Blackbird Code
Games and Animations Curriculum
Standards Alignment

In this curriculum, composed of 10 Stages created for grades 6-12, students engage in learning experiences that address standards and practices identified in the Next Generation Science Standards (NGSS), the Common Core State Standards (CCSS) for Math and English as well as the Computer Science Teachers Association (CSTA) Computer Science standards. Standards shared in common across all stages are described below. Several examples of learning activities that address these standards are identified and annotated. Particular CCSS Mathematics standards are identified as well.

Common Core State Standards for Mathematics

- Mathematical Practices addressed throughout curriculum
    - *CCSS.MATH.PRACTICE.MP1: Make sense of problems and persevere in solving them.*
        - Students asked to position shapes precisely. Use examples, documentation, and trial and error to persevere.
    - *CCSS.MATH.PRACTICE.MP2: Reason abstractly and quantitatively.*
        - Students learn how to apply mathematical operators to lines of code to solve arithmetic problems. Later students learn to create programs that solve for complex calculations like compound interest.
    - *CCSS.MATH.PRACTICE.MP4: Model with mathematics.*
        - Students learn how computer programs can help them with repeatable tasks, like generating random numbers. Later students learn to model phenomena like the creation of a normal distribution curve, or bell curve, with a computer program.
    - *CCSS.MATH.PRACTICE.MP5: Use appropriate tools strategically.*
        - Students learn about a variety of number functions built-in to the programming language they learn. Students are taught to use these functions to do things like round numbers, convert data into different data types, use exponential functions, or find the square root of a number. Students are provided examples and rationale for using these tools in the context of a computer program.
    - *CCSS.MATH.PRACTICE.MP6: Attend to precision.*
        - Writing lines of computer code requires intense attention to precision. Our tool helps students identify when their code isn't quite right in a friendly, and immediate, way. Students later learn that the sequence of their lines of code matter in the context of an entire computer program.
    - *CCSS.MATH.PRACTICE.MP7: Look for and make use of structure.*

        - Students learn the pattern of single lines of code and later the structure required to compose whole computer programs.

    - *CCSS.MATH.PRACTICE.MP8: Look for and express regularity in repeated reasoning.*
        - Students are provided with models of how to structure programs and solve problems within them. Later students learn more elegant ways of solving those same problems using more advanced programming techniques.

- Mathematical Content Standards
    - Ratios and Proportional Relationships
        - *CCSS.MATH.CONTENT.6.RP.A.1: Understand the concept of a ratio and use ratio language to describe a ratio relationship between two quantities.*
        - *CCSS.MATH.CONTENT.7.RP.A.1: Compute unit rates associated with ratios of fractions, including ratios of lengths, areas and other quantities measured in like or different units.*
            - Example lessons: 1.4, 4.6, 5.9

        - *CCSS.MATH.CONTENT.6.RP.A.2: Understand the concept of a unit rate a/b associated with a ratio a:b with b ≠ 0, and use rate language in the context of a ratio relationship.*
        - *CCSS.MATH.CONTENT.7.RP.A.2: Recognize and represent proportional relationships between quantities.*
            - Example lessons: 1.4, 5.1, 6.4, 9.2

        - *CCSS.MATH.CONTENT.6.RP.A.3.D: Use ratio reasoning to convert measurement units; manipulate and transform units appropriately when multiplying or dividing quantities.*
        - *CCSS.MATH.CONTENT.7.RP.A.3: Use proportional relationships to solve multistep ratio and percent problems.*
            - Example lessons: 1.4, 6.5, 6.7, 8.7

    - The Number System
        - *CCSS.MATH.CONTENT.6.NS.A.1: Interpret and compute quotients of fractions, and solve word problems involving division of fractions by fractions, e.g., by using visual fraction models and equations to represent the problem.*
        - *CCSS.MATH.CONTENT.7.NS.A.1: Apply and extend previous understandings of addition and subtraction to add and subtract rational numbers; represent addition and subtraction on a horizontal or vertical number line diagram.*
            - Example lessons: 3.3, 4.5, 5.3

        - *CCSS.MATH.CONTENT.6.NS.C.6: Understand a rational number as a point on the number line. Extend number line diagrams and coordinate axes familiar from previous grades to represent points on the line and in the plane with negative number coordinates.*
            - Example lessons: 1.1, 1.3, 3.3

        - *CCSS.MATH.CONTENT.7.NS.A.2.A: Understand that multiplication is extended from fractions to rational numbers by requiring that operations continue to satisfy the properties of operations, particularly the distributive property, leading to products such as (-1)(-1) = 1 and the rules for multiplying signed numbers. Interpret products of rational numbers by describing real-world contexts.*

