Course Overview:
8th grade Computer Science Exploration focuses on algorithms and programming. By the end of the course, students should be able to create an interactive animation or game that includes basic programming concepts such as control structures, variables, user input, and randomness. They should manage this by working with others to break it down using objects (sprites) and functions. Throughout the process, they should give and respond constructively to peer feedback and work with their teammates to complete a project.

Course Outcomes:

Program Development
Students should be able to collaborate with peers to develop a piece of software. This process involves defining the needs of the program, designing a program to meet those needs, and breaking down the design into manageable pieces. Student code should be written so that others can read and understand it, and they should give and receive feedback on their work, as well as test and revise the program.

Assessment
Interactive Card -
Students will give thoughtful feedback to peers and respond to peer feedback by making appropriate changes to their program. (Rubric in Appendix B)

Game-
Students will design a program guide and reflect on peer suggestions. (Rubric in Appendix C) (Peer Guide in Appendix D)

Modularity
Students should be able to break down complex problems into their component parts, both to increase readability and organization of code and to allow them to reuse portions of code many times. Algorithms should be broken into functions, and screen elements into sprites/objects. They should also recognize and use abstraction as it is built into programming languages.

Assessment
Interactive Card -
Students will design a card with multiple sprites, with multiple properties updated in the draw loop. (Rubric in Appendix B)

Game-
Students will design at least three functions used to organize code into logical segments. At least one of these functions is called multiple times in the program. (Rubric in Appendix C)
Algorithms and Control
Students should be able to use basic programming constructs to create a wide range of behaviors in their programs. These constructs should be combined to create complex behaviors, such as screen elements that move according to user input, or properties that change after a certain threshold has been reached. Programs should run differently each time according to user input or random chance.

Assessment
Interactive Card -
Student’s program responds to multiple types of user input and uses at least one random number. The student’s program is well sequenced and properly separates code in and out of the draw loop. The card uses multiple conditionals inside the draw loop, at least one of which is triggered by a variable or sprite property. (Rubric in Appendix B)

Game -
Students will design a game that has at least three backgrounds that are displayed during run time, and at least one change is triggered automatically through a variable (e.g. score). The student’s game also includes multiple different interactions between sprites, responds to multiple types of user input (e.g. different arrow keys). (Rubric in Appendix C)

Position and Movement
Students should use the coordinate plane to place and move screen elements. They should be able to model various types of motion, including acceleration, linear movement and simulating gravity.

Assessment
Interactive Card -
Student’s card contains multiple elements that are placed on the screen using the coordinate system, and that move in different ways. (Rubric in Appendix B)

Game -
Student’s games will include complex movements such as acceleration, moving in a curve, or jumping in multiple places in the program. (Rubric in Appendix C)

Variables/Storing Information
Students should be able to create new variables as needed in their programs, and update and access the variable values as the program runs.

Assessment
Interactive Card -
Student’s card contains multiple variables that are used and their values are updated during the program. At least one variable or property uses the counter pattern. (Rubric in Appendix B)
Game -
The game includes multiple variables that are updated during the game and affect how the game is played. (Rubric in Appendix C)

Assessment Rubrics in Appendix E (Interactive Card) Appendix G (Game). Students will be given scores of 1-4 based on their performance on the standards.
## Unit 3 Chapter 1 Project Rubric

