Glendale Unified School District

Middle School

October 6, 2020

Department:	Career Technical Education
Course Title:	Computer Science Discoveries
Course Code:	5128GV
Grade Level(s):	7-8
School(s) Course Offered:	Rosemont Middle School
UC/CSU Approved (Y/N, Subject):	N/A
Course Credits:	*5*
Recommended Prerequisite:	N/ A
Recommended Textbook:	https://curriculum.code.org/csd-20/
Course Overview:	Computer Science Discoveries (CS Discoveries) is an introductory computer science course that empowers students to create authentic artifacts and engage with computer science as a medium for creativity, communication, problem solving, and fun.

Unit 1: **Problem Solving and Computing** (*4 weeks*)

A. 1B-AP-08 - Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
1B-AP-11 - Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
1B-AP-16 - Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
2-AP-10 - Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-15 - Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
2-AP-17 - Systematically test and refine programs using a range of test cases.
2-AP-18 - Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
1B-CS-01 - Describe how internal and external parts of computing devices function to form a system.
1B-CS-02 - Model how computer hardware and software work together as a system to accomplish tasks.
2-CS-02 - Design projects that combine hardware and software components to collect and exchange data.

B. Unit 1 introduces core practices and frameworks that students will use throughout the course. By the end of the unit, students should be able to identify the defining characteristics of a computer and how it is used to solve information problems. They should be able to use a structured problem solving process to address problems and design solutions that use computing technology. Students learn how computers input, output, store, and process information to help humans solve problems within the context of apps.

C. The Problem Solving and Computing unit is a highly interactive and collaborative introduction to the field of computer science, as framed within the broader pursuit of solving problems. Through a series of puzzles, challenges, and real world scenarios, students are introduced to a problem solving process that they will return to repeatedly throughout the course. The unit concludes with students designing an app that helps solve a problem of their choosing.

Unit 2: **Web Development** (*4 weeks*)

A. 1B-IC-20 - Seek diverse perspectives for the purpose of improving computational artifacts.
1B-IC-21 - Use public domain or creative commons media and refrain from copying or using material created by others without permission.
2-IC-20 - Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
2-IC-22 - Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
2-IC-23 - Describe tradeoffs between allowing information to be public and keeping information private and secure.
1B-NI-05 - Discuss real-world cybersecurity problems and how personal information can be protected.
1B-AP-11 - Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
1B-AP-12 - Modify, remix or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
1B-AP-14 - Observe intellectual property rights and give appropriate attribution when creating or remixing programs.
1B-AP-15 - Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
1B-AP-16 - Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation and review stages of program development.
2-AP-13 - Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-15 - Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
2-AP-16 - Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 - Systematically test and refine programs using a range of test cases.
2-AP-18 - Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
2-AP-19 - Document programs in order to make them easier to follow, test, and debug.
3A-AP-20 - Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.

B. Unit 2 introduces computer languages and how students can use these languages to create digital artifacts. By the end of the unit, students should be able to create a digital artifact that uses multiple computer languages to control the structure and style of their content. They should understand that different languages allow them to solve different problems, and that these solutions can be generalized across similar problems. They are also introduced to problem solving as it relates to programming, as they learn valuable skills such as debugging, using resources, and teamwork. Lastly, they should understand their responsibilities as both creators and consumers of digital media.

C. In the Web Development unit, students are empowered to create and share the content on their own web pages. They begin by thinking about the role of the web, and how it can be used as a medium for creative expression before creating their own personal web pages. As students develop their pages and begin to see themselves as programmers, they are encouraged think critically about the impact of sharing information online and how to be more critical content consumers. At the conclusion of the unit, students work together to create a website to address a problem.

Unit 3: **Interactive Animations and Games** (*6 weeks*)

A. 2-IC-21 - Discuss issues of bias and accessibility in the design of existing technologies.
2-AP-10 - Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-11 - Create clearly named variables that represent different data types and perform operations on their values.
2-AP-12 - Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
2-AP-13 - Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-16 - Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 - Systematically test and refine programs using a range of test cases.
2-AP-18 - Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
2-AP-19 - Document programs in order to make them easier to follow, test, and debug.

B. Unit 3 focuses on algorithms and programming. By the end of the unit, 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 task 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. Along the way, they practice design, testing, and iteration, as they come to see that failure and debugging are an expected and valuable part of the programming process.

C. In the Animations and Games unit, students build on their coding experience as they create programmatic images, animations, interactive art, and games. Starting off with simple, primitive shapes and building up to more sophisticated sprite-based games, students become familiar with the programming concepts and the design process computer scientists use daily. They then learn how these simpler constructs can be combined to create more complex programs. In the final project, students develop a personalized, interactive program.

Unit 4: **The Design Process** (*6 weeks*)

A. 2-CS-01 - Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
2-CS-02 - Design projects that combine hardware and software components to collect and exchange data.
2-IC-20 - Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
2-IC-21 - Discuss issues of bias and accessibility in the design of existing technologies.
2-IC-22 - Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
2-AP-10 - Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-13 - Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-14 - Create procedures with parameters to organize code and make it easier to reuse.
2-AP-15 - Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
2-AP-16 - Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17 - Systematically test and refine programs using a range of test cases.
2-AP-18 - Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
2-AP-19 - Document programs in order to make them easier to follow, test, and debug.
2-DA-08 - Collect data using computational tools and transform the data to make it more useful and reliable.

2-DA-09 - Refine computational models based on the data they have generated.

B. Unit 4 extends the problem solving process to incorporate user centered design and software development. By the end of the unit, students should see the design process as a form of problem solving that prioritizes the needs of a user. They should be able to identify user needs and assess how well different designs address them. In particular they know how to develop a paper and digital prototypes, gather and respond to feedback about a prototype, and consider ways different user interfaces do or do not affect the usability of their apps. Students should leave the unit with a basic understand of other roles in software development, such as product management, marketing, design, and testing, and to use what they have learned as a tool for social impact.

C. The Design Process unit transitions students from thinking about computer science as a tool to solve their own problems towards considering the broader social impacts of computing. Through a series of design challenges, students are asked to consider and understand the needs of others while developing a solution to a problem. The second half of the unit consists of an iterative team project, during which students have the opportunity to identify a need that they care about, prototype solutions both on paper and in App Lab, and test their solutions with real users to get feedback and drive further iteration.