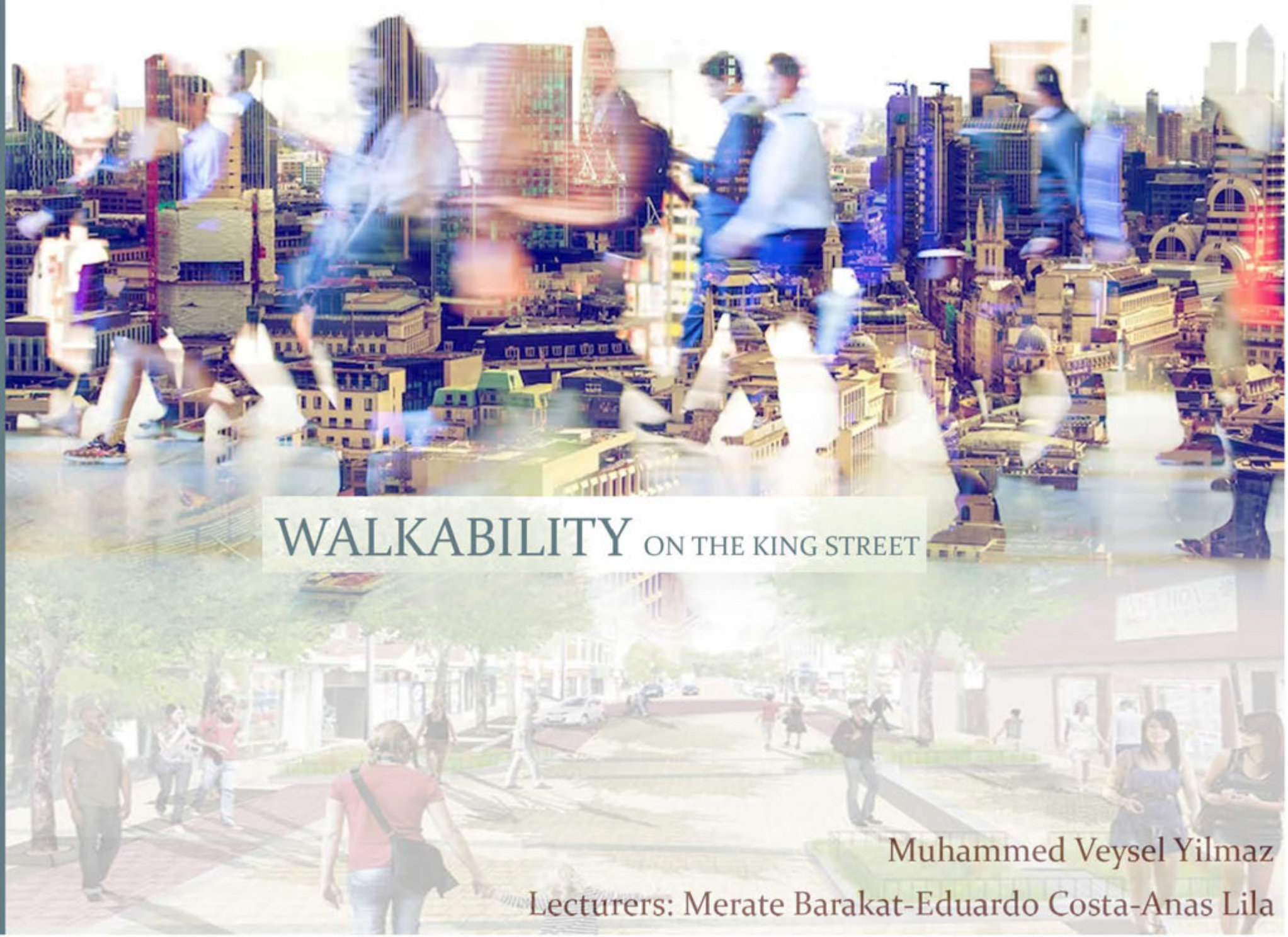
COMPUTING COMPLEXITY FINAL PROJECT



WALKABILITY ON THE KING STREET

Muhammed Veysel Yilmaz

Lecturers: Merate Barakat-Eduardo Costa-Anas Lila

Project Location
King Street, Bristol

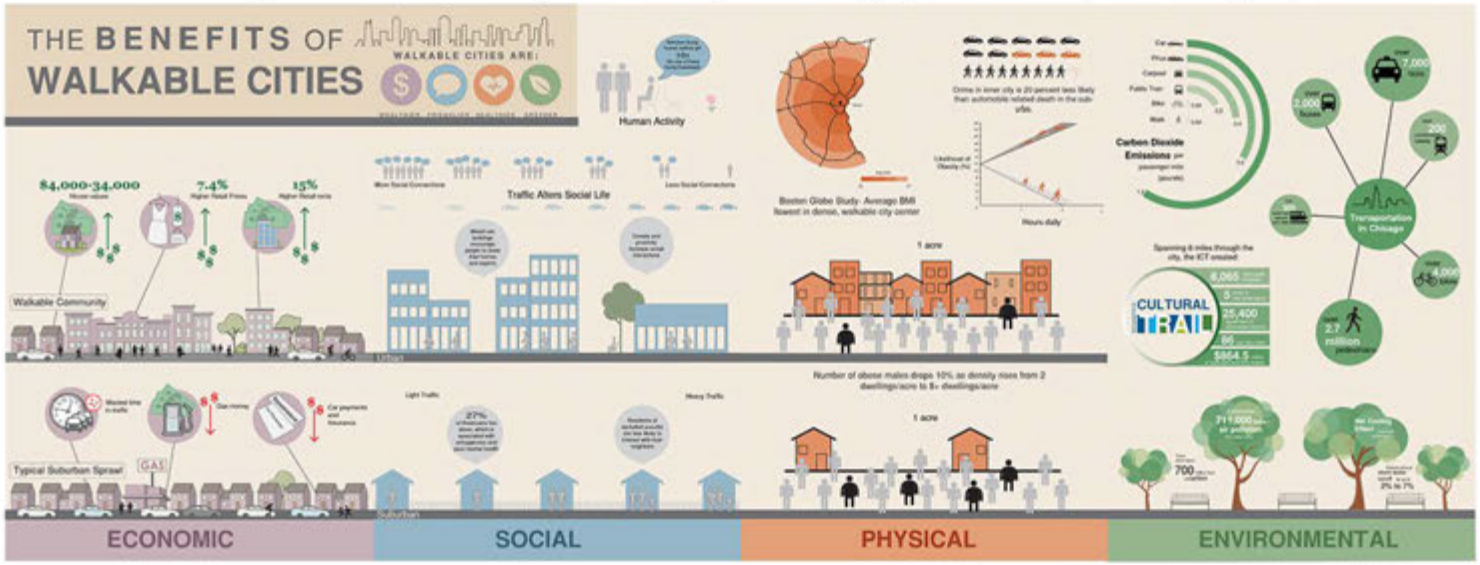
CONTENTS

WALKABILITY

Spatial configuration in cities has a direct effect on walking and cycling potentials and accessibility; which in turn have direct impact on viability of many retail businesses, social integration, public health, social safety and security (social segregation and crime havens), and environment (car dependency and its consequences).

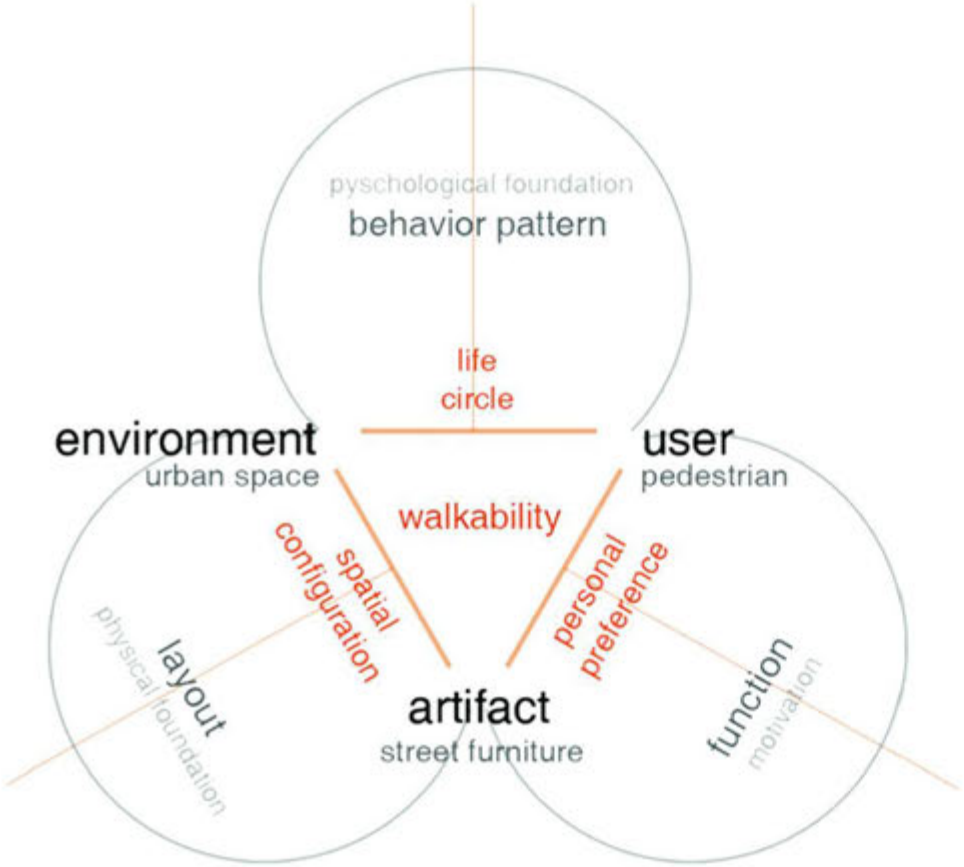
Walking has been identified as the most influenceable behavior; it is also the most environmental-friendly mode of transport, social and health. From the planning view, the concept of walkability therefore aims at a built environment facilitating physical activity. It is increasingly recognized that walkability has become an important topic in the field of planning, urban design and health, since the built environment affects certain behaviors. From practice, concrete guidance is demanded as to the type of urban design features to be captured or applied to evaluate the walkability or to create active cities. The measurement of features of the street environment plays a special role in this context.

There is a need for spatial configuration analysis in order to capture walking and accessibility. In the late 1970s, British architects Bill Hillier and Julienne Hanson hit on the idea that any space within a city – or the entire city itself – could be analyzed in terms of connectivity and movement. They reasoned that a city's success depended largely on how easy it was for people to move about on foot.

