Scratch Game Development:
Hackathon Technical Manual

This guide outlines the implementation of core mechanics found in popular Scratch projects to facilitate rapid development during a hackathon.

1. Physics and Movement (Platformers)

High-quality platformers rely on a variable-based physics system rather than simple coordinate changes. This allows for smooth jumping and realistic gravity.

- **Gravity Implementation:** * Initialize a variable y_velocity.
    - In a perpetual loop, decrement y_velocity by a set gravity constant (e.g., -1).
    - Update the vertical position by the current y_velocity.
- **Collision Detection:** * If the sprite contacts the "Ground" layer, reset y_velocity to 0.
    - Adjust the position slightly upward to prevent the sprite from sinking into the floor.
- **Tutorial:** https://www.youtube.com/watch?v=D16hTnDGweo

2. Economic Systems and Scaling (Clickers)

Idle games require precise variable management to handle "Price Inflation" and "Income per Second."

- **Core Click Logic:** Increment a Currency variable upon sprite interaction.
- **The Shop System:** Create a logic gate that checks if Currency is greater than or equal to the Item_Cost. If true, subtract Item_Cost from Currency and increment the Multiplier or Auto_Clicker variable.
- **Scaling Difficulty:** Increase the Item_Cost by a percentage (e.g., 1.15x) after every purchase to maintain game balance.
- **Tutorial:** https://www.youtube.com/watch?v=R38_FxQ3VgA

3. Visual Feedback and Particle Systems

Visual "juice" improves the perceived quality of a project. Cloning is the primary method for creating these effects.

- **Trailing Effects:** Use the create clone of [myself] block within a movement loop.
- **Clone Behavior:** Set clones to gradually increase their ghost effect or decrease in size before executing the delete this clone block. This creates a fading trail similar to professional animations.
- **Dynamic UI:** Use the change size blocks to create a momentary "bulge" effect when buttons are pressed, providing immediate tactile feedback to the player.
- **Tutorial:** https://www.youtube.com/watch?v=2s27Y1k634Y https://www.youtube.com/watch?v=3audNZs7uV8

4. Boss AI and Combat Patterns

Complex enemy behavior is managed through "Broadcasts" and "States."

- **State Management:** Use a State variable to determine if the boss is "Idle," "Attacking," or "Moving."
- **Attack Patterns:** Use a pick random function to trigger different Broadcast signals. Each signal corresponds to a unique script:
  - **Pattern A:** Launch projectiles toward the player's coordinates.
  - **Pattern B:** Move to a designated screen location and execute a radial attack.
- **Tutorial:** https://www.youtube.com/watch?v=iU9ntt_v3jY https://www.youtube.com/watch?v=3ddUAR7LK6A https://www.vibelf.com/questions/4/boss-fight-mechanics/

5. Development Efficiency

- **The Backpack Tool:** Utilize the Scratch Backpack to store verified scripts for smooth movement or inventory systems. This allows for the rapid deployment of complex code across different sprites.
- **Clean Architecture:** Use Broadcasts to trigger events rather than long strings of code under a single "Green Flag" block. This modular approach simplifies debugging under tight deadlines.

Examples of Scratch Drone Games and Key Mechanics

1. Physics and Movement (Platformer Engine)

Reference Project: 880747509 High-quality platformers rely on a variable-based physics system rather than simple coordinate changes. This allows for smooth jumping and realistic gravity.

- **Gravity Implementation:**
    - Initialize a variable y_velocity.
    - In a perpetual loop, decrement y_velocity by a set gravity constant (e.g., -1).
    - Update the vertical position by the current y_velocity.
- **Collision Detection:**
    - If the sprite contacts the "Ground" layer, reset y_velocity to 0.
    - Adjust the position slightly upward to prevent the sprite from sinking into the floor.
- **Tutorial:** Advanced Platforming Physics

2. Economic Systems and Scaling (Clicker/Idle)

Reference Project: 1153438654 Idle games require precise variable management to handle price inflation and income generation.

- **Core Click Logic:** Increment a Currency variable upon sprite interaction.
- **The Shop System:** Create a logic gate that checks if Currency is greater than or equal to the Item_Cost. If true, subtract Item_Cost from Currency and increment the Multiplier or Auto_Clicker variable.
- **Scaling Difficulty:** Increase the Item_Cost by a percentage (e.g., 1.15x) after every purchase to maintain game balance.
- **Tutorial:** https://www.youtube.com/watch?v=bfTXxRX_CQw

3. Objective-Based Movement (Catcher)

Reference Project: 1165877212 This archetype involves randomized spawning and collision-based scoring.

- **Randomized Spawning:** Set a sprite to Hide, then use a Forever loop to Create clone of [myself] at set intervals.

- **Falling Logic:** For the clone: When I start as a clone -> Go to random position -> Set y to 180 -> Show.
- **The Loop:** Use Repeat until y position < -170. Inside, Change y by -5. If Touching [Player], increment the score and Delete this clone.
- **Tutorial:** Catch Game Basics

4. Visual Feedback and Particle Systems (Art/Mouse Effects)

Reference Project: 436458930 Visual "juice" improves the perceived quality of a project. Cloning is the primary method for creating these effects.

- **Trailing Effects:** Use the create clone of [myself] block within a movement loop or tracking the mouse pointer.
- **Decay Logic:** Set clones to gradually increase their ghost effect or decrease in size before executing the delete this clone block.
- **Dynamic UI:** Use change size blocks to create a momentary "bulge" effect when buttons are pressed, providing immediate feedback to the player.
- **Tutorial:** https://www.youtube.com/watch?v=cenBN8I62FQ

5. Boss AI and Combat Patterns (Action/Combat)

Reference Project: 482508113 Complex enemy behavior is managed through "Broadcasts" and "States."

- **State Management:** Use a State variable to determine if the boss is "Idle," "Attacking," or "Moving."
- **Attack Patterns:** Use a pick random function to trigger different Broadcast signals. Each signal corresponds to a unique script:
  - **Pattern A:** Launch projectiles toward the player's coordinates.
  - **Pattern B:** Move to a designated screen location and execute a radial attack.
- **Tutorial:** https://www.youtube.com/watch?v=vMQg1DZdW6o