<table>
<thead>
<tr>
<th>Key Concept</th>
<th>Extensive Evidence</th>
<th>Convincing Evidence</th>
<th>Limited Evidence</th>
<th>No Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Development</td>
<td>Gave thoughtful feedback to peers and responded to peer feedback by making appropriate changes to program</td>
<td>Gave and responded to peer feedback.</td>
<td>Gave some feedback to peers.</td>
<td>Did not give feedback to peers</td>
</tr>
<tr>
<td>Modularity</td>
<td>Multiple sprites, with multiple properties updated in the draw loop</td>
<td>Multiple sprites, each with at least one property updated inside the draw loop</td>
<td>At least one sprite, with at least one property updated after sprite creation.</td>
<td>No sprites, or no sprite properties are updated after the sprite is created.</td>
</tr>
<tr>
<td>Algorithms and Control Structures</td>
<td>Program responds to multiple types of user input and uses at least one random number</td>
<td>Program responds to user input and uses at least one random number</td>
<td>Program uses a random number or responds to a user input.</td>
<td>Program does not use random numbers or respond to user input.</td>
</tr>
<tr>
<td>Algorithms and Control Structures</td>
<td>Program is well sequenced and properly separates code in and out of the draw loop.</td>
<td>Program correctly separates code in and out of the draw loop to create animation. May contain some incorrectly sequenced code.</td>
<td>Program is animated through the draw loop, but some code is improperly placed in or out of the loop.</td>
<td>Draw loop is not used to create animation</td>
</tr>
<tr>
<td>Algorithms and Control Structures</td>
<td>Uses multiple conditionals inside the draw loop, at least one of which is triggered by a variable or sprite property</td>
<td>Uses a conditional that is triggered by a variable or sprite property inside the draw loop.</td>
<td>Uses at least one conditional inside the draw loop.</td>
<td>No conditionals.</td>
</tr>
<tr>
<td>Position and Movement</td>
<td>Multiple elements are placed on the screen using the coordinate system, and move in different ways.</td>
<td>At least one element is placed on the screen using the coordinate system and moves during the program.</td>
<td>At least one element is placed on the screen using the coordinate system.</td>
<td>No elements (sprites or shapes) are placed on the screen using the coordinate system.</td>
</tr>
<tr>
<td>Variables</td>
<td>Multiple variables are used and their values are updated during the program. At least one variable or property uses the counter pattern.</td>
<td>At least one variable is used, and its value is updated during the program.</td>
<td>At least one variable is used in the program.</td>
<td>No variables.</td>
</tr>
</tbody>
</table>
## Unit 3 Chapter 2 Project Rubric

<table>
<thead>
<tr>
<th>Key Concept</th>
<th>Extensive Evidence</th>
<th>Convincing Evidence</th>
<th>Limited Evidence</th>
<th>No Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Development</td>
<td>The project guide is complete and reflects the project as submitted.</td>
<td>The project guide is mostly complete and generally reflective of the submitted project.</td>
<td>The project guide is filled out, but is not complete or does not reflect the submitted project.</td>
<td>The project guide is incomplete or missing.</td>
</tr>
<tr>
<td>Program Development</td>
<td>The program code effectively uses whitespace, good naming conventions, indentation and comments to make the code easily readable.</td>
<td>The program code makes use of whitespace, indentation, and comments.</td>
<td>The program code has few comments and does not consistently use formatting such as whitespace and indentation.</td>
<td>The program code does not contain comments and is difficult to read.</td>
</tr>
<tr>
<td>Modularity</td>
<td>At least three functions are used to organize code into logical segments. At least one of these functions is called multiple times in the program.</td>
<td>At least two functions are used in the program to organize code into logical segments.</td>
<td>At least one function is used in the program.</td>
<td>There are no functions in the program.</td>
</tr>
<tr>
<td>Algorithms and Control</td>
<td>The game has at least three backgrounds that are displayed during run time, and at least one change is triggered automatically through a variable (e.g. score).</td>
<td>The game has multiple backgrounds that are displayed during run time (e.g. main background and &quot;end game&quot; screen)</td>
<td>The game has multiple backgrounds.</td>
<td>The game does not have multiple backgrounds.</td>
</tr>
<tr>
<td>Structures</td>
<td>The game includes multiple different interactions between sprites, responds to multiple types of user input (e.g. different arrow keys).</td>
<td>The game includes at least one type of sprite interaction and responds to user input.</td>
<td>The game responds to user input through a conditional.</td>
<td>The game includes no conditionals.</td>
</tr>
<tr>
<td>Position and Movement</td>
<td>Complex movement such as acceleration, moving in a curve, or jumping is included in multiple places in the program.</td>
<td>The program includes some complex movement, such as jumping, acceleration, or moving in a curve.</td>
<td>The program includes simple independent movement, such as a straight line or rotation.</td>
<td>There is no movement in the program, other than direct user control.</td>
</tr>
<tr>
<td>Variables</td>
<td>The game includes multiple variables that are updated during the game and affect how the game is played.</td>
<td>The game includes at least one variable that is updated during the game and affects the way the game is played</td>
<td>There is at least one variable used in the program.</td>
<td>There are no variables, or they are not updated.</td>
</tr>
</tbody>
</table>
## Peer Review - Interactive Card

