```
#include <LiquidCrystal.h>
#include "pitches.h"
#include <Adafruit_NeoPixel.h>

//light strip
#define litestrip 38
Adafruit_NeoPixel strip = Adafruit_NeoPixel(40, litestrip);

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(23,25,27,29,31,33);
const int rows = 4;
const int colms = 20;

//input pins from PIC, pin 3 is lowest bit 6 is highest
int picin[] = {22,24,26,28};
//variable for key input
int key = 0;

// create elements for dice roll
int roll = random(1, 7);
char* diedots[] = {"|       |", "|   o   |", "| o     |","|     o |","| o
o |"};

//pins for stepper
int turn1 = 52;
int dir1 = 50;
int turn2 = 53;
int dir2 = 51;

//pins for preiferials
int lit = 45;
int mag = 43;
int lim = 49;
int zer = 47;

//sounds
int speaker = 41;

const int pause = 50;

//game setup
String colors[] = {" Red", " Blue", "Green", "Yellow"};
int peices [4][4][3];
int posR = 0;
int posT = 0;
int players = 4;
int player = 0; // current player

void setup() {
  //start lights
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'

  // initialize the inputs:
```

```
  pinMode(picin[0], INPUT);
  pinMode(picin[1], INPUT);
  pinMode(picin[2], INPUT);
  pinMode(picin[3], INPUT);

  //initialize motor pins
  pinMode(dir1, OUTPUT);
  pinMode(turn1, OUTPUT);
  pinMode(dir2, OUTPUT);
  pinMode(turn2, OUTPUT);

  //initialize prefierials
  pinMode(lit, OUTPUT);
  pinMode(mag, OUTPUT);
  pinMode(lim, INPUT);
  pinMode(zer, INPUT);
  pinMode(speaker, OUTPUT);

  // initialize lcd
  lcd.begin(colms, rows);
  lcd.print("Welcome!");
  introSong();

  //select number of players
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Enter the number of");
  lcd.setCursor(0,1);
  lcd.print("players 2-4.");
  lcd.setCursor(0,2);
  lcd.print("Push * to confirm");
  while(keyin(picin) != 10){
    lcd.setCursor(0,3);
    lcd.print("Players: ");
    lcd.setCursor(11, 3);
    lcd.print(players);
    uint16_t i, j;
    for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
      for(i=0; i< strip.numPixels(); i++) {
        strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j)
& 255));
      }
      if(keyin(picin) != 0) break;
      strip.show();
      delay(5);
    }
    if( (keyin(picin)>0) && (keyin(picin)<5))  players = keyin(picin);
  }

  //initialize peice positions
  for(int pl=0; pl<players; pl++){
    int theta = 5*pl+2;
    //theta
    peices[pl][0][0] = theta;
```

```
    peices[pl][1][0] = theta+1;
    peices[pl][2][0] = theta;
    peices[pl][3][0] = theta+1;
    //radius
    peices[pl][0][1] = 1;
    peices[pl][1][1] = 1;
    peices[pl][2][1] = 0;
    peices[pl][3][1] = 0;
    //state -1 start, 0 in play, 1 home
    peices[pl][0][2] = -1;
    peices[pl][1][2] = -1;
    peices[pl][2][2] = -1;
    peices[pl][3][2] = -1;
  }

  zero();
  delay(600);
  moveOver(true);
}



void loop(){
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(colors[player]);
  lcd.setCursor(7, 0);
  lcd.print("players turn");
  lcd.setCursor(0, 1);
  lcd.print("Push & hold to roll,");
  lcd.setCursor(0, 2);
  lcd.print("release to stop. . .");
  lcd.setCursor(0, 3);
  lcd.print("Use the 0 key.");
  colorSet(player);
  strtSong();
  while(keyin(picin) != 11) delay(pause); //pause till push
  while(keyin(picin) == 11) {
    // start rolling:
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Rolling . . . ");
    roll = random(1, 7);
    diceRoll(roll, diedots);
    rollSong();
  }
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("You rolled a:");
  diceRoll(roll, diedots);
  delay(1000);

  boolean gone = false;
  while(!gone){
```

```
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Select piece 1-4");
    lcd.setCursor(0,1);
    lcd.print("Push # to check");