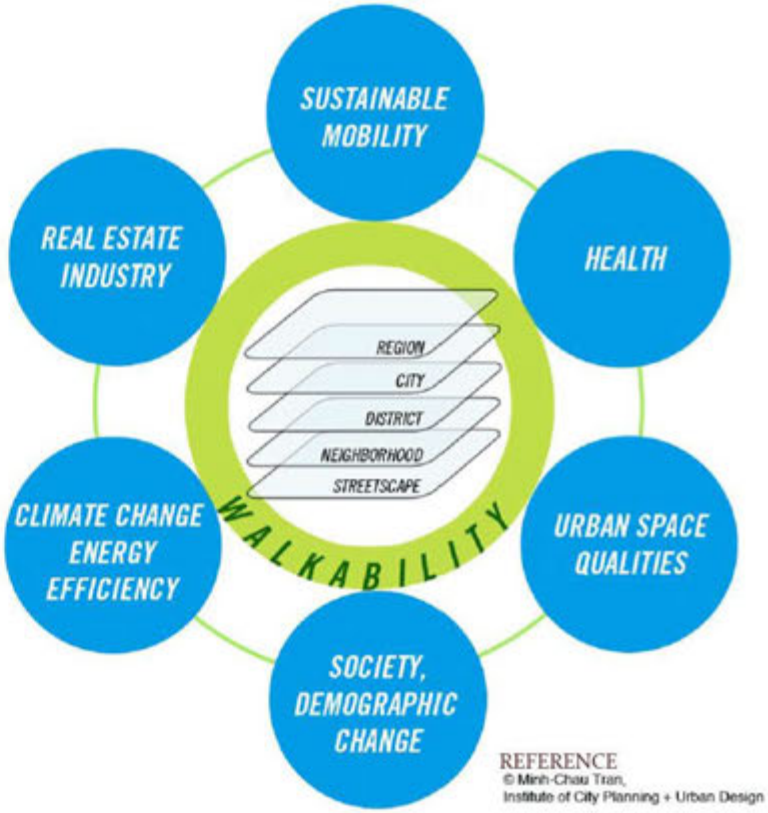


https://trtrail.files.wordpress.com/2014/12/walkabillyee.jpgA

WALKABILITY



PARAMETERS FOR WALKABILITY



BENEFITS OF WALKABILITY

FROM CITY LEVEL TO STREET LEVEL

Project Context and Scope

In this project, walkability of the king street will be analysed. The king street is one of the most busy streets in Bristol. After observing the street, it was noticed that there is street furniture which causes to poor walkability and permeability.

Problem

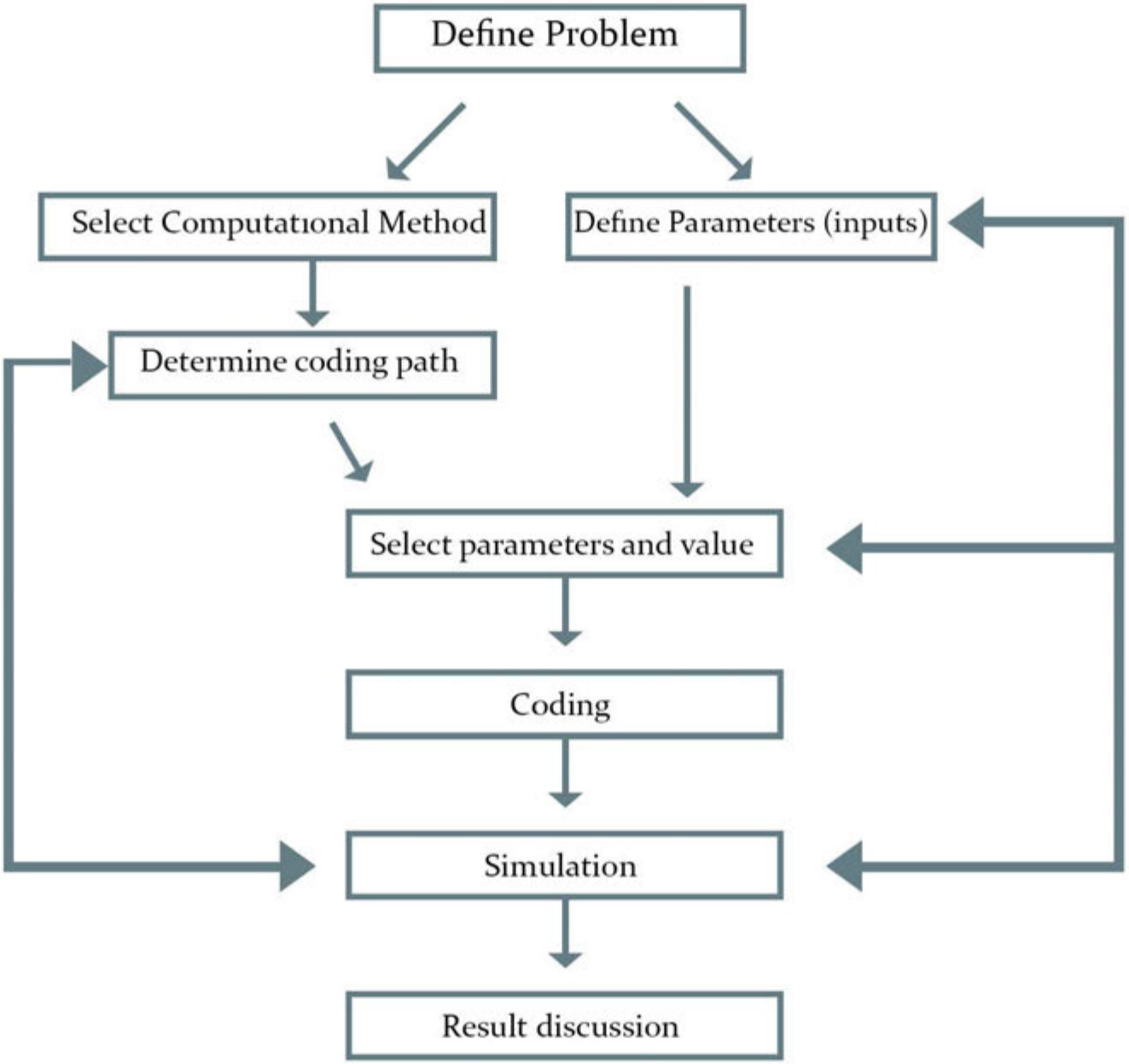
- Poor walkability on the king street due to improper placement of street furniture.

Research Question

- How pedestrian movement can be simulated in the processing?
- Does street furniture affect walkability of the king street?
- Which parameters lead to poor walkability of the king street? (Furniture, day/night , weekdays/weekend etc.)

working area

WORK FLOW


Define Problem
Select Computational Method
Define Parameters (inputs)
Determine coding path
Select parameters and value
Coding
Simulation
Result discussion

The Processing will be used to analyze and simulate walkability on the king street by using particles and a couple of behaviors. I preferred to do everything with the "Processing" to challenge myself about the "Processing".

PARAMETERS

-Morning/Night density
-Types of path
-People's age
-Velocity
-Forces
-Desired Separation
-Obstacles
-Boundaries

FUNCTIONS

Particles
Sworm behaviour
Separation/cohesion
Path following
Steering
Obstacle
Closeness
Seek

OUTPUTS

Walkability
Density
Accesibility
Congestion

Processing...

First of all, I'm gonna create particle which represents people

MAIN TAB

```
People p;

void setup() {
  size(1600, 800);
  p = new People(width/2, height/2);

}
void draw() {
  background(255);
  p.run();
  p.edgeBehaviour();
}
```

PEOPLE CLASS

```
// The "People" class
class People {
  PVector location;
  PVector velocity;
  PVector acceleration;
  float r;
  float maxspeed;

  People(float x, float y) {
    acceleration = new PVector(5, 5);
    velocity = new PVector(3, -2);
    velocity.mult(5);
    location = new PVector(x, y);
    r = 6;
    maxspeed = 6;
  void run() {
    update();
    display();
  }
  // Method to update position
  void update()  // Method to update position
    { velocity.add(acceleration); // Update velocity
    velocity.limit(maxspeed); // Limit speed
    location.add(velocity);
    acceleration.mult(0); // Reset accelertion to  each cycle
  }
```

continuation of the class

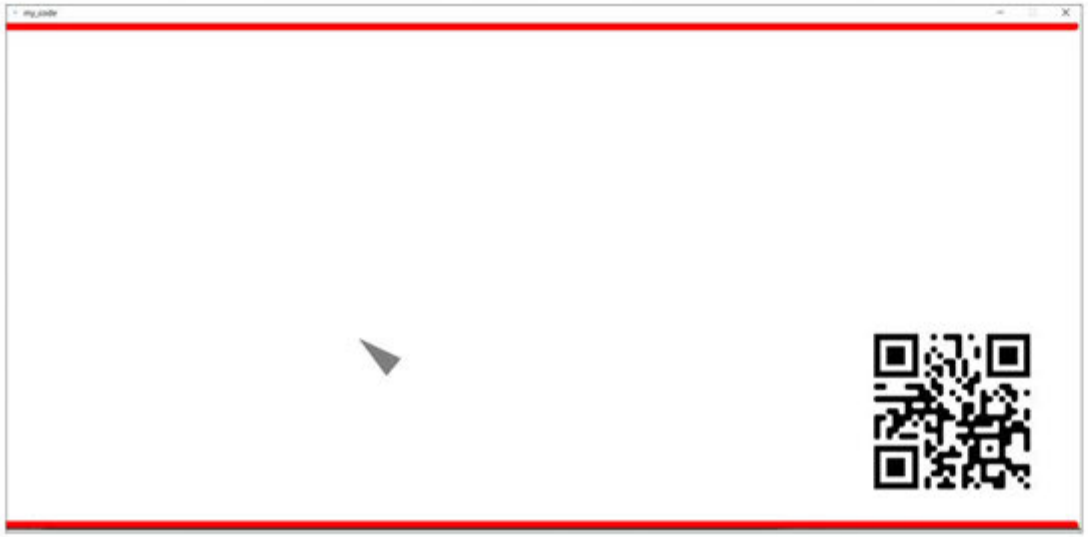
```
void edgeBehaviour()
 {
  if (location.x >width)
  {
   velocity.add (new PVector (-1, random(-1, 1)));
  }
  if (location.x <0)
  {
   velocity.add (new PVector (1, random(-1, 1)));
  }
  if (location.y >height)
  {
   velocity.add (new PVector (random (-1, 1), -1));
  }
  if (location.y <0)
  {
   velocity.add (new PVector (random (-1, 1), 1));
  }
 }
 void display() {
   // Draw a triangle rotated in the direction of ve-
locity
   float theta = velocity.heading2D() + radians(90);
   fill(127);
   stroke(0);
   pushMatrix();
   translate(location.x, location.y);
   rotate(theta);
   beginShape(TRIANGLES);
   vertex(0, -r*2);
   vertex(-r, r*2);
   vertex(r, r*2);
   endShape();
   popMatrix();
 }
}
```

Single particle which represents human

Processing...

