FLOCKING BEHAVIOR OF CROWDS

Shashi Bhushan
M.Eng, Electrical and Computer Engineering
University of Ottawa
Ottawa, Canada
Email: sbhus066@uottawa.ca

***Abstract*: Simulation of crowd in real time is a challenging task demanding careful consideration of efficiency and accuracy tradeoff. Crowd scenarios simulations using the particle based method has have been successful in various applications like military, robotics, film, gaming and architecture. The paper focusses on local dynamics and capture the entity's personal space infringement captured by area based penalty force while focusing on local dynamics. The method provides data parallel implementation that can simulate thousands of entities at interactive frame rates without the need of nearest neighbor search which ca be costly to implement. Moving crowd personal space compression around motion barriers as well as around point of interest for static crowds can be successfully reproduced using the algorithm [1]. The paper also discusses the flocking behavior of crowd and the behavior of a flock under various conditions is compared.**

***Keywords:* Crowd; penalty force; interactive; [1] flock; boids; cohesion force; separation force; alignment force**

I. Introduction

The modelling and simulation is an important technique to study the complex systems that are not practical to be studied in actual physical conditions, at least before the expected behavior is studied. One of the important areas using this technique is in the field of crowd simulation, especially in architectural and urban projects. This helps the designers and programmers to visualize and understand the projects utility and its functionality under various desired scenarios thus facilitating the feedback, discussion and helps in making decisions about how and where to implement them and making changes to the design, if any is required.

One of the demanding areas for crowd simulation is in the gaming and entertainment industry. The industry focusses more on the theoretical accuracy as compared to other practical scenarios where the stability in various unpredictable scenarios, visual believability and other interactive experiences in real time.
The one of the most demanding areas for crowd simulation consists of civil applications where people safety has to be analyzed under scenarios and contingency planning has to be done. The simulations helps concerned agencies to plan measures to deal with possible contingencies like bottleneck, large number of crows in smaller areas with fewer access points and other problems.

The huge variety of application requirements demands an equally varied modelling approach to tackle them. The modelling technique used is a basic java code that simulates human crowd behavior and helps us to model and simulate the behavior in various scenarios under the influence of various forces.

The current crowd simulation model is modelled using already existing centroid particles which is a discrete java based engine for specifically modelling crowd simulations and processing, which is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Crowd flocking behavior simulation capability has been added to the current engine and various forces like separation force, cohesion force and alignment force has been added to simulate the steering behavior of entities, called boids [2], which

are part of a flock. Various scenarios are modelled and the behavior of single entity and flock is simulated, and the results are compared to better understand the behavior. In this paper we also discuss the behavior of flock under various scenarios. These are modelled by calculating the time taken by flock to travel a certain distance under various conditions. This can be seen in various real world scenarios such as: a group of people meeting someone standing at a specific point, an important person moving towards the stage along with his/her colleagues, group of friends going to a restaurant and various other scenarios.

II. Background

Centroid particle crowd simulation is a personal area based method that mimics physical interactions, pressure propagation waves and compression of personal space when provided with dense enough scenarios. This happens due to the interest of crowd towards same area and it starts behaving like compressible fluid even in stationary conditions. The engine was developed by Prof. Gabriel Wainer and Omar Hesham [1]. The algorithm design overview is provided below.

A. Overview

A combined personal space violation map is computed first to simulate the local pedestrian interactions. The penalty forces are then computed. Personal space contribution of each entity is modifiable and can be provided through parameterized weight maps. Each entity's new location can then be calculated by acceleration force's integration over several frames. [1]

B. Personal Space

A contiguous area of personal space (PS) footprint surrounding a 2D particle represents an entity. The personal space increases in the direction in which the entity travels, with the increase in entity's velocity. This models that the humans expect an increase in empty area as they gain speed in any direction.

The entity is moved towards its destination under the influence of a primary directional force and is given by global pathing scheme. The entity is assumed to reach the destination in the absence of any obstacle and if it follows the vector. The reactive behavior of the crowd is maintained by adding a penalty force to the global pathing force. The force is shown in Figure 1.

