# ARCH 4342 ΠΛΗΡΟΦΟΡΙΚΗ

## Υπολογιστικές Προσεγγίσεις στις Δημιουργικές Τέχνες και Επιστήμες
## Σειρά Ασκήσεων 3

ΤΗΣ ΧΟΥΤΣΙΣΒΙΛΙ
ΜΑΓΚΝΤΑ



ΔΙΔΑΚΤΙΚΗ ΟΜΑΔΑ: Δ. ΠΑΠΑΝΙΚΟΛΑΟΥ
ΕΠΙΚΟΥΡΙΚΟ ΕΡΓΟ: ΦΩΤΕΙΝΗ ΚΟΡΡΕ

```
let gridData = [];
let rows = 10;
let cols = 10;

function setup() {
  createCanvas(800, 800);
  gridData = buildGrid(rows, cols);
  noLoop(); // Με αυτή την εντολή ελέγχουμε εμείς πότε αλλάζει το χρώμα των ορθογωνίων
  drawGrid(); // Αρχικός χρωματισμός πλέγματος
}

function draw() {
  drawGrid();
}

// Function για να φτιάξουμε το πλέγμα
function buildGrid(rows, cols) {
  let grid = [];
  let cellWidth = width / cols;
  let cellHeight = height / rows;

  for (let a = 0; a < rows; a++) {
    grid[a] = [];
    for (let b = 0; b < cols; b++) {
      let x = b * cellWidth;
      let y = a * cellHeight;

      let subRows = int(random(1, 15)); //με αυτην την εντολή φτιαχνουμε τυχαίες γραμμές
      let subCols = int(random(1, 11)); //με αυτην την εντολή φτιαχνουμε τυχαίες στήλες

      grid[a][b] = [];

// Εντολή για τον σχεδιάσμο γράμμων και στήλων στο εσωτέρικο των κελιών
      for (let aa = 0; aa < subRows; aa++) {
        grid[a][b][aa] = [];
        for (let bb = 0; bb < subCols; bb++) {
//Μεταβλητές που ορίζουν τις διαστάσεις, την θέση και το χρώμα των κελιών
          let subX = x + bb * (cellWidth / subCols);
          let subY = y + aa * (cellHeight / subRows);
          let w = cellWidth / subCols;
          let h = cellHeight / subRows;
          let col = getRandomColour();

          grid[a][b][aa][bb] = [subX, subY, w, h, col];
        }
      }
    }
  }
  return grid;
}

// Function για τον σχεδιασμό του πλέγματος και των εσωτερικών κελιών
function drawGrid() {
  background(255);
  for (let a = 0; a < gridData.length; a++) {
    for (let b = 0; b < gridData[a].length; b++) {
      for (let aa = 0; aa < gridData[a][b].length; aa++) {
        for (let bb = 0; bb < gridData[a][b][aa].length; bb++) {
          let subcell = gridData[a][b][aa][bb];
          fill(subcell[4]);
          noStroke();
          rect(subcell[0], subcell[1], subcell[2], subcell[3]);
        }
      }
    }
  }
}

// Εντολή για τυχαίο χρώμα
function getRandomColour() {
  let g= random(0, 200)
  let r = random(0, 300)
  let b = random(0, 200);
  return color(r, g, b);
}
```

```
// Εντολή για την αλλάγη χρώματος με το κλικ του ποντικιού
function mousePressed() {
  for (let a = 0; a < gridData.length; a++) {
    for (let b = 0; b < gridData[a].length; b++) {
      for (let aa = 0; aa < gridData[a][b].length; aa++) {
        for (let bb = 0; bb < gridData[a][b][aa].length; bb++) {
          let subcell = gridData[a][b][aa][bb];
          let x = subcell[0];
          let y = subcell[1];
          let w = subcell[2];
          let h = subcell[3];

          if (mouseX >= x && mouseX <= x + w && mouseY >= y && mouseY <= y + h) {
            subcell[4] = getRandomColour(); // Κάθε φορά που πατάμε το κέλι αλλάζει χρώμα
            redraw();
            return;
          }
        }
      }
    }
  }
}
```