Building Boundaries

The red lines represent the buildings boundaries in the king street



What I need is I want this particle to spin softly when it gets close to the walls, without hitting the wall for make our analysis more real. For this I need to update edge behaviour function

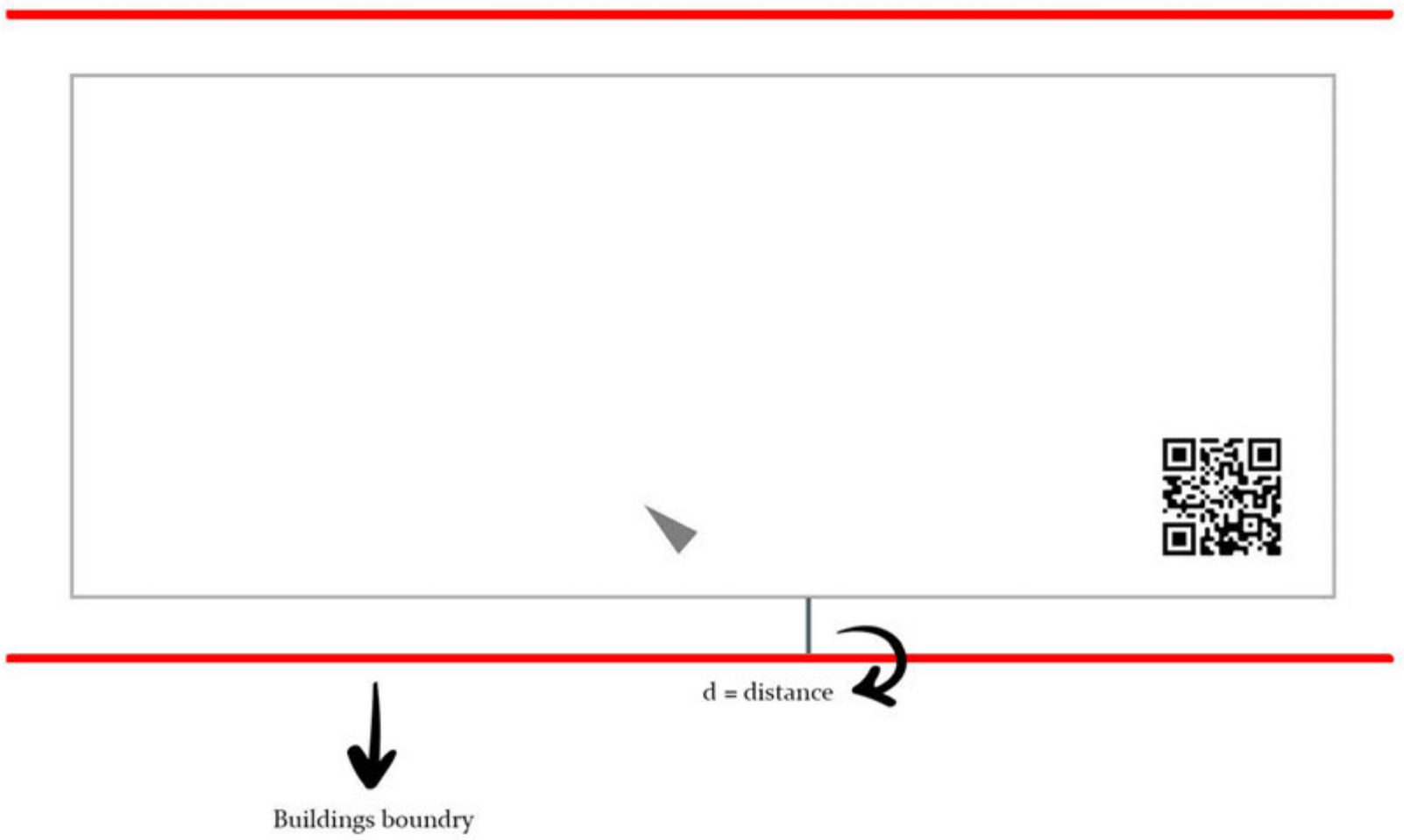
d = distance value between people and buildings

Code of Stay away from building

```
void edgeBehaviour() {

  PVector desired = null;

  if (location.x < d) {
    desired = new PVector(maxspeed, velocity.y);
  }
  else if (location.x > width -d) {
    desired = new PVector(-maxspeed, velocity.y);
  }

  if (location.y < d) {
    desired = new PVector(velocity.x, maxspeed);
  }
  else if (location.y > height-d) {
    desired = new PVector(velocity.x, -maxspeed);
  }
  if (desired != null) {
    desired.normalize();
    desired.mult(maxspeed);
    PVector steer = PVector.sub(desired, velocity);
    steer.limit(maxforce);
    applyForce(steer);
  }
}
```

Processing...

Stay away from building



Processing...

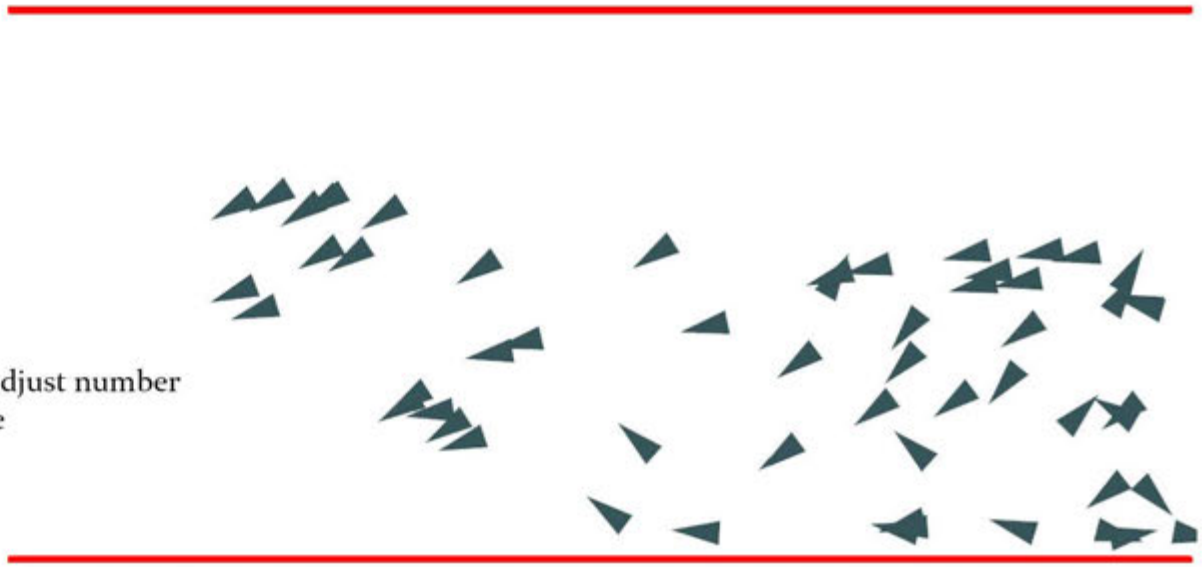
ArrayList People

Now, I'm gonna create people array instead of single particle

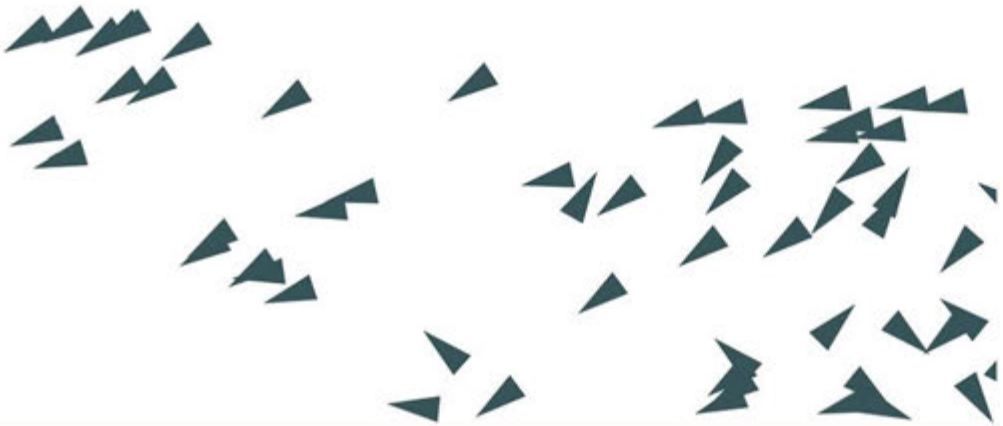
```
ArrayList<People> people;
boolean isRecording = false;
float d = 80;
boolean debug = true;

void setup() {
  size(1920, 900);
  people = new ArrayList<People>();
  for (int i = 0; i < 50; i++){
  people.add(new People(ran-
dom(width),random(height)));
  }
}
void draw() {

  background(255);

  for (People p : people){

  p.run();
  p.edgeBehaviour();
```

Create array list

We can adjust number of people

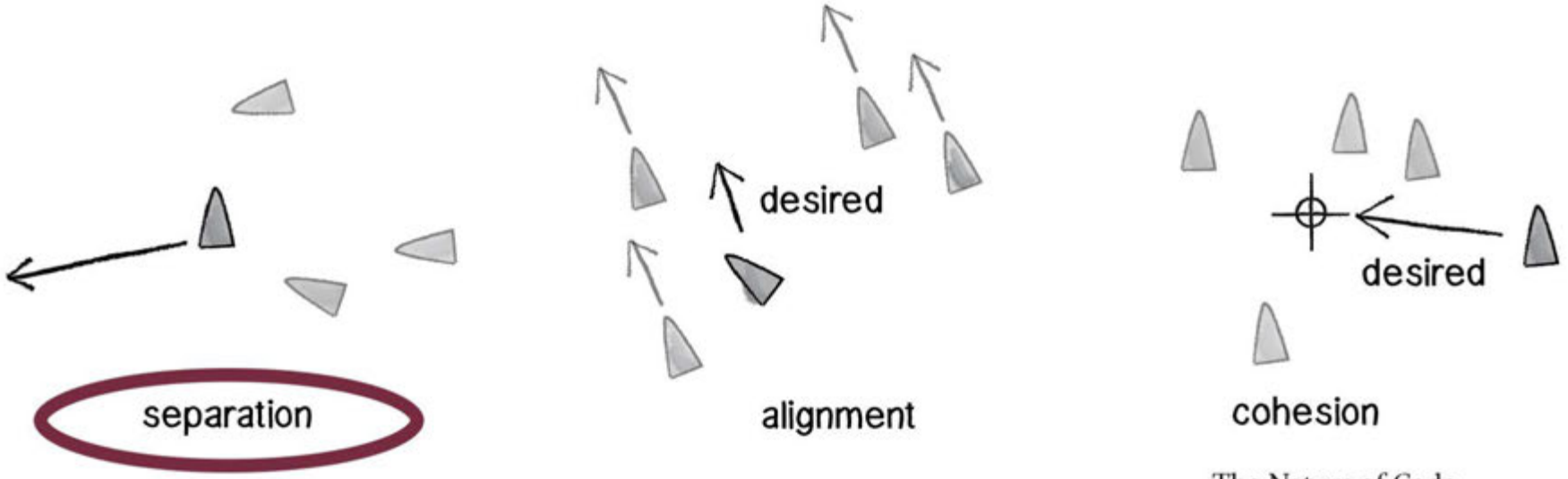


People Behaviour- Separation



As seen in the picture, particles pass through each other. to avoid this, to make our code look more real, I'm gonna add human behaviour.