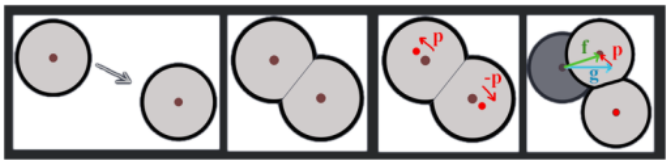


Figure 1. The net force (*f*) experienced by an entity is a linear combination of the global pathing force (*g*), and the penalty force (*p*) which falls along the direction of the new centroid. [1]

C. Particle Parameterization

Personal space is not just a uniformly weighted homogeneous area. Changing the footprint's geometry can be used to modify the local dynamics. This can also be achieved by using a map to influence the weight. Figure 2 shows the artificial variation for the personal space footprint in response to global events or in proximity to point of interest. The examples of these can be a fire alarm evacuation or slowing down near a shop while shopping respectively. The entity type can also cause the change in footprint. For example, an adult's footprint would be different from the footprint of a child. [1]

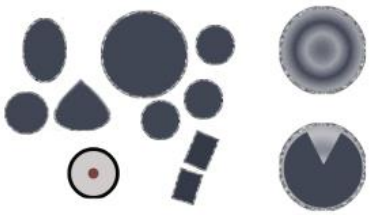


Figure 2. Variations of PS footprints via shape and weight maps. [1]

D. Comfort Speed

Fundamental diagram [3] notion can be employed where, on average, the pedestrian speed and local density are inversely related to each other. Studying the empirical data shows that context of the crowds (indoors vs. crosswalks) or cultures can lead to difference in fundamental diagram. This can be used

to determine individual's comfort speed throughout the simulation. [1]

E. Flocking

The flocking behavior is a behavior shown by a group of entities. The behavior can be simulated where the simulated flock is an elaboration of a particle system with the simulating entities, called boids being the particle [3]. The entities that are part of the flock follow three simple steering behaviors namely Separation, Cohesion and Alignment. Alignment behavior causes a particular entity to line up with the entities near it. Cohesion causes the entities to steer towards the average position of entities in within the certain radius that determines whether the entity is a part of a flock or not. Separation behavior causes an entity to steer away from other neighboring entities. Figure 3 shows all the three steering behaviors.

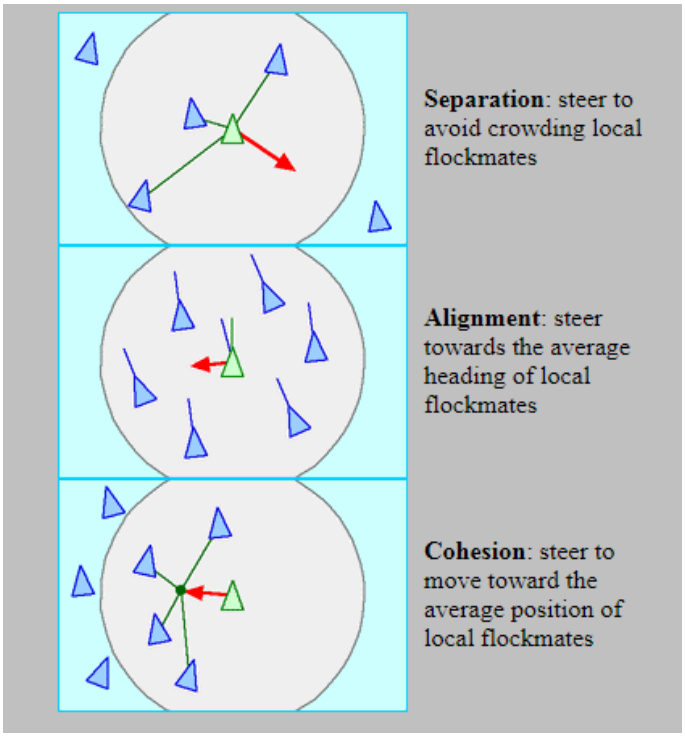


Figure 3. Steering behaviors of entities of a flock.