            - Example lessons: 5.1, 9.5

    - Expressions and Equations
        - *CCSS.MATH.CONTENT.6.EE.A.2: Write, read, and evaluate expressions in which letters stand for numbers.*
            - Example lessons: 1.1, 2.2, 3.3, 4.1, 4.2,

        - *CCSS.MATH.CONTENT.7.EE.B.4: Use variables to represent quantities in a real-world or mathematical problem, and construct simple equations and inequalities to solve problems by reasoning about the quantities.*
            - Example lessons: 4.4, 5.1, 5.5

        - *CCSS.MATH.CONTENT.6.EE.B.5: Understand solving an equation or inequality as a process of answering a question: which values from a specified set, if any, make the equation or inequality true? Use substitution to determine whether a given number in a specified set makes an equation or inequality true.*
            - Example lessons: 3.4, 3.5, 7.1, 7.2, 7.3

    - Geometry
        - *CCSS.MATH.CONTENT.7.G.A.1: Solve problems involving scale drawings of geometric figures, including computing actual lengths and areas from a scale drawing and reproducing a scale drawing at a different scale.*
            - Example lessons: 2.1, 2.3, 4.4

        - *CCSS.MATH.CONTENT.7.G.B.4: Know the formulas for the area and circumference of a circle and use them to solve problems; give an informal derivation of the relationship between the circumference and area of a circle.*
            - Example lessons: 4.4, Guided Project - Area calculator 2

    - Statistics and Probability
        - *CCSS.MATH.CONTENT.7.SP.C.6: Approximate the probability of a chance event by collecting data on the chance process that produces it and observing its long-run relative frequency, and predict the approximate relative frequency given the probability.*
            - Example lessons: 7.C, Guided Project - Bell curve

        - *CCSS.MATH.CONTENT.7.SP.C.8: Find probabilities of compound events using organized lists, tables, tree diagrams, and simulation.*
            - Example lessons: 6.2, 7.2, 7.C, 8.7

Next Generation Science Standards

- Science and Engineering Practices
    - Practice 1 - Defining Problems
        - *Middle School 6-8: Define a design problem that can be solved through the development of a computer program.*
        - Example Lessons: 1.3, 2.1

            - Students learn how to place lines on the canvas and then how to arrange them into a square - a time consuming process. They are later introduced to the Rectangle object - a better tool for creating squares on the canvas.
        - Example Lessons: 4.1, 4.3
            - Students first learn to define simple variables, which result in long lists of seemingly unrelated information. Students then learn they can organize this data within objects.

- Practice 2 - Developing and Using Models
    - *Middle School 6-8: Evaluate limitations of a model for a proposed tool.*
    - Example Lessons: 5.1, 9.5
        - Students learn to create an animation of a single shape bouncing off the edges of their computer screen. In order to add a second object to their program, using techniques they've learned so far, they would have to nearly double the length of the program. In 9.5 students use new techniques they've learned to add as many shapes as they like to their program in a much more effective way.

- Practice 4 - Planning and Carrying Out Investigations
    - *Middle School 6-8: Analyze and interpret data to provide evidence for phenomena.*
    - Example Lessons: 3.1, 3.3, 3.4
        - In this lesson sequence, students are introduced to their first *run errors*. These errors produce messages that students must interpret in order to fix the code that is causing the error.

- Practice 5 - Using Mathematics and Computational Thinking
    - *Middle School 6-8: Create algorithms (a series of ordered steps) to solve a problem.*
    - Example Lessons: 1.1-1.5
        - The first lesson sequence in the curriculum sets the pattern for the rest of the Games and Animations curriculum. Each lesson in this curriculum introduces learners to the construction of algorithms that computers are able to interpret. Along the way they are shown why the sequence of those steps matter.
    - *Middle School 6-8: Apply mathematical concepts and/or processes (e.g., ratio, rate, percent, basic operations, simple algebra) to scientific and engineering questions and problems.*
    - Example Lessons: 4.4, 6.7, 7.8, 8.7
        - Students learn to store equations in variables to solve complex mathematical problems, starting with calculating the area of a rectangle. In 6.7 students calculate the effects of compound interest. Later students learn how to use operators and logic statements in their programs to determine if a number is evenly divisible by another. Eventually students use their programs to determine the mean of a list of randomly generated numbers.