At this point, I'm gonna use swarm behaviours



People Behaviour- Separation

```
void separate (ArrayList<People> people) {
```

We'll loop through each people inside of this function to check whether any are too near.

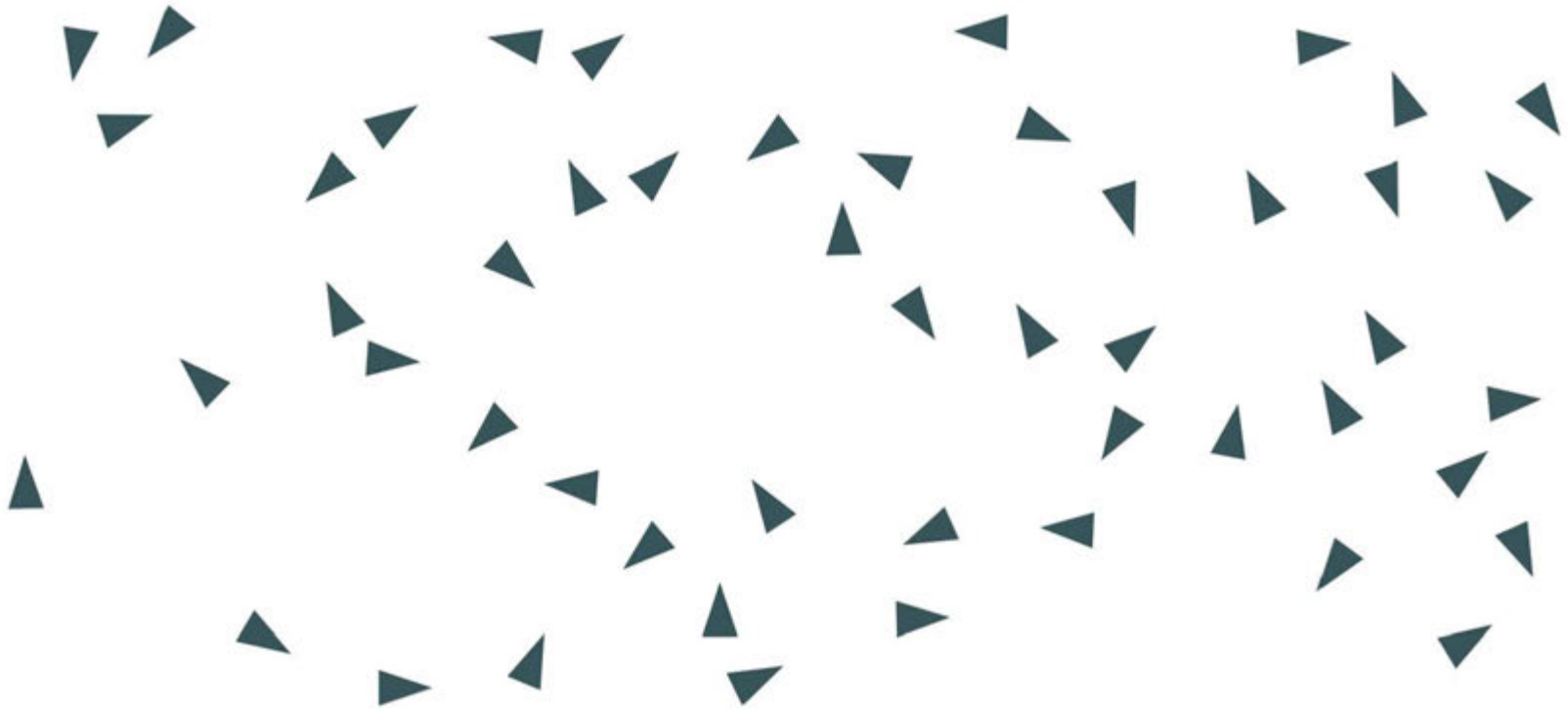
```
float desiredseparation = 30;          This variable show that how close  is too close

 for (People other : people) {

   float d = PVector.dist(location, other.location);  What is the distance between people?

   if ((d > 0) && (d < desiredseparation)) {
PVector pointing away from the other's location
PVector diff = PVector.sub(location, other.location);
    diff.normalize();
sum.add(diff);   Add all the vectors together and increment the count.
   count++;          Keep track of how many
```

```
   if (count > 0) {
       // Our desired vector is moving away maximum speed
       sum.setMag(maxspeed);
       // Implement Reynolds: Steering = Desired - Velocity
       PVector steer = PVector.sub(sum, velocity);
       steer.limit(maxforce);
       applyForce(steer);
     }
```

I set the magnitude of velocity to 0 for examine the separation behaviour

```
velocity = new PVector(0, 0);
```

People Behaviour- Separation

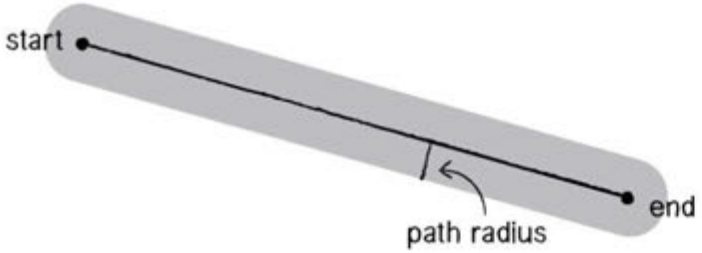


No one is touching the other's

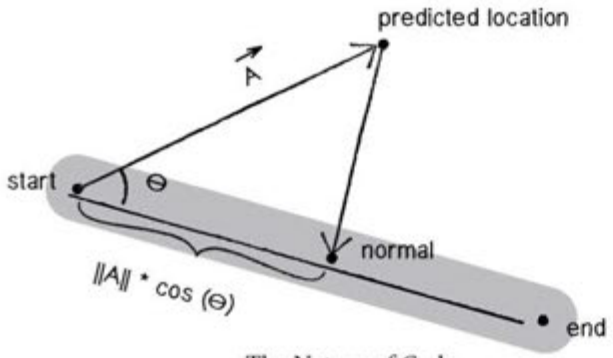
Path Following

With path following, I'm gonna specify people's way based on coordinate of obstacles(strret furniture)

Simple way will be to define a path as a series of connected

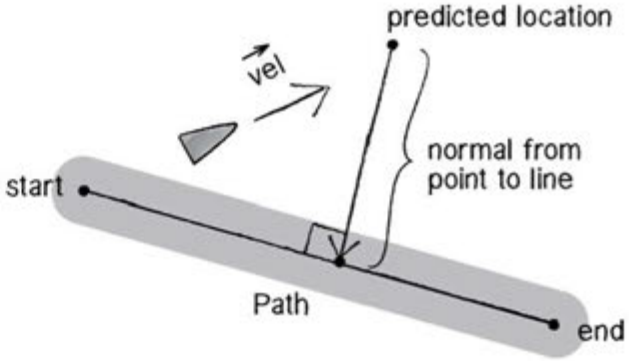


Path radius determines the road's width

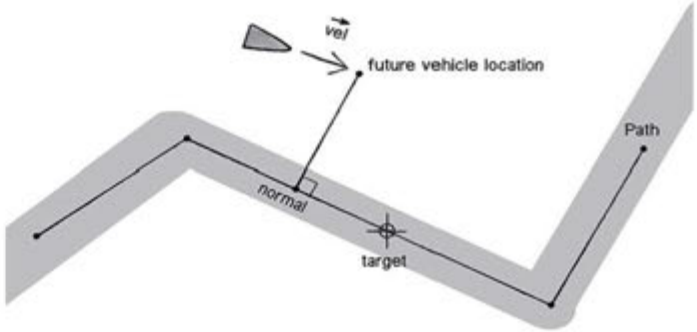


The Nature of Code

What we need to know for path following



The Nature of Code



The Nature of Code

Overall algorithm for path following

Processing...

Path Following

I'm gonna start with coding "path following". After defining coordinate of obstacles, I'm gonna re-create path. This part is so wide for explain every steps, so I'm gonna show generally what I did.
My reference is the book "The Nature of Code / Chapter 6- path following "

1

```
/ Path Following

class Path {

 // A Path is an arraylist of points (PVector objects)
 ArrayList<PVector> points;
 // A path has a radius, i.e how far is it ok for the boid to wander off
 float radius;
```

2

```
  Path() {
    // Arbitrary radius of 30
    radius = 30;
    points = new ArrayList<PVector>();
  }

  // Add a point to the path
  void addPoint(float x, float y) {
    PVector point = new PVector(x, y);
    points.add(point);
  }

  // Draw the path
  void display() {
    strokeJoin(ROUND);
```

3

```
    // Draw thin line for center of path
    stroke(255);
    strokeWeight(1);
    noFill();
    beginShape();
    for (PVector v : points) {
      vertex(v.x, v.y);
    }
    endShape(CLOSE);
  }
}
```

4

```
// Draw thick line for radius
  stroke(155);
  strokeWeight(radius*2);
  noFill();
  beginShape();
  for (PVector v : points) {
    vertex(v.x, v.y);
  }
endShape(CLOSE);
```

Processing...