**https://editor.p5js.org/Magda_Khutsishvili/sketches/7sBTvI-TK**

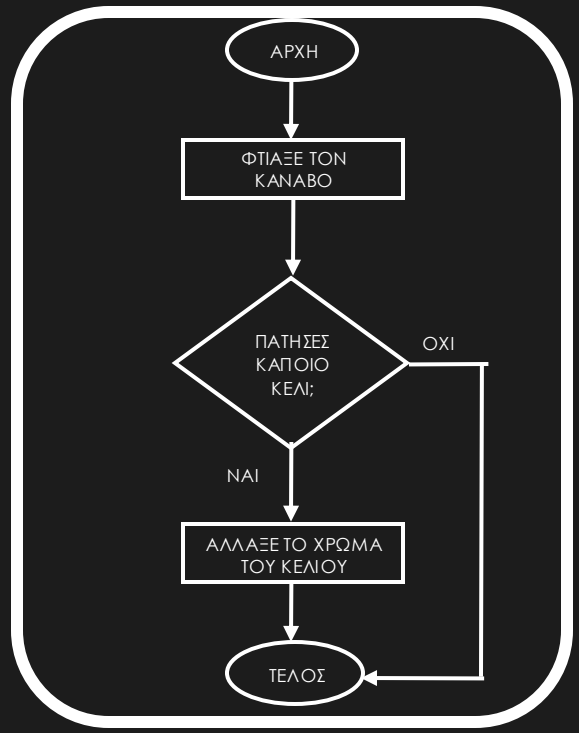
# ΠΡΟΒΛΗΜΑ 1

Στο πρόβλημα αυτό καλούμαστε να φτιάξουμε έναν κάναβο ο οποίος θα αποτελείται από μικρότερα κελιά ποικίλων μεγεθών και θερμών χρωμάτων. Η ιδέα ήταν να ακολουθήσω όσο το δυνατόν περισσότερο την εικόνα της εκφώνησης, ωστόσο προέκυψαν κάποιες διαφοροποιήσεις.

Επιλέγω πρώτα απ' όλα να μεγαλώσω τον καμβά μου για να βλέπω καλύτερα το σχέδιο(κοίτα εντολή 6. 800x800), καθώς και να γίνουν τα κελιά μου λίγο μεγαλύτερα για να μπορώ να αναγνωρίζω πιο εύκολα όταν αλλάζω χρώμα σε ένα κελί πατώντας το με το ποντίκι(κ.ε. 28-29 όπου βάζω 15 γραμμές και 11στήλες). Ορίζω μεταβλητές για γραμμές, στήλες, εσωτερικές γραμμές και εσωτερικές στήλες με τις μεταβλητές a, b, aa, bb αντίστοιχα.

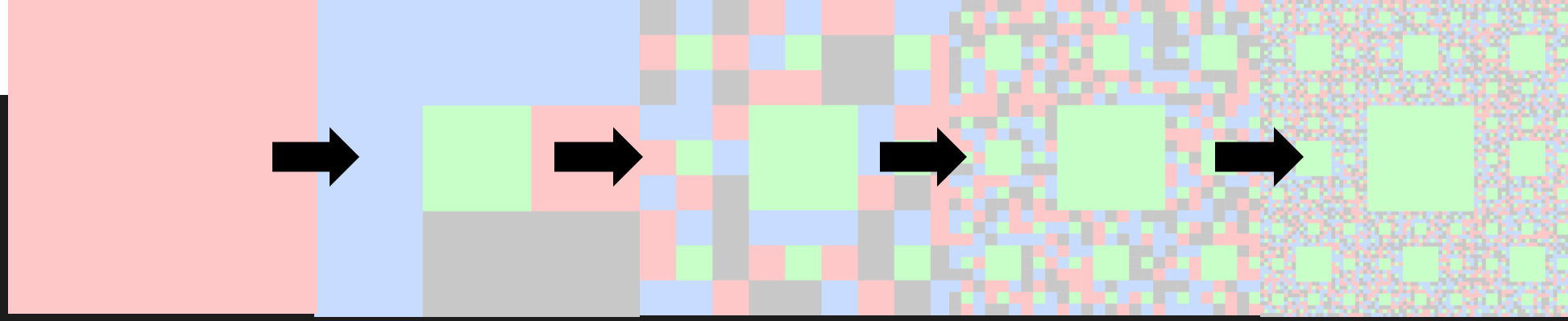
Μετέπειτα δίνω την εντολή να διαιρούνται τα κελιά μου σε μικρότερα κελιά και με κάθε update να παίρνουν ένα τυχαίο χρώμα από το εύρος που έχω ορίσει παραπάνω.
Τέλος ορίζω ότι μετά από κάθε πάτημα του ποντικιού μου πάνω στα κελιά, πρέπει να αλλάζει το χρώμα τους.

