ReflexMaster:
Blink & Beat ; Dont Blink or you’ll miss !!

C code for Tinkercad Simulation

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

// ---------- Hardware pins ----------
const int NUM_BTNS = 5;
const int ledPins[NUM_BTNS]    = {3, 4, 5, 6, 7};
const int buttonPins[NUM_BTNS] = {9, 10, 11, 12, 13};
const int buzzer = A2;
const int resetButton = A3; // menu / confirm / quit

// ---------- Game settings & state ----------
int numPlayers = 1;
int numTries = 5;
int mode = 1; // 1=Easy,2=Medium,3=Hard

int scores[6]; // 1..5
int score = 0;
int streak = 0;
int wrongAttempts = 0;
bool quitGame = false;

// timing & UI
unsigned long lastInputTime = 0;
unsigned long buttonDownTime = 0;
bool buttonHeld = false;

// constants
const unsigned long MENU_IDLE_MS = 2000; // 2s auto-confirm
const unsigned long QUIT_HOLD_MS = 3000; // 3s quit hold
const int MAX_WRONGS = 5;

// ---------- Helpers: buzzer / UI ----------
void beepShort(int f=1000, int ms=80) { tone(buzzer, f, ms); delay(ms+20); noTone(buzzer); }
void beepLong(int f=400, int ms=300)  { tone(buzzer, f, ms); delay(ms+20); noTone(buzzer); }

void showSplash() {
  lcd.clear();
  lcd.setCursor(2,0); lcd.print("ReflexMaster");
  lcd.setCursor(1,1); lcd.print("Blink & Beat!");
  tone(buzzer, 1200, 200);
  delay(250);
  tone(buzzer, 1500, 200);
  delay(250);
  tone(buzzer, 2000, 300);
  delay(1500);
}

void showSmallHUD_LivesTopRight(int currentWrong) {
  // prints small lives indicator on top-right: "<3xN"
  lcd.setCursor(12, 0);
  lcd.print("<3x");
  lcd.print(MAX_WRONGS - currentWrong); // remaining lives
}

// safe read button pressed (active LOW due to INPUT_PULLUP)
bool btnPressed(int pin) {
  return digitalRead(pin) == LOW;
}

// wait for release (debounce)
void waitButtonRelease(int pin) {
  while (digitalRead(pin) == LOW) delay(5);
  delay(60);
}

// ---------------- Menu selectors (auto-confirm after idle) ----------------
int selectPlayers() {
  int players = 1;
  lastInputTime = millis();
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("Select Players");
  lcd.setCursor(0,1); lcd.print("Players: ");
  lcd.print(players);

  while (true) {
    if (btnPressed(resetButton)) {
      unsigned long pressStart = millis();
      while (btnPressed(resetButton)) {
        if (millis() - pressStart >= QUIT_HOLD_MS) {
          beepLong(400,600);
          return players;  // quit to menu
        }
      }
      // short press → cycle players
      players++;
      if (players > 5) players = 1;
      lcd.setCursor(0,1); lcd.print("Players:    ");
      lcd.setCursor(9,1); lcd.print(players);
      beepShort(1200,80);
      lastInputTime = millis();
    }

    if (millis() - lastInputTime > MENU_IDLE_MS) {
      lcd.setCursor(12,1); lcd.print("....");
      beepShort(1500,120);
      delay(400);
      return players;
    }
  }
}


int selectTries() {
  int tries = 5;
  lastInputTime = millis();
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("Select Tries");
  lcd.setCursor(0,1); lcd.print("Tries: ");
  lcd.print(tries);

  while (true) {
    if (btnPressed(resetButton)) {
      unsigned long pressStart = millis();
      while (btnPressed(resetButton)) {
        if (millis() - pressStart >= QUIT_HOLD_MS) {
          beepLong(400,600);
          return tries;  // quit to menu
        }
      }
      // short press → cycle tries
      tries += 5;
      if (tries > 50) tries = 5;
      lcd.setCursor(0,1); lcd.print("Tries:     ");
      lcd.setCursor(7,1); lcd.print(tries);
      beepShort(1000,80);
      lastInputTime = millis();
    }