Path Following

1

```
I changed constructor
People(PVector l,float ms, float mf) {
  acceleration = new PVector(0, 0);
  velocity = new PVector(5, 0);
  velocity.mult(5);
  location = l.get();
  r = 10;
  maxspeed = ms;
  maxforce = mf;
```

2

```
// A function to deal with path following and separation
 void applyBehaviors(ArrayList people, Path path) {
  // Follow path force
  PVector f = follow(path);
  // Separate from other boids force
  PVector s = separate(people);
  // Arbitrary weighting
  f.mult(6);
  s.mult(4);
  // Accumulate in acceleration
  applyForce(f);
  applyForce(s);
 }
```

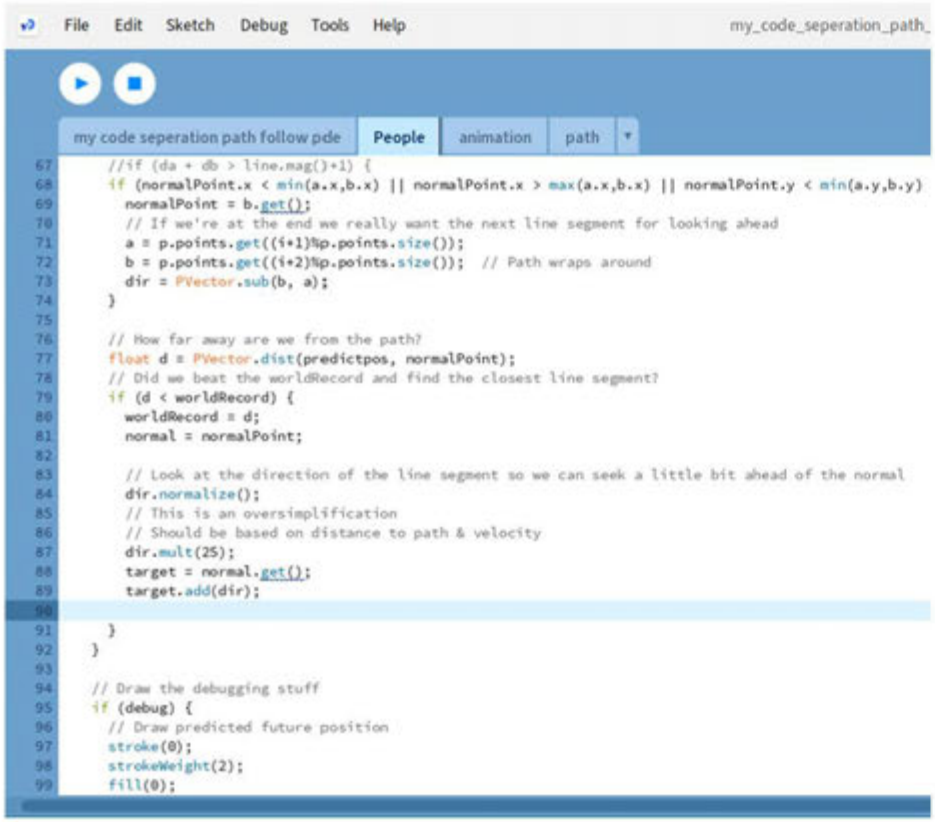
3

```
PVector follow(Path p) {

  // Predict position 25 (arbitrary choice) frames ahead
  PVector predict = velocity.get();
  predict.normalize();
  predict.mult(25);
  PVector predictpos = PVector.add(location, predict);
  // looking at the normal for each line segment and pick out the closest one
  PVector normal = null;
  PVector target = null;
  float worldRecord = 1000000;  // Start with a very high worldRecord distance that can easily be beaten
  // Loop through all points of the path
  for (int i = 0; i < p.points.size()-1; i++) {
    // Look at a line segment
    PVector a = p.points.get(i);
    PVector b = p.points.get((i+1)%p.points.size()); // Note Path has to wrap-around
    // Get the normal point to that line
    PVector normalPoint = getNormalPoint(predictpos, a, b);

    // Check if normal is on line segment
    PVector dir = PVector.sub(b, a);
    // If it's not within the line segment, consider the normal to just be the end of the line segment (point b)

    if (normalPoint.x < min(a.x,b.x) || normalPoint.x > max(a.x,b.x) || normalPoint.y < min(a.y,b.y) || normalPoint.y > max(a.y,b.y)) {
      normalPoint = b.get();
```

Processing...

Path Following

Continuation of follow path function 4

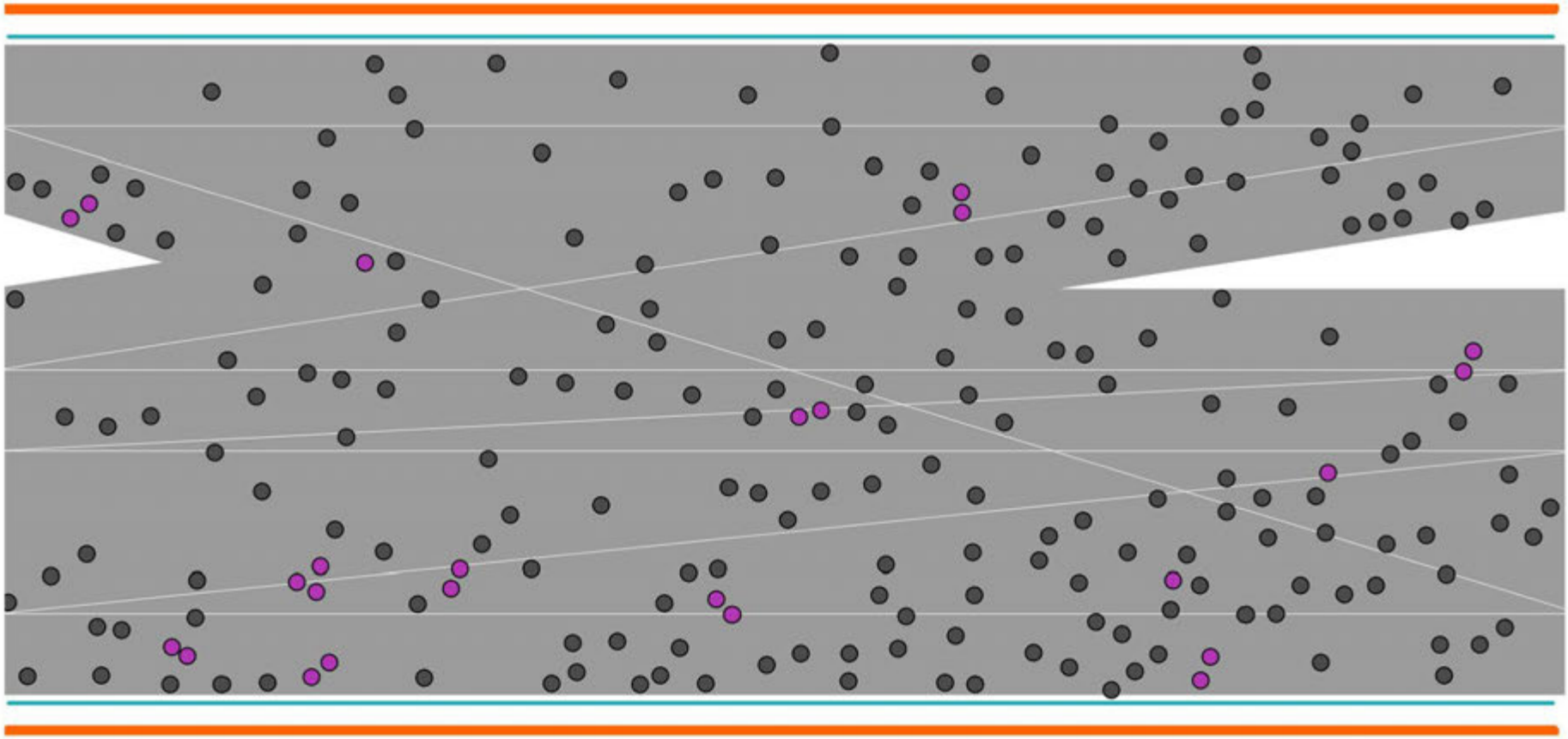


Updating separate function

```
PVector separate (ArrayList people) {
  float desiredseparation = r*3;
  PVector steer = new PVector(0, 0, 0);
  int count = 0;
  // For every PEOPLE in the system, check if it's too
close
  for (int i = 0 ; i < people.size(); i++) {
    People other = (People) people.get(i);
    float d = PVector.dist(location, other.location);
    // If the distance is greater than 0 and less than an
arbitrary amount
    if ((d > 0) && (d < desiredseparation)) {
      // Calculate vector pointing away from neighbor
      PVector diff = PVector.sub(location, other.loca-
tion);
      diff.normalize();
      diff.div(d);       // Weight by distance
      steer.add(diff);
      count++;           // Keep track of how many
```

Processing...

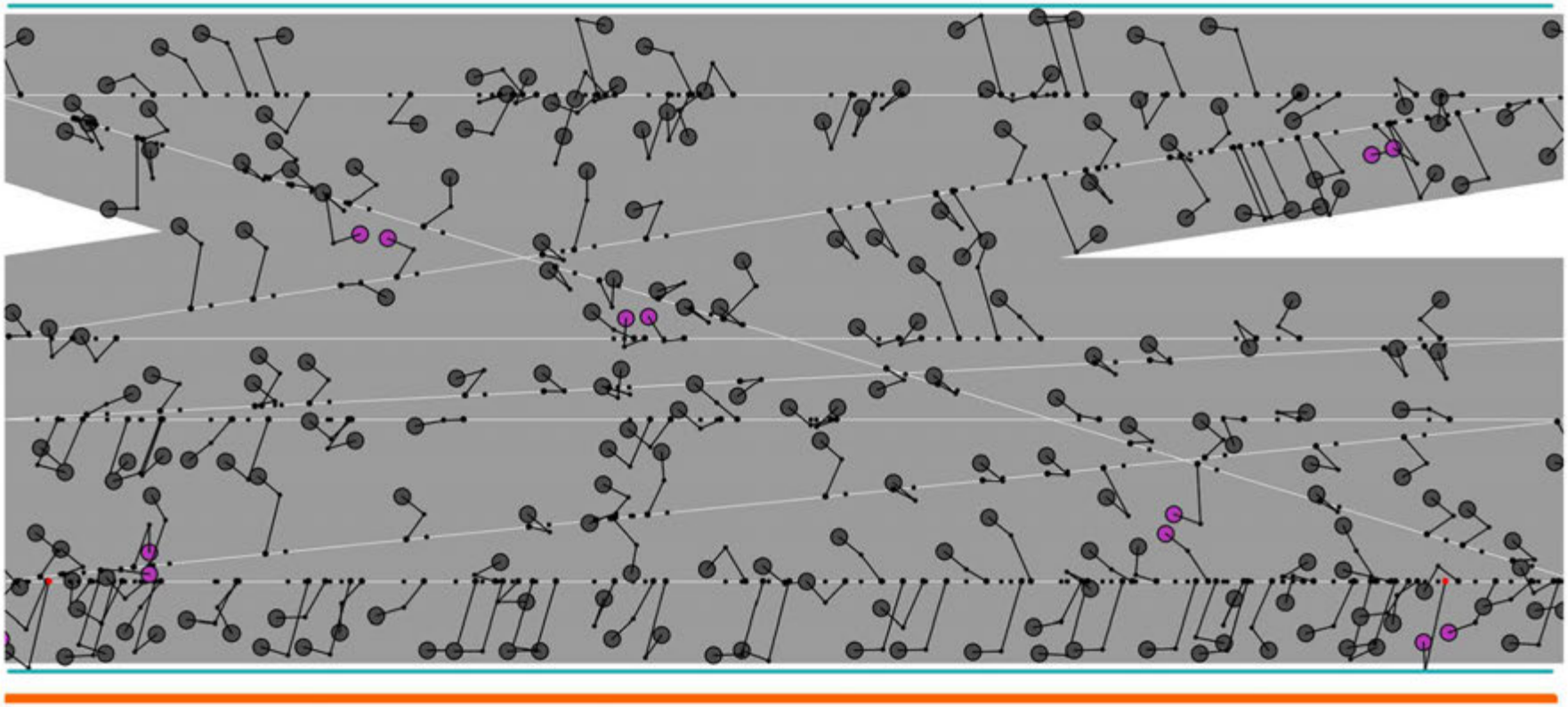
First I create path without obstacles throughout the street for compare with the path with obstacles. For seeing how close people each other and density, I used highlight code, I encode that if people close to each other than 30 pixel display purple



I changed peoples shape to circle for observing more easy

Processing...

