2025

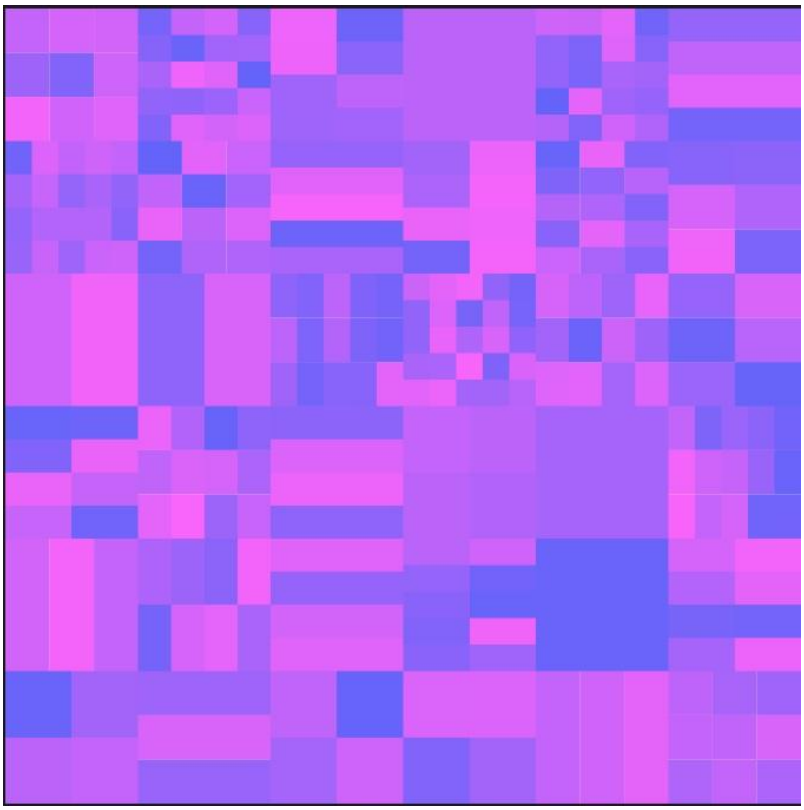
Εργασία 3

ΠΛΗΡΟΦΟΡΙΚΗ 2ΟΥ ΕΞΑΜΗΝΟΥ

ΕΛΙΣΑΒΕΤ ΚΑΖΑΝΗ-AR24612

Πρόβλημα 1

Στην παρούσα εργασία δημιουργήθηκε κώδικας σε p5.js για τη δημιουργία της παρακάτω διαδραστικής εικόνας.



Στον κώδικα χρησιμοποιήθηκαν οι μέθοδοι όπως παρακάτω:

- `createCanvas(w, h)` Δημιουργεί τον καμβά
- `noStroke()` Αφαιρεί περίγραμμα από σχήματα
- `background(c)` Ορίζει το χρώμα φόντου
- `fill(h, s, b)` Ορίζει το χρώμα γεμίσματος
- `rect(x, y, w, h)` Σχεδιάζει ορθογώνιο
- `random(a, b)` Επιστρέφει τυχαίο αριθμό στο [a, b)
- `int(n)` Στρογγυλοποιεί αριθμό προς τα κάτω σε ακέραιο

Παρακάτω παρουσιάζονται αναλυτικά ο ψευδοκώδικας, ο κώδικας και το link του online editor (editor.p5js.org).