III. Models

This section shows the existing centroid particle model and the improvements done to simulate the flocking behavior of people under various conditions. The simulation shows few scenarios to show how people behave while moving as a flock. The time taken by a group of people to travel a certain distance while moving as a flock is computed in the presence of crowd and in the absence of crowd. It is then also compared to the time taken by single entity to cover the same distance. All the improvements done in the existing code are under ModFlocking comment.

The first model implements the flocking behavior of crowds using the three steering forces namely

- Separation force: Steering force avoiding local flockmates
- Alignment force: Steers an entity (boid) towards the average heading of local flockmates
- Cohesion force: Steers an entity (boid) towards the average position of local flockmates

The following twelve models are then made to Study the behavior of people under various conditions:

- Time taken by 10 entities displaying flocking behavior to travel 400 units in horizontal direction in the absence of crowd
- Time taken by 10 entities displaying flocking behavior, but with cohesion and separation force off, to travel 400 units in horizontal direction in the absence of crowd
- Time taken by 10 entities displaying flocking behavior to travel 400 units in horizontal direction in the presence of 990 other entities acting as crowd moving in bidirectional flow
- Time taken by 10 entities displaying flocking behavior, but with cohesion and separation force off, to travel 400 units in horizontal direction in the presence of 990 other entities acting as crowd moving in bidirectional flow
- Time taken by 10 entities displaying flocking behavior to travel 400 units in the vertical direction in the absence of crowd
- Time taken by 10 entities displaying flocking behavior, but with cohesion and

separation force off, to travel 400 units in vertical direction in the absence of crowd

- Time taken by 10 entities displaying flocking behavior to travel 400 units in vertical direction in the presence of 990 other entities acting as crowd moving in bidirectional flow
- Time taken by 10 entities displaying flocking behavior, but with cohesion and separation force off, to travel 400 units in vertical direction in the presence of 990 other entities acting as crowd moving in bidirectional flow
- Time taken by one entity to travel 400 units in horizontal direction in the absence of crowd
- Time taken by one entity to travel 400 units in horizontal direction in the presence of 990 other entities acting as crowd moving in bidirectional flow
- Time taken by one entity to travel 400 units in vertical direction in the absence of crowd
- Time taken by one entity to travel 400 units in vertical direction in the presence of 990 other entities acting as crowd moving in bidirectional flow

A. *Code with flocking functionality*

The code is divided into various sections each with its own different functions. For our simulation, all of the changes are done in mods.pde file. The only change apart from this file is in the CentroidalSofticles.pde file in which only the CROWDCOUNT value is changes as per the requirement of the model. The various improvements done in the code are discussed below.

1) *Global variables Section:* This section declares and initializes various global variables. The Variables under ModFlocking have been added to implement flocking behavior. Flock radius value is used to check whether the entities are part of the flock or not. The three flags are used to turn the forces on or off separately. The weight variables decides how strong each of the steering force will be. The models discussed always use this particular weight and these values can be played with to get various behaviors as required.

```
//****************** Global Variables Section *********************//
//*****************************************************************//
int[] specialIDs;               // A list of IDs of the modified entities
PVector[] initialPositions;     // A list of their initial positions
PVector groupAvgPosition;       // Keeps track of the average of special entity positions
PVector mousePosition;          // Keeps track of mouse position
boolean meetingOccured;         // Becomes true if the special entities have met/gathered
PImage obstacleMap;             // Holds an image of the obstacle map
int obstacleChoice;             // Choice of multiple available obstacles
boolean recordTimeToMeeting;    // Enable printing of Time-To-Meeting (in seconds)

int WIDTH = 500, HEIGHT = 600;   // Canvas dimensions (1 pixel = 10cm in real life)
// Non-special entities initial distribution (see EntitySystem.initializePositions for options)
String NONSPECIAL_INIT_DISTRIBUTION = "bidirectional";
// Are they in bidirectional flow, or standing still
boolean NONSPECIAL_BIDIRECTIONAL = true;
// Simulation starts paused?
boolean START_PAUSED = false;
```

```
//ModFlocking
int flockRadius = 50;
boolean cohesionFlag = true; // Flag to control the cohesion force On/Off
boolean separationFlag = true; // Flag to control the separation force On/Off
boolean alignmentFlag = true; // Flag to control the alignment force On/Off
PVector[] cohesionForcesArray;  // array to hold all cohesion forces
PVector[] separationForcesArray;  // array to hold all separation forces
PVector[] alignmentForcesArray;   // array to hold all alignment forces
float cohesionWeight = 0.5;  // Weight of cohesionForce. change in the force changes the strength of cohesion force
float separationWeight = 0.4;  //Weight of separationForce. change in the force changes the strength of cohesion force
float alignmentWeight = 0.5;  //Weight of alignmentForce. change in the force changes the strength of cohesion force
```