    lcd.setCursor(0,2);
    lcd.print("Hold * to confirm");
    int choice = 0;
    boolean selected = false;
    while(!selected){    //peice selection
      while(keyin(picin) != 12){
        if( (keyin(picin) != 0) && (keyin(picin) < 5) )  choice =
keyin(picin)-1;
        if(choice == -1) choice = 0;
        if(keyin(picin) == 10) {
          selected = true;
          break;
        }
        if(keyin(picin) == 9){    //developer option, used to reset while
in play
          zero();
          delay(600);
          moveOver(true);
        }
        lcd.setCursor(0,3);
        lcd.print("Piece: ");
        lcd.setCursor(7,3);
        lcd.print(choice+1);
      }
      straitTo(peices[player][choice][0], peices[player][choice][1]);
      digitalWrite(lit, HIGH);
      delay(500);
      digitalWrite(lit, LOW);
    }

    boolean valid=false;
    int start = peices[player][choice][0];
    int dest = peices[player][choice][0]+roll;
    if(dest>21) dest = dest-21;

    if(peices[player][choice][2]==1){   //peice is home
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Invalid selection");
      lcd.setCursor(0,1);
      lcd.print("Already home!");
      lcd.setCursor(0,2);
      lcd.print("Choose another piece");
      delay(2000);
    }
    else if(peices[player][choice][2] == -1){  //starting area
      if( (choice>1) && (peices[player][(choice-2)][2]==-1)){
          lcd.clear();
          lcd.setCursor(0,0);
```

```
          lcd.print("Invalid selection");
          lcd.setCursor(0,1);
          lcd.print("Can't reach track");
          lcd.setCursor(0,2);
          lcd.print("Choose another piece");
          delay(2000);
      }
      else{  // move peice from start to track
        // check if knocks out another peice
        for(int pl = 0; pl<=players; pl++){
          for(int pc = 0; pc <=3; pc++){
            if( (peices[pl][pc][2]==0) &&
(peices[pl][pc][0]==(5*player+2)) ){
              //remove peice
              lcd.setCursor(0,3);
              lcd.print("Piece knocked back!!");
              int strR = 0;
              int strT = 5*pl+2;
              if((pc == 0) || (pc == 1))  strR = 1;
              if((pc == 1) || (pc == 3))  strT++;
              straitTo(peices[pl][pc][0], peices[pl][pc][1]);
              outSong();
              peices[pl][pc][0] = strT;
              peices[pl][pc][1] = strR;
              peices[pl][pc][2] = -1;
              moveTo( peices[pl][pc][0], peices[pl][pc][1]);
              straitTo(peices[player][choice][0],
peices[player][choice][1]);
            }
          }
        }
        peices[player][choice][0] = 5*player+2;
        peices[player][choice][1] = 3;
        peices[player][choice][2] = 0;
        moveTo( peices[player][choice][0], peices[player][choice][1]);
        gone = true;
      }
    }
    else if(peices[player][choice][2] == 0){  //on track
      // check if knocks out another peice
      for(int pl = 0; pl<=players; pl++){
        for(int pc = 0; pc <=3; pc++){
          if( (peices[pl][pc][2]==0) && (peices[pl][pc][0]==(dest)) ){
            //remove peice
            lcd.setCursor(0,3);
            lcd.print("Piece knocked back!!");
            int strR = 0;
            int strT = 5*pl+2;
            if((pc == 0) || (pc == 1))  strR = 1;
            if((pc == 1) || (pc == 3))  strT++;
            straitTo(peices[pl][pc][0], peices[pl][pc][1]);
            outSong();
            peices[pl][pc][0] = strT;
            peices[pl][pc][1] = strR;
```

```
            peices[pl][pc][2] = -1;
            moveTo( peices[pl][pc][0], peices[pl][pc][1]);
            straitTo(peices[player][choice][0],
peices[player][choice][1]);
            valid = true;