Ψευδοκώδικας

ΑΡΧΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

1. ΟΡΙΣΕ gridData ως 5-διάστατο array
2. ΟΡΙΣΕ rows = 5
3. ΟΡΙΣΕ cols = 5
4. ΣΥΝΑΡΤΗΣΗ setup()
5. Δημιούργησε καμβά 600 x 600 pixels
6. Απενεργοποίησε το περίγραμμα των σχημάτων
7. ΚΑΛΕΣΕ createGrid(rows, cols) και ΑΠΟΘΗΚΕΥΣΕ το αποτέλεσμα σε gridData
8. ΣΥΝΑΡΤΗΣΗ draw()
    a. Καθάρισε τον καμβά (λευκό φόντο)
    b. ΚΑΛΕΣΕ drawGrid(gridData)
9. ΣΥΝΑΡΤΗΣΗ createGrid(rows, cols)
    a. ΑΡΧΙΚΟΠΟΙΗΣΕ κενό πίνακα grid
    b. ΥΠΟΛΟΓΙΣΕ cellW = πλάτος καμβά / cols
    c. ΥΠΟΛΟΓΙΣΕ cellH = ύψος καμβά / rows
    d. ΓΙΑ κάθε γραμμή i από 0 έως rows-1
        i. ΓΙΑ κάθε στήλη j από 0 έως cols-1
            1. ΔΗΜΙΟΥΡΓΗΣΕ κενό array grid[i][j]
            2. ΕΠΙΛΕΞΕ τυχαίους ακέραιους subRows και subCols από 1 έως 4
            3. Για κάθε υπογραμμή m από 0 έως subRows-1
                a. ΓΙΑ κάθε υποστήλη n από 0 έως subCols-1
                b. ΥΠΟΛΟΓΙΣΕ x θέση υποκελιού
                c. ΥΠΟΛΟΓΙΣΕ y θέση υποκελιού
                d. ΥΠΟΛΟΓΙΣΕ w (πλάτος υποκελιού)
                e. ΥΠΟΛΟΓΙΣΕ h (ύψος υποκελιού)
                f. ΔΗΜΙΟΥΡΓΗΣΕ τυχαία απόχρωση hue (π.χ. 100–150)
                g. ΑΠΟΘΗΚΕΥΣΕ [x, y, w, h, hue] στο grid[i][j][m][n]
10. ΕΠΕΣΤΡΕΨΕ τον πίνακα grid
11. ΣΥΝΑΡΤΗΣΗ drawGrid(grid)
    a. ΓΙΑ κάθε κελί grid[i][j][m][n] στο 5-διάστατο array
        i. ΑΝΑΚΤΗΣΕ x, y, w, h, hue
        ii. ΟΡΙΣΕ χρώμα γεμίσματος με βάση το hue
        iii. ΣΧΕΔΙΑΣΕ ορθογώνιο στις συντεταγμένες (x, y) με διαστάσεις (w, h)
12. ΣΥΝΑΡΤΗΣΗ mousePressed()
    a. ΓΙΑ κάθε υποκελί grid[i][j][m][n]
        i. ΑΝ το mouseX, mouseY είναι μέσα στο ορθογώνιο (x, y, w, h)
        ii. ΤΟΤΕ άλλαξε την τιμή hue με νέα τυχαία τιμή (π.χ. 200–250)

ΤΕΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
//Δημιουργία 5-διάστατου πίνακα για αποθήκευση όλων των πληροφοριών
let gridData = [];

let rows = 6; //Πλήθος γραμμών του αρχικού κανάβου
let cols = 6; //Πλήθος στηλών του αρχικού κανάβου

function setup() {
  createCanvas(600, 600); //Δημιουργία καμβά
  noStroke(); //Χωρίς περίγραμμα στα σχήματα

    //Κλήση συνάρτησης δημιουργίας κανάβου και αποθήκευση του πίνακα
    gridData = createGrid(rows, cols);
}

function draw() {
  background(255); //Καθαρισμός καμβά με λευκό
  drawGrid(gridData); //Σχεδίαση του κανάβου με βάση τα δεδομένα του πίνακα
}

function createGrid(rows, cols) {
  let grid = [];
  let cellW = width / cols; //Πλάτος κάθε κύριου κελιού
  let cellH = height / rows; //Ύψος κάθε κύριου κελιού

  //Βρόχοι για κάθε γραμμή και στήλη του βασικού κανάβου
  for (let i = 0; i < rows; i++) {
    grid[i] = [];
    for (let j = 0; j < cols; j++) {
        grid[i][j] = [];

        //Βήμα 2: Τυχαίος αριθμός υπογραμμών και υποστηλών για κάθε κύριο κελί
        let subRows = int(random(1, 6));
        let subCols = int(random(1, 6));