FLOWCHART



# ΠΡΟΒΛΗΜΑ 2

```
let maxDepth = 0;   // η αρχη, ξεκινάμε από το τίποτα και με κάθε κλικ ξεδιπλώνεται η πόλη σαν το χαλί του Σιερπίνσκι
let maxAllowedDepth = 5;  // σκέψου το σαν τα level λεπτομέρειας που μπορεί να φτάσει ο κωδικας
let fractalColors = [];  // αρχή του array


function setup() {
  createCanvas(900, 900);
  noStroke();
  generateFractalColors(); }


function draw() {
  background(200, 255, 200);   // ας θεωρήσουμε αυτούς τους χώρους το αστικό πράσινο
  drawFractal(0, 0, width, maxDepth, fractalColors);}


function mousePressed() {    // η εντολή που μας επιτέπει να αλλάζουμε το μοτίβο με κάθε κλικ
  maxDepth++;
  if (maxDepth > maxAllowedDepth) {
    maxDepth = 0;
  }
  generateFractalColors(); }


function generateFractalColors() {
  fractalColors = [];
  storeColors(fractalColors, maxDepth);}


function storeColors(container, level) { // Εδώ θα ορίσουμε το χρώμα και το μέγεθος των κελιών μας ,στη συγκεκριμένη περιπτωσ
ας σκεφτούμε το γκρι σαν κατοικίες, το κοκκινο σαν εμπορικοί χώροι και το μπλέ σαν δημόσιους χώρους/κτήρια
  if (level === 0) {
    let colorOptions = [
      color(200),  // γκρι για τις κατοικιες
      color(255, 200, 200), // κοκκινο για τους εμπορικούς χώρου
      color(200, 220, 255)
    ];
    container.color = random(colorOptions);
  } else {
    container.smallparks = [];
    for (let dx = 0; dx < 3; dx++) {
      container.smallparks[dx] = [];
      for (let dy = 0; dy < 3; dy++) {
        if (dx === 1 && dy === 1) {
          container.smallparks[dx][dy] = null; // Το μεγάλο πάρκο της γειτονιάς μας, το άλσος με έναν τρόπο
        } else {
          container.smallparks[dx][dy] = {};
          storeColors(container.smallparks[dx][dy], level - 1);
}}}}


function drawFractal(x, y, size, level, data) {  // Εδώ θα ορίσουμε το χρώμα και το μέγεθος των κελιών μας
  if (level === 0) {
    fill(data.color);
    rect(x, y, size, size);
  } else {
    let newSize = size / 3;
    for (let dx = 0; dx < 3; dx++) {
      for (let dy = 0; dy < 3; dy++) {
        if (dx === 1 && dy === 1) continue;
        let smallpark = data.smallparks[dx][dy];
        if (smallpark) {
          drawFractal(x + dx * newSize, y + dy * newSize, newSize, level - 1, smallpark); }}}}
```



