```
#include <Adafruit_CircuitPlayground.h>

// Game constants
const int MAX_SEQUENCE_LENGTH = 50;
const int TIMEOUT_MS = 10000; // 10 seconds
const int FLASH_DURATION = 500;
const int INPUT_FLASH_DURATION = 200;

// Game states
enum GameState {
 IDLE,
 STARTING,
 SHOWING_SEQUENCE,
 WAITING_FOR_INPUT,
 GAME_OVER
};

// Game variables
GameState currentState = IDLE;
int sequence[MAX_SEQUENCE_LENGTH];
int sequenceLength = 1;
int currentInputIndex = 0;
int score = 0;
unsigned long lastInputTime = 0;
unsigned long gameStartTime = 0;

// Sound notes (frequencies in Hz)
const int NOTE_C4 = 262;
const int NOTE_D4 = 294;
const int NOTE_DS4 = 311;
const int NOTE_E4 = 330;
const int NOTE_G4 = 392;
const int NOTE_A4 = 440;
const int NOTE_C5 = 523;

void setup() {
 CircuitPlayground.begin();
 Serial.begin(9600);
 randomSeed(analogRead(0)); // Seed random number generator
  // Initialize to green (idle state)
 setAllLEDs(0, 255, 0); // Green
  Serial.println("Memory Game Ready!");
```

```
 Serial.println("Press both buttons to start");
}

void loop() {
 switch (currentState) {
   case IDLE:
     handleIdleState();
     break;
   case STARTING:
     handleStartingState();
     break;
   case SHOWING_SEQUENCE:
     handleShowingSequence();
     break;
   case WAITING_FOR_INPUT:
     handleWaitingForInput();
     break;
   case GAME_OVER:
     handleGameOver();
     break;
 }
  delay(50); // Small delay to prevent excessive polling
}

void handleIdleState() {
 // Check if both buttons are pressed to start the game
 if (CircuitPlayground.leftButton() && CircuitPlayground.rightButton()) {
   // Wait for buttons to be released
   while (CircuitPlayground.leftButton() || CircuitPlayground.rightButton()) {
     delay(10);
   }

   // Start the game
   currentState = STARTING;
   score = 0;
   sequenceLength = 1;
   Serial.println("Game Starting!");
 }
}

void handleStartingState() {
 // Flash white 3 times
```

```
 for (int i = 0; i < 3; i++) {
   setAllLEDs(255, 255, 255); // White
   delay(300);
   setAllLEDs(0, 0, 0); // Off
   delay(300);
 }
  // Play starting melody
 if (CircuitPlayground.slideSwitch()) { // Only if switch is on
   playStartingMelody();
 }
  // Generate first sequence
 generateNewSequence();
 currentState = SHOWING_SEQUENCE;
}

void handleShowingSequence() {
 // Turn off all LEDs
 setAllLEDs(0, 0, 0);
 delay(500);
  // Show the sequence
 for (int i = 0; i < sequenceLength; i++) {
   if (sequence[i] == 0) { // Red (left)
     setAllLEDs(255, 0, 0); // Red
   } else { // Blue (right)
     setAllLEDs(0, 0, 255); // Blue
   }
   delay(FLASH_DURATION);
   setAllLEDs(0, 0, 0); // Off
   delay(200);
 }
  // Start waiting for input
 currentState = WAITING_FOR_INPUT;
 currentInputIndex = 0;
 lastInputTime = millis();
}

void handleWaitingForInput() {
 // Check for timeout
 if (millis() - lastInputTime > TIMEOUT_MS) {
   Serial.println("Timeout!");
   currentState = GAME_OVER;
   return;
```

```
 }
  // Check for button presses
 bool leftPressed = CircuitPlayground.leftButton();
 bool rightPressed = CircuitPlayground.rightButton();
  if (leftPressed && !rightPressed) {
   processInput(0); // Left = Red = 0
 } else if (rightPressed && !leftPressed) {
   processInput(1); // Right = Blue = 1
 }
}

void processInput(int input) {
 // Flash corresponding color briefly to show input registered
 if (input == 0) {
   setAllLEDs(255, 0, 0); // Red flash
 } else {
   setAllLEDs(0, 0, 255); // Blue flash
 }
 delay(INPUT_FLASH_DURATION);
 setAllLEDs(0, 0, 0);
  // Check if input is correct
 if (input == sequence[currentInputIndex]) {
   currentInputIndex++;
   lastInputTime = millis(); // Reset timeout

   // Check if sequence is complete
   if (currentInputIndex >= sequenceLength) {
     // Sequence completed correctly!
     score++;
     Serial.print("Score: ");
     Serial.println(score);

     // Play success chime
     if (CircuitPlayground.slideSwitch()) {
       playSuccessChime();
     }

     // Play happy melody every 5 points
     if (score % 5 == 0 && CircuitPlayground.slideSwitch()) {
       playHappyMelody();
     }
```

```
      // Increase sequence length and generate new sequence
      sequenceLength++;
      if (sequenceLength > MAX_SEQUENCE_LENGTH) {
        sequenceLength = MAX_SEQUENCE_LENGTH;
      }

      generateNewSequence();
      delay(1000); // Brief pause before next sequence
      currentState = SHOWING_SEQUENCE;
    }
  } else {
    // Wrong input - game over
    Serial.print("Final Score: ");
    Serial.println(score);
    currentState = GAME_OVER;
  }
   // Wait for button release
  while (CircuitPlayground.leftButton() || CircuitPlayground.rightButton()) {
    delay(10);
  }
}

void handleGameOver() {
  // Flash yellow 3 times
  for (int i = 0; i < 3; i++) {
    setAllLEDs(255, 255, 0); // Yellow
    delay(300);
    setAllLEDs(0, 0, 0); // Off
    delay(300);
  }
   // Play sad melody
  if (CircuitPlayground.slideSwitch()) {
    playSadMelody();
  }
   // Reset to idle state
  setAllLEDs(0, 255, 0); // Green
  currentState = IDLE;
  Serial.println("Game Over! Press both buttons to play again.");
}

void generateNewSequence() {
  // Keep existing sequence and add one new element
```

```
 sequence[sequenceLength - 1] = random(0, 2); // 0 = red/left, 1 = blue/right
}

void setAllLEDs(int red, int green, int blue) {
 for (int i = 0; i < 10; i++) {
   CircuitPlayground.setPixelColor(i, red, green, blue);
 }
}

// Sound functions
void playStartingMelody() {
 CircuitPlayground.playTone(NOTE_C4, 150);
 delay(50);
 CircuitPlayground.playTone(NOTE_E4, 150);
 delay(50);
 CircuitPlayground.playTone(NOTE_G4, 200);
 delay(150);
}

void playSuccessChime() {
 CircuitPlayground.playTone(NOTE_G4, 50);
 delay(30);
 CircuitPlayground.playTone(NOTE_C5, 50);
 delay(50);
}

void playSadMelody() {
 CircuitPlayground.playTone(NOTE_E4, 150);
 delay(50);
 CircuitPlayground.playTone(NOTE_DS4, 150);
 delay(50);
 CircuitPlayground.playTone(NOTE_D4, 200);
 delay(150);
}

void playHappyMelody() {
 CircuitPlayground.playTone(NOTE_C4, 100);
 delay(50);
 CircuitPlayground.playTone(NOTE_D4, 100);
 delay(50);
 CircuitPlayground.playTone(NOTE_E4, 150);
 delay(150);
```

```
}
```