          }
        }
      }
      if( ((dest-roll)<(5*player+1)) && (dest>=(5*player+1))){  //reached
home
          lcd.setCursor(0,3);
          if(player == 0)  moveTo( 22, 3 );
          else if( (player == 1) && (roll == 6))  moveTo( 27, 3 );
          else   moveTo( (5*player+1), 3 );
          delay(1000);
          int homeR = 1;
          int homeT = 5*player;
          if(choice == 0 || choice == 1)  homeR = 0;
          if(choice == 1 || choice == 3)  homeT++;
          peices[player][choice][0] = homeT;
          peices[player][choice][1] = homeR;
          peices[player][choice][2] = 1;
          moveTo( peices[player][choice][0], peices[player][choice][1]);
          gone = true;
          homeSong();
          lcd.setCursor(0,3);
          lcd.print("Piece Home!!");
          delay(1000);
      }
      else valid = true;
    }
    else{
      lcd.print("error in game flow");
    }

    if(valid){
      peices[player][choice][0] += roll;
      moveTo( (peices[player][choice][0]), 3);
      gone = true;
    }

    //take care of board handoff
    if(peices[player][choice][0] > 21)  peices[player][choice][0] -= 21;
  }

  //check for win
  int home = 0;
  for(int pc = 0; pc <=4; pc++){
    home += peices[player][pc][2];
  }
  if(home == 4){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(colors[player]);
```

```
    lcd.setCursor(7, 0);
    lcd.print("Player Wins!");
    lcd.setCursor(0, 2);
    lcd.print("Congradulations !!");
    while(true){
      colorSet(player);
      delay(100);
      colorSet(4);
      delay(50);
    }
  }

  //next player
  player++;
  if(player == players) player = 0;
}



void diceRoll(int roll, char* diedots[]){
  switch(roll) {
  case 1:
    lcd.setCursor(0,1);
    lcd.print(diedots[0]);
    lcd.setCursor(0, 2);
    lcd.print(diedots[1]);
    lcd.setCursor(0,3);
    lcd.print(diedots[0]);
    break;
  case 2:
    lcd.setCursor(0, 1);
    lcd.print(diedots[3]);
    lcd.setCursor(0,2);
    lcd.print(diedots[0]);
    lcd.setCursor(0, 3);
    lcd.print(diedots[2]);
    break;
  case 3:
    lcd.setCursor(0, 1);
    lcd.print(diedots[2]);
    lcd.setCursor(0, 2);
    lcd.print(diedots[1]);
    lcd.setCursor(0, 3);
    lcd.print(diedots[3]);
    break;
  case 4:
    lcd.setCursor(0, 1);
    lcd.print(diedots[4]);
    lcd.setCursor(0,2);
    lcd.print(diedots[0]);
    lcd.setCursor(0, 3);
    lcd.print(diedots[4]);
    break;
  case 5:
```

```
    lcd.setCursor(0, 1);
    lcd.print(diedots[4]);
    lcd.setCursor(0, 2);
    lcd.print(diedots[1]);
    lcd.setCursor(0, 3);
    lcd.print(diedots[4]);
    break;
  case 6:
    for(int thisrow = 1; thisrow < rows; thisrow ++){
      lcd.setCursor(0, thisrow);
      lcd.print(diedots[4]);
    }
    break;
  }
  delay(pause);
}

int keyin(int picin[]){
  int val = 0;
  if(digitalRead(picin[0]) == HIGH ){
    val = val + 1;
  }
  if(digitalRead(picin[1]) == HIGH ){
    val = val + 2;
  }
  if(digitalRead(picin[2]) == HIGH ){
    val = val + 4;
  }
  if(digitalRead(picin[3]) == HIGH ){
    val = val + 8;
  }
  return val;
}

// be carful that the degree argument given will be short if not
perfectly divisible by step size
void move(int stp, boolean drct, int turn, int dir, int spd){  //drct is
false for ccw, true for cw
  if(drct) digitalWrite(dir, HIGH);
  else digitalWrite(dir, LOW);

  for(int i = 1; i < stp; i++){
    digitalWrite(turn, HIGH);
    delay(spd);               // wait for a second
    digitalWrite(turn, LOW);
    delay(spd);               // wait for a second
  }
}

void moveOut(boolean in){ // true for out
  move(130, !in, turn2, dir2, 5);
  if(in) posR++;
  if(!in) posR--;
}
```

```
void moveOver(boolean inc){ //true for right
