ROBOT DRONE LEAGUE
2024 Challenge: MINESHAFT

Standards Alignment with the Tennessee Middle Grades 6-8 and High School 9-12 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.

Middle School: Computer Science Standards

MS.FC: Foundational Concepts
1) Analyze the advantages and limitations of existing computing devices to improve user experience.
2) Demonstrate skills in identifying and solving hardware and software problems that can occur during regular usage.
3) Apply computational thinking to a variety of problems across multiple disciplines.
4) Understand how collaboration is essential to computer science and apply collaborative skills to develop computational solutions.

MS.AT: Algorithmic Thinking
1) Use clearly named variables of various data types to create generalized algorithms.
2) Create algorithms which include methods of controlling the flow of computation using "if…then…else" type conditional statements to perform different operations depending on the values of inputs.
3) Identify algorithms that make use of sequencing, selection, or iteration.
4) Describe how algorithmic processes and automation increase efficiency.

MS.PC: Programming Concepts
1) Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2) Create procedures with parameters that hide the complexity of a task and can be reused to solve similar tasks.

3) Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
4) Provide proper attribution when incorporating existing code, media, and libraries into original programs.
5) Use the iterative design process to systematically test and refine programs to improve performance and eliminate errors.
6) Document programs using comments and/or README files to make them easier to follow, test, and debug.
7) Design a function using a programming language.

High School: Computer Science Standards

CS.AT: Algorithmic Thinking
1) Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
2) Systematically design and develop programs for broad audiences by incorporating feedback from users.
3) Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
4) Use effective communication and accurate computer science terminology to explain problem solving when completing a task.

CS.PC: Programming Concepts
1) Choose and apply an appropriate iterative design process to systematically test and refine a program to increase performance.
2) Develop a plan to manage and assign data values of different types (strings, numeric, character, integer, and date) to a variable
3) Create and refine programs with Boolean conditionals to demonstrate the use of branches and logical operators.
4) Design and develop iterative programs that combine control structures, including nested loops and compound conditionals.
5) Create parameters to organize a program to make it easier to follow, test, and debug.
6) Incorporate existing code, media, and libraries into original programs, and give proper attribution.
7) Debug (identify and fix) errors in an algorithm or program that includes sequences and simple and complex loops following a two-step debugging process.