### Pre-Review
Creator's Name: ____________________________

One thing I want feedback on is... ________________________________________________________

### Reviewer Section
Reviewer’s Name: __________________________

<table>
<thead>
<tr>
<th>Evidence I Found</th>
<th>Types of Evidence</th>
<th>Ideas for More</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple sprites, with multiple properties updated in the draw loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program responds to multiple types of user input and uses at least one random number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program is well sequenced and properly separates code in and out of the draw loop.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uses multiple conditionals inside the draw loop, at least one of which is triggered by a variable or sprite property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple elements are placed on the screen using the coordinate system, and move in different ways.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple variables are used and their values are updated during the program. At least one variable or property uses the counter pattern.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An overview of each lesson and a closing activity to formatively assess students and/or opportunity to provide feedback to students.

Lesson 1: Programming for Entertainment
The class is asked to consider the "problems" of boredom and self expression, and to reflect on how they approach those problems in their own lives. From there, they will explore how Computer Science in general, and programming specifically, plays a role in either a specific form of entertainment or as a vehicle for self expression.

*Live Coding: Students will be able to code in the stamp and alien lab for ‘fun’.

*Journal: Based on what you saw today, both in your research and the example apps, what kinds of programs are you most interested in learning to create?

Lesson 2: Plotting Shapes
This lesson explores the challenges of communicating how to draw with shapes and use a tool that introduces how this problem is approached in Game Lab. The class uses a Game Lab tool to interactively place shapes on Game Lab’s 400 by 400 grid. Partners then take turns instructing each other how to draw a hidden image using this tool, accounting for many of the challenges of programming in Game Lab.

*Live Coding: Students will draw shapes in Game Lab.

*Journal: Have students reflect on each of the following prompts

  ● What things were important in communicating about position, color, and order of the shapes in this activity?
  ● What’s a way you have seen similar problems solved in the past?

Lesson 3: Drawing in Game Lab
The class is introduced to Game Lab, the programming environment for this unit, and begins to use it to position shapes on the screen. The lesson covers the basics of sequencing and debugging, as well as a few simple commands. At the end of the lesson, the class creates an online version of the image they designed in the previous lesson.

Live Coding: Students will begin block programming in Game Lab.

*Exit Ticket
Goal: Students share tricks they learned as they went through levels.
Prompt: Today you learned how to draw in Game Lab for the first time. What type of advice would you share with a friend who was going to learn about drawing in Game Lab to make it easier for them? Write it down on a piece of paper.
Collect: Collect answers from students and pick out a few that might be helpful for all students to hear. Share those at the beginning of the next class.

**Lesson 4: Shapes and Randomization**
This lesson extends the drawing skills to include width and height and introduces the concept of random number generation. The class learns to draw with versions of ellipse() and rect() that include width and height parameters and to use the background() block to fill the screen with color. At the end of the progression the class is introduced to the randomNumber() block and uses the new blocks to draw a randomized rainbow snake.

*Live Coding: Students begin making shapes in Game Lab. They are introduced to functions.*

*Journal*
Prompt: Have students reflect on their development of the five practices of CS Discoveries (Problem Solving, Persistence, Creativity, Collaboration, Communication). Choose one of the following prompts as you deem appropriate.
- Choose one of the five practices in which you believe you demonstrated growth in this lesson. Write something you did that exemplified this practice.
- Choose one practice you think you can continue to grow in. What’s one thing you’d like to do better?
- Choose one practice you thought was especially important for the activity we completed today. What made it so important?

**Lesson 5: Variables**
This lesson introduces variables as a way to label a number in a program or save a randomly generated value. The class begins the lesson with a very basic description of the purpose of a variable and practices using the new blocks. Afterwards, the class uses variables to save a random number, allowing the programs to use the same random number multiple times.