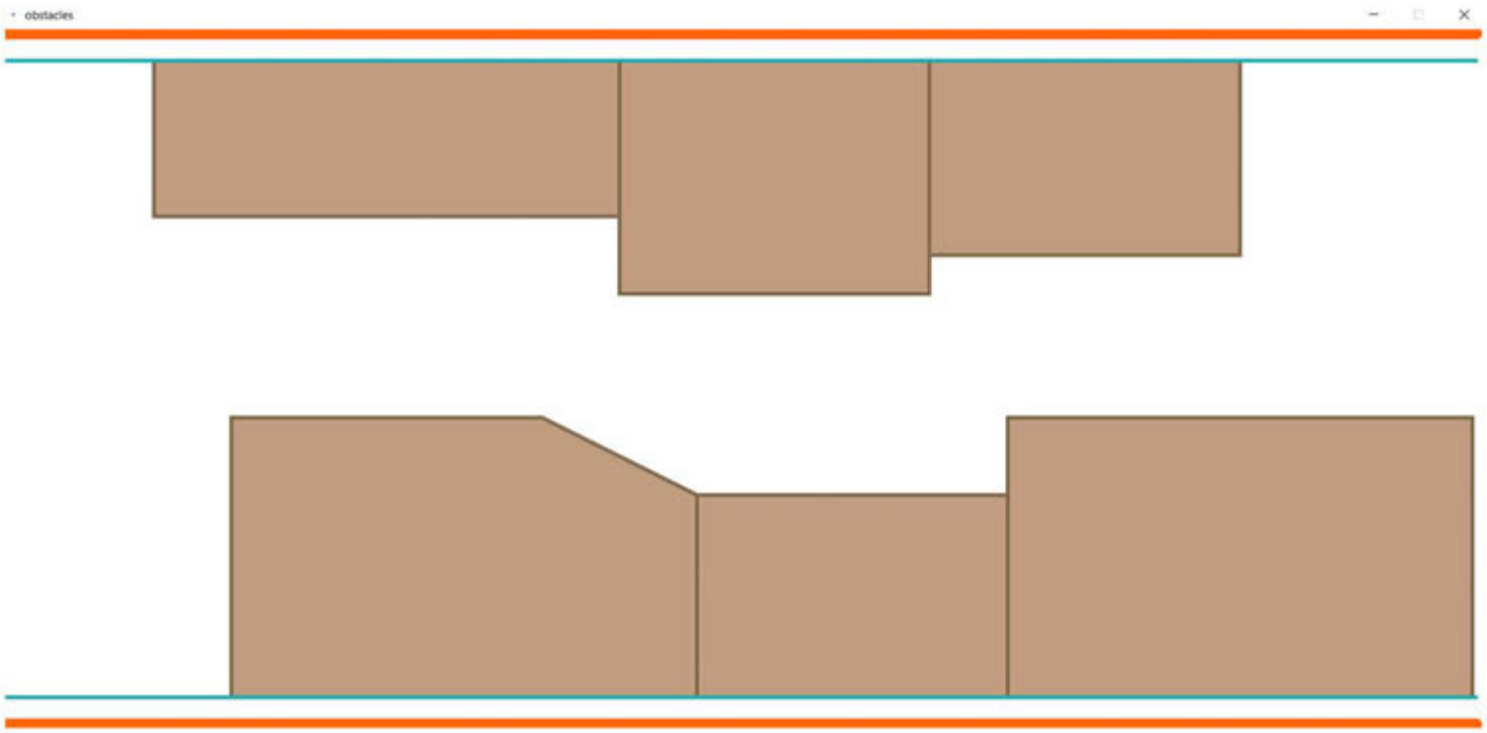
We can see path following vectors for each people in this image



orange line : building boundry
light blue line: to avoid crashing to building
purple: how we close

Processing...

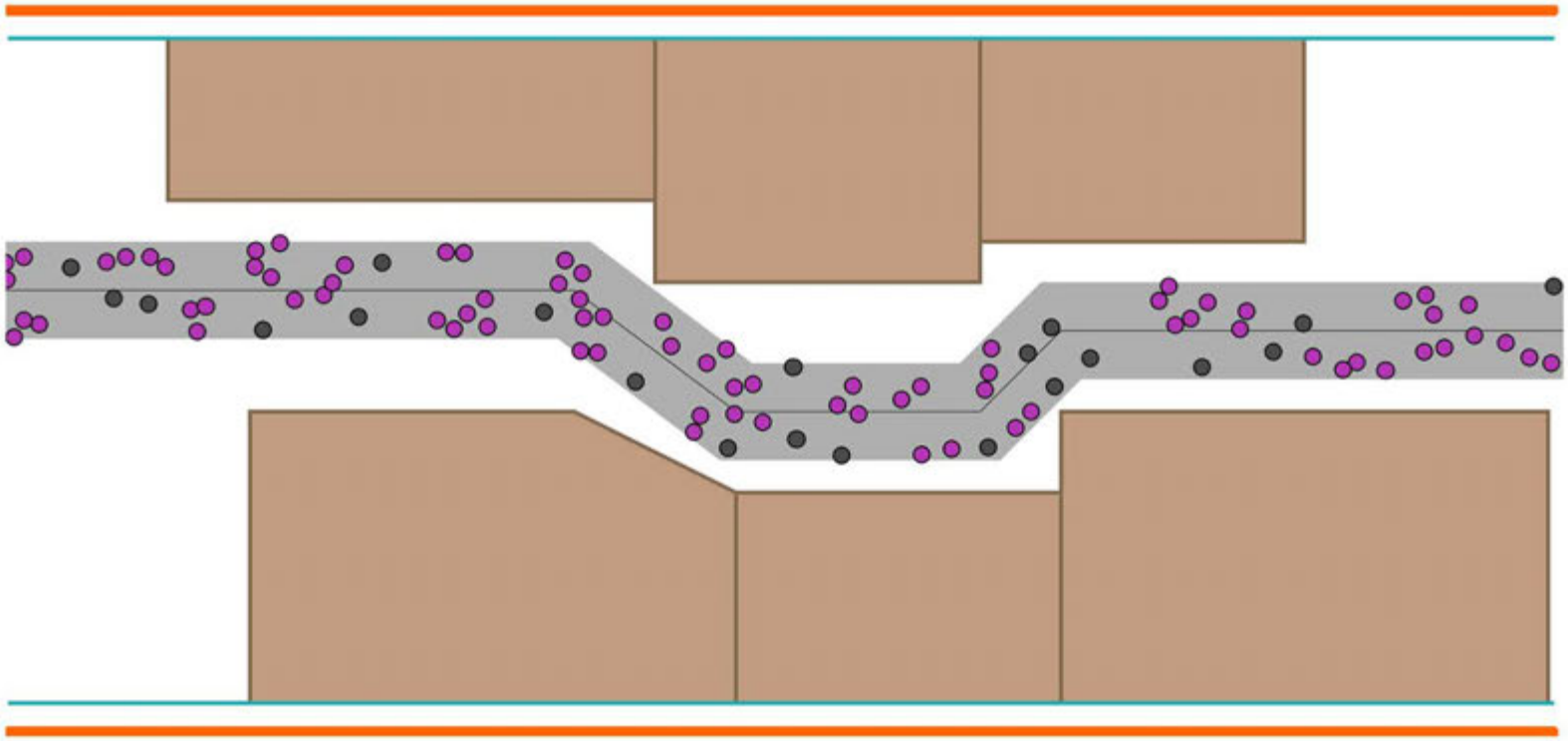
Now, the boundaries of obstacles are defined. I'm gonna merge it with my project and re-define path




```
PShape s;

void Obstacles()
{
  fill(193, 156, 129);
  stroke(130, 106, 78);
  strokeWeight(5);
  rect(200, 40, 600, 200);
  rect(800, 40, 400, 300);
  rect(1200, 40, 400, 250);
  rect(900, 860, 400, -260);
  rect(1300, 860, 600, -360);


s = createShape();
  s.beginShape();
  s.fill(193, 156, 129);
  s.stroke(130, 106, 78);
  s.strokeWeight(5);
  s.vertex(300, 860);
  s.vertex(300, 500);
  s.vertex(700, 500);
  s.vertex(900, 600);
  s.vertex(900, 860);
  s.endShape(CLOSE);
  shape(s,0,0);
```

Processing...

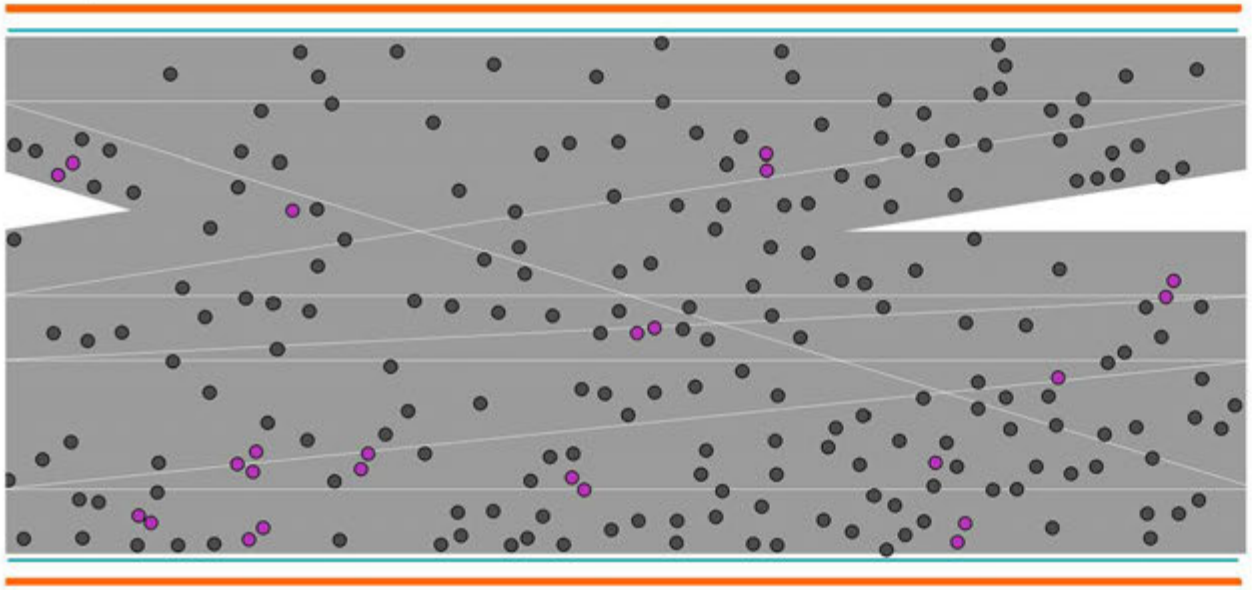
RE-PATH ACCORDING TO OBSTACLES



Number of People: 100

Processing...

