ROBOT DRONE LEAGUE
2024 Challenge: MINESHAFT

Standards Alignment with the South Carolina Computer Science Standards

RDL Introduction

Creativity and innovation are key elements to advancing the fields of science, technology, engineering, and mathematics (STEM) into the future. Robot Drone League (RDL) has been designed to provide students with open-ended challenges that allow for creation and innovation by engaging in hands-on design, engineering, and programming of interactive robots and drones. Students are presented with the opportunity to develop real-world connections to classroom learning. Working with robots in a collaborative game format can be a very powerful tool to engage students and enhance math and science skills through hands-on, student-centered learning. Through participation in RDL, students can develop the essential life skills of teamwork and collaboration, as well as critical thinking, project management, and communication required to become the next generation of innovators and problem-solvers in our global society.

The South Carolina Computer Science and Digital Literacy Process Standards should be integrated into every grade level within the South Carolina Computer Science and Digital Literacy Content Standards. Because the Process Standards drive the pedagogical component of teaching and serve as the means by which students should demonstrate understanding of the content standards, the process standards must be incorporated as an integral part of overall student expectations when assessing content understanding.

A computer science literate student can:
1. Foster an inclusive computing culture.
a. Recognize that equitable access to computing benefits society as a whole.
b. Consider others' perspectives as well as one's own perspective when developing computational solutions.
c. Consider the needs of a variety of end users regarding accessibility and usability.
2. Collaborate around computing.
a. Select appropriate technological tools that can be used to collaborate on a project.
b. Collaborate productively with individuals of varying perspectives, skills, and backgrounds.
c. Set and implement equitable expectations and workloads when working in teams.
d. Integrate constructive feedback while working in teams.
3. Recognize, define, and analyze computational problems.
a. Recognize when it is appropriate to solve a problem computationally.
b. Make sense of computational problems and persevere in solving them.
c. Relate computational problems to prior knowledge.
d. Recognize that there may be multiple approaches to solving a problem.
e. Approach problem solving iteratively, using a cyclical process.
4. Create, test, and refine computational artifacts.
a. Consider the purpose of computational artifacts for practical use, personal expression, and/or

societal impact.
b. Recognize when to use the same solution for multiple problems.
c. Test computational artifacts systematically by considering multiple scenarios and using test cases.
d. Approach troubleshooting systematically.
e. Consider performance, reliability, usability, and accessibility when evaluating and refining computational artifacts.
5. Communicate about computing.
a. Select and use appropriate technological tools to convey solutions to computing problems.
b. Communicate about computational processes and solutions using appropriate terminology consistent with the intended audience and purpose.
c. Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.

Grade 6:
Computing Systems
Standard 1: Analyze the use of computing to solve relevant problems.
The student will:
6.CS.1.1 Identify and describe the key functional components (e.g., input devices, output devices, processor, operating system, software applications, memory, storage) of a computer.
6.CS.1.2 Identify relevant problems and how they are solved using computer science and various types of computing devices(e.g., directions to a location can be obtained through Global Position Systems (GPS) and/or online maps).
Standard 2: Examine how computing devices function.
The student will:
6.CS.2.1 Understand various ways software is acquired and installed.
Standard 3: Evaluate various solutions to common hardware and software problems.
The student will:
6.CS.3.1 Identify the source of a problem using a systematic process (i.e., troubleshooting).

Algorithms and Programming
Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich;
steps to a popular dance; steps for sending an email).
The student will:
6.AP.1.1 Recognize that there are multiple ways to sequence instructions that can lead to the same result.
6.AP.1.2 Interpret pseudocode and flowcharts.
Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
The student will:
6.AP.2.1 Select appropriate coding control structures to skip or repeat instructions.
Standard 3: Decompose problems into subproblems and write code to solve the subproblems

(i.e., break down a problem into smaller parts).
The student will:
6.AP.3.1 Discuss the parts of a program (e.g., components of creating a video game include keeping score, determining winners/losers, moving characters, designing game art, and advancing levels).
Standard 4: Design and code programs to solve problems.
The student will:
6.AP.4.1 Use a beginner coding language (e.g., drag-and-drop, block-based) to design and code a simple program that solves a problem.
Standard 5: Identify variables and compare the types of data stored as variables.
The student will:
6.AP.5.1 Recognize variables that represent information (e.g., age, first name).
6.AP.5.2 Recognize variables can represent different types of data (e.g., numbers, words, colors, images).

