Adam Atienza
27 May 2025
CSS 385
Project - Cool Math Game

This project is a beginner-friendly recreation of the popular Coolmath game “Run” using the Unity game engine. The main goal was to build a simple endless runner where the player continuously moves forward and must jump across platforms while avoiding gaps that end the game.

Controls:
The player automatically moves forward at a constant speed. The only control the player has is the spacebar, which makes the character jump. Pressing space causes the player to leap over gaps between ground tiles. If the player falls into a gap or hits a designated death zone, the game resets or ends.

Character Behavior:
The player character has a Rigidbody2D component controlling its physics. The character runs automatically to the right at a fixed horizontal speed. Jumping is enabled by applying an upward force when the spacebar is pressed, allowing the player to clear gaps between platforms. Gravity pulls the character back down to the ground. The jump height and speed are tuned so the player can successfully navigate most of the generated gaps.

Environment and Ground Spawning:
Ground tiles are spawned dynamically ahead of the player to create an infinite runner effect. Tiles have a fixed width, and the spawner randomly inserts small gaps between tiles to challenge the player’s timing and jumping skills. To ensure the game is fair and playable, gaps are controlled to be of a fixed, jumpable width, preventing impossible jumps. The spawner also ensures no consecutive gaps appear, avoiding unfair difficulty spikes.

Challenges:
The most challenging part of the project was balancing the procedural generation of ground tiles and gaps so that the gameplay felt fair yet challenging. Initially, gaps were too wide or randomly spaced, making some jumps impossible. This was resolved by creating a separate gap prefab with a fixed width and adjusting the spawning logic to prevent multiple gaps in a row. Tuning the player’s jump force, movement speed, and gravity also required experimentation to create smooth, responsive controls.

Development Time:
This project took approximately 4-5 hours to complete, including learning basic Unity concepts such as prefabs, scripting in C#, Rigidbody2D physics, and scene management. Additional time was spent debugging player movement, jump physics, and procedural ground generation.