// PROJECT  :TicTacToeAutonomousRandom

// PURPOSE  :The base code for the eventual two player serial communication game

// DEVICE   :Single Arduino

// AUTHOR   :C. D'Arcy

// DATE     :2019 01 17

// uC       :328

// COURSE   :ICS3U

// STATUS   :Working

// REFERENCE:http://darcy.rsgc.on.ca/ACES/TEI3M/CommunicationProtocols.html#SC2

char EMPTY = ' ';      //character to be used for the empty TTT cell

char ttt[3][3];        //the tictac-toe game board

const char SYMBOLS[] = {'X', 'O'}; //player pieces

const uint8\_t winner0 = 3 \* SYMBOLS[0]; //row or column totals indicating a

const uint8\_t winner1 = 3 \* SYMBOLS[1]; //row or column totals indicating a

char buff[8] = "        ";              //output buffer supports serial display of game

uint8\_t turn;                           //turn: even/odd alternation between players

uint8\_t over;                           //game state: 0 (playing), 1 (winner0), 2 (winner1)

void setup() {

  **Serial**.begin(9600);                   //invoke serial communications

  while (!**Serial**);                      //wait to instantiate

  **Serial**.println("Tic-Tac-Toe. Autonomous and Random. Just let it run...");

  randomSeed(analogRead(A0));           //different every time...

}

void loop() {                           //maximizes use of **functions** (modular)

  **newGame();**                            //start a new game

  while (!over) {                       //play game until outcome determined

    **Serial**.println("\nTurn: " + String(turn % 2 ? 'O' : 'X'));

    **randomPlay();**                       //randomly take a turn

    **displayBoard();**                     //display game board on terminal

    over = **checkBoard();**                //determine possible winner or draw

    if (over)                           //well, is **over** non-zero? ie, there’s a winner

      **Serial**.println("The winner is " + String(SYMBOLS[over - 1]));

    else if (turn == 8) {               //a draw!

      **Serial**.println("Draw");           //say so...

      over = -1;                        //trigger new game

    } else {

      turn++;                           //keep playing

      delay(1000);                      //1 second between moves

    }

  }

  **Serial**.println("\nNew Game...");      //announce the start of a new game

}

void newGame() {

  for (uint8\_t row = 0; row < 3; row++)     //empty the game board

    for (uint8\_t col = 0; col < 3; col++)

      ttt[row][col] = EMPTY;

  turn = 0;                                 //zero the turn

  over = 0;                                 //outcome to be determined

  delay(3000);                              //catch your breath...so exciting☺

}

uint8\_t randomPlay() {                      //respective player takes a turn

  uint8\_t row = random(3);

  uint8\_t col = random(3);

  while (ttt[row][col] != EMPTY) {          //not efficient but keeps code simple

    row = random(3);

    col = random(3);

  }

  ttt[row][col] = SYMBOLS[turn % 2];

}

uint8\_t checkBoard() { //check the 16 ways a win could happen

  uint8\_t total; //0:continue; 1:player0: 2:player1

  **//check rows**

  for (uint8\_t row = 0; row < 3; row++) {

    total = ttt[row][0] + ttt[row][1] + ttt[row][2];

    if (total == winner0) return 1;

    if (total == winner1) return 2;

  }

 **//check cols**

  for (uint8\_t col = 0; col < 3; col++) {

    total = ttt[0][col] + ttt[1][col] + ttt[2][col];

    if (total == winner0) return 1;

    if (total == winner1) return 2;

  }

  **//check main diagonal**

  total = ttt[0][0] + ttt[1][1] + ttt[2][2];

  if (total == winner0) return 1;

  if (total == winner1) return 2;

  **//check minor diagonal**

  total = ttt[0][2] + ttt[1][1] + ttt[2][0];

  if (total == winner0) return 1;

  if (total == winner1) return 2;

  return 0;

}

void displayBoard() {                     //render a game board on the serial terminal

  for (uint8\_t row = 0; row < 3; row++) {

    sprintf(buff, "|%c|%c|%c|", ttt[row][0], ttt[row][1], ttt[row][2]);

    **Serial**.println(buff);

  }

}