Scratch Unit

Content Area: **Math**
Course(s): **Introduction to Computer Science**
Time Period: **Marking Period 1**
Length: **8 weeks**
Status: **Published**

Unit Overview

This unit introduces students to programming through the Scratch programming language. Scratch allows students to quickly develop fully functional programs quickly without all the deatails required in normal languages. This is abstraction and is an important part of computer science. Abstraction is hiding details from the user, ao students are using instruction blocks to making things hapen on the screen without exactly understanding how the block works from a coding standpoint. Students will complete 8 programs which are based around game design. All of the major topics that will be intorduced later in the year will be explored here.

Project 1: Racing Game: At-A-Glance

During this class, students create a two-player racing game in which players control movement with the keyboard.

Topics introduced:

- Events
- Movement
- "Repeat" blocks

Project 2: Maze Game: At-A-Glance

In this activity, students create a game in which the player guides a sprite through a maze.

Topics introduced

- If-then statements (control flow)
- Boolean blocks (conditions)

Project 3: Platform Game: At-A-Glance

In this activity, students create and learn about platform games. Students program a player sprite to move and jump across platforms when the arrow keys are pressed.

Topics introduced

- If-then statements
- Platform games

Project 4: Escape Game: At-A-Glance

In this activity, students create an escape game in which a player must avoid other sprites that move randomly. Users increase their score by avoiding these sprites.

Topics introduced

- Variables
- Randomness

Project 5: Launcher Game: At-A-Glance

In this activity, students create a launcher game using key press events, clones, and variables. In this game, a player must launch and navigate a sprite across a screen of bouncing enemies.

Topics introduced

- Cloning
- Increasing game difficulty

Project 6: Quest Game: At-A-Glance

In this activity, students learn how to use storytelling in video game design while building an RPG style Quest Game.

Topics introduced

- Broadcast (events)

Project 7: Cave Surfing Game: At-A-Glance

In this activity, students create a game with a side scrolling background (similar to the popular game Flappy Bird). In this game, the player sprite moves up and down to avoid obstacles. This class ends with a certificate of recognition
and a closing reflective discussion.

Topics introduced

- If-else statements
- Scrolling backgrounds

Project 8: Student led project

In this activity students will work in pairs to exhibit the skills they have learned in a unit project by their own design.

Enduring Understandings

Objectives / Topics Covered

- Commands
- Separating code into different tasks
- Designing blocks of code
- Program entry points
- Control flow
- Looping
- Conditionals
- Top Down Design

Essential Questions

General Questions

- How do you break a bigger task into smaller (more easily programmed) tasks?
- How can smaller tasks be used to solve more complex problems?
- How can your coding be done in a way that another reader understands the process? (Abstraction)
- Can you give specific instructions (algorithms) to solve a specific problem?

Project 1 :

What do events do in computer science?

How do new tools (coding skills) imporve program functionality.

Project 2 :

What do if statements do in computer science?

Project 3 :

How did you interact with your sprite? How did you use math to figure out how to make the player move and fall?

Project 4 :

How did you use randomness in the project?

How did you use variables?

Project 5 :

How did you use events in your project?

How did you use randomness?

What concept did you use to store the player's score?

Project 6 :

How do you control a program with different screens but with the reality being only one screen?

Project 7 :

What do computer scientists do?

Does anyone have any questions for me about what we've worked on in this class or about computer science in general?

New Jersey Student Learning Standards (No CCS)

8.1.8.AP.3: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. Programs use procedures to organize code and hide implementation details. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability.

8.1.8.AP.4: Decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.

8.1.8.AP.6: Refine a solution that meets users' needs by incorporating feedback from team members and users.

8.1.8.AP.7: Design programs, incorporating existing code, media, and libraries, and give attribution.

8.1.8.AP.8: Systematically test and refine programs using a range of test cases and users.

8.1.8.AP.9: Document programs in order to make them easier to follow, test, and debug.

Technology Standards

List specific standards that are relevant
No general statements

TECH.8.1.12	Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaborate and to create and communicate knowledge.
TECH.8.1.12.A.3	Collaborate in online courses, learning communities, social networks or virtual worlds to discuss a resolution to a problem or issue.
TECH.8.1.12.A.CS1	Understand and use technology systems.
TECH.8.1.12.A.CS2	Select and use applications effectively and productively.
TECH.8.1.12.B.2	Apply previous content knowledge by creating and piloting a digital learning game or tutorial.
TECH.8.1.12.B.CS1	Apply existing knowledge to generate new ideas, products, or processes.
TECH.8.1.12.B.CS2	Create original works as a means of personal or group expression.
TECH.8.1.12.C.CS4	Contribute to project teams to produce original works or solve problems.
TECH.8.2.12.C.CS1	The attributes of design.
TECH.8.2.12.E.1	Demonstrate an understanding of the problem-solving capacity of computers in our world.
TECH.8.2.12.E.3	Use a programming language to solve problems or accomplish a task (e.g., robotic functions, website designs, applications, and games).
TECH.8.2.12.E.CS1	Computational thinking and computer programming as tools used in design and engineering.

21st Century Themes/Careers

List specific standards that are relevant

No general statements

CAEP.9.2.12.C.3	Identify transferable career skills and design alternate career plans.
CAEP.9.2.12.C.5	Research career opportunities in the United States and abroad that require knowledge of world languages and diverse cultures.

Instructional Strategies & Learning Activities

Formative Assessments

Summative Assessment

Alternate Assessments