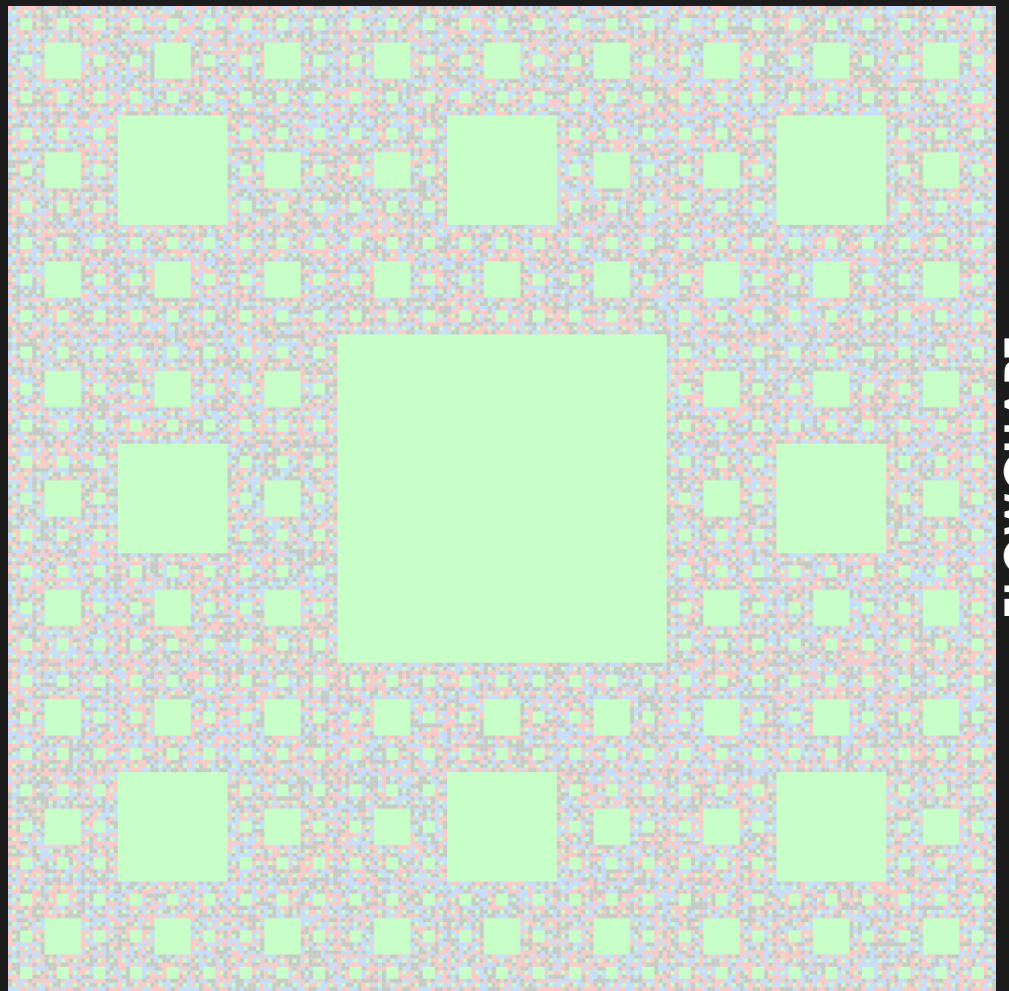
Στο πρόβλημα αυτό καλούμαστε να αναπαράγουμε το φράκταλ του Σιερπίνσκι με το σκεπτικό να ενσωματωθεί η κάτοψη μια πόλης.

Αρχικά ορίζω τα επίπεδα στα οποία θα υποδιαιρείται αυτό το μοτίβο(ο κατακρεματισμός) με maximum το 5 γιατί πάνω από αυτό το νούμερο το site κρασάρει.
Μετά από κάθε πάτημα του ποντικιού μου πάνω στην εικόνα, πρέπει να αλλάζει το επίπεδο κατακρεματισμού. Ορίζω το backround σαν πράσινους χώρους ενσωματώνοντας φρακταλς χρωμάτων για τα πάρκα, τις κατοικίες, τους εμπορικούς και δημόσιους χώρους.
Ορίζω το κεντρικό μεγάλο κενό σαν το άλσος μου.

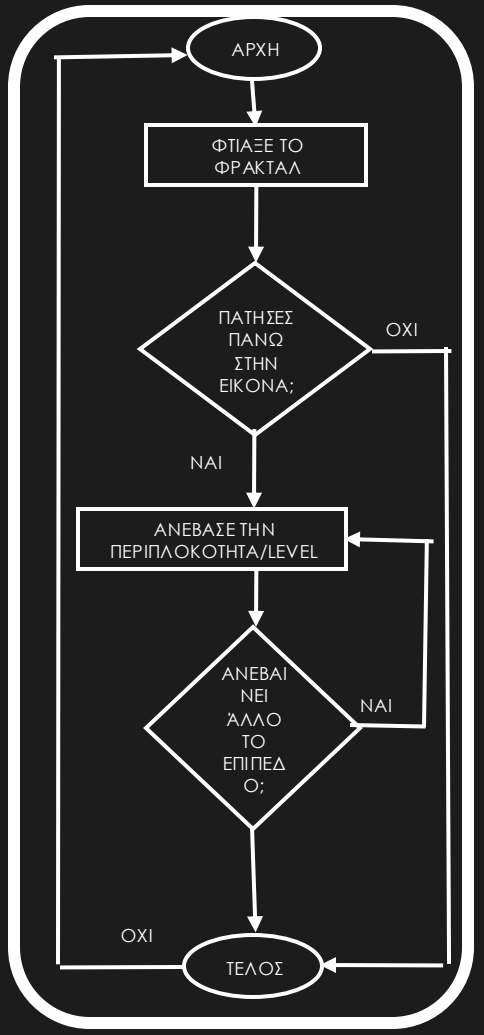
Μετέπειτα δίνω την εντολή να διαιρούνται τα κελιά μου(οι χώροι δηλαδή και τα κτήρια) σε μικρότερα κελιά και μετά από κάθε πάτημα του ποντικιού μου να υποδιαιρούνται ακόμα περισσότερο μέχρι το τελικό επίπεδο(5).