- Practice 6 - Constructing Explanations and Designing Solutions
    - *Middle School 6-8: Undertake a design project, engaging in the design cycle, to construct and/or implement a solution that meets specific design criteria and constraints.*
    - Examples: Guided Projects
        - At the end of each stage students are led to a menu of Guided Projects in the Workshop. These projects provide students with a target program to create. These

projects are able to be reviewed by their instructors to ensure that they meet the specified criteria.

- Cross Cutting Concepts
    - Patterns
        - Examples of lessons and projects that address this cross cutting concept include: Stage 1, All Guided Projects
        - Students engage in the practice of constructing lines of computer code, which follows a very particular pattern. For example, the structure of a variable declaration in the programming language used in Blackbird Code, and any computer language, follows the following pattern:
            - [variable keyword] [variable name] [assignment operator] [value]
        - On a broader scale, students engage in the construction of entire computer programs, which also require students to learn and use the following pattern:
            - [data to be used in a program] --> [instructions for how to use that data]

    - Cause and Effect
        - Examples of lessons and projects that address this cross cutting concept include: Lessons 3.2-3.4, 6.1, 8.4
        - As students start writing code they quickly start to see the effect their instructions have on the outcome of a program. Our curriculum requires students to run the code that they write regularly in order to make progress.
        - Making changes, or additions, to computer programs often as unintended effects. Understanding which changes caused the effect is a process known as debugging.

    - Scale, Proportion and Quantity
        - Examples of lessons and projects that address this cross cutting concept include: 1.4, 2.4, 4.1, 5.1, 6.1, 6.10
        - Students learn how to create computer programs that give instructions to a graphical space known as the canvas. The canvas is defined by a width and a height described in pixels. Throughout the curriculum students learn how to use these properties in their code.
        - For example, students learn to place a Point (a small circle) in the *exact center* of the canvas by dividing the width and height of the canvas by two:
            - p.x = canvas.width / 2;
            - p.y = canvas.height / 2;

Common Core State Standards for English

- Science and Technical Subjects
    - *CCSS.ELA-LITERACY.RST.6-8.3: Follow precisely a multistep procedure when carrying out experiments, taking measurements, or performing technical tasks.*

    - *CCSS.ELA-LITERACY.RST.6-8.4: Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant to grades 6-8 texts and topics.*

- *CCSS.ELA-LITERACY.RST.6-8.7: Integrate quantitative or technical information expressed in words in a text with a version of that information expressed visually (e.g., in a flowchart, diagram, model, graph, or table).*

CSTA Computer Science Standards

Blackbird Code addresses the CS standards listed below iteratively throughout the curriculum. The place where these content standards are first addressed are listed below with commentary.

- Algorithms and Programming
  - *2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.*
    - First introduced: Lesson 1.1
    - Students are provided models of variable naming conventions throughout the Games and Animations curriculum.
  - *2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.*
    - First introduced: 1.2
    - Our lessons demonstrate the iterative nature of programming by adding and refining portions of code written in previous lessons.
  - *2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.*
    - First introduced: 3.2
    - 'Step tasks' guide learners, line-by-line, through complex portions of code to demonstrate how the program works, and to help uncover problems in the code.
  - *2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.*
    - First introduced: 1.1
    - Students master the use of a limited set of functions, built-in objects and data structures so they are able to employ those skills in work of their own.
  - *2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.*
    - First introduced: Stage 1 Guided Projects
    - All work completed in the Workshop on Blackbird School is viewable by instructors who have the ability to leave detailed code review. Projects are also viewable by friends and classmates on the platform to encourage collaboration and refinement of programs.
  - *2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.*
    - First introduced: 1.1
    - Blackbird Code features a small set of built-in objects and functions that students use to create animations and games. They develop models of these programs in lessons and practice their skills in the Workshop. Examples of these built-in objects include shape, image and text objects, the animate function, and the collide function.

- *2-AP-17: Systematically test and refine programs using a range of test cases.*
  - First introduced: 3.3
  - Lessons in the curriculum deliberately include bugs and highlight how a programmer can solve these issues in their code. With the help of step mode, a linter, and helpful error messages, students are able to uncover issues present in their code. Students are able to quickly run and test changes they make to their code.

- *2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.*
  - First introduced: Stage 1 Guided Projects
  - The Workshop in Blackbird School allows students to easily access the work of their collaborators. Guided projects in the workshop provide opportunities for students to work on a common task with their classmates. Teachers are able to monitor this work and review code - all within the same application.

- *2-AP-19: Document programs in order to make them easier to follow, test, and debug.*
  - First introduced: 1.1
  - Lessons in the Games and Animation curriculum are all completely commented, provided ample models of proper documentation of code. With Code Review, teachers are able to monitor and reinforce proper documentation practices with their students.