Grade 7:
Computing Systems
Standard 1: Analyze the use of computing to solve relevant problems.
The student will:
7.CS.1.1 Explore an expanded definition of computing devices (e.g., "Internet of Things," wearable technology, robotics).
7.CS.1.2 Analyze relevant problems and how they are solved using computer science and various types of computing devices (e.g., Global Positioning System (GPS) and online maps provide guided step-by-step directions to locations).
Standard 2: Examine how computing devices function.
The student will:
7.CS.2.1 Describe processing speed and storage capacity using standard units of measure (e.g., 3 TB hard drive, 256 GB cell phone, 3.8 GHz processor).
Standard 3: Evaluate various solutions to common hardware and software problems.
The student will:
7.CS.3.1 Understand and communicate solutions to various computing problems (e.g., computing device is frozen; webpage does not load; application does not launch; keyboard does not work).
7.CS.3.2 Understand how rebooting a computing device can solve problems.

Algorithms and Programming
Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email).
The student will:
7.AP.1.1 Write sequences of instructions for others to perform tasks.
7.AP.1.2 Suggest changes to the sequence of instructions that can lead to the same result (e.g., explore different ways to tying shoes).
7.AP.1.3 Write clear instructions using pseudocode.
Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).

The student will:
7.AP.2.1 Write code using control structures to skip or repeat instructions.
Standard 3: Decompose problems into subproblems and write code to solve the subproblems (i.e., break down a problem into smaller parts).
The student will:
7.AP.3.1 Decompose a problem into smaller parts.
7.AP.3.2 Identify the parts of a program (e.g., components of creating a video game include keeping score, determining winners/losers, moving characters, designing game art, and advancing level).
Standard 4: Design and code programs to solve problems.
The student will:
7.AP.4.1 Use a beginner coding language (e.g., drag-and-drop, block-based) to design and code a moderately complex program that solves a problem.
Standard 5: Identify variables and compare the types of data stored as variables.
The student will:
7.AP.5.1 Identify variables as a representation for information.
7.AP.5.2 Discuss the differences between the types of data (e.g., characters, integers, decimals).

Grade 8:
Computing Systems
Standard 1: Analyze the use of computing to solve relevant problems.
The student will:
8.CS.1.1 Compare and contrast relevant problems and how they are solved using computer science and various types of computing devices (e.g., Global Positioning System (GPS) and online maps include different features, including real-time traffic, satellite images, construction and accident notifications).
Standard 2: Examine how computing devices function.
The student will:
8.CS.2.1 Understand that computers receive and process data as a series of on and off signals (i.e., binary data).
8.CS.2.2 Determine appropriate hardware, operating systems, and software based upon the needs of users in various career fields (e.g., computing devices used by professional video producers and students differ).
Standard 3: Evaluate various solutions to common hardware and software problems.
The student will:
8.CS.3.1 Understand computer hardware and software compatibility (e.g., applications designed for Android devices cannot run on iOS devices).
8.CS.3.2 Identify appropriate resources for troubleshooting hardware and software problems (e.g., user manuals, online searches, technology support services).

Algorithms and Programming
Standard 1: Design, evaluate, and modify simple algorithms (e.g., steps to make a sandwich; steps to a popular dance; steps for sending an email).

The student will:
8.AP.1.1 Modify a sequence of instructions to solve problems.
8.AP.1.2 Make changes to the sequence of instructions that can lead to the same result.
8.AP.1.3 Write clear instructions using flowcharts.
Standard 2: Use and compare simple coding control structures (e.g., if-then, loops).
The student will:
8.AP.2.1 Modify an algorithm using conditionals and iteration.
Standard 3: Decompose problems into subproblems and write code to solve the subproblems
(i.e., break down a problem into smaller parts).
The student will:
8.AP.3.1 Decompose a problem into functional parts.
8.AP.3.2 Compose a program with multiple parts.
Standard 4: Design and code programs to solve problems.
The student will:
8.AP.4.1 Use a beginner coding language (e.g., drag-and-drop, block-based) to design and code a complex program that solves a problem.
Standard 5: Identify variables and compare the types of data stored as variables.
The student will:
8.AP.5.1 Compare and contrast variables that change or are constant.
8.AP.5.2 Identify the variables needed to solve a given problem (i.e., information that needs to be tracked).