People amount : 100



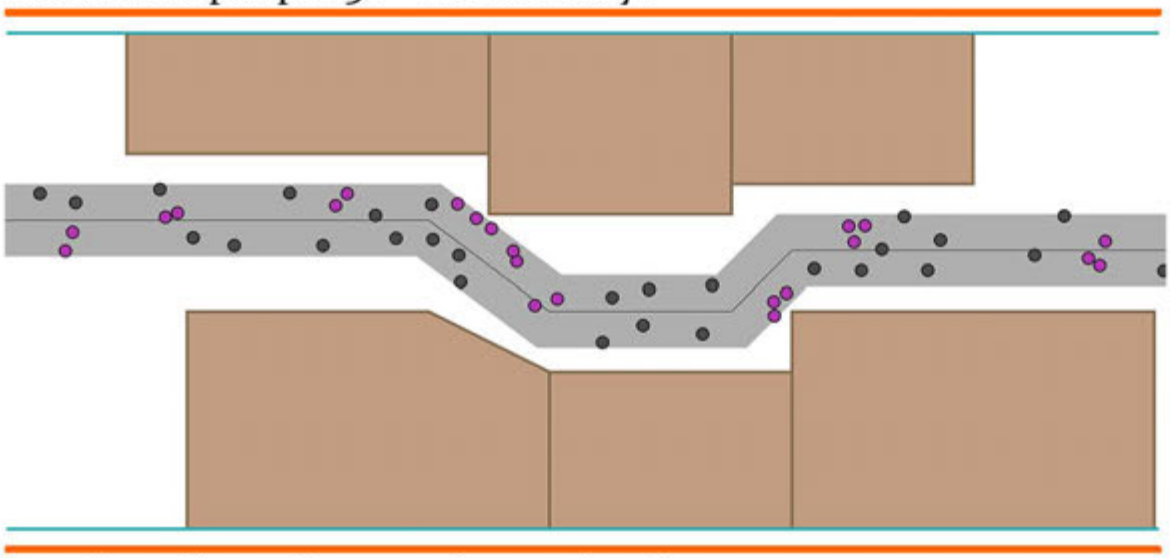
People amount : 230

When we compare the paths, its obvious that in the path which is with the obstacles is more packed and there is more congestion than other although it has less people.

Processing...

WEEKEND-WEEKDAYS

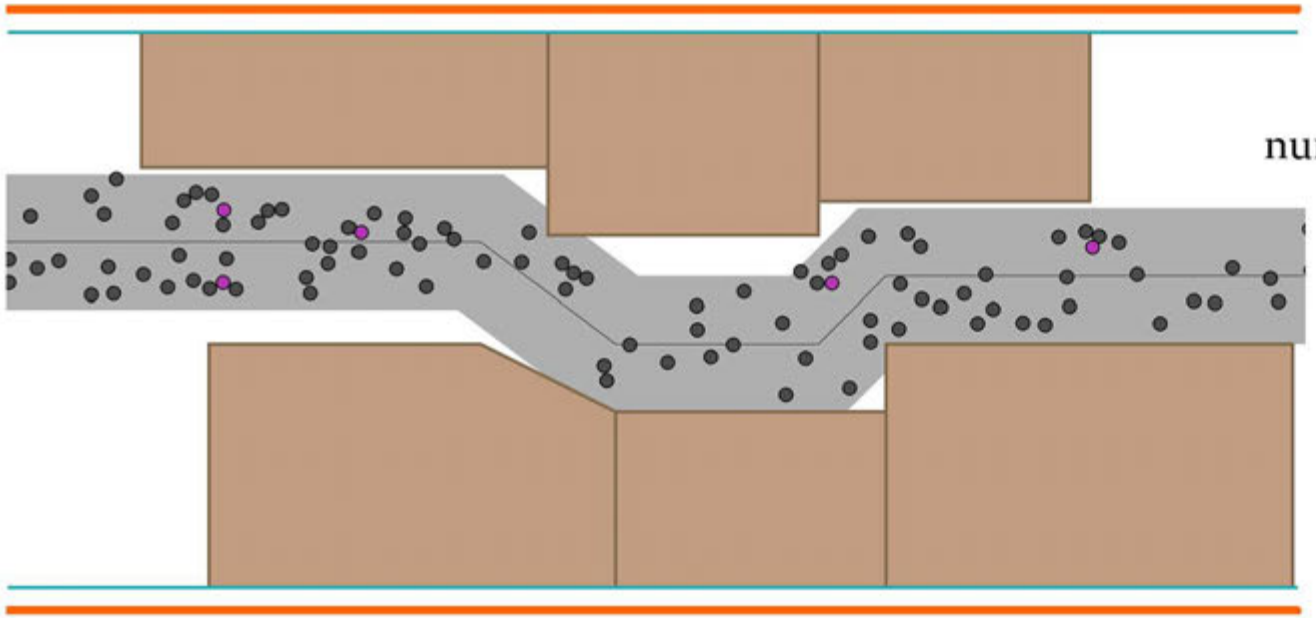
number of people:50 for weekdays



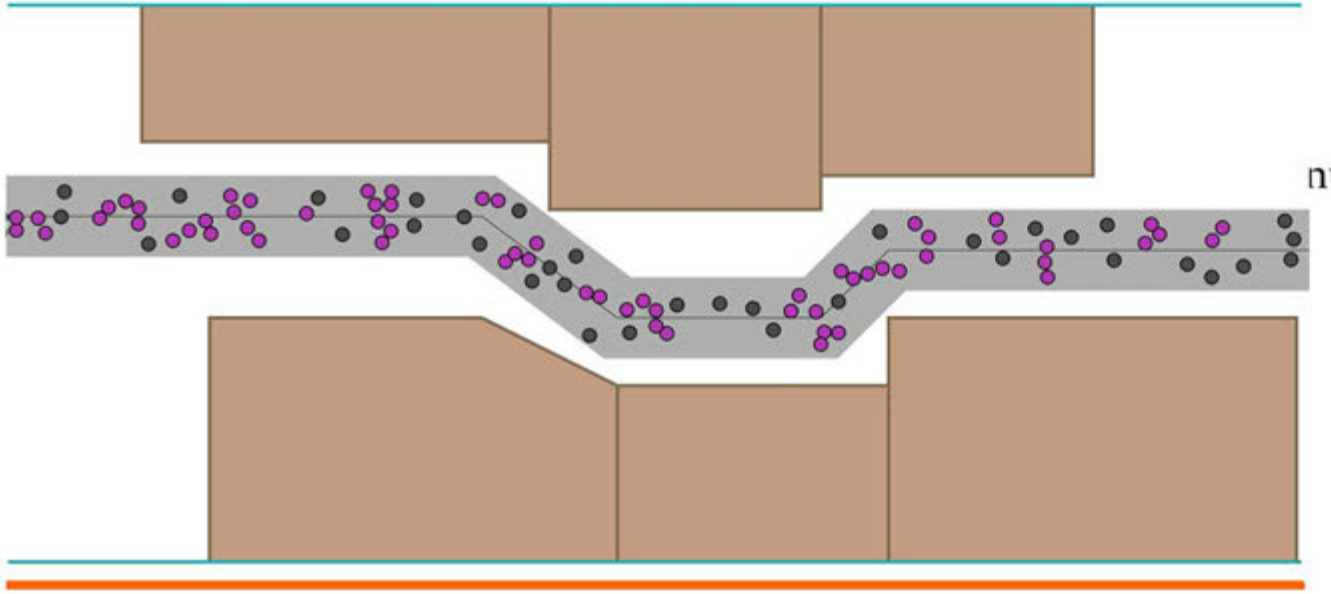
number of people:200 for weekend

Processing...