Δεν κατάφερα με τίποτα να βάλω τους δρόμους, ότι και να έκανα άλλαζε το μοτίβο και δεν μπόρεσα να το λύσω ούτε μέσω Chatgbt, ουτε με Reddit, οπότε το παράτησα έτσι

https://editor.p5js.org/Magda_Khutsishvili/sketches/ImQoS0oH4



FLOWCHART



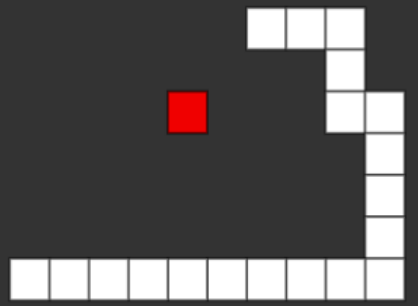
# ΠΡΟΒΛΗΜΑ 3

index.html
sketch.js
snake.js
style.css

```
var snake;       //Ορίζουμε τις αρχές του παιχνιδιού, το φίδι, το μεγεθός του και την τροφή του
var scl = 30
var food;

function setup() {
  createCanvas(900, 900);
  snake = new Snake();
  frameRate(5);
 pickLocation();
}


function pickLocation() {
  var cols= floor(width/scl);
  var rows = floor(height/scl);
   food = createVector(floor(random(cols)), floor(random(rows)))
  food.mult(scl);
}


function draw() {
  background(51);



 if (snake.eat(food)) {
   pickLocation();
 }
  snake.update();
  snake.show();
  snake.death();



  fill(255, 0, 0)
  rect(food.x, food.y, scl, scl)
}


// Χρησιμοποιούμε Global Function για να βάλουμε Key Controls
function keyPressed() {
  if (keyCode === UP_ARROW) {
    snake.dir(0, -1);
  } else if (keyCode === DOWN_ARROW) {
    snake.dir(0, 1);
  } else if (keyCode === RIGHT_ARROW) {
    snake.dir(1, 0);
  } else if (keyCode === LEFT_ARROW) {
    snake.dir(-1, 0);
  }
}
```

index.html
sketch.js
snake.js
style.css

```
function Snake() {  //χρησιμοποιούμε Constructor Function(this.κατι) για να μας "φτιαξει" το αντικείμενο (το φιδάκι μας)
  this.x = 0    // η αρχική θέση, πάνω αριστερά, η αρχή του παιχνιδιού ουσιαστικά
  this.y = 0
  this.xspeed = 1   // στο ξεκινημα να πάει προς τα δεξιά
  this.yspeed = 0
  this.total = 0
  this.tail = []  // το array της ουράς μας



  this.eat= function(pos) {    // Εντολή για το πως θα τρώει το φιδάκι μας
     var d= dist(this.x, this.y, pos.x, pos.y);
     if (d < 1) {
       this.total++
       return true;
     } else {
       return false;
     }
   }


  this.dir = function(x, y) {
    this.xspeed = x;
    this.yspeed = y;
  }

    this.death = function() {        // Ετσι πεθαινει το φιδακι μας, ώστε να ξαναξεκινάει το παιχνίδι
       for (var i = 0; i < this.tail.length; i++) {
         var pos = this.tail[i]
         var d = dist(this.x, this.y, pos.x, pos.y)
         if (d < 1) {
           this.total = 0
           this.tail = []
         }
       }
     }
  this.update = function() {       // Με αυτό το τρόπο μεγαλώνει το φιδάκι μας, κάνει δηλαδή update το tail length
     if (this.total === this.tail.length) {
     for(var i = 0; i < this.tail.length-1; i++)  {
       this.tail[i] = this.tail[i+1];
     }

   }
   this.tail[this.total-1] = createVector(this.x, this.y)


     this.x = this.x + this.xspeed*scl;
     this.y = this.y + this.yspeed*scl;

     this.x = constrain(this.x, 0, width-scl)
     this.y = constrain(this.y, 0, height-scl)

   }
  }


  this.show = function() {     // Εδώ μας δειχνει πως θα φαινεται το φιδι
     fill(255)
     for (var i = 0; i < this.total; i++) {
       rect(this.tail[i].x, this.tail[i].y, scl, scl);
     }
     rect(this.x, this.y, scl, scl)
   }
}
```