  move(20, inc, turn1, dir1, 8);
  if(inc) posT++;
  if(!inc) posT--;
}

void moveTo(int theta, int rad){
  digitalWrite(mag, HIGH);
  while(posR != 2){
    if(posR > 2) moveOut(false);
    else moveOut(true);
  }
  while(posT != theta){
    if(theta > posT) moveOver(true);
    else moveOver(false);
  }
  while(posR != rad){
    if(posR > rad) moveOut(false);
    else moveOut(true);
  }
  digitalWrite(mag, LOW);
}

void straitTo(int theta, int rad){
  while(posR != rad){
    if(posR > rad) moveOut(false);
    else moveOut(true);
  }
  while(posT != theta){
    if(theta > posT) moveOver(true);
    else moveOver(false);
  }

}

void zero(){
  int spd = 7;
  digitalWrite(dir1, LOW);
  digitalWrite(dir2, LOW);
  while(digitalRead(lim) == LOW){
    digitalWrite(turn2, HIGH);
    delay(spd);               // wait for a second
    digitalWrite(turn2, LOW);
    delay(spd);               // wait for a second
  }
  while(digitalRead(zer) == HIGH){
    digitalWrite(turn1, HIGH);
    delay(spd);               // wait for a second
    digitalWrite(turn1, LOW);
    delay(spd);               // wait for a second
  }
  posT = 0;
  posR = 3;
```

```
}

void colorSet(int player){
  switch (player){
    case 0:
      colorWipe(strip.Color(127, 0, 0), 10); // Red
      break;
   case 1:
      colorWipe(strip.Color(0, 0, 127), 10); // Blue
      break;
   case 2:
      colorWipe(strip.Color(0, 127, 0), 10); // Green
      break;
   case 3:
      colorWipe(strip.Color(127, 127, 0), 10); // Yellow
      break;
   case 4:
      colorWipe(strip.Color(0, 0, 0), 10); // Yellow
      break;
  }
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, c);
      strip.show();
      delay(wait);
  }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else if(WheelPos < 170) {
    WheelPos -= 85;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  }
}

void introSong(){
  int melody1[] = {NOTE_A4, NOTE_C5, NOTE_E5, NOTE_E5, NOTE_F5, NOTE_C5,
NOTE_E5};
  int noteDurations1[] = {8, 8, 8, 8, 8, 8, 8};
  for (int thisNote = 0; thisNote < 7; thisNote++) {
    int noteDuration = 2000 / noteDurations1[thisNote];
    tone(speaker, melody1[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
```

```
    delay(pauseBetweenNotes);
    noTone(speaker);
  }
}

void strtSong(){
  int melody1[] = {NOTE_A5, NOTE_B5, NOTE_C6};
  int noteDurations1[] = {16, 16, 16};
  for (int thisNote = 0; thisNote < 3; thisNote++) {
    int noteDuration = 2000 / noteDurations1[thisNote];
    tone(speaker, melody1[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(speaker);
  }
}

void homeSong(){
  int melody1[] = {NOTE_C6, NOTE_D6, NOTE_F6, NOTE_E6, NOTE_D6};
  int noteDurations1[] = {4, 4, 16, 16, 4};
  for (int thisNote = 0; thisNote < 5; thisNote++) {
    int noteDuration = 2000 / noteDurations1[thisNote];
    tone(speaker, melody1[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(speaker);
  }
}

void rollSong(){
  int melody1[] = {NOTE_C3, NOTE_D3};
  int noteDurations1[] = {32, 32, 32};
  for (int thisNote = 0; thisNote < 3; thisNote++) {
    int noteDuration = 2000 / noteDurations1[thisNote];
    tone(speaker, melody1[thisNote], noteDuration);
//    int pauseBetweenNotes = noteDuration/2;
//    delay(pauseBetweenNotes);
    noTone(speaker);
  }
}

void outSong(){
  int melody2[] = {NOTE_A2, NOTE_A2, NOTE_A2, NOTE_A2, NOTE_C3, NOTE_B2,
NOTE_PAUSE, NOTE_B2, NOTE_A2, NOTE_A2, NOTE_A2, NOTE_A2};
  int noteDurations2[] = {4, 6, 8, 4, 6, 8, 120, 6, 8, 6, 8, 2};
  for (int thisNote = 0; thisNote < 12; thisNote++) {
    int noteDuration = 2000 / noteDurations2[thisNote];
    tone(speaker, melody2[thisNote], noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(speaker);
  }
}
```