**Coding and Geometry**

**Teacher Version**

**Authors:**

Michelle Hills

Fernando Alegre

Juana Moreno

Summer 2015

 

**Coding and Geometry:**

**Table of Contents**

Introduction

Overview of coding

Lesson 1:

* Points
* Segments
* Lines

Lesson 2: Intersections

* Line-Line
* Line-Circle
* Circle-Circle

Lesson 3:

* Midpoint
* Distance

Lesson 4: Angles

* Angle Measure
* Angle Bisector

Lesson 5: Lines

* Perpendicular Lines
* Parallel Lines

Lesson 6:

* Perimeter
* Area of Triangles

Lesson 7: Equilateral Triangle

Lesson 8: Transformations

* Translate
* Rotate
* Reflect
* Dilate

End of the Year Project

* Sample rubric accessible at <http://rubistar.4teachers.org/index.php?screen=ShowRubric&rubric_id=2547984&>

Reference Sheet

**Coding and Geometry**

**What?**

Coding and Geometry is a method of illustrating the use of coding in the context of high school geometry. The students will learn skills to manipulate programs step by step throughout the lessons. From basic skills such as drawing points on a display to more advanced functionality such as calculating the perimeter of an arbitrary triangle, these lessons encompass the goal of incorporating computer science skills into the high school mathematics classroom. The lessons are hands-on, and so they will require student computers and teacher facilitation.

**Why?**

Computer Science is a growing field. According to the Huffington Post, a study showed that from 2010-2020 there is a projected 19% increase in employment in computer and mathematical occupations compared to 14.3% on average for other growing occupations. Also according to Code.org, “more than 1.4 million computer jobs will be demanded by 2020, yet only 400,000 students will go on to study computer science in college.” In talking with several current college students and recent graduates, many stated that the first time they had seen any type of programming language was when they were required to take a programming class in college. Many of them stated that it was flat out scary even in a beginner’s course. Many discussions have led to the conclusion that if computer science is implemented into the basics of some high school courses, many more students may feel propelled to pursue a career in computer science or at least not be so overwhelmed by it in college.

The programs that are placed throughout the lessons are designed to be used as supplementary materials to help students have a visual representation of what is taking place in the classroom. Geometry is a visual concept to that allowing the programs to compute random examples for the students will give the students a better conceptual understanding. Also, using the non-random further applications will allow students to better understand how computer/calculator programs are set up. With the technological advances in our society, it is important to help our students take advantage of developing more efficient methods of calculations. Another factor to take into account is differentiated learning. Because the world is so diverse, it is important to cater to the individual learning styles of students. Coding will give the students another outlet to help them further understand the concepts being taught in the classroom.

**How?**

In order to effectively implement the lessons, go through each lesson carefully to make sure the coding is fully understood before trying to implement in the classroom. Each classroom is different so it will be important to carefully analyze the content and how it relates to your specified scope and sequence. It will not be in the best interest of the student if lessons are skipped as each lesson builds on each other. If it is decided to rearrange/skip lessons, it will be important to carefully dissect what coding knowledge or geometry content will need further explanation. Each lesson is designed for teacher/student interaction. It is important to facilitate each lesson by asking probing questions to help the students be able to draw their own conclusions. Then further direction on the correctness or completion of their conclusion should be given.

**Materials Needed:**

* Computers for every one to two students (will be best to have one computer per student)
* Teacher computer
* Teacher Version PDF
* Student Handouts (If you cannot make copies for each student, make a class set and have students use loose leaf paper to take notes.)
* Reference Sheet
* Downloadable Haskell Files
  + k12math-student—Files the student will need
  + k12math-teacher—Answers to the Exercises and Further Applications
* Programs installed on each computer being used
  + Windows:
    - Haskell Platform

(<https://www.haskell.org/downloads>)

Choose 64bit unless your computer is over 8 years old.

* + - Git (GitBash (terminal) is what you will open to run programs)

(<http://www.git-scm.com/downloads>)

* + - Notepad++ (Editor)

(<https://notepad-plus-plus.org/download/v6.7.9.2.html>)

* + MAC
    - Haskell Platform

http://www.haskell.org/platform/mac.html

* + - Terminal is already installed on Macs.

To access it open applications then the utilities folder. Open the terminal. Type /usr/bin/ld

If it ask you to install developer tools, click install.

* + - Atom (Editor)

<https://atom.io>

Click download. Then in the Welcome Guide you need to install a package. Click Install a package. Then in the search box type Haskell. Install package language-haskell.

**How to access Haskell Files:**

**Windows:**

1. Create a folder in Documents named Haskell
2. Download the files from the website.
3. Open the zip file.
4. Click Extract All Files.
   1. Browse
   2. Save it under Documents/Haskell
5. Extract

**Mac:**

1. Create a folder in Documents named Haskell
2. Download the files from the website.
3. Drag the files from downloads into the Haskell folder.
4. Then double click to unzip the file.

As you manipulate and create your own files (exercises), you will want to save them in a folder (YourName\_Coding) in the Haskell/k12math-student folder on the computer. At the end of the lesson if the computer is not your own, you will want to copy your folder onto your USB drive.

When you come back to manipulate and create files again in another lesson, just copy the folder from your USB to the Haskell/k12math-student folder on the computer. Repeat this process when using a school or someone else’s computer.

**Getting started with the terminal:**

After installing the three programs listed above, open the terminal and perform the following commands:

Teacher & Student

1. Type cd Documents/Haskell/k12math-student
2. Type util/prepare

Teacher only

1. Type cd ..
2. Type cd k12math-teacher
3. Type util/prepare

**Running Code:**

**Windows:**

1. Open GitBash
2. cd Documents/Haskell/k12math-\_\_\_\_\_\_\_\_\_\_\_ (the blank should either be student or teacher)
3. ls
   1. If running a program already installed in k12math-\_\_\_\_\_\_\_\_\_\_\_, type

rungeo prog/\_\_\_\_\_\_\_\_\_\_ /lesson1a.hs ( the blank should either be student or teacher)

* 1. If running a program you manipulated and saved in your personal folder, type

rungeo yourname\_coding/lesson1a.hs

**MAC:**

1. Open the terminal (applications/utilities)
2. Open GitBash
3. cd Documents/Haskell/k12math-\_\_\_\_\_\_\_\_\_\_\_ (the blank should either be student or teacher)
4. ls
   1. If running a program already installed in k12math-\_\_\_\_\_\_\_\_\_\_\_, type

./rungeo prog/\_\_\_\_\_\_\_\_\_\_ /lesson1a.hs ( the blank should either be student or teacher)

* 1. If running a program you manipulated and saved in your personal folder, type

./rungeo yourname\_coding/lesson1a.hs