```
#include "U8glib.h"

const byte backlight = 11;
const byte joyX = A0;
const byte joyY = A1;
const byte joyClick = 3;
const byte randomPin = A5;

volatile bool buttonPressed = false;

U8GLIB_PCD8544 u8g(8, 4, 7, 5, 6);  // CLK=8, DIN=4, CE=7, DC=5, RST=6

typedef struct{
  byte xCoor;
  byte yCoor;
} point;

short joyXCenter = 510;
short joyYCenter = 510;

bool boolRegisters[5] = {false, false, false, false, false};

unsigned long lastTimer = 0;

unsigned short highScore = 0;
unsigned short score = 0;
byte snakeLength = 10;
point snake[80] = {{21, 24}, {21, 25}, {21, 26}, {21, 27}, {21, 28}, {21, 29}, {21, 30},
{21, 31}, {21, 32}, {21, 33}};
point pointRegisters[10] = {{0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0}, {0, 0},
{0, 0}, {0, 0}};

//0 is up, 1 is left, 2 is down, 3 is right
byte snakeDirection = 0;


void setup() {
  u8g.begin();
  pinMode(backlight, OUTPUT);
  digitalWrite(backlight, HIGH);
  pinMode(joyClick, INPUT_PULLUP);
  //interrupt for joystick button on pin 3
  attachInterrupt(1, joyInterrupt, FALLING);
  // put your setup code here, to run once:
  //u8g.setFont(u8g_font_profont10);
  //u8g.setFont(u8g_font_courR08);
  //u8g.setFont(u8g_font_6x10);
  u8g.setFont(u8g_font_trixel_square);
  setupJoy();
  Serial.begin(9600);
  //use analog pin a5 as the random number seed
  randomSeed(analogRead(randomPin));
}



void loop() {
  u8g.firstPage();
  // put your main code here, to run repeatedly:
  do {
    draw();
  } while( u8g.nextPage() );
```

```
      myPoint -> xCoor = 0;
      myPoint -> yCoor = 0;
      snakeLength = 10;
      //set snake back to original position
      for(byte countSnake = 0; countSnake < 10; countSnake++){
        snake[countSnake].xCoor = 21;
        snake[countSnake].yCoor = 24 + countSnake;
      }
      myPoint -> xCoor = 0;
      myPoint -> yCoor = 0;
    }
  }
}



//depending on the directin the snake is currently heading, increment by 1 unit
void snakeMove(byte snakeDirection){
  for(byte countSnake = snakeLength - 1; countSnake > 0; countSnake--){
    snake[countSnake] = snake[countSnake - 1];
  }
  switch(snakeDirection){
   //up
   case 0: snake -> yCoor = snake -> yCoor - 1;
           break;
   //left
   case 1: snake -> xCoor = snake -> xCoor - 1;
           break;
   //down
   case 2: snake -> yCoor = snake -> yCoor + 1;
           break;
   //right
   case 3: snake -> xCoor = snake -> xCoor + 1;
           break;
   }

   if(score < 50) delay(150);
   else if(score < 80) delay(130);
   else if(score < 100) delay(120);
   else if(score < 300) delay(100);
   else if(score < 500) delay(80);
   else if(score < 1000) delay(50);
   else if(score < 2000) delay(30);
   else delay(20);
}



//generates a random point if the flag is true and saves it in pointRegisters[0] and sets
the flag to false
//parameter: flag the flag that signals the main code that a point need to be generated
//returns true if new point is generated, else false
bool genRandomPoint(bool* flag){
  if(*flag == true){
    //generate a new point
    point* myPoint = pointRegisters;
    pointRegisters -> xCoor = random(2, 42);
```

```
  modeAction();
}


//uses boolRegisters[0] as game start
//uses boolRegisters[1] as game pause
//uses boolRegisters[2] as game over
//uses boolRegisters[3] as generateNewPoint flag
//use pointRegisters[1] as point destination
void modeAction(){
  bool* gameStart = &boolRegisters[0];
  bool* gamePause = &boolRegisters[1];
  bool* gameOver = &boolRegisters[2];
  bool* genNewPoint = &boolRegisters[3];
  point* myPoint = pointRegisters;

  //check if the game has ended
  (*gameOver) = gameOverCheck();
  (*genNewPoint) = checkRandomPoint(myPoint);

  //when game first starts
  if(buttonPressed == true && *gameStart == false && *gameOver == false && *gamePause ==
false){
    *genNewPoint = true;
    *gameStart = true;
    buttonPressed = false;
    genRandomPoint(genNewPoint);

  //if game is running and not over yet and button is pressed, then pause and unpause the
game
  } else if(buttonPressed == true && *gameStart == true && *gameOver == false){
    *gamePause = !(*gamePause);
    buttonPressed = false;
  } else if(*gameStart == true && *gamePause == false && *gameOver == false){

    //determine if the direction of the joystick is changed
    if(analogRead(joyX) < 5 && snakeDirection != 3){
      snakeDirection = 1;
    }else if(analogRead(joyX) > 1020 && snakeDirection != 1){
      snakeDirection = 3;