2) *State Initialization Section:* This section runs only once on startup and specifies special entity IDs, initial positions etc. Here we specify our 10 special entities and initialize our forces array used for drawing the force vectors. We can see that the entities start from a specific x-axis and within a range of y-axis. The range is equal to the neighbor radius range to maintain the flock.

```
//****************** State Initialization Section *************************//
// Specify special entity IDs, intitial positions, etc. Runs once on startup.
// This section is similar to CellDEVS inital value (.val) files.
//*************************************************************************//
void modification_Initialization(){
  // Specify the IDs of the special entities using an explicit list
  specialIDs = new int[]{0,1,2,3,4,5,6,7,8,9};


//    //Or procedurally in a loop
   initialPositions = new PVector[specialIDs.length]; // allocate empty array
   for(int i=0; i<specialIDs.length; i++){
     // ModFlocking : specific location cluster
     initialPositions[i] = new PVector(random(20,20), random(275,325));
   }
```

```
//ModFlocking
separationForcesArray = new PVector[specialIDs.length]; // initialize separation forces array
cohesionForcesArray = new PVector[specialIDs.length]; // initialize cohesion forces array
alignmentForcesArray = new PVector[specialIDs.length]; // initialize alignment forces array
```

3) *Update Section:* The values in this section are computed once every frame. Various values are computed and may not be used. It is up to the user whether to use the computed values or not. There are no changes in the section for normal flocking behavior. But for computing the time taken by the entities to travel 400 units, additions have been added that can be slightly modified for various cases.

```
//ModFlocking: Above code is commented out as meeting occured condition is changed for flocking. New logic is added for flocking
//ModFlocking: To see when all the special entities (boids) cross the desired distance
for (int i=0; i<specialIDs.length; i++) {
  PVector pos = crowd.getEntities()[specialIDs[i]].getPosition();
  if(pos.x >= travelDistance){
  specialIDsMeet[i] = true;
  }
}

//ModFlocking: meetingOccured is true only if all the boids have reached or crossed the desired distance
if(specialIDsMeet[0] == true && specialIDsMeet[1] == true && specialIDsMeet[2] == true && specialIDsMeet[3] == true && specialIDsMeet[4] == true &&
specialIDsMeet[5] == true && specialIDsMeet[6] == true && specialIDsMeet[7] == true && specialIDsMeet[8] == true && specialIDsMeet[9] == true ){
   meetingOccured = true;
}

// Print the Time-to-Meeting?
```