PATH RADİUS



number of people:100 path radius : 100

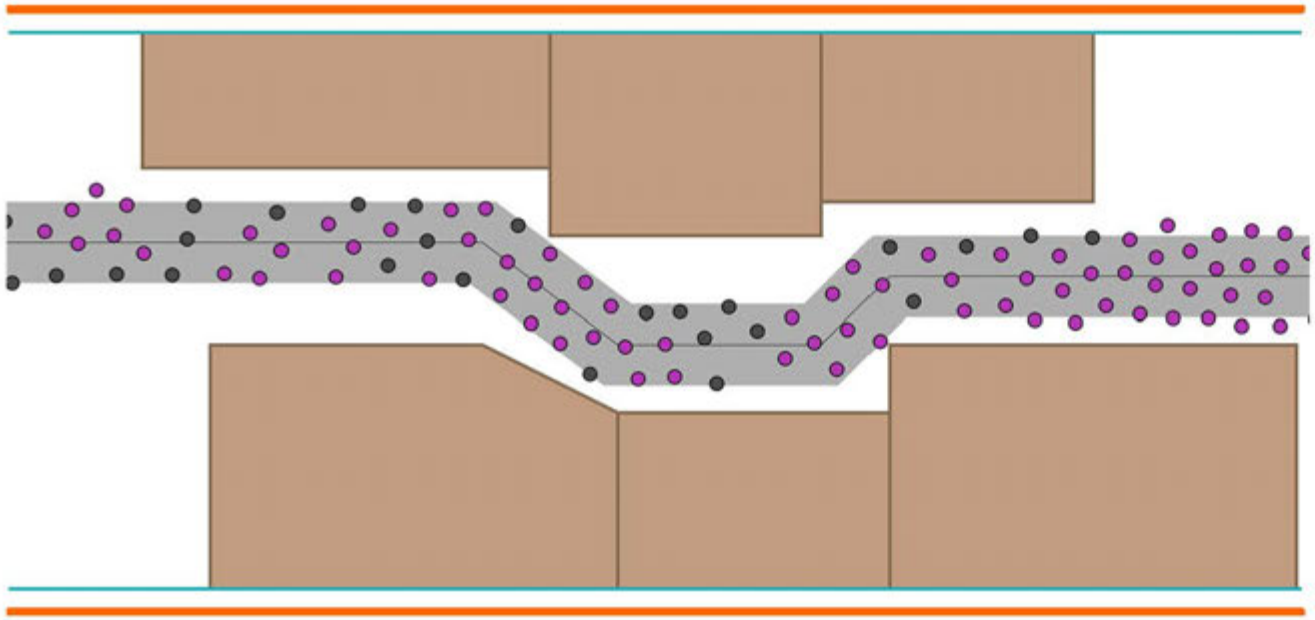


number of people:100 path radius : 60

MORE CONGESTION

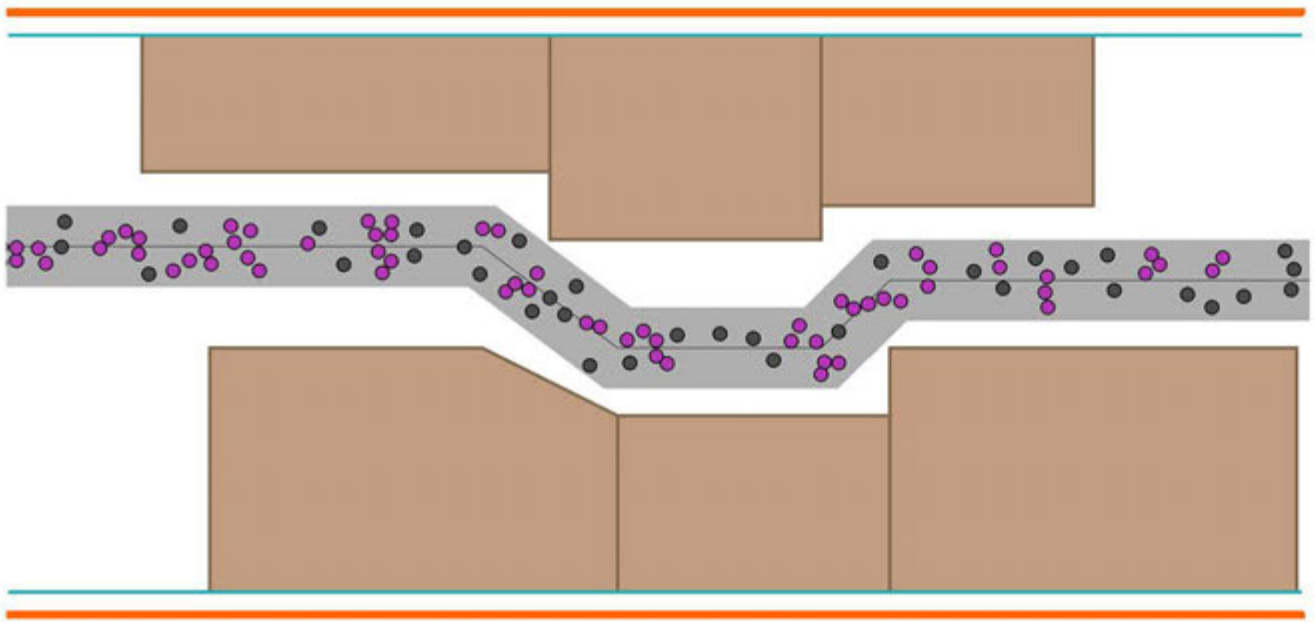
Processing...

SEPARATION DISTANCE



separation distance : 50
(pandemic distance :)

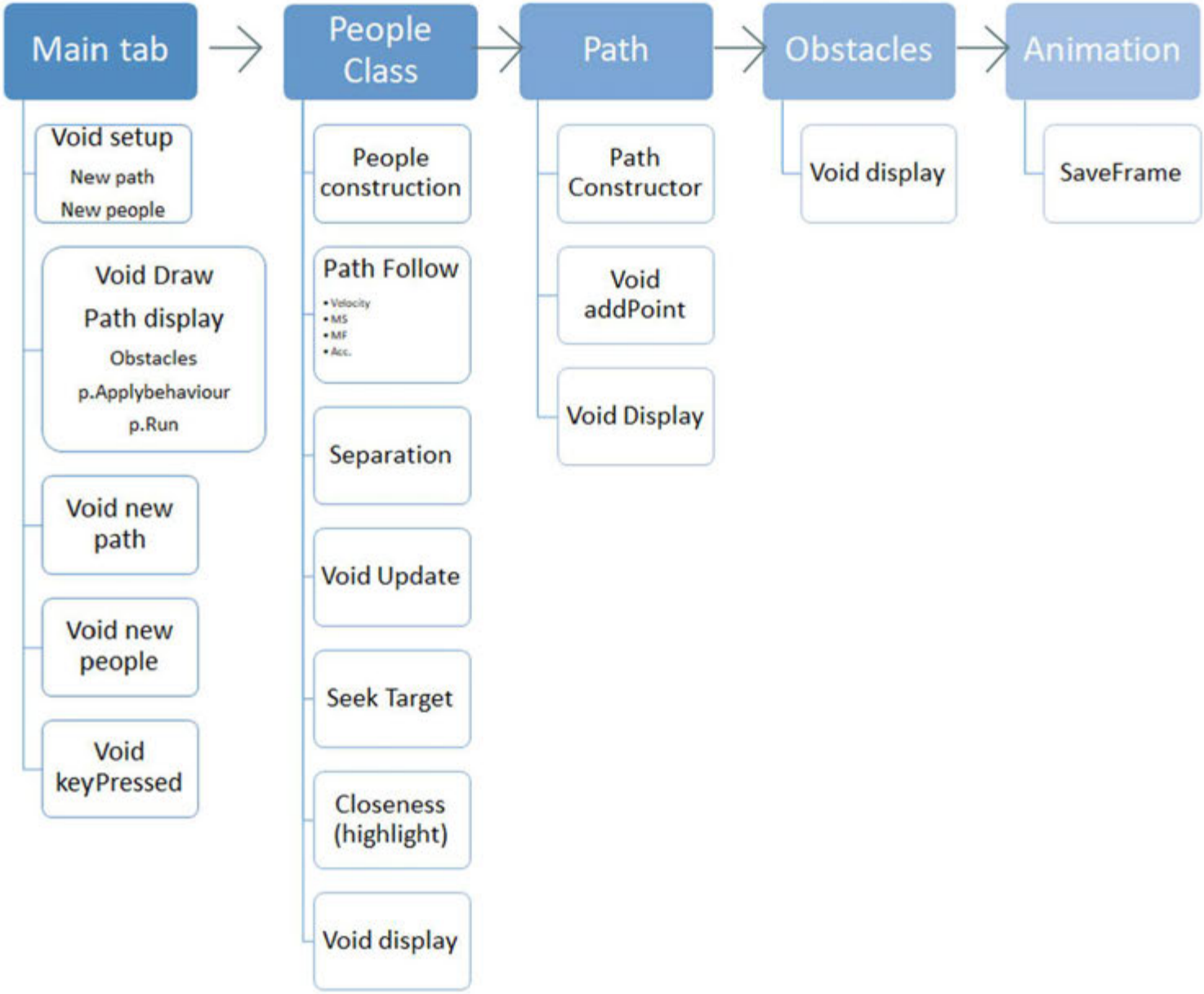
more organized



separation distance : 20

Processing...

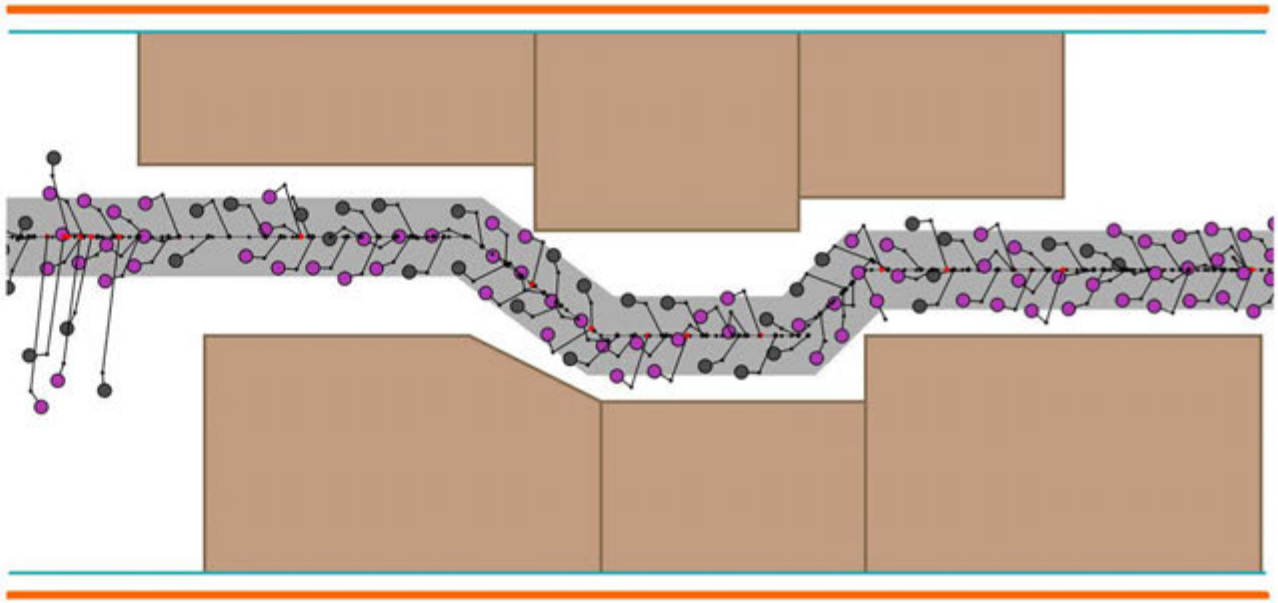
WHAT I DID (FLOWCHART)



AddPeople

With this code in the main tab, we can add the people as much as we want

```
void newPeople(float x, float y) {
  float maxspeed = 3;
  float maxforce = 0.3;
  people.add(new People(new PVector(x,y),-
maxspeed,maxforce));
}


void mousePressed() {
  newPeople(mouseX,mouseY);
}
```



Link for the all animations, I also added relevant animation with each topic via QR

https://youtube.com/playlist?list=PL-f-f7LpYegvGHUup6W36IEnb-5WpkXWy

DISCUSSION

- To summarise, exist walkability of the King Street was simulated.
- Different scenarios were compared to each other.
- In these scenarios, some parameters were changed like path radius, with/without obstacles, weekdays-weekend density by playing a number of people, desired separation distance, etc.
- Street furniture has a big effect on poor walkability on the King Street.
- The density of people is also an important factor because based on my observation, people more prefer businesses on King Street on weekends rather than on weekdays.

REFERENCES

- The Nature of Code, Daniel Shiffman
- Healthy cities — walkability as a component of health-promoting urban planning and design,Minh-Chau Tran,Institute of City Planning and Urban Design, University of Duisburg-Essen, 45141 Essen, Germany
- Walkable Environment in Increasing the Liveability of a City, Article in Procedia - Social and Behavioral Sciences · December 2012
- https://processing.org/reference
- https://www.youtube.com/@TheCodingTrain
- Configraphics,Graph Theoretical Methods for Design and Analysis of Spatial Configurations, Pirouz Nourian, Delft University of Technology, Faculty of Architecture and the Built Environment, Department of Architectural Engineering + Technology / Department of Urbanism
- Bill Hillier's Legacy: Space Syntax—A Synopsis of Basic Concepts, Measures, and Empirical Application, Article