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.

Grading Outline

<table>
<thead>
<tr>
<th>Generic Unit Grading Scale</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>4  Meets all requirements for 3; Demonstrates skills outlined in optional examples/readings</td>
<td>A+</td>
</tr>
<tr>
<td>3  Demonstrates all skills outlined in unit project description; Knowledgably discusses code in one on one code review with the instructor</td>
<td>A/A-</td>
</tr>
<tr>
<td>2.5 Demonstrates some skills outlined in unit project description; Knowledgably discusses code in one on one code review with the instructor</td>
<td>B+/B</td>
</tr>
<tr>
<td>2  Demonstrates all skills outlined in unit project description; Requires assistance from instructor while discussing code in one on one code review with the instructor</td>
<td>B-/C+</td>
</tr>
<tr>
<td>1.5 Demonstrates some skills outlined in unit project description; Requires assistance from instructor while discussing code in one on one code review with the instructor</td>
<td>C/C-</td>
</tr>
<tr>
<td>1  Demonstrates some skills outlined in unit project description; Cannot knowledgably discuss code in one on one code review with instructor. (Even with assistance.)</td>
<td>D+/D/D-</td>
</tr>
<tr>
<td>0  Has not demonstrated skills outlined in unit project description</td>
<td>F</td>
</tr>
</tbody>
</table>
Code review interviews will include the questions below. Students submit written answers then instructor follows up it more clarification is needed.

<table>
<thead>
<tr>
<th>Code Review Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. What was the most difficult part of this unit/project? Where did you struggle?</td>
</tr>
<tr>
<td>2. What successes did you have working on this project? What are you most proud of?</td>
</tr>
<tr>
<td>3. Walk me through the portion of the code that meets &lt;insert unit requirement&gt;. Continue with probing questions to evaluate understanding of how the code works.</td>
</tr>
<tr>
<td>4. Which of the learning activities best helped you understand the new concepts/skills?</td>
</tr>
<tr>
<td>5. How would you explain this content/skills to someone who was struggling with understanding this material?</td>
</tr>
<tr>
<td>6. &lt;include other content specific questions for the unit&gt;</td>
</tr>
</tbody>
</table>

Students may resubmit unit projects and repeat code reviews to improve their scores.

**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.
3. Students will be able to document and discuss the programs that they write.

**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. (interview)

**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. Meets requirements below:
  1.1 Program includes header and imbedded documentation and naming conventions that allows reader to understand logic/structure.
  1.2 All shapes and locations are coded relative to a central point within the character using appropriately typed variables
  1.3 Program generates random numbers within the appropriate range.
- Students complete a code review interview in which they discuss their code with the instructor. Students demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.
Learning Activities

• Lecture/Live coding: Example program that draws a ball bouncing around the screen. Focus on introductory/IDE specifics, println, variables, variable type and assignments, RGB color mixing (Knowledge)
• Reading: Text 1.1-1.5 (Knowledge)
• Optional Readings: Drawing Curves and beginShape function. (Optional readings are for students who are more experience with programming or want to do more advanced things than the course requires.) (Knowledge)
• Handwritten programming: Document requiring student to hand write snippets of code. Will include descriptions of data to be stored in the program; students need to correctly create, and initialize a primitive variable. Practice with function syntax as well, void functionName() {
} (Practice)
• Exercises/Pair Programming:
  o Write a program that uses basic shapes to draw a smiley face to the screen. All location values should use variables rather than hard numbers
  o Update your smiley program to look crazy. Adjust shape attributes like color and Stroke Weight to make your smiley face unique (Practice)

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.
  2.1 Program includes header and imbedded documentation and naming conventions that allows reader to understand logic/structure.
  2.2 Program is broken down into functions.
  2.3 Program handles mouse and keyboard events with mousePressed() and keyPressed() functions
  2.4 Program utilizes built in variables mouseX/mouseY.
• 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.

Learning Activities

• Lecture/Live coding: Continue bouncing ball program making a new ball that will chase the mouse. Focus on functions that handle mouse/keyboard functions, easing a target example. (Knowledge)
• Reading: Text 3.1-3.4 (Knowledge)
• Handwritten programming: Document requiring student to hand write snippets of code. Will include example programs and descriptions of desired results. Student replace values in code with mouse/keyboard variables to achieve desired results.
• Exercises/Pair Programming:
  o Write update the Smiley program that follows the mouse around the screen. The eyes of the smiley will ease their target as the smiley moves around. (Practice)
  o Write an alternate smiley program that changes the colors based on screen position. A key press clears the screen. (Practice)

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 character.  
  3.1 Program includes header and imbedded documentation and naming conventions that allows reader to understand logic/structure.
  3.2 Program contains conditional logic that detects collisions between character and obstacles.
  3.3 Program uses logical operator (and/or) and nested conditionals to check for specific conditions
• 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.

Learning Activities
• Lecture/Live coding: Continue bouncing ball program making the ball bounce off the edges of the screen. Focus on Boolean expressions, random(), if/else/else if, and &&/|| (Knowledge)
• Reading: Text p.49-57 & 59-67(Knowledge)
• Handwritten programming: Document requiring student to hand write snippets of code. Will include exercises where students write Boolean test for cards drawn from a deck. Example: Write a test to check if a drawn playing card is red? Or black? Or a face card? Etc. (Practice)
• Exercises/Pair Programming:
  o Write update the Smiley that randomly changes color when the mouse is inside and background changes color when the mouse is outside the smiley. Focus on the conditional logic of the location of the mouse if/else if/ else (Practice)
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. (Code review)

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.)