4) *Forces Section*: In this section, forces for every special entities is customized. Three steering forces are computed in this section for modelling flocking behavior.
   a. *Separation force:* Steering force avoiding local flockmates
   b. *Alignment force:* Steers an entity (boid) towards the average heading of local flockmates
   c. *Cohesion force:* Steers an entity (boid) towards the average position of local flockmates

```
//****************** Forces Section *********************************//
// Here, we customize the forces for every special entities.
//   The net modification force (i.e. due to Princy and Walter's
//   contributions) is stored in the variable modForce.
//   This is similar to DEVS state transition functions.
//*******************************************************************//
void modification_Forces(){
  // For each special entity:
  for(int i=0; i<specialIDs.length; i++){
    Entity entity = crowd.getEntities()[specialIDs[i]];    // get the special entity
    PVector pos   = entity.getPosition();                  // get its current position
    PVector modForce = new PVector(0, 0, 0);               // modification net force (starts empty)

    int num_flockmates = 0;                                // holds the number of neighbours/flockmates based on flockRadius
    PVector separationForce = new PVector(0, 0, 0);        // holds the separation force
    PVector cohesionForce = new PVector(0, 0, 0);          // holds the cohesion force
    PVector alignmentForce = new PVector(0, 0, 0);         // holds the alignment force

    //ModFlocking loop: query my neighbours
```

```
//ModFlocking loop: query my neighbours
for(int j=0; j<specialIDs.length; j++){                // this loop is required as the forces are calculated depending on all the neighbours
  if(i==j) continue;                                   // do not calculate for self
  Entity flockmate = crowd.getEntities()[specialIDs[j]]; // get the special entities

  // calculate the distance to determine whether the boid can be consideres flockmate depending on flockRadius
  PVector posMate  = flockmate.getPosition();
  float dist = PVector.sub(posMate, pos).mag();

  if(j != i && dist > 0 && dist < flockRadius ){       // do not calculate for self

    // add the forces of the neighbours
    separationForce.x += flockmate.getPosition().x - entity.getPosition().x;
    separationForce.y += flockmate.getPosition().y - entity.getPosition().y;

    cohesionForce.x += flockmate.getPosition().x;
    cohesionForce.y += flockmate.getPosition().y;

    alignmentForce.x += flockmate.getVelocity().x;
    alignmentForce.y += flockmate.getVelocity().y;

    num_flockmates++;
  }
}
```

```
// Calculate flocking forces
if(num_flockmates != 0){
  separationForce.x /= num_flockmates;  //get the average of all the neighbour/flockmates(j)  positions from selected boid(i)
  separationForce.y /= num_flockmates;

  separationForce.x *= -1;            // multiply by -1 as we want separation
  separationForce.y *= -1;

  cohesionForce.x /= num_flockmates;  //get the average positions depending on all the neighbour/flockmates(j)  positions
  cohesionForce.y /= num_flockmates;

  cohesionForce.x -= entity.position.x;  // get the position of boid(i) from the avarage position
  cohesionForce.x -= entity.position.x;

  alignmentForce.x /= num_flockmates;    // get the average of all neighbor/flockmates(j) velocities
  alignmentForce.y /= num_flockmates;

  // Normalize all three vectors calculated
  separationForce.normalize();  // final separation force separating the selected boid(i) from all neighbors/flockmates(j)
  separationForcesArray[i] = separationForce;  // used for visualization in modification_Visualization()

  cohesionForce.normalize();    // final cohesion force bringing the selected boid(i) close to neighbors/flockmates(j)
  cohesionForcesArray[i] = cohesionForce; // used for visualization in modification_Visualization()

  alignmentForce.normalize();   // final alignment force aligning the velocity of selected boid(i) to all the neighbors/flockmates(j)
  alignmentForcesArray[i] = alignmentForce; // used for visualization in modification_Visualization()

}
```

```
//ModFlocking
// The flags gives the option of disabling and re-enabling any of three steering forces.
// Can be controlled by prsessing keyboard keys: Cntrl for cohesionForce, Shift for separationForce and Alt for alignmentForce
if(separationFlag){
  modForce.add(separationForce.mult(separationWeight));
}
if(cohesionFlag){
modForce.add(cohesionForce.mult(cohesionWeight));
}
if(alignmentFlag){
modForce.add(alignmentForce.mult(alignmentWeight));
}
// Apply the net mod force to the entity, and update its position
entity.setForce(modForce.mult(0.1)).updatePositionMod();
}
```

5) *Visualization Section:* This section is used to describe how the special entities are displayed on the screen. In this section, modifications have been done to show the flocking forces for debugging and visualization.

```
//ModFlocking : draw the flocking forces for debugging and visualization
for(int i=0; i<specialIDs.length; i++){
  PVector pos = crowd.getEntities()[specialIDs[i]].getPosition(); // get entity position
  color redArrow = color(250,50,30);  // create a red color (R:250, G:50, B:30)
  stroke(redArrow);                   // set the stroke color to red
  line(pos.x, pos.y, pos.x+20*separationForcesArray[i].x, pos.y+20*separationForcesArray[i].y);
  line(pos.x, pos.y, pos.x+20*cohesionForcesArray[i].x, pos.y+20*cohesionForcesArray[i].y);
  line(pos.x, pos.y, pos.x+20*alignmentForcesArray[i].x, pos.y+20*alignmentForcesArray[i].y);
  noStroke();     //reset stroke color
}
```