        //Βρόχοι για κάθε υπογραμμή και υποστήλη
        for (let m = 0; m < subRows; m++) {
          grid[i][j][m] = [];
          for (let n = 0; n < subCols; n++) {
              //Υπολογισμός θέσης και διαστάσεων του υποκελιού
              let x = j * cellW + (n * cellW) / subCols;
              let y = i * cellH + (m * cellH) / subRows;
              let w = cellW / subCols;
              let h = cellH / subRows;

              //Δημιουργία τυχαίας απόχρωσης (π.χ. μπλε τόνων)
              let hue = random(100, 250);
```

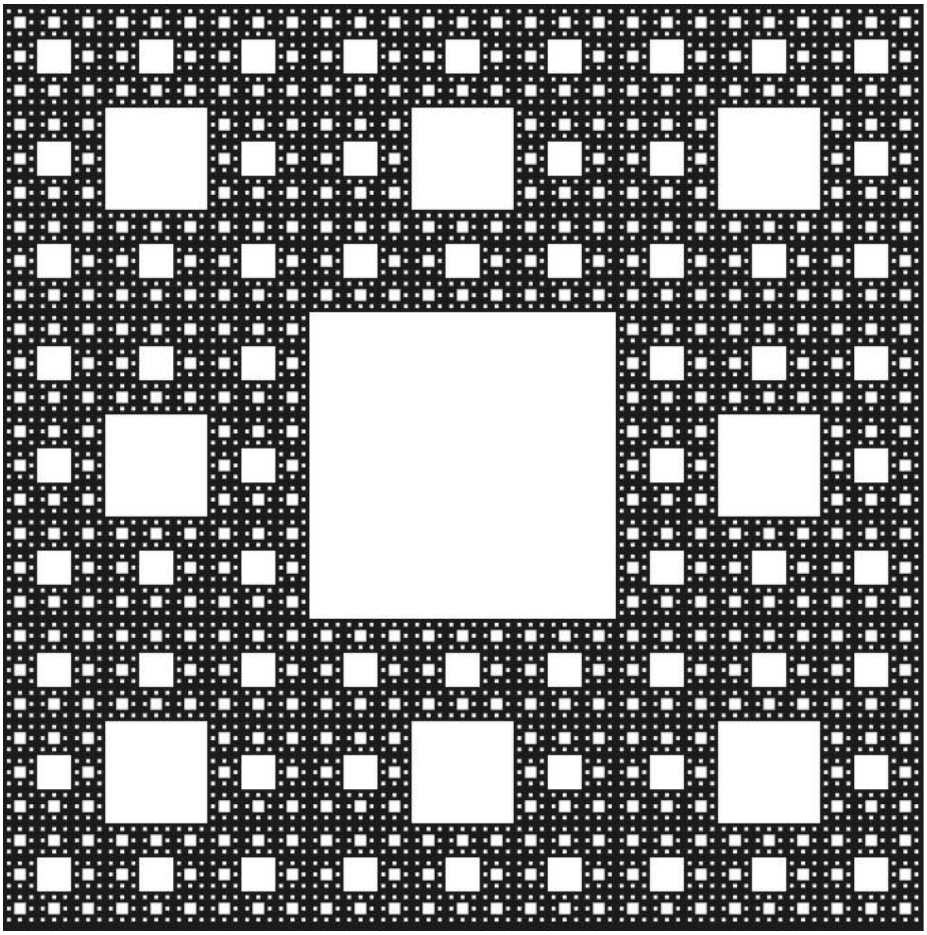
```
                    //Αποθήκευση πληροφοριών για κάθε υποκελί: x, y, πλάτος, ύψος, χρώμα
                    grid[i][j][m][n] = [x, y, w, h, hue];
                }
            }
        }
    }
  return grid; //Επιστροφή του πίνακα
}