    if (millis() - lastInputTime > MENU_IDLE_MS) {
      lcd.setCursor(12,1); lcd.print("....");
      beepShort(1200,120);
      delay(400);
      return tries;
    }
  }
}

int selectMode() {
  int m = 1;
  lastInputTime = millis();
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("Select Mode");
  lcd.setCursor(0,1); lcd.print("Easy");

  while (true) {
    if (btnPressed(resetButton)) {
      unsigned long pressStart = millis();
      while (btnPressed(resetButton)) {
        if (millis() - pressStart >= QUIT_HOLD_MS) {
          beepLong(400,600);
          return m;  // quit to menu
        }
      }
      // short press → cycle mode
      m++;
      if (m > 3) m = 1;
      lcd.setCursor(0,1); lcd.print("                ");
      lcd.setCursor(0,1);
      if (m == 1) lcd.print("Easy");
      else if (m == 2) lcd.print("Medium");
      else lcd.print("Hard");
      beepShort(1100,80);
      lastInputTime = millis();
    }

    if (millis() - lastInputTime > MENU_IDLE_MS) {
      lcd.setCursor(12,1); lcd.print("....");
      beepShort(1400,120);
      delay(400);
      return m;
    }
  }
}



// ---------------- Utility: show small two-line messages (fits 16x2) ----------------
void showCentered2lines(const char *line1, const char *line2, int toneFreq=0, int toneMs=0,
int holdMs=1200) {
  lcd.clear();
  int pad1 = max(0, (16 - (int)strlen(line1)) / 2);
  int pad2 = max(0, (16 - (int)strlen(line2)) / 2);
  lcd.setCursor(pad1,0); lcd.print(line1);
  lcd.setCursor(pad2,1); lcd.print(line2);
  if (toneFreq) tone(buzzer, toneFreq, toneMs);
  delay(holdMs);
}

// ------------------ Gameplay modes ------------------

// Small helper: display HUD row with Try and lives and score
void displayHUD(int roundNum, int totalTries, int currentScore, int currentWrong) {
  // First row: "Try x/y <3xN" ; ensure fits
  lcd.setCursor(0,0);
  char buf[17];
  snprintf(buf, 17, "Try %d/%d", roundNum, totalTries);
  lcd.print(buf);
  // lives at right
  lcd.setCursor(11,0);
  lcd.print("<3x");
  lcd.print(MAX_WRONGS - currentWrong);
  // second row: Score
  lcd.setCursor(0,1);
  lcd.print("Score:");
  lcd.print(currentScore);
  // pad rest blank to avoid leftovers
  lcd.print("     ");
}

// wait for a button press and return pressed button index (0..NUM_BTNS-1) or -1 on quit or
timeout
int waitForButtonPress(unsigned long timeoutMs) {
  unsigned long start = millis();
  while (millis() - start < timeoutMs) {
    // check quit hold
    if (digitalRead(resetButton) == LOW) {
      unsigned long s = millis();
      while (digitalRead(resetButton) == LOW) {
        if (millis() - s >= QUIT_HOLD_MS) {
          // forced quit
          quitGame = true;
          return -1;
        }
      }
    }
    for (int i=0;i<NUM_BTNS;i++) {
      if (digitalRead(buttonPins[i]) == LOW) {
        waitButtonRelease(buttonPins[i]);
        return i;
      }
    }
    delay(6);
  }
  return -1; // timeout
}

// ---------- EASY mode ----------
bool playEasyRound(int roundIndex, int &reactionTime) {
  // show HUD
  displayHUD(roundIndex+1, numTries, score, wrongAttempts);

  // --- Ready–Set–Go ---
  lcd.clear(); lcd.setCursor(0,0); lcd.print("Ready...");
  delay(700);
  lcd.clear(); lcd.setCursor(0,0); lcd.print("Set...");
  delay(700);
  lcd.clear(); lcd.setCursor(0,0); lcd.print("Go!");
  beepShort(1400,120);
  delay(200);