6) *User Interface section:* This section handle user inputs, especially keyboard inputs. The keys Alt, Shift and Ctrl can be used to turn the Alignment force, Separation force and Cohesion force respectively. This is

controlled by toggling the particular force flags.

```
void keyPressed() {
  // If pressed key is special (see: https://processing.org/reference/keyCode.html)
  if(key == CODED) {
    switch(keyCode){
      case LEFT:    /* do something interesting when left arrow pressed */; break;
      case RIGHT:   /* do something interesting when right arrow pressed*/; break;
      case UP:      /* do something interesting when up arrow pressed*/;    break;
      case DOWN:    /* do something interesting when down arrow pressed*/;  break;
      //ModFlocking
      case ALT:     alignmentFlag = !alignmentFlag;  println("Alignment: "+ alignmentFlag); break;
      case SHIFT:   separationFlag = !separationFlag; println("Separation: "+ separationFlag); break;
      case CONTROL: cohesionFlag = !cohesionFlag; println("Cohesion: "+ cohesionFlag); break;
    }
  }
```

B. *For travelling 400 units horizontally/vertically*

To calculate the time taken by the entities to travel 400 units in a direction, few changes are required. First, the initial position of the entities has to be defined. Changes for horizontal and vertical cases respectively are shown.

```
initialPositions = new PVector[specialIDs.length]; // allocate empty array
for(int i=0; i<specialIDs.length; i++){
  // ModFlocking : specific location cluster
  initialPositions[i] = new PVector(random(20,20), random(275,325));
}
```

```
initialPositions = new PVector[specialIDs.length]; // allocate empty array
for(int i=0; i<specialIDs.length; i++){
  // ModFlocking : specific location cluster
  initialPositions[i] = new PVector(random(225,275), random(20,20));
}
```

Then a position is set at the horizontal end in the update section to attarct the entities towards it.

```
//ModFlocking
horizontalEnd = new PVector (500, 300, 0);
```

```
//ModFlocking
//horizontalEnd = new PVector (500, 300, 0);
verticalEnd = new PVector (250, 600, 0);
```

Now a force is calculated in the forces section that forces the entities to move towarsd the particular point defined.

```
//ModFlocking
 PVector crossHorizontalForce = PVector.sub(horizontalEnd, pos).limit(1);
 modForce.add(crossHorizontalForce.mult(1));
```

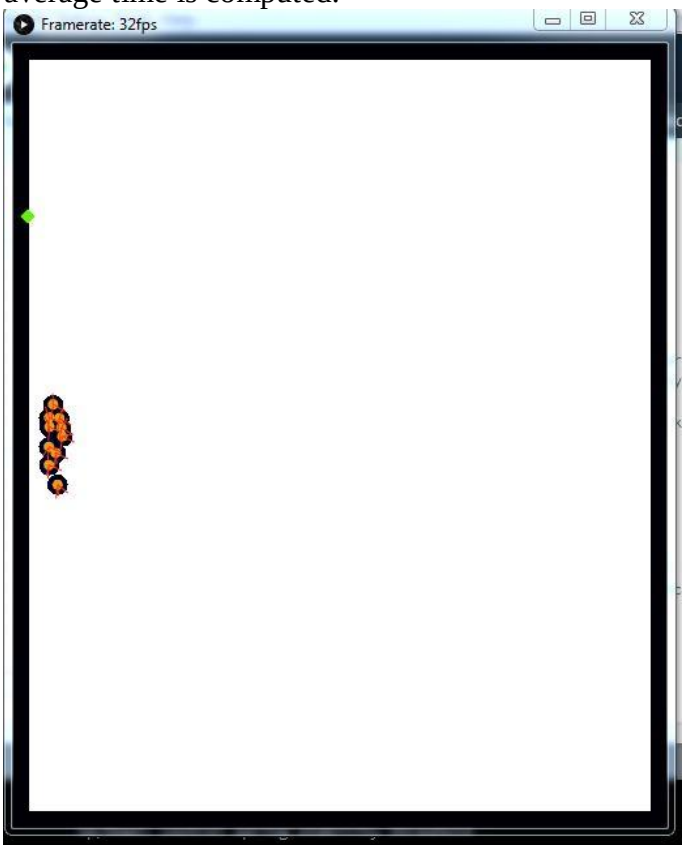
```
PVector crossVerticalForce = PVector.sub(verticalEnd, pos).limit(1);
modForce.add(crossVerticalForce.mult(1));
```