function drawGrid(grid) {
  //Πλήρης διάσχιση του 5D πίνακα για σχεδίαση κάθε υποκελιού
  for (let i = 0; i < grid.length; i++) {
    for (let j = 0; j < grid[i].length; j++) {
      for (let m = 0; m < grid[i][j].length; m++) {
        for (let n = 0; n < grid[i][j][m].length; n++) {
            //Ανάκτηση τιμών υποκελιού
          let [x, y, w, h, hue] = grid[i][j][m][n];
          fill(hue, 100, 250); //Ορισμός χρώματος με βάση την απόχρωση
          rect(x, y, w, h); //Σχεδίαση ορθογωνίου
        }
      }
    }
  }
}

function mousePressed() {
  //Έλεγχος σε όλα τα υποκελιά για το αν έχει γίνει κλικ πάνω τους
  for (let i = 0; i < gridData.length; i++) {
    for (let j = 0; j < gridData[i].length; j++) {
      for (let m = 0; m < gridData[i][j].length; m++) {
        for (let n = 0; n < gridData[i][j][m].length; n++) {
          let cell = gridData[i][j][m][n];
          let [x, y, w, h, hue] = cell;

          //Έλεγχος αν οι συντεταγμένες του ποντικιού είναι μέσα στο υποκελί
          if(mouseX > x && mouseX < x + w && mouseY > y && mouseY < y + h) {
            //Αλλαγή απόχρωσης σε νέα τιμή για να γίνει επανασχεδίαση με νέο χρώμα
            cell[4] = random(100, 250);
          }
        }
      }
    }
  }
}
```

Link online editor

https://editor.p5js.org/elisakazani/sketches/JoSwlR9S3 (κώδικας)
https://editor.p5js.org/elisakazani/full/JoSwlR9S3 (αποτέλεσμα)

Πρόβλημα 2

Στην παρούσα εργασία δημιουργήθηκε κώδικας σε p5.js για τη δημιουργία του παρακάτω καμβά.



Στον κώδικα χρησιμοποιήθηκαν οι μέθοδοι όπως παρακάτω:

- **createCanvas(600, 600)**: Δημιουργεί έναν καμβά 600x600 pixels.
- **noStroke()**: Απενεργοποιεί το περίγραμμα στα σχήματα (ώστε να φαίνονται καθαρά τα "κενά" τετράγωνα).
- **fill(255)**: Ορίζει το χρώμα γεμίσματος των τετραγώνων (λευκό = RGB(255)).
- **drawCarpet(0, 0, width)**: Ξεκινά την αναδρομική σχεδίαση από το πάνω αριστερό σημείο (0,0) με αρχικό μέγεθος όσο το πλάτος του καμβά.

Παρακάτω παρουσιάζονται αναλυτικά ο ψευδοκώδικας, ο κώδικας και το link του online editor (editor.p5js.org).

Ψευδοκώδικας

ΑΡΧΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Συνάρτηση setup()

Δημιουργία καμβά με διαστάσεις 600 x 600

Απενεργοποίηση περιγράμματος σχημάτων

Ορισμός χρώματος γεμίσματος σε λευκό

Κλήση drawCarpet(x = 0, y = 0, μέγεθος = πλάτος καμβά)

Συνάρτηση drawCarpet(x, y, s)

Αν s < 5 ΤΟΤΕ

ΕΠΙΣΤΡΟΦΗ (τερματισμός αναδρομής)

ΤΕΛΟΣ_ΑΝ

Ορισμός newS = s / 3 // νέο μέγεθος κάθε υπο-τετραγώνου

// Ζωγράφισε το κεντρικό τετράγωνο (το "κενό")

Ζωγράφισε τετράγωνο στη θέση (x + newS, y + newS) με διαστάσεις newS x newS

// Επανέλαβε τη διαδικασία για τα 8 υπόλοιπα τετράγωνα

ΓΙΑ dx από 0 μέχρι 2

ΓΙΑ dy από 0 μέχρι 2

Αν dx = 1 ΚΑΙ dy = 1 ΤΟΤΕ

ΣΥΝΕΧΙΣΕ (παράλειψη του κέντρου)

ΤΕΛΟΣ_ΑΝ

Κλήση drawCarpet(x + dx * newS, y + dy * newS, newS)

ΤΕΛΟΣ_ΓΙΑ

ΤΕΛΟΣ_ΓΙΑ

ΤΕΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Κώδικας