  // Light one random LED
  int led = random(0, NUM_BTNS);
  digitalWrite(ledPins[led], HIGH);

  unsigned long start = millis();
  int pressedBtn = -1;
  bool pressed = false;
  while (millis() - start < (unsigned long)reactionTime) {
    // handle quit
    if (digitalRead(resetButton) == LOW) {
      unsigned long s = millis();
      while (digitalRead(resetButton) == LOW) {
        if (millis() - s >= QUIT_HOLD_MS) {
          quitGame = true;
          digitalWrite(ledPins[led], LOW);
          return false;
        }
      }
    }

    for (int i=0;i<NUM_BTNS;i++) {
      if (digitalRead(buttonPins[i]) == LOW) {
        waitButtonRelease(buttonPins[i]);
        pressed = true; pressedBtn = i;
        break;
      }
    }
    if (pressed) break;
    delay(4);
  }
  digitalWrite(ledPins[led], LOW);

  if (pressed && pressedBtn == led) {
    score += 10;
    streak++;
    beepShort(1000,100);
    if (streak % 3 == 0) {
      score += 3;
      showCentered2lines("STREAK BONUS!","+3 Points",1600,200,900);
    } else {
      showCentered2lines("Clean Hit!","",1000,100,700);
    }

    //  shrink only when correct
    reactionTime -= 50;
    if (reactionTime < 500) reactionTime = 500;

  } else if (pressed && pressedBtn != led) {
    score -= 5;
    streak = 0;
    wrongAttempts++;
    showCentered2lines("Wrong btn!","",300,300,900);
  } else {
    // timed out (miss)
    score -= 2;
    streak = 0;
    wrongAttempts++;
    showCentered2lines("Too slow!","",500,200,900);
  }

  // show lives
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("Score:");
  lcd.print(score);
  lcd.setCursor(10,0); lcd.print("<3x"); lcd.print(MAX_WRONGS - wrongAttempts);
  delay(700);

  // check wrong attempts
  if (wrongAttempts >= MAX_WRONGS) {
    showCentered2lines("GAME OVER","Too many wrongs",300,1000,1400);
    return false;
  }
  return true;
}

bool playEasy(int baseTime) {
  int reactionTime = baseTime;
  for (int r=0; r < numTries; r++) {
    if (quitGame) return false;
    bool ok = playEasyRound(r, reactionTime);
    if (!ok) return false;
  }
  return true;
}


// ---------- MEDIUM mode: sequence forward ----------
// seq length increases only when beaten correctly
void playSequence(int seq[], int len, int blinkMs) {
  // play sequence with tone per LED
  for (int i=0;i<len;i++) {
    int led = seq[i];
    digitalWrite(ledPins[led], HIGH);
    tone(buzzer, 900, 80);
    delay(blinkMs);
    digitalWrite(ledPins[led], LOW);
    delay(150);
  }
}

bool playMediumSequenceGame() {
  int seqLen = 3;
  int baseBlink = 700;
  int minBlink = 500;
  int blinkMs = baseBlink;

  for (int attempt = 0; attempt < numTries; attempt++) {
    if (quitGame) return false;

    int seq[20];
    for (int i=0;i<seqLen;i++) seq[i] = random(0, NUM_BTNS);

    // HUD
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Seq "); lcd.print(attempt+1); lcd.print("/"); lcd.print(numTries);
    lcd.setCursor(10,0); lcd.print("<3x"); lcd.print(MAX_WRONGS - wrongAttempts);
    lcd.setCursor(0,1); lcd.print("Memorise...");
    delay(600);