    }else if(analogRead(joyY) < 5 && snakeDirection != 2){
      snakeDirection = 0;
    }else if(analogRead(joyY) > 1020 && snakeDirection != 0){
      snakeDirection = 2;
    }

    genRandomPoint(genNewPoint);
    snakeMove(snakeDirection);
  //when game over
  }else if(*gameOver == true){
    *gameStart = false;
    *genNewPoint = false;
    //if button is pressed again, restart the game
    if(buttonPressed == true){
      //resets all values
      *gameStart = false;
      *gameOver = false;
      *gamePause = false;
      *genNewPoint = false;
      buttonPressed = false;
      score = 0;
```

```
    pointRegisters -> yCoor = random(2, 33);
    *flag = false;

    return true;
  }

  return false;
}



//determines if point is acquired, and increments point if it is
//parameter: the location of the point as a point pointer
//returns true is point is acquired, false otherwise
bool checkRandomPoint(point* myPoint){

  if(snake -> xCoor == myPoint -> xCoor && snake -> yCoor == myPoint -> yCoor){
    snake[snakeLength].xCoor = snake[snakeLength - 1].xCoor;
    snake[snakeLength].yCoor = snake[snakeLength - 1].yCoor;
    snakeLength = snakeLength + 1;

    //score growth is proportional to the score
    //score = score + map(score, 0, 600, 2, 30);
    if(score <= 10) score = score + 3;
    else if(score <= 30) score = score + 10;
    else if(score <= 100) score = score + 25;
    else if(score <= 250) score = score + 50;
    else if(score <= 500) score = score + 90;
    else if(score <= 1000) score = score + 150;
    else if(score <= 1500) score = score + 250;
    else score = score + 500;

    return true;
  }

  return false;
}



//check if the game is over, checks both the boundary and the snake
bool gameOverCheck(){
  //checks if the snake intersects itself
  for(byte countSnake = 1; countSnake < snakeLength; countSnake++){
    if(snake -> xCoor == snake[countSnake].xCoor && snake -> yCoor ==
snake[countSnake].yCoor)
      return true;
  }

  //if the head of the snake goes beyond boundary
  /*
  if(snake -> xCoor <= 20 || snake -> xCoor >= 62 || snake -> yCoor <= 0 || snake -> yCoor
>= 34){
    return true;
  }
  */
  if(snake -> xCoor <= 0 || snake -> xCoor >= 42 || snake -> yCoor <= 0 || snake -> yCoor >=
34){
    return true;
  }
```

```
  return false;
}


//canvas area of the snake is 0 to 43 for x, 0 to 35 for y
void draw() {
    bool* gameStart = &boolRegisters[0];
    bool* gamePause = &boolRegisters[1];
    bool* gameOver = &boolRegisters[2];

    point* myPoint = pointRegisters;

    u8g.drawStr(44, 6, F("SCORE:"));
    u8g.setPrintPos(44, 13);
    u8g.print(score);
    u8g.drawStr(44, 21, F("HIGHSCORE:"));
    u8g.setPrintPos(44, 29);
    u8g.print(highScore);

    //u8g.drawFrame(20,0,43,35);
    u8g.drawFrame(0,0,43,35);

    //game hasn't started yet
    if(*gameStart == false && *gameOver == false){
      u8g.drawStr(21, 44, F("PUSH TO START"));
      for(byte countSnake = 0; countSnake < snakeLength; countSnake++)
        u8g.drawPixel(snake[countSnake].xCoor, snake[countSnake].yCoor);

    //game started but not paused or game over
    } else if((*gamePause) == false && (*gameOver) == false){
      for(byte countSnake = 0; countSnake < snakeLength; countSnake++)
        u8g.drawPixel(snake[countSnake].xCoor, snake[countSnake].yCoor);

    //game paused
    } else if((*gamePause) == true){
     u8g.drawStr(23, 44, F("GAME PAUSED"));
     for(byte countSnake = 0; countSnake < snakeLength; countSnake++)
        u8g.drawPixel(snake[countSnake].xCoor, snake[countSnake].yCoor);

    //game over
    } else if((*gameOver) == true){
      u8g.drawStr(25, 44, F("GAME OVER"));
      for(byte countSnake = 0; countSnake < snakeLength; countSnake++)
        u8g.drawPixel(snake[countSnake].xCoor, snake[countSnake].yCoor);
    }

    if(*gameStart == true) u8g.drawPixel(myPoint -> xCoor, myPoint -> yCoor);
}


//triggered by interrupt sets the button press flag to true
void joyInterrupt(){
  if(millis() - lastTimer > 800)
    buttonPressed = true;
  lastTimer = millis();
}


void setupJoy(){
```

```
  pinMode(joyX, INPUT);
  pinMode(joyY, INPUT);
  pinMode(joyClick, INPUT_PULLUP);

  //calibrate the centre positions of the joystick
  short sumJoy = 0;
  for(byte countJoy = 0; countJoy < 10; countJoy++) sumJoy = sumJoy + analogRead(joyX);
  joyXCenter = sumJoy / 10;
  sumJoy = 0;
  for(byte countJoy = 0; countJoy < 10; countJoy++) sumJoy = sumJoy + analogRead(joyY);
  joyYCenter = sumJoy / 10;
}
```