```
function setup() {
  createCanvas(600, 600); // Δημιουργεί έναν καμβά 600x600 pixels
  noStroke();             // Απενεργοποιεί το περίγραμμα των σχημάτων
  fill(255);              // Ορίζει το χρώμα γεμίσματος σε λευκό (χρησιμοποιείται για τα "κενά")
  drawCarpet(0, 0, width); // Ξεκινά το σχέδιο από το πάνω αριστερό σημείο του καμβά με πλήρες μέγεθος
}

function drawCarpet(x, y, s) {
  if (s < 5) {
    return; // Αν το μέγεθος του τετραγώνου είναι πολύ μικρό, σταματά η αναδρομή
  }

  let newS = s / 3; // Διαιρεί το τετράγωνο σε 9 μικρότερα τετράγωνα, άρα το μέγεθος του καθενός είναι το 1/3

  // Ζωγραφίζει το κεντρικό τετράγωνο ως "κενό" (λευκό)
  rect(x + newS, y + newS, newS, newS);

  // Επαναλαμβάνει τη διαδικασία αναδρομικά για τα 8 τετράγωνα γύρω από το κέντρο
  for (let dx = 0; dx < 3; dx++) {
    for (let dy = 0; dy < 3; dy++) {
      if (dx === 1 && dy === 1) continue; // Αν βρισκόμαστε στο κέντρο, το προσπερνάμε (ήδη σχεδιάστηκε ως "κενό")
      // Κλήση της ίδιας συνάρτησης για κάθε από τα 8 υπόλοιπα τετράγωνα
      drawCarpet(x + dx * newS, y + dy * newS, newS);
    }
  }
}
```

Link online editor

https://editor.p5js.org/elisakazani/sketches/PLRejZk8g (Κώδικας)
https://editor.p5js.org/elisakazani/full/PLRejZk8g (Αποτέλεσμα)

Πρόβλημα 3

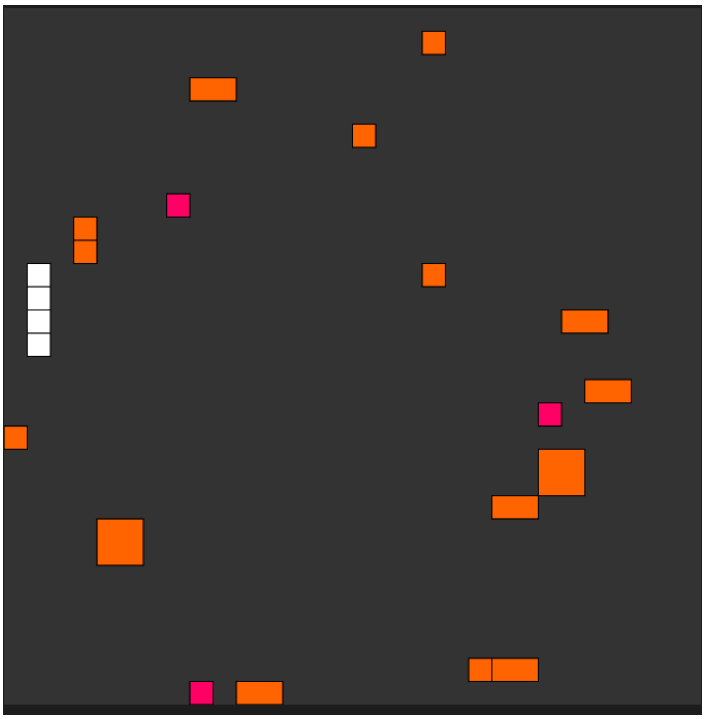
COMPUTER GAME

Παιχνίδι Φιδάκι με Εμπόδια και Πολλαπλά Είδη Τροφής

Στόχος στην παρούσα εργασία είναι η δημιουργία παιχνιδιού τύπου Φιδάκι.Το παιχνίδι "Φιδάκι με Πολλαπλά Είδη Τροφής και Εμπόδια" είναι υλοποιημένο με χρήση της βιβλιοθήκης **p5.js** και αποτελείται από **δύο αρχεία κώδικα**:

- **sketch.js**: περιέχει τον κύριο κορμό του παιχνιδιού (ρύθμιση παραμέτρων, εμφάνιση, σύγκρουση με φαγητό και εμπόδια).
- **snake_class.js:** περιέχει την κλάση Snake, δηλαδή τη λογική για τη συμπεριφορά και κίνηση του φιδιού.

Ο διαχωρισμός αυτός βοηθά στη **σαφή δομή του προγράμματος**, καθώς διαχωρίζεται η λογική του "κόσμου του παιχνιδιού" από τη λογική του "φιδιού".



Στον κώδικα χρησιμοποιήθηκαν μέθοδοι όπως οι παρακάτω:

- **frameRate()**: Για τον ορισμό της ταχύτητας παιχνιδιού
- **pickLocation()** : Για την δημιουργία συγκεκριμένου αριθμού τροφών σε τυχαίες θέσεις
