Mission to Mars - Coding Unit (Year 5)

Week 1: Solar System Animation

Your Challenge

Create planets that orbit around the sun using **move on path** blocks and **forever loops**. You'll build a working solar system where planets move at different speeds just like in real space. By the end of this lesson, you should have planets that continuously orbit without stopping, with closer planets moving faster than distant ones.

Learning Objectives

Students will create orbital animations using **path-based movement** and **continuous loops** to demonstrate understanding of object movement and timing.

Computing Concepts

- **Paths:** Objects move along predetermined routes
- **Forever loops:** Create continuous animation
- **Parallel processing:** Multiple objects move simultaneously
- **Animation timing:** Different durations create varied speeds

Your Challenge

Create planets that orbit around the sun using **move on path** blocks and **forever loops**.

Differentiated Challenge Levels

Mild: Sun-Earth-Moon System

- 3 objects with simple circular paths
- **Assessment:** Earth and Moon move smoothly in continuous loops
- **Success Criteria:** Objects follow paths without stopping

Medium: Inner Solar System

- 5-6 planets with varied orbital periods
- **Assessment:** Logical speed progression (closer planets faster)
- **Success Criteria:** Multiple objects with different timing values

Hot: Full Solar System + Rotation

- All 8 planets with realistic speed relationships
- **Assessment:** Complex animations with additional rotation effects
- **Success Criteria:** Planet rotation while orbiting, creative enhancements

Technical Skills

- Object placement and scaling
- Path creation and assignment
- **Forever loop** implementation
- **Parallel execution** of multiple animations

Week 2: Controlling Gravity with Velocity Control and Boolean Logic

Learning Objectives

Students will control object movement using **velocity** and implement **conditional logic** with **boolean variables** to create interactive systems.

Computing Concepts

- **Velocity:** Force and direction that moves objects
- **Event-driven programming:** Button clicks trigger actions
- **Boolean variables:** True/false states for tracking conditions
- **Conditional statements:** If/else logic for decision making

Your Challenge

Control an astronaut in space using **push blocks** and create toggle systems with **boolean logic**. You'll master how to move objects with precise control in all directions, then learn to create smart systems that remember states and respond to clicks. Your astronaut should float realistically in low gravity, and your toggle system should change object behavior and appearance based on true/false logic.

Differentiated Challenge Levels

Foundation Level: Directional Velocity Control

- 6-direction movement system with stop control
- **Assessment:** Successful directional control with appropriate velocity values
- **Success Criteria:** Objects respond to all direction buttons and stop command

Intermediate Level: Enhanced Control Systems

- Multiple objects with varied velocity experiments
- **Assessment:** Understanding of velocity differences and multi-object control
- **Success Criteria:** Demonstrates speed variations and control mechanisms

Advanced Level: Boolean Toggle Systems

- **Boolean variables** with **conditional logic** (if/else statements)
- **Assessment:** Working toggle system with visual feedback and state management
- **Success Criteria:** Object changes behavior based on true/false conditions

Technical Skills

- **Push block** syntax and parameters
- **Boolean variable** creation and modification
- **If/else conditional statements**
- Visual feedback through color changes

Week 3: Control Room Gravity - Lists, Functions, and Collision Physics

Your Challenge

Build an enclosed gravity control room using **functions**, **lists**, and **collision physics**. You'll create a realistic space environment where objects have different weights and bounce differently when they hit walls. Your control room should demonstrate how coding can organize complex behaviors, with functions that affect multiple objects at once and collision boundaries that contain your physics experiments.

Learning Objectives

Students will organize code using **functions** and **lists** while implementing **collision detection** and **physics properties** to create contained environments.

Computing Concepts

- **Lists:** Organize multiple objects (`create empty list`, `add to list`)
- **Functions:** Reusable code blocks for organization
- **For each loops:** Apply actions to all list items
- **Collision detection:** Objects interact with boundaries
- **Physics properties:** Mass, bounciness, and friction effects

Your Challenge

Build an enclosed gravity control room using **functions**, **lists**, and **collision physics**.