    // Ready-Set-Go
 lcd.clear(); lcd.setCursor(0,0); lcd.print("Ready...");
 delay(700);
 lcd.clear(); lcd.setCursor(0,0); lcd.print("Set...");
 delay(700);
 lcd.clear(); lcd.setCursor(0,0); lcd.print("Go!");
 beepShort(1400,120);
 delay(200);
  // play sequence
    playSequence(seq, seqLen, blinkMs);
    bool perfect = true;
    for (int i=0;i<seqLen;i++) {
      lcd.clear();
      lcd.setCursor(0,0); lcd.print("Input "); lcd.print(i+1); lcd.print("/"); lcd.print(seqLen);
      lcd.setCursor(0,1); lcd.print("Score:"); lcd.print(score);
      lcd.setCursor(10,1); lcd.print("<3x"); lcd.print(MAX_WRONGS - wrongAttempts);

      int pressed = waitForButtonPress(2000);
      if (quitGame) return false;
      if (pressed == -1 || pressed != seq[i]) { perfect = false; break; }
      beepShort(1000,70);
    }

    if (perfect) {
      score += 10;
      streak++;
      if (streak % 3 == 0) { score += 3; showCentered2lines("STREAK! +3","+3
Points",1600,200,900); }
      else showCentered2lines("Seq OK!","+10 pts",1200,140,800);
      // only now increase length & speed
      seqLen += 2; if (seqLen > 15) seqLen = 15;
      if (blinkMs > minBlink) blinkMs = max(minBlink, blinkMs - 100);
    } else {
      streak = 0;
      wrongAttempts++;
      showCentered2lines("Wrong Seq","- life",300,400,900);
    }

    if (wrongAttempts >= MAX_WRONGS) {
      showCentered2lines("GAME OVER","Too many wrongs",200,1000,1400);
      return false;
    }
  }
  return true;
}

// ---------- HARD mode: sequence reverse ----------
// seq length increases only when beaten correctly
bool playHardSequenceGame() {
  int seqLen = 3;
  int baseBlink = 500;
  int minBlink = 300;
  int blinkMs = baseBlink;

  for (int attempt = 0; attempt < numTries; attempt++) {
    if (quitGame) return false;

    int seq[20];
    for (int i=0;i<seqLen;i++) seq[i] = random(0, NUM_BTNS);

    // HUD
    lcd.clear(); lcd.setCursor(0,0);
    lcd.print("Seq "); lcd.print(attempt+1); lcd.print("/"); lcd.print(numTries);
    lcd.setCursor(10,0); lcd.print("<3x"); lcd.print(MAX_WRONGS - wrongAttempts);
    lcd.setCursor(0,1); lcd.print("Memorise rev");
    delay(500);

    playSequence(seq, seqLen, blinkMs);

    // Ready-Set-Go
    lcd.clear(); lcd.setCursor(0,0); lcd.print("Ready...");
    delay(350);
    lcd.clear(); lcd.setCursor(0,0); lcd.print("Set...");
    delay(350);
    lcd.clear(); lcd.setCursor(0,0); lcd.print("Go!");
    beepShort(1500,120);
    delay(200);

    bool perfect = true;
    for (int i=seqLen-1; i>=0; i--) {
      lcd.clear();
      lcd.setCursor(0,0); lcd.print("Input "); lcd.print(seqLen-i); lcd.print("/"); lcd.print(seqLen);
      lcd.setCursor(0,1); lcd.print("Score:"); lcd.print(score);
      lcd.setCursor(10,1); lcd.print("<3x"); lcd.print(MAX_WRONGS - wrongAttempts);

      int pressed = waitForButtonPress(1500);
      if (quitGame) return false;
      if (pressed == -1 || pressed != seq[i]) { perfect = false; break; }
      beepShort(1200,70);
    }

    if (perfect) {
      score += 15;
      streak++;
      if (streak % 3 == 0) { score += 3; showCentered2lines("STREAK! +3","+3
Points",1800,200,900); }
      else showCentered2lines("Perfect! +15","",1400,160,800);
      // only now increase length & speed
      seqLen += 2; if (seqLen > 15) seqLen = 15;
      if (blinkMs > minBlink) blinkMs = max(minBlink, blinkMs - 100);
    } else {
      streak = 0;
      wrongAttempts++;
      showCentered2lines("Wrong Rev","- life",300,500,900);
    }

    if (wrongAttempts >= MAX_WRONGS) {
      showCentered2lines("GAME OVER","Too many wrongs",200,1000,1400);
      return false;
    }
  }
  return true;
}