- **placeObstacles()** : Για την τοποθέτηση εμποδίων σε τυχαία θέση και μέγεθος
- **console.log(obstacles()**: Για τον έλεγχο των εμποδίων στο κονσόλα
- **keyPressed()**: Για τον έλεγχο του παιχνιδιού με τα βελάκια
- Σχεδίαση και έλεγχος σύγκρουσης με εμπόδια
- Ανίχνευση σύγκρουσης (Axis-Aligned Bounding Box)
- Σχεδίαση και έλεγχος για φαγητό
- Έλεγχος αν το φίδι έφαγε τον εαυτό του
- Μεγαλώνει το φίδι χειροκίνητα (δοκιμαστικά)

Παρακάτω παρουσιάζονται αναλυτικά ο ψευδοκώδικας, ο κώδικας και το link του online editor (editor.p5js.org).

Ψευδοκώδικας για κάθε αρχείο: sketch.js, snake_class.js

sketch.js (Κύριος έλεγχος και ροή του παιχνιδιού):

Setup():

1. Δημιούργησε καμβά 600x600
2. Ρύθμισε **frameRate** (ταχύτητα παιχνιδιού)
3. Δημιούργησε νέο αντικείμενο Snake
4. Τοποθέτησε 3 τυχαίες τροφές
5. Τοποθέτησε 10 εμπόδια σε τυχαία σημεία

Draw():

1. Καθάρισε την οθόνη
2. Κάλεσε update() και show() του Snake
3. Για κάθε εμπόδιο:
    a. Σχεδίασέ το
    b. Ανιχνεύεται σύγκρουση με το φίδι (AABB) → σταμάτα παιχνίδι
4. Για κάθε τροφή:
    a. Σχεδίασέ την
    b. Αν την φάει το φίδι:
        i. Αύξησε μήκος
    c. Τοποθέτησε ξανά 3 τροφές
5. Κάλεσε **death()** του **Snake** (έλεγχος αυτοσύγκρουσης) **keyPressed()**
6. Αν πατηθεί UP, DOWN, LEFT, RIGHT → άλλαξε κατεύθυνση φιδιού

mousePressed()

1. Πρόσθεσε χειροκίνητα μήκος στο φίδι (για δοκιμή)

snake_class.js (Βοηθητική Συνάρτηση Φιδιού):

Constructor()

1. Αρχικοποιεί x, y
2. Θέτει αρχική ταχύτητα προς τα δεξιά
3. Ουρά = άδειος πίνακας, total = 0

dir(x, y) ()

- Θέτει νέα κατεύθυνση μετακίνησης

eat(pos)()

1. Αν απόσταση (φίδι, τροφή) < 1:
    a. αύξησε total
    b. true
2. αλλιώς επέστρεψε false

death()

1. Για κάθε κομμάτι της ουράς:
    a. Αν απόσταση από το κεφάλι < 1:
        i. Επανεκκίνηση παιχνιδιού (tail = [], total = 0)

update()

1. Μετακινεί την ουρά (όταν μεγαλώνει)
2. Προσθέτει νέα θέση κεφαλής στην ουρά
3. Μετακινεί φίδι σύμφωνα με κατεύθυνση
4. Περιορίζει μέσα στο καμβά

show()

1. Σχεδιάζει κάθε τμήμα ουράς με rect()
2. Σχεδιάζει την κεφαλή του φιδιού

ΕΠΕΞΗΓΗΣΗ ΣΥΓΚΡΟΥΣΗΣ (AABB) - Axis-Aligned Bounding Box:

Αν το ορθογώνιο του φιδιού (π.χ. 20x20) επικαλύπτει το ορθογώνιο του εμποδίου:

