Programming Course Elements

Eric Grabe

Course Overview:

This course is the first in a sequence of two courses designed to prepare high school students for the AP Computer Science A exam. At the same time, the material in this class is intended to be a simple introduction to programming that can be enjoyed by students with no programming experience whatsoever. By the end of the course, students will be proficient in basic programming concepts including variables, conditionals, iteration, functions, object-oriented programming, arrays and more. Given a solid introduction to these base concepts, students are well prepared to extend their programming skills into other languages and more complicated programming techniques that will be covered in the second course of the sequence. The course is 18 weeks over which students develop a game using the programming environment Processing. The Processing environment is Java based and allows for relatively easy drawing and manipulation of graphics which makes it convenient for creating games. Note on the assessment portion; the code review interviews students will complete no less than 4 code reviews in a one on one interview with the instructor. Students may complete more than 4 code reviews if they choose.

Course Outcomes:

1. Students will be able to write programs that utilize basic programming techniques including primitive data types, variables, conditionals, iteration, and composite data types.
2. Students will be able to write programs that utilize basic programming syntax and structure including functions, file I/O, classes, and objects.

Intermediate Outcomes:

Unit 1: Drawing Shapes

- I can represent data in my program using variables of the appropriate type
- I can explain how data can be lost when casting data types

Assessment:

- Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
- Unit project requiring students to write a program that draws their hero character to the screen. All locations of component parts of the character must be given relative to a central control point for the character. This is scored on a rubric.
- Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.
Unit 2: Interaction

- I can use built in functions to handle events such as key presses and mouse clicks
- I can use built in variables to control/change the location of objects on the screen

Assessment:

- Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
- Unit project requiring students to write a program that allows the user to interact with their hero character through mouse movement, key presses, and mouse clicks. These key presses and mouse clicks must be handled by the keyPressed() and mousePressed() functions built into Processing. This is scored on a rubric.
- Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Unit 3: Conditionals

- I can write Boolean conditional statements to test for multiples states in my program.
- I can write complex conditionals utilizing if, else if, and else and/or by nesting conditionals.

Assessment:

- Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
- Unit project requiring students to write a program that randomly generates an obstacle for the hero to avoid. Conditionals are written to detect collisions between the obstacle and the hero. This is scored on a rubric.
- Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.
Unit 4: Iteration

- I can write functional looping statements to simplify repetitive events in my code. (for & while)
- I can nest looping statements within each other when repetition is based on multiple conditions
- I can differentiate between local and global scope of a variable.

Assessment:

- Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
- Unit project requiring students to write a program that draws a background (scenery) behind the game. Nested loops create a randomly scattered pattern of objects in the background. (Examples: stars, tree, buildings etc.) This is scored on a rubric.
- Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Unit 5: Advanced Functions

- I can write functions that accept arguments as parameters
- I can write functions that return values of appropriate type

Assessment:

- Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
- Unit project requiring students to write a program that includes a more advanced function that detects collisions in their game. The collision function accepts locations of the hero and the obstacle then returns true if they have collided. This is scored on a rubric.
- Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Unit 6: Objects

- I can create classes and objects from these classes
- I can use constructor arguments to add variability to the objects in a class

Assessment:

- Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
- Unit project requiring students to refactor the code for their game to be object oriented. Student identify the data and functions for the components of the game and write class files for each component. A carrot is added to the game that the hero character collects for points. This is scored on a rubric.
• Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Unit 7: Arrays

• I can declare and initialize values in an array using a loop
• I can create multiple variations of the same objects using an array
• I can sort/search an array

Assessment:

• Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
• Unit project requiring students to write a program that increases the complexity/difficulty of the game by adding multiple obstacles for the hero to dodge. Students must create an array of obstacle/carrot objects for the hero to dodge/collect. This is scored on a rubric.
• Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Unit 8: File I/O and Strings

• I can read/write data from a file
• I can use join/split to manipulate a String

Assessment:

• Students will take a quiz with multiple types of questions (fill in the blank, matching, ordering lists, error analysis etc.) Student must score at least 80% to advance to the unit project.
• Unit project requiring students to write a program that records player initials and high score for their game. The program must read/write to a separate file that contains high score info. The user must be able to input their initials at the end of the game if a high score has been achieved. This is scored on a rubric.
• Students complete a code review interview in which they discuss their code with the instructor. Student demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.