* Live Coding: Students block code again in Game Lab but today they are introduced to variables.

*Reflection*
Prompt: Give students the following prompts
- What is your own definition of a variable?
- Why are variables useful in programs?
Discuss: Have students silently write their ideas before sharing in pairs and then as a whole group.
Journal: What connections do you see between variables and what you learned about the Input-Output-Store-Process model of a computer?

**Lesson 6: Sprites**
In order to create more interesting and detailed images, the class is introduced to the sprite object. Every sprite can be assigned an image to show, and sprites also keep track of multiple values about themselves, which will prove useful down the road when making animations. At the end of the lesson, everyone creates a scene using sprites.

*Live Coding: Multi-day unit on sprites. Students create functions to draw sprites.

*Share Out
Share: Allow students to share their Sprite Scenes. Encourage students to reflect on their scenes and identify ways in which they'd like to improve.

Lesson 7: The Draw Loop
This lesson introduces the draw loop, one of the core programming paradigms in Game Lab. The class combines the draw loop with random numbers to manipulate some simple animations with dots and then with sprites. Afterwards, everyone uses what they learned to update the sprite scene from the previous lesson.

*Live Coding: Students are introduced to a built in function for drawing called the 'draw' loop. This is used to teach iteration.

*Prompt: Have students respond to the following prompts
  ● What is an animation?
  ● Why does the draw loop help us make animations?
  ● What are some common errors or mistakes we should look out for as we keep programming with the draw loop?

Lesson 8: Counter Pattern Unplugged
This unplugged lesson explores the underlying behavior of variables. Using notecards and string to simulate variables within a program, the class implements a few short programs. Once comfortable with this syntax, the class uses the same process with sprite properties, tracking a sprite’s progress across the screen.

*Prompt: We saw some clues today of how we might program the types of movement that we want for our sprites. What are some different ideas for how to program movement that you have after this activity? What are some problems that we still need to solve to make the sprite look like it's moving in the way that you want?
Allow students to brainstorm problems and list them on the board.
*Prompt: Choose one or two of these problems and start to think of some ways you could solve this problem.
Allow students time to brainstorm individually before sharing out their solutions.

Lesson 9: Sprite Movement
By combining the Draw Loop and the Counter Pattern, the class writes programs that move sprites across the screen, as well as animate other sprite properties.

*Live Coding: Students pull in all elements they have learned to animate their sprites.

*Journal
Prompt: Have students reflect on their development of the five practices of CS Discoveries (Problem Solving, Persistence, Creativity, Collaboration, Communication). Choose one of the following prompts as you deem appropriate.
  - Choose one of the five practices in which you believe you demonstrated growth in this lesson. Write something you did that exemplified this practice.
  - Choose one practice you think you can continue to grow in. What’s one thing you’d like to do better?
  - Choose one practice you thought was especially important for the activity we completed today. What made it so important?

Lesson 10: Booleans Unplugged
This lesson introduces boolean values and logic, as well as conditional statements. The class starts by playing a simple game of Stand Up, Sit Down in which the boolean (true/false) statements describe personal properties (hair or eye color, clothing type, age, etc). The class then groups objects based on increasingly complex boolean statements, then looks at how conditionals can impact the flow of a program.

*Prompt: Ask students some boolean questions about that single object AND give students something to do if that question is true. For example:
  - If sides is equal to 4, do a dance
  - If pattern is equal to striped, sit down
  - If width is equal to height, hop on one foot

Lesson 11: Booleans and Conditionals
The class starts by using booleans to compare the current value of a sprite property with a target value, using that comparison to determine when a sprite has reached a point on the screen, grown to a given size, or otherwise reached a value using the counter pattern. After using booleans directly to investigate the values or sprite properties, the class adds conditional if statements to write code that responds to those boolean comparisons.

*Live Coding: Students begin outputing results of boolean comparisions to the console.