High School
South Carolina Computer Science High School Process Standards
The South Carolina Computer Science High School Process Standards should be integrated into every level within the South Carolina Computer Science High School Content Standards. Because the Process Standards drive the pedagogical component of teaching and serve as the means by which students should demonstrate understanding of the content standards, the process standards must be incorporated as an integral part of overall student expectations when assessing content understanding.
A computer science literate student can:
1. Foster an inclusive computing culture.
a. Recognize that equitable access to computing benefits society as a whole.
b. Consider others' perspectives as well as one's own perspective when developing computational solutions.
c. Consider the needs of a variety of end users regarding accessibility and usability.
2. Collaborate around computing.
a. Select appropriate technological tools that can be used to collaborate on a project.
b. Collaborate productively with individuals of varying perspectives, skills, and backgrounds.
c. Set and implement equitable expectations and workloads when working in teams.
d. Integrate constructive feedback while working in teams.
3. Recognize, define, and analyze computational problems.
a. Recognize when it is appropriate to solve a problem computationally.

b. Make sense of computational problems and persevere in solving them.
c. Relate computational problems to prior knowledge.
d. Recognize that there may be multiple approaches to solving a problem.
e. Approach problem solving iteratively, using a cyclical process.
4. Create, test, and refine computational artifacts.
a. Consider the purpose of computational artifacts for practical use, personal expression, and/or societal impact.
b. Recognize when to use the same solution for multiple problems.
c. Test computational artifacts systematically by considering multiple scenarios and using test cases.
d. Approach troubleshooting systematically.
e. Consider performance, reliability, usability, and accessibility when evaluating and refining computational artifacts.
5. Communicate about computing.
a. Select and use appropriate technological tools to convey solutions to computing problems.
b. Communicate about computational processes and solutions using appropriate terminology consistent with the intended
audience and purpose.
c. Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.

Computing Systems
Standard 1: Examine how hardware and software contribute to computing devices solving relevant problems.
Level 4: HS4.CS.1.1 Develop a solution to a given problem using appropriate hardware and software (e.g., sensor devices, Wi-Fi capabilities,specialized displays, runtime modules, operating systems, application programming interfaces (APIs)).

Algorithms and Programming
Standard 1: Design algorithms that can be adapted to express an idea or solve a problem.
HS4.AP.1.1Evaluate algorithms in terms of efficiency, correctness, and clarity (CSTA, 2017).

Standard 2: Build a combination of control structures that supports complex execution, readability, and program performance.
Level 1: HS1.AP.2.1 Trace the flow of execution of a program that uses a combination of control structures (e.g., conditionals, loops, event handlers, recursion).
Level 2: HS2.AP.2.1 Design and iteratively develop programs that combine control structures (e.g., conditionals, loops, event handlers, recursion).
Level 3: HS3.AP.2.1 Justify the selection of specific control structures explaining the benefits and drawbacks of choices made (e.g., tradeoffs involving implementation, readability, and program performance).

Standard 3: Divide a task into sets of functional units that can be reused to compose a complex solution.
Level 1: HS1.AP.3.1 Decompose tasks into smaller, reusable parts to facilitate the design, implementation, and review of programs.
Level 2: HS2.AP.3.1 Develop code to solve the smaller parts of a decomposed task that can be reused to solve similar problems (e.g., procedures, functions, objects).
Level 3: HS3.AP.3.1 Build a complex solution to a problem that incorporates reusable code (e.g., student created, application programming interfaces (APIs), libraries).
Level 4: HS4.AP.3.1 Justify the selection of modular parts in the creation of a complex solution.

Standard 4: Plan, build, test, refine, and document programs using text-based coding languages to solve problems with varying degrees of difficulty.
Level 1: HS1.AP.4.1 Plan and develop programs for a variety of audiences using a process that incorporates development, feedback, and revision.
Level 2: HS2.AP.4.1 Plan and develop a program that addresses potential security issues.
Level 3: HS3.AP.4.1 Plan and develop a program that is accessible across multiple computing platforms (e.g., iOS, Unix, Windows, web-based).
Level 4: HS4.AP.4.1 Evaluate a program through a review process (e.g., code review, beta testing, pilot group).

Level 1: HS1.AP.4.2 Seek and incorporate feedback to refine a solution (e.g., users, team members, code review, teachers).
Level 2: HS2.AP.4.2 Systematically test programs using a range of test cases to meet design specifications (e.g., specific outcomes, functionality, user interface, error handling) (CSTA, 2017).
Level 3: HS3.AP.4.2 Evaluate and refine programs to make them more usable, functional, and accessible.
Level 4: HS4.AP.4.2 Implement version control to track program refinements.

Level 1: HS1.AP.4.3 Recognize the variety of documentation methods available while developing a program (e.g., inline comments, procedure header, purposeful naming).
Level 2: HS2.AP.4.3 Document programs in order to make them easier to follow, test, and debug.
Level 3: HS3.AP.4.3 Document programs that use non-user-created resources (e.g., code, media, libraries) giving attribution to the original creator.
Level 4: HS4.AP.4.3 Justify design decisions by documenting the design process of complex programs (e.g., developer journal, digital portfolio, presentation).