Το τελευταίο πρόβλημα αυτής της σειράς ασκήσεων αφορά την δημιουργία ενός παιχνιδιού που όλοι γνωρίζουμε, το θρυλικό φιδάκι.Σε αυτό το πρόβλημα επιλέγω πρώτον να δουλέψω σε δυο διαφορετικά αρχεία για να επιτρέψω στο κώδικα να λειτουργήσει χωρίς να μου κρασάρει όπως έκανε όταν έβαζα τις λειτουργίες του φιδιού στο ίδιο αρχείο(sketch.js). Τα δυο αυτά αρχεία δουλεύονται ταυτόχρονα για να φτάσουμε στο τελικό στάδιο.

Ξεκινάω ορίζοντας τις "αρχές" του παιχνιδιού , ουσιαστικά τα σημαντικότερα κομμάτια του , ως variations, το φιδάκι μας, το μεγεθός του και το φαγητό του. Να σημειωθεί ότι το παιχνίδι ελέγχεται με τους δείκτες τους πληκτρολογίου και λειτουργεί κανονικά (κοιτα αρ.εικονα 40-50)
Μετέπειτα θέτω την αρχική θέση του "φιδιού ", σε αυτή την φάση είναι μόνο ένα pixel, αποφασίζω ότι θα ξεκινάει πηγαίνοντας στα δεξιά αυτομάτως και δημιουργώ την τροφή του(κοίτα δεξ.εικ.1-17). Σημαντικό μέρος του χρόνου που αφιερώθηκε σε αυτήν την εργασία παίρνει η ιδέα του θανάτου στο παιχνίδι μου.  Η τελική απόφαση καθορίζει ότι το φιδάκι θα "πεθαίνει" όταν θα πέφτει πάνω στους τοίχους και όταν θα πέφτει πάνω στον εαυτό του (κοίτα 27-33). Επιπλέον προστίθεται η εντολή για να μεγαλώνει το φιδάκι με κάθε κυβάκι που τρώει. Το κομμάτι που δεν καταφέρνω να κάνω στο τέλος είναι η δημιουργία των εμποδίων για πολλαπλούς λόγους δυστυχώς.

https://editor.p5js.org/Magda_Khutsishvili/sketches/N47E9inki

# ΠΗΓΕΣ

Για το **ΠΡΟΒΛΗΜΑΤΑ 1,2,3** συμβουλεύτηκα τις παρακάτω πηγές

1. rect()
https://p5js.org/reference/p5/rect/

2. Διαλέξεις του Δρ. Παπανικολαου,
https://drive.google.com/drive/u/0/folders/1IZIKgRJW3HY29Sm3fFRKSoS878An0cOS

3. CHATGBT, OPENAI,
https://chatgpt.com/g/g-Yqey5fkGs-ellenika-gpt

4. mitosis, cells, cell growth by Chudroy
https://editor.p5js.org/Chudroy/sketches/KaQ6NQ5eI

5. sierpinski-carpet by inoha_naito
https://editor.p5js.org/inoha_naito/sketches/w5A4eHJQ9

6. Sierpinski Carpet by maxgoodrich
https://editor.p5js.org/maxgoodrich/sketches/LslGTMWVN

7. Snake
https://p5js.org/examples/games-snake/

8. Snake Game by Viv-Galinari
https://editor.p5js.org/Viv-Galinari/sketches/H1FqIMT5Z

https://editor.p5js.org/Magda_Khutsishvili/sketches