// ---------- runPlayer: pick mode ----------
int runPlayer(int playerIndex) {
  score = 0; streak = 0; wrongAttempts = 0; quitGame = false;

  lcd.clear(); lcd.setCursor(0,0); lcd.print("Player "); lcd.print(playerIndex);
  lcd.setCursor(0,1); lcd.print("Get Ready!");
  tone(buzzer, 900, 160);
  delay(1000);

  // mode instruction (short)
  if (mode == 1) {
    showCentered2lines("EASY MODE","Press as Shown",1000,120,1400);
    int base = 2000; int minB = 200;
    // stronger dynamic reduction in easy for challenge
    bool ok = playEasy(base);
    (void)ok;
  } else if (mode == 2) {
    showCentered2lines("MEDIUM MODE","Watch & Repeat",1000,120,1400);
    bool ok = playMediumSequenceGame();
    (void)ok;
  } else {
    showCentered2lines("HARD MODE","Repeat Reverse",1200,140,1400);
    bool ok = playHardSequenceGame();
    (void)ok;
  }

  // final per-player summary
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("P"); lcd.print(playerIndex); lcd.print(" Final:");
  lcd.setCursor(0,1); lcd.print("Score: "); lcd.print(score);
  delay(1500);
  return score;
}

// ---------- Winner & tie logic ----------
void showWinnersAndEnd() {
  // determine best and ties
  int best = scores[1];
  for (int p=2; p<=numPlayers; p++) if (scores[p] > best) best = scores[p];

  int cnt = 0;
  for (int p=1; p<=numPlayers; p++) if (scores[p] == best) cnt++;

  if (cnt > 1) {
    // tie
    lcd.clear();
    lcd.setCursor(0,0); lcd.print("It's a TIE!");
    lcd.setCursor(0,1); lcd.print("Score: "); lcd.print(best);
    tone(buzzer, 1400, 400);
    delay(2000);
  } else {
    int winner = 1;
    for (int p=1; p<=numPlayers; p++) if (scores[p] == best) winner = p;

    // winner show: flash 3 times
    for (int i=0;i<3;i++) {
      lcd.clear();
      lcd.setCursor(0,0); lcd.print("Winner: P"); lcd.print(winner);
      lcd.setCursor(0,1); lcd.print("Score: "); lcd.print(best);
      tone(buzzer, 1600 + i*200, 200);
      delay(500);
      lcd.clear(); delay(200);
    }
  }

  // final dramatic tagline
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("Don't Blink...,");
  lcd.setCursor(0,1); lcd.print("or u’ll Miss.!");
  tone(buzzer, 1000, 700);
  delay(2200);
}

// ------------------ main flow ------------------
void setup() {
  // pins
  for (int i=0;i<NUM_BTNS;i++) {
    pinMode(ledPins[i], OUTPUT);
    pinMode(buttonPins[i], INPUT_PULLUP); // buttons active LOW
    digitalWrite(ledPins[i], LOW);
  }
  pinMode(resetButton, INPUT_PULLUP);
  pinMode(buzzer, OUTPUT);

  lcd.init(); lcd.backlight();
  randomSeed(analogRead(0));
  showSplash();
}

void loop() {
  // main menu flow each loop
  // select players
  lcd.clear(); lcd.setCursor(0,0); lcd.print("Press Reset to"); lcd.setCursor(0,1); lcd.print("start
selection");
  // wait for first press to begin menu selection (so accidental restarts don't immediately start)
  while (!btnPressed(resetButton)) { delay(30); }
  waitButtonRelease(resetButton);
  beepShort(1200,100);

  numPlayers = selectPlayers();
  numTries = selectTries();
  mode = selectMode();

  // run each player
  for (int p=1; p<=numPlayers; p++) {
    scores[p] = runPlayer(p);
    if (quitGame) break;
    delay(600);
  }

  if (!quitGame) showWinnersAndEnd();

  // reset some flags and go back to splash/menu
  quitGame = false;
  delay(800);
}
```