IV. Simulation Results

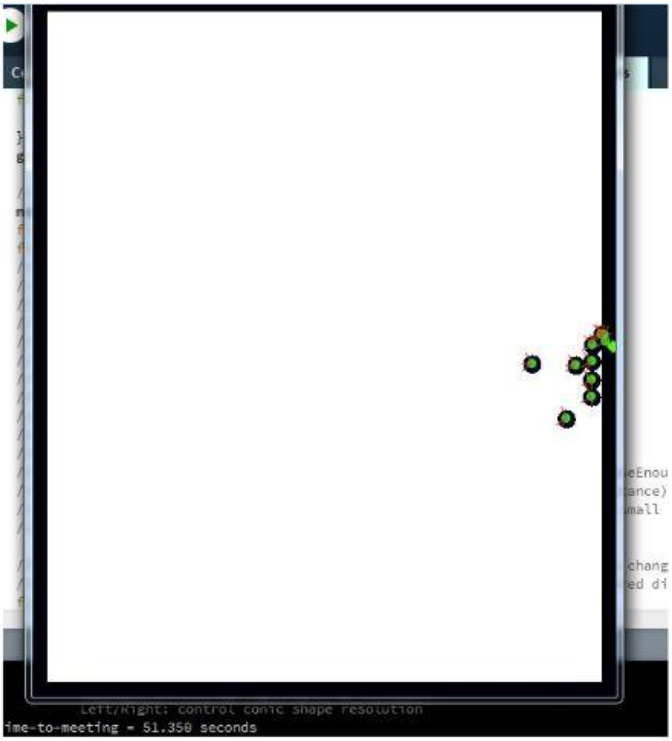
Simulation results of all twelve cases discussed in section III are discussed. For all cases, time taken by entities to travel 400 units is computed. Three simulations are performed for each case and the average time is computed. A comparison of all the cases is then made to see the behavior of flocks under various conditions

A. *Entities as flock travelling in horizontal direction with no crowd*

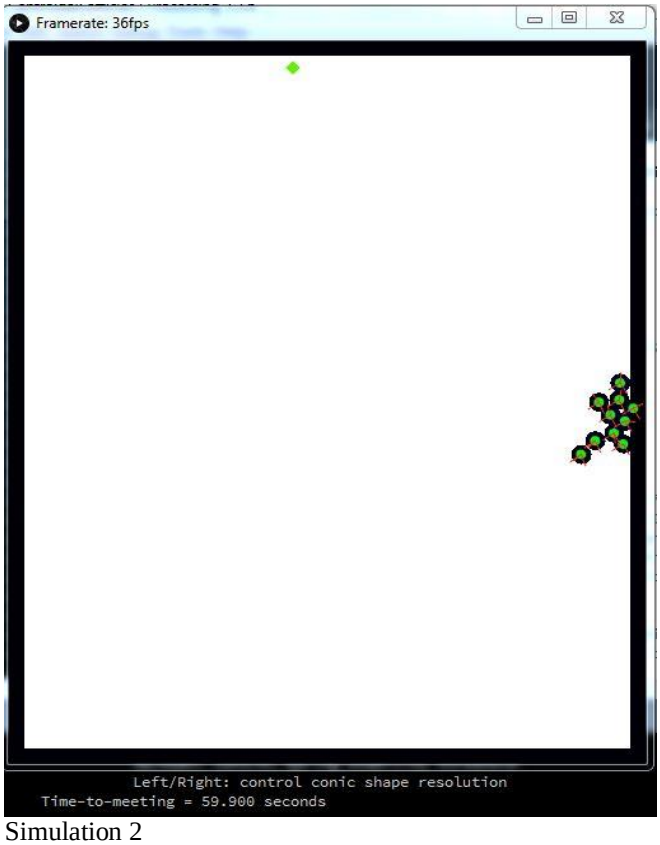
Simulation is performed for ten entities moving as a flock in the horizontal direction in the absence of any crowd. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.