*Journal: Think back to all of the programs you’ve written so far; how might you use conditionals to improve one of your programs from past lessons? What condition would you check, and how would you respond to it?
Lesson 12: Conditionals and User Input
Following the introduction to booleans and if statements in the previous lesson, students are introduced to a new block called keyDown() which returns a boolean and can be used in conditionals statements to move sprites around the screen. By the end of this lesson students will have written programs that take keyboard input from the user to control sprites on the screen.

*Live Coding: This module teaches the students about user input and how to deal with key presses and run them through some boolean logic.

*Prompt: To get students to continue thinking about how conditionals can be used in programming, prompt them to come up with scenarios in games or programs they use regularly that might be triggered by conditionals.

Lesson 13: Other Forms of Input
The class continues to explore ways to use conditional statements to take user input. In addition to the simple keyDown() command learned in the previous lesson, the class learns about several other keyboard input commands as well as ways to take mouse input.

*Live Coding: Students are introduced to conditionals.

*Prompt: You now have many different ways to detect user input. With a partner, choose three different user input commands and think of an example of when you might use them.

Lesson 14: Project - Interactive Card
In this cumulative project for Chapter 1, the class plans for and develops an interactive greeting card using all of the programming techniques they've learned to this point.

*Live Coding: Students will work on their cards.

*Sharing Cards
Goal: Students share their creations with the class.
Share: Find a way for students to share their cards with each other, and with the intended recipient. It will likely be helpful to use the share link for the project so that students can share the project with other students.

Lesson 15: Velocity
After a brief review of how the counter pattern is used to move sprites, the class is introduced to the properties that set velocity and rotation speed directly. As they use these new properties in different ways, they build up the skills they need to create a basic side scroller game.

*Live Coding: Students create counting loops to track velocity.
*Prompt: You learned a few new blocks today. As first glance, these blocks did the same sorts of things we'd already done with the counter pattern, but made it simpler for us to do them. As you went through the puzzles, though, you started doing some interesting movements that we hadn't been able to do before.

- Describe one of those movements, and how you made it.
- Describe another block that you'd like to have.
  - What would you name it?
  - What would it do?
  - What code would it hide inside?
  - How would it help you?

Lesson 16: Collision Detection
The class learns about collision detection on the computer. Pairs explore how a computer could use sprite location and size properties and math to detect whether two sprites are touching. The class then uses the isTouching() block to create different effects when sprites collide, including playing sounds. Last, they use their new skills to improve the side-scroller game that they started in the last lesson.

*Live Coding: This lab uses variables and conditionals to teach the kids if two sprites are touching each other.

*Prompt: At the beginning of the lesson, you saw that it’s possible to do everything that the isTouching block does without using the block at all. What makes this block useful?

Lesson 17: Complex Sprite Movement
The class learns to combine the velocity properties of sprites with the counter pattern to create more complex sprite movement, such as simulating gravity, making a sprite jump, and allowing a sprite to float left or right. In the final levels the class combine these movements to animate and control a single sprite and build a simple game in which a character flies around and collects coins.

*Live Coding: Students learn X,Y movements of sprites in Game Lab.

*Prompt: Today we built lots of new sprite movements like gravity and jumping, but none of this required us to learn new blocks. How were you able to do new things without learning any new blocks?

Lesson 18: Collisions
The class programs their sprites to interact in new ways. After a brief review of how they used the isTouching block, the class brainstorms other ways that two sprites could interact. They then use isTouching to make one sprite push another across the screen before practicing with the four collision blocks (collide, displace, bounce, and bounceOff).
Lesson 19: Functions
This lesson covers functions as a way to organize their code, make it more readable, and remove repeated blocks of code. The class learns that higher level or more abstract steps make it easier to understand and reason about steps, then begins to create functions in Game Lab. At the end of the lesson the class uses these skills to organize and add functionality to the final version of their side scroller game.

*Live Coding: Students build their own functions.

*Prompt: Why would we say that functions allow us to "create our own blocks?" Why is this something we'd want to do?
Prompt: Write down your own definition of an abstraction? Why would a function count as an abstraction?

Lesson 20: The Game Design Process
This lesson introduces the process the class will use to design games for the remainder of the unit. The class walks through this process in a series of levels. As part of this lesson the class also briefly learned to use multi-frame animations in Game Lab. At the end of the lesson they have an opportunity to make improvements to the game to make it their own.

