach the cat how to walk to the girl without bumping into the fire-eating dragon
- Using the Scratch visual programming system

Setting the Scene

- scratch.mit.edu



5:20 - 5:30 Slide 3 plus Scratch demonstration

Scratch: set desert backdrop, move cat below cactus, add Alex facing the cat

Mathematical note: two ways to have Alex face the cat, rotation and reflection

Coding: snap *point towards*, *set rotation style* blocks together

Sequence is the first of three fundamental ways to specify the flow of control

Warm-Up Exercise

- Teach the cat how to move to the girl

5:30 - 5:45 Slide 4: Warm-up exercise
Show slides 5-7 as participants work in pairs on exercise

Warm-Up Exercise …

- Teach the cat how to move to the girl
- Make the cat appear to walk

Warm-Up Exercise …

- Teach the cat how to move to the girl
- Make the cat appear to walk
- Make the cat start walking when you click it

Warm-Up Exercise …

- Teach the cat how to move to the girl
- Make the cat appear to walk
- Make the cat start walking when you click it
- Don’t depend on where the cat starts

Warm-Up Exercise
One Solution

5:45 - 6:00 Debriefing

10 minutes: Project participants' solutions on the smart board

5 minutes: Slides 8-10

Basic Structures for Code

Flow

- Sequence (straight-line code)
- Choice (branching or conditional code)
- Repetition (loop)

Solution uses sequence and repetition
Choice is coming soon!

Basic Structures for Code

Flow

- Sequence (straight-line code)
- Choice (branching or conditional code)
- Repetition (loop)

Organization

- Procedure (method, function, subroutine)

More later about another important structure

Enter, the dragon!

6:00 - 6:10 Slides 11-12, followed by brainstorming and Slide 13
Notice use of choice

The Problem

- Teach the cat how to walk to the girl without bumping into the fire-eating dragon
- But how?

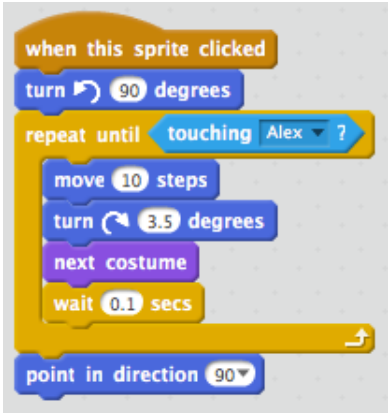
Some Things to Try

- Walk in a semicircle.
- Jump over the dragon.
- Bounce off the top of the screen.
- Hug the edges of the screen.
- … and lots more!

Semicircular Path

Why 3.5 degrees?

Experimentation
or
some simple math

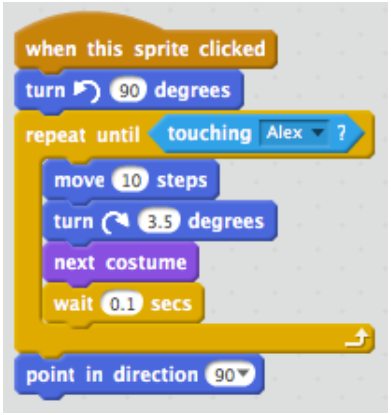


6:30 - 6:45 Debriefing (Slides 14-18 held in reserve)
First project participants' solutions on the smart board
Use Slides 14-18 to discuss connections to mathematics

Semicircular Path

Why 3.5 degrees?

Path diameter: 326
Path length: 512
Turn: 10*180/512



6:30 - 6:45 Debriefing (Slides 14-16 held in reserve)
First project participants' solutions on the smart board
Use Slides 14-16 to discuss connections to mathematics

Jump Over Dragon

Why 17.5?

Experimentation
or
some simple math

Jump Over Dragon

Why 17.5?

32 times through loop.
Height after 16 times should be about 160.

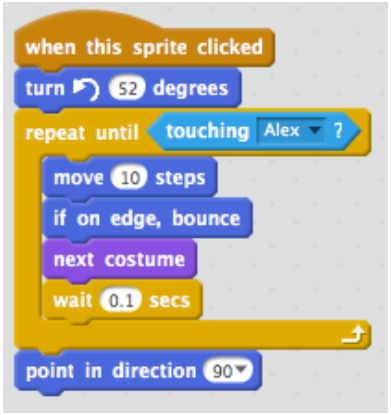
160 = v + (v-1) + … (v-15)
=16v - 15*16/2
=16v - 120

v = 17.5

Bounce Off Top Edge

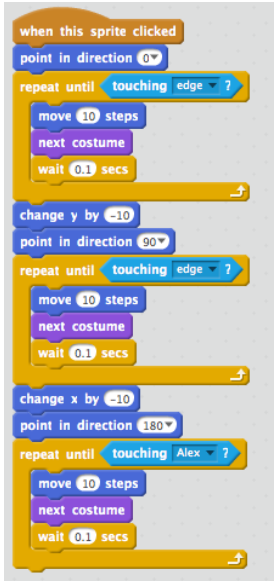
Why 52?

Need some trigonometry!



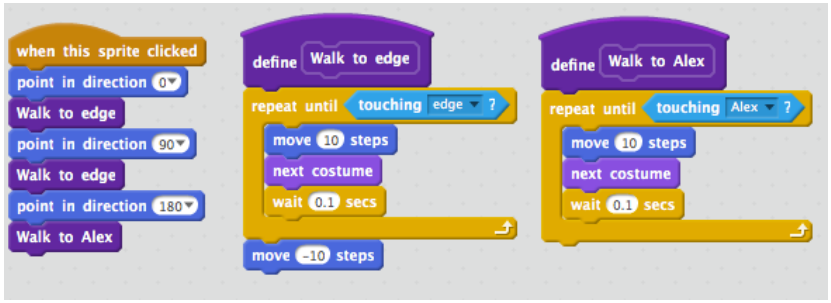
Hug Edges of the Screen

Repetitive code is hard to understand!



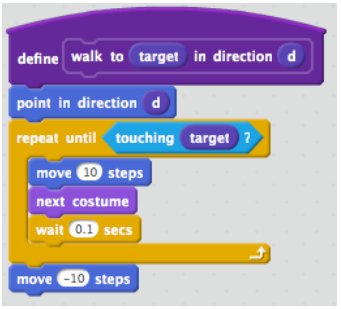
6:45 - 6:50 Debriefing, continued
Slides 19-21

Hug Edges of the Screen



- Defining new blocks makes code easier to understand

Hug Edges of Screen



- Defining a more general block makes the code even better

Connections to Common Core Math Standards

- Graph points on the coordinate plane to solve real-world and mathematical problems (5.G)
- Reason about and solve one-variable equations and inequalities (6.EE)
- Know the formulas for the area and circumference of a circle (7.G)
- Understand congruence and similarity using physical models, …, or geometry software; understand and apply the Pythagorean Theorem (8.G)

6:50 Wrap-up, Slides 22-24, leaving time for door prizes, etc.

Math Practices

MP1. Make sense of problems and persevere in solving them

MP2. Reason abstractly and quantitatively

MP3. Construct viable arguments and critique the reasoning of others

MP4. Model with mathematics

MP5. Use appropriate tools strategically

MP6. Attend to precision

MP7. Look for and make use of structure

CSTA Practices

1. Recognizing and representing computational problems
2. Developing and using abstractions
3. Creating computational artifacts
4. Testing and iteratively refining
5. Fostering an inclusive computing culture
6. Communicating about computing