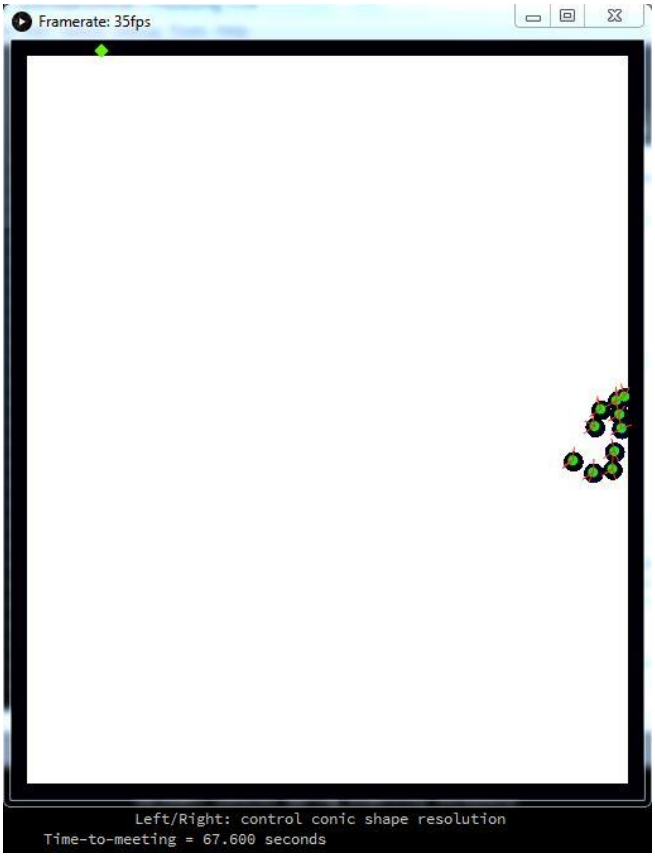
Start



Simulation 1



Simulation 2

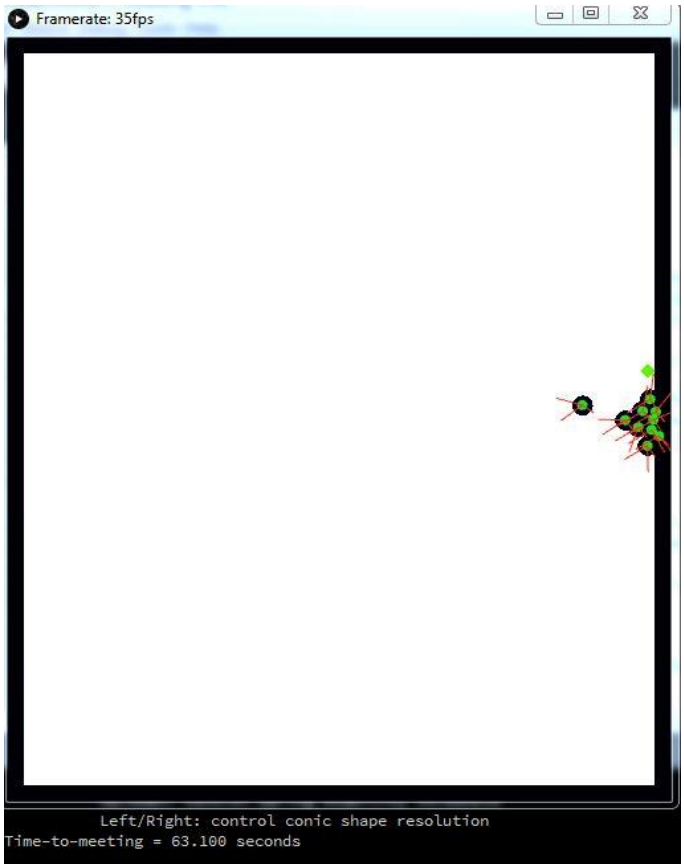


Simulation 3