4.1 Program includes header and imbedded documentation and naming conventions that allow reader to understand logic/structure.
4.2 Program uses iterative logic to simplify repetitive parts of the code
4.3 Program includes nested loops to achieve the desired results
4.4 Programs includes variables of local and global scope

- Students complete a code review interview in which they discuss their code with the instructor. Students demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Learning Activities

- Lecture/Live coding: Continue bouncing ball program making a line of bouncing balls that follow each other around the screen. Demonstrate while() and for() loops and nesting of loops for a grid pattern of bouncing balls (Knowledge)
- Reading: Ch. 6 p 81-95 (Knowledge)
- Handwritten programming: Document requiring student to hand write snippets of code. Student are given geometric patterns of shapes/lines. Ask to write loops/nested loops that would recreate the patterns. Begin with simple patterns and fill in the blanks that grow more complicated and give less code to start. (Practice)
- Exercises/Pair Programming:
  - Write update the Smiley that repeats the smiley to a grid of smileys. Nested loop to make rows/columns of smileys

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.
5.1 Program includes header and imbedded documentation and naming conventions that allows reader to understand logic/structure.
5.2 Main program consists of appropriate number of function calls
5.3 Program includes functions that accept arguments that control their behavior
5.4 Program includes functions that return useful values
   • 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.

Learning Activities
   • Lecture/Live coding: Continue bouncing ball program making two balls bounce off each other. Write a function that accepts the location of the two balls as arguments and changes direction when they collide. Include a check edges function that returns true when the ball is on the edge of the screen. The ball changes color when it hits the edge. (Knowledge)
   • Reading: Text p. 101-116 (Knowledge)
   • Handwritten programming: Document requiring student to hand write snippets of code. Will include exercises where students re-write simple given functions to accept parameters or return certain values. (Practice)
   • Exercises/Pair Programming:
     o Rewrite the color changing smiley from unit 3 to use functions that accept mouse location as arguments and return True or False when inside the smiley. (Practice)
     o I may add another activity here.

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. Students 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.
6.1 Program includes header and imbedded documentation and naming conventions that allows reader to understand logic/structure.
6.2 Program includes appropriate number class files that include data and functions for modular parts of the code.
6.3 Class files include constructor functions that utilize pass by copy method to create objects of the same class with different attributes
   • Students complete a code review interview in which they discuss their code with the instructor. Students demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.
Learning Activities

- Lecture/Live coding: Refactor the bouncing ball code. Write a class file for a ball object that bounces around the screen. Call two ball objects that can bounce off of one another. Focus is on reorganization of the code and writing constructors to make multiple unique ball objects. (Knowledge)
- Reading: Text p. 129-139 (Knowledge)
- Handwritten programming: Document requiring student to hand write snippets of code. Will include exercises where student rewrite linear code to be object oriented. Also includes practice at writing constructors from sample code. (Practice)
- Exercises/Pair Programming:
  - Make an OOP version of the smiley face code. Code should produce 5 unique smiley faces. (Five smiley objects that have unique values passed in to make them unique.)

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.
  7.1 Program includes header and imbedded documentation and naming conventions that allows reader to understand logic/structure.
  7.2 Program utilizes iterative logic to initialize an array.
  7.3 Program creates multiple variations of the same objects using an array
  7.4 Program uses array methods to organize or adjust values in the array
- Students complete a code review interview in which they discuss their code with the instructor. Students demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Learning Activities

- Lecture/Live coding: Create an array of bouncing ball objects of various sizes. At the start of the program the balls line up by size on the left-hand side of the screen. (Knowledge)
- Reading: Text p. 141-151 (Knowledge)
- Handwritten programming: Document requiring student to hand write snippets of code. Will include exercises where students initialize arrays or sort given arrays. Also, revisit same assignment from Unit 6 but make them arrays of objects. (Practice)
- Exercises/Pair Programming:
Revisit the Unit 6 exercise with the Smiley object. Make an array of smiley objects that are arranged in a grid pattern. Every time the mouse is clicked an new unique smiley is added to the grid.

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.

8.1 Program includes header and imbedded documentation and naming conventions that allows reader to understand logic/structure.
8.2 Program inputs/outputs data from external source
8.3 Program uses split/join to create and manipulate arrays of Strings

- Students complete a code review interview in which they discuss their code with the instructor. Students demonstrate understanding of the code and are asked to explain the decisions they have made in their code. This is scored on a rubric.

Learning Activities

- Lecture/Live coding: Another version of the bouncing balls program. Each ball has a name that follows it around the screen. The names are pulled in from a separate file. (Knowledge)
- Reading: Text p. 305, p. 325 (Knowledge)
- Handwritten programming: Document requiring student to hand write snippets of code. Given a list of data hand write code that loads the data into a usable form (String/Array of strings) a program. Also hand write code that will alter text/organize data in some way. (Practice)
- Exercises/Pair Programming:
  - Write a program that counts the number of times the word “ham” is used in the book Green Eggs and Ham.
  - Given a list of students and a generic form letter. Insert each students name into an individualize version of the form letter. Program outputs a single file for each student in the class.