Room Setup Requirements

- **Collision-enabled** walls, ceiling, and floor
- Objects with different **mass values** and **bounciness settings**
- **Lists** to organize multiple objects

Differentiated Challenge Levels

Mild: Functions with Lists

- Simple **functions** controlling multiple objects via **lists**
- **Assessment:** Successfully organizes velocity skills using lists and functions
- **Success Criteria:** Function affects multiple objects, demonstrates code reusability

Medium: Physics Integration

- **Collision boundaries** with weight-differentiated objects
- **Assessment:** Demonstrates collision understanding with mass effects
- **Success Criteria:** Objects bounce appropriately based on mass and bounciness

Hot: Boolean Functions with Collision Logic

- **Boolean logic** integrated with **list management** through **functions**
- **Assessment:** Sophisticated functions combining Week 2 conditional logic with lists
- **Success Criteria:** Complex toggle systems affecting multiple objects

Technical Skills

- **Function** definition and calling
- **List** creation and manipulation
- **Collision detection** setup
- **Physics properties** configuration (mass: 0.1-10.0kg, bounciness: 0.1-0.8)

Physics Configuration

- **Light objects (0.1-0.5kg):** Bounciness 0.6-0.8
- **Medium objects (1.0-3.0kg):** Bounciness 0.3-0.5
- **Heavy objects (5.0-10.0kg):** Bounciness 0.1-0.2

Week 4: Mars Base Construction

Your Challenge

Design a Mars base using all previous coding skills with **camera tours** and **interactive elements**. You'll combine everything you've learned to create an immersive Martian environment complete with realistic structures and guided exploration. Your finished base should showcase animated paths, physics-controlled objects, interactive systems, and creative design that demonstrates mastery of multiple coding concepts working together.

Learning Objectives

Students will integrate **path animation**, **velocity control**, **functions**, and **collision physics** to create complex interactive environments.

Computing Concepts

- **Integration:** Combining multiple programming concepts
- **Camera paths:** Guided movement for user experience
- **Interactive objects:** Objects responding to coded behaviors
- **Environmental design:** 3D space creation with physics

Your Challenge

Design a Mars base using all previous coding skills with **camera tours** and **interactive elements**.

Differentiated Challenge Levels

Mild: Basic Mars Base

- 2-3 areas with simple **camera path** tour
- **Assessment:** Successful integration of basic animation and physics concepts
- **Success Criteria:** Working camera movement, basic interactive elements

Medium: Interactive Mars Base

- 4-5 areas with **interactive objects** and longer tours
- **Assessment:** Multiple features demonstrating varied coding concepts
- **Success Criteria:** Complex camera paths, interactive doors/equipment

Hot: Complex Systems Integration

- 6+ areas with sophisticated **function** and **physics** integration
- **Assessment:** Advanced implementation of all learned concepts
- **Success Criteria:** Multiple camera tours, complex interactive systems

Technical Skills

- **Camera path** creation and timing
- **Environment design** with realistic physics
- **Interactive object** programming
- Integration of **functions**, **lists**, and **collision physics**

Week 5: Conditional Logic and User Input

Your Challenge

Add educational quizzes using **if/else logic** and **user input** to your Mars base. You'll create interactive learning experiences that respond intelligently to user answers, providing helpful feedback and branching paths based on responses. Your quiz system should demonstrate how conditional logic can create engaging educational content that adapts to different user inputs.

Learning Objectives

Students will implement **conditional statements** and **user input** systems to create educational interactive experiences.

Computing Concepts

- **Conditional logic:** If/else statements for decision making
- **User input:** Text input and response systems
- **String comparison:** Checking user answers against correct responses
- **Feedback systems:** Providing appropriate responses

Your Challenge

Add educational quizzes using **if/else logic** and **user input** to your Mars base.

Technical Skills

- **If/else statement** implementation
- **Text input** handling
- **String comparison** for answer checking
- **Feedback system** design

Assessment Criteria

- Working **conditional logic** with appropriate responses
- Clear educational content integration
- User-friendly **input/output** systems

Week 6: Documentation and Peer Assessment

Your Challenge

Create a video tour of your Mars base and provide constructive **peer assessment** using technical criteria. You'll document your coding journey by recording a presentation that showcases all your technical achievements and explains your problem-solving process. Your final presentation should demonstrate clear understanding of computing concepts while providing helpful feedback to classmates using specific technical vocabulary and assessment criteria.

Learning Objectives

Students will document their learning process and provide constructive **peer assessment** using technical criteria.

Computing Concepts

- **Code documentation:** Explaining programming choices
- **Technical communication:** Using appropriate computing vocabulary
- **Peer assessment:** Evaluating others' work against success criteria
- **Reflection:** Analyzing problem-solving processes

Assessment Focus Areas

- **Technical Implementation:** Correct use of functions, lists, collision physics
- **Code Organization:** Clear, readable, and reusable code structure
- **Problem Solving:** Evidence of debugging and iterative improvement
- **Integration:** Successful combination of multiple programming concepts

Technical Vocabulary Assessment

Students demonstrate understanding of: **velocity**, **boolean variables**, **functions**, **lists**, **collision detection**, **physics properties**, **conditional statements**, **loops**