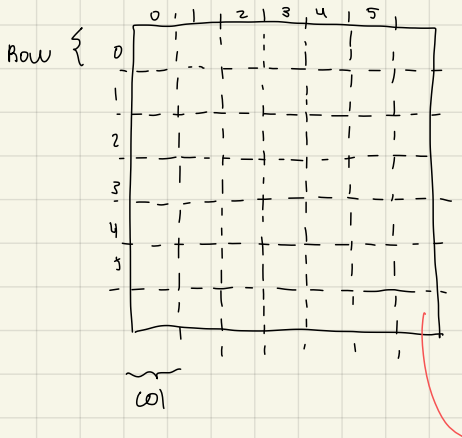
# Game Programming

Maze generating algorithm

1. Make the initial cell the current cell and mark it as visited
2. While there are unvisited cells
   - If the current cell has any neighbours which have not been visited
     - choose randomly one of the unvisited neighbours
     - add the current cell to the stack → an array of Cells
     - remove the wall between the current cell and the choosen cell
     - make the choosen sell the current cell and mark it as visited.
   - Else if the stack is not empty
     - pop a cell from the stack
     - make it the current cell

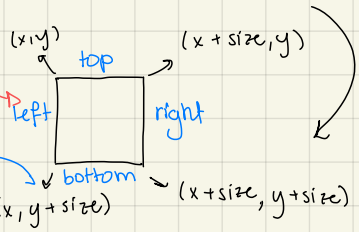
## Maze

Canva




Idea: the program figures out which walls have to be removed to create a certain path

- Cell object (global variables)
  - needs to know which column and row it is in
  - needs to know about its walls.




order in which I wrote the boolean values for the wall (clockwise)

Global variables
- col, rows
- size of the square

col = width/size of square
row = height/size of square

Walls

create a boolean array for the walls

walls = [true, true, true, true] → [top, right, bottom left]

draw the walls if they exist (true)

We can remove walls by turning one of the wall values to false.

Note: I plan to do the forward tracking part first

## Game Design and Functions:

void maze(); ⟶ draw/show the maze

Create a PGraphic and generate a random maze ⟶ PGraphic maze;
Display the maze image(maze, 0, 0);

Should create a loading page to wait for maze to load. (wait until all cells have been visited).

if this fails
Generate a random maze
Save it as a png
Display the PNG as the background.

void fieldofView(); ⟶ Generates a circle that displays the field of view

Make a circle of a certain size around the player
↳ circle(playerPositionX, playerPositionY, size);
↳ fill(150, viewBrightness)
↳ will control the transparency of the circle

void flicker(); ⟶ makes the field of view flicker

create a variable called lightValue or something, initialize to -5 or something
global variable flicker, global variable viewBrightness;
↳ boolean, initialize to true. ↳ float, initialize to 255

if flicker is greater than 255, then flicker is true
if flicker is less than 0, flicker is false

if flicker is false increase viewBrightness by n
if flicker is false decrease viewBrightness by n

Idea: Create a PGraphic and use the fieldview function to draw it.
Use mask method and apply it to the maze PGraphic.

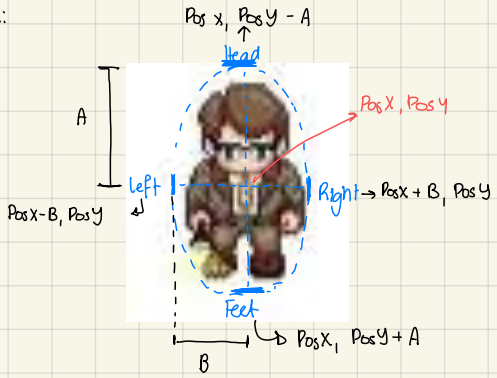
Another Idea:
Have the field view become smaller when an enemy is approaching
Kind of like: while(enemy is some amount close to the player)
fieldOf view is ½ its size.

## To check the borders of the maze:

→ Since I don't know the color of the line (because it might not be pure white, something I can do is instead check if the color of the pixel is Not the color of the walkable part of the maze.

```
if (pixel.getColor != yellow)
{ stopWalk
}
```

Visual Idea:




How should I stop the sprite?

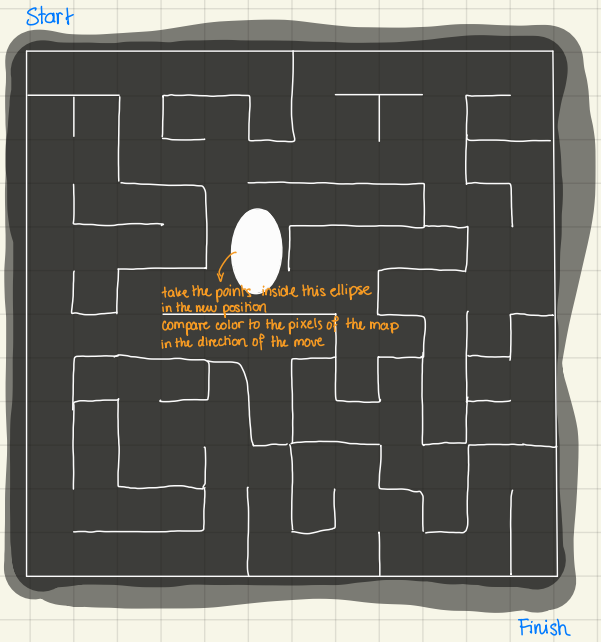
→ maybe if pixel color is yellow, change direction
or if pixel color is yellow, decrease position in direction by 1 pixel.

## Professor's Recomendations:

- Don't figure out the walls by xy coordinates

What you should do:
- make a black & white version of the maze (don't draw it on screen)
- place the ellipse (no stroke) on a PGraphic
- check if any same xy location pixels between the PGraphic and the black & white map are both white
- if they are white, that means there is a collision and you don't want to proceed.

Notes: - test the new proposed location first before moving the character.




When keyPressed
→ check new position of ellipse
→ check that new position in the image (color value)
→ if the pixel is the same color
either return false
or just don't increase the position value.

↓

An option would be to create a function that checks the white values and returns true if the player can move to that position or false if they can't.

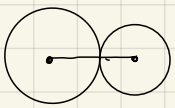
## For Lives

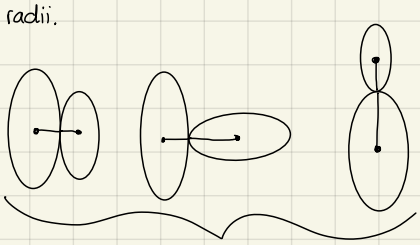
Create variable numLives (can be a global variable or inside Player class)
Assign a value of 3 to it.

While ( numLives >0)
  { play current game}
Else
  display Game Over

## For Enemy Collision

I want to use ellipsis to detect the collision between the player and the enemies, but I would have to use a different formula than the distance between two circles been less than the sum of their radii.

Basically this →  is not the same as 

these do work though

Idea: Use Pixaphics to create ellipsis around the enemies.
  ↳ Use the existing Pixaphic for the player to detect collision by using pixels
    (Don't forget to loadPixels(); )