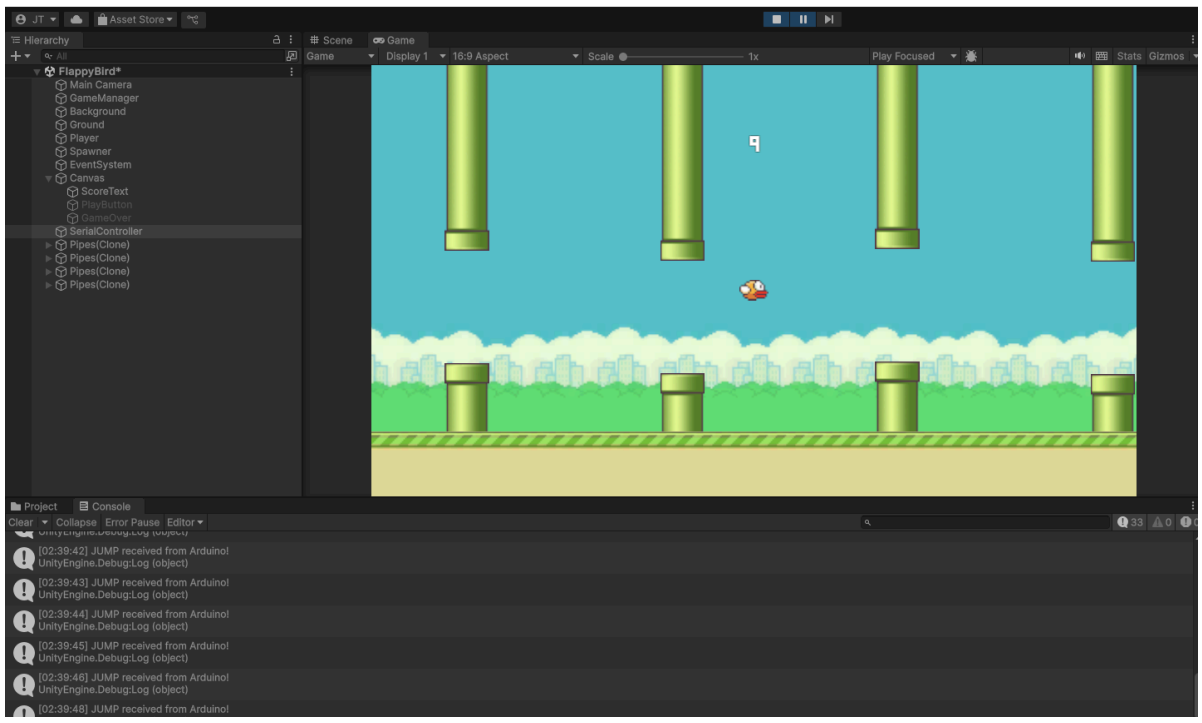
FLAPPY BUTTON: Flap Like You Mean It

This game reimagines Flappy Bird using a physical button controller made with Arduino. The player must press the large physical button to flap, making the game more active and tactile.



Flappy Button is a custom-built version of Flappy Bird where the player controls the bird using a real physical button instead of the keyboard. Each press of the button sends a signal from an Arduino board to Unity, causing the bird to flap upward.

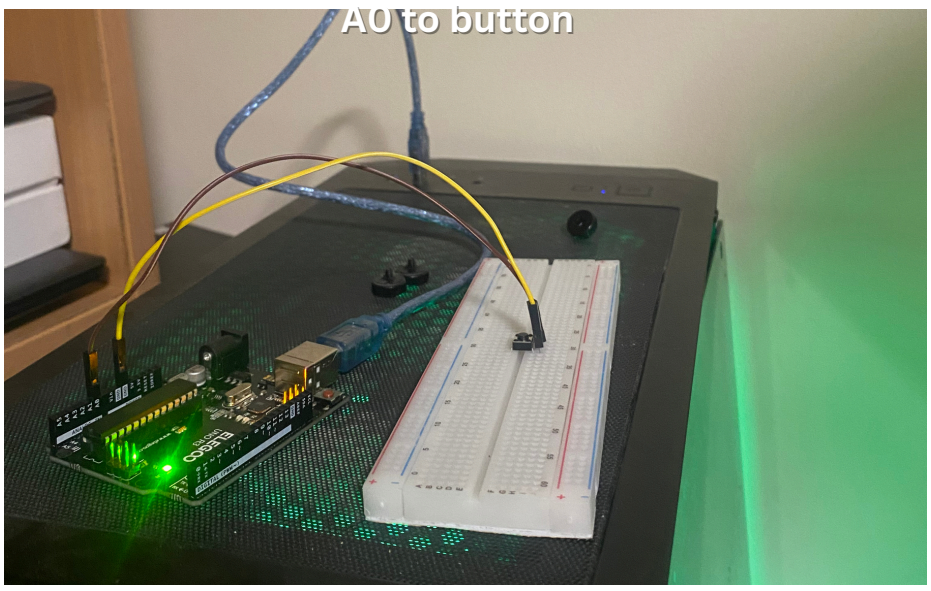
The objective is simple:
Survive as long as possible by avoiding obstacles and carefully timing your jumps. If the bird hits a pipe or the ground, it’s game over!

This physical input makes the game more engaging and tactile, blending classic arcade gameplay with hands-on electronics.

```
const int buttonPin = A0;
bool pressed = false;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  int buttonState = analogRead(buttonPin);

  if (buttonState < 100 && pressed == false) {
    Serial.println("JUMP");
    Serial.println(buttonState);
    pressed = true;

  }
  else {
    pressed = false;
  }

  delay(100);
}
```

The game uses a physical push button connected to an Arduino instead of a keyboard. When the button is pressed, the Arduino sends a "JUMP" signal to Unity via USB. Unity reads this signal and makes the bird flap, just like pressing the spacebar.
This setup makes the game more engaging and physical, standing out to spectators and encouraging interaction. It's simple, fun, and ideal for a public event or expo.

Wire connecting GND to button, Wire connecting A0 to button



The system was developed using Unity for the game and an Arduino Uno to create the custom input device. A momentary push button was wired to a digital input pin on the Arduino. The button is connected using a configuration to ensure reliable digital reads (HIGH when pressed, LOW when released).
The Arduino code listens for button presses and sends the keyword "JUMP" over the serial port when the button is pressed. This signal is picked up in Unity by a C# script (SerialReader), which listens to the correct COM port using the System.IO.Ports namespace.
When "JUMP" is received, the script directly triggers the Flap() method in the Player script, causing the bird to rise. This replicates the spacebar functionality through physical interaction.
The Arduino is powered and connected via USB, allowing both serial communication and power delivery. All components were placed securely on a solderless breadboard during development. The setup is simple, stable, and integrates easily with Unity’s update cycle, ensuring responsive input with minimal delay.

James Thornburgh JT316721 Game Development: Programming