```
if (
  s.x < obs.x + obs.w &&
  s.x + scl > obs.x &&
  s.y < obs.y + obs.h &&
  s.y + scl > obs.y
)
=> υπάρχει σύγκρουση!
```

```
//skets.js My snake_game_4
let s;
let scl = 20;
let food = [];
let obstacles = [];// πίνακας για τα εμπόδια

function setup() {
  createCanvas(600, 600)
  s = new Snake();  //δημιουργία του φιδιού
  frameRate(5);// ορίστε την ταχύτητα του παιχνιδιού
  pickLocation(3);// τοποθετούμε το φαγητό
  placeObstacles(15);// τοποθετούμε 15 εμπόδια
  console.log(obstacles); // Δες εδώ αν δημιουργήθηκαν σωστά
  }

function pickLocation(count) {
  food = [];
  let cols = floor(width/scl);
  let rows = floor(height/scl);
```

```
  for (let i = 0; i < count; i++) {
    let f = createVector(floor(random(cols)), floor(random(rows)));
    f.mult(scl);
    food.push(f);
  }
}

function placeObstacles(count) {
  for (let i = 0; i < count; i++) {
    let cols = floor(width / scl);// υπολογισμός στηλών
    let rows = floor(height / scl);// υπολογισμός γραμμών
    let pos = createVector(floor(random(cols)), floor(random(rows)));
    pos.mult(scl);

    let w = random([1, 2]) * scl;
    let h = random([1, 2]) * scl;
    obstacles.push({ pos: pos, w: w, h: h });
  }
}

function mousePressed() {
  s.total++;// αυξανει το μεγεθος του με το πατημα του πληκτρολογίου
}

function draw () {
  background(51);// καθορισμός φόντου

   s.update();// έλεγχος για σύγκρουση με το φίδι
   s.show();// ενημέρωση της θέσης του φιδιού

 for (let i = 0; i < obstacles.length; i++) {
  fill(255, 100, 0); // χρώμα εμποδίων
  rect(obstacles[i].pos.x, obstacles[i].pos.y, obstacles[i].w, obstacles[i].h);


    if (
      s.x < obstacles[i].pos.x + obstacles[i].w &&
      s.x + scl > obstacles[i].pos.x &&
      s.y < obstacles[i].pos.y + obstacles[i].h &&
      s.y + scl > obstacles[i].pos.y
    ) {
      console.log("Χτύπησες εμπόδιο!");
      noLoop();
    }
  }
  // Φαγητά
  for (let i = 0; i < food.length; i++) {
    fill(255, 0, 100);
    rect(food[i].x, food[i].y, scl, scl);
    if (s.eat(food[i])) {
      pickLocation(3);
      break;
    }
```

```
  }

  s.death();
}

function mousePressed() {
  s.total++;
}

function keyPressed() {
  if (keyCode === UP_ARROW) {
       s.dir(0, -1);
  }  else if (keyCode === DOWN_ARROW) {
       s.dir(0, 1);
          } else if (keyCode === LEFT_ARROW) {
      s.dir(-1, 0);
    } else if (keyCode === RIGHT_ARROW) {
      s.dir(1, 0);
    };
}
```

Link online editor

https://editor.p5js.org/elisakazani/sketches/KEsEaT4Lk (Κώδικας)
https://editor.p5js.org/elisakazani/full/KEsEaT4Lk (Αποτέλεσμα)