*Live Coding: Students fix broken code in Game Lab.

*Prompt: Today, you used a filled-out project guide as you completed your program.
    ● How did the project guide help you as you coded?
    ● What do you think will be important to remember when you fill out your own project guide?

Lesson 21: Using the Game Design Process
In this multi-day lesson, the class uses the problem solving process from Unit 1 to create a platform jumper game. After looking at a sample game, the class defines what their games will look like and uses a structured process to build them. Finally, the class reflects on how the games could be improved, and implements those changes.

*Live Coding: Students create code to build a platform jumping game.
*Journal
Prompt: Have students reflect on their development of the five practices of CS Discoveries (Problem Solving, Persistence, Creativity, Collaboration, Communication). Choose one of the following prompts as you deem appropriate.

- Choose one of the five practices in which you believe you demonstrated growth in this lesson. Write something you did that exemplified this practice.
- Choose one practice you think you can continue to grow in. What’s one thing you’d like to do better?
- Choose one practice you thought was especially important for the activity we completed today. What made it so important?

Lesson 22: Project - Design a Game
The class plans and builds original games using the project guide from the previous two lessons. Working individually or in pairs, the class plans, develops, and gives feedback on the games. After incorporating the peer feedback, the class shares out the completed games.

*Live Coding: Students build their video game. This is a multi-day project and their summative assessment for understanding of coding elements.

*Share: Give students a chance to share their games.
## Interactive Card - Grading Assessment

<table>
<thead>
<tr>
<th>Standard</th>
<th>1 - Little/No Evidence</th>
<th>2 - Emerging</th>
<th>3 - Convincing</th>
<th>4 - Mastery</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-AP-11 - Create clearly named variables that represent different data types and perform operations on their values.</td>
<td>No variables are present</td>
<td>One variable is used</td>
<td>At least one variable is used and it updated through the program</td>
<td>Multiple variables are used and are updated throughout the program</td>
</tr>
<tr>
<td>2-AP-13 - Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.</td>
<td>The draw loop is not present.</td>
<td></td>
<td></td>
<td>Program is well sequenced and properly separates code in and out of the draw loop</td>
</tr>
<tr>
<td>2-AP-15 - Seek and incorporate feedback from team members and users to refine a solution that meets user needs.</td>
<td>Did not give feedback to peers.</td>
<td>Gave some feedback to peers</td>
<td>Gave and responded to peers.</td>
<td>Gave thoughtful feedback to peers and responded to peer feedback by making appropriate changes to program.</td>
</tr>
</tbody>
</table>
## Game - Grading Assessment

<table>
<thead>
<tr>
<th>Standard</th>
<th>1 - Little/No Evidence</th>
<th>2 - Emerging</th>
<th>3 - Convincing</th>
<th>4 - Mastery</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-AP-12 - Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.</td>
<td>Game has no loops</td>
<td></td>
<td></td>
<td>Game has multiple loops</td>
</tr>
<tr>
<td>2-AP-15 - Seek and incorporate feedback from team members and users to refine a solution that meets user needs.</td>
<td>Did not give feedback to peers.</td>
<td>Gave some feedback to peers</td>
<td>Gave and responded to peers.</td>
<td>Gave thoughtful feedback to peers and responded to peer feedback by making appropriate changes to program.</td>
</tr>
<tr>
<td>2-AP-11 - Create clearly named variables that represent different data types and perform operations on their values.</td>
<td>No variables are present</td>
<td>One variable is used</td>
<td>At least one variable is used and it updated through the program</td>
<td>Multiple variables are used and are updated throughout the program</td>
</tr>
<tr>
<td>2-AP-13 - Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.</td>
<td>The game includes no conditionals.</td>
<td>The game responds to user input through a conditional.</td>
<td>The game includes at least one type of sprite interaction and responds to user input.</td>
<td>The game includes multiple different interactions between sprites, responds to multiple types of user input (e.g. different arrow keys).</td>
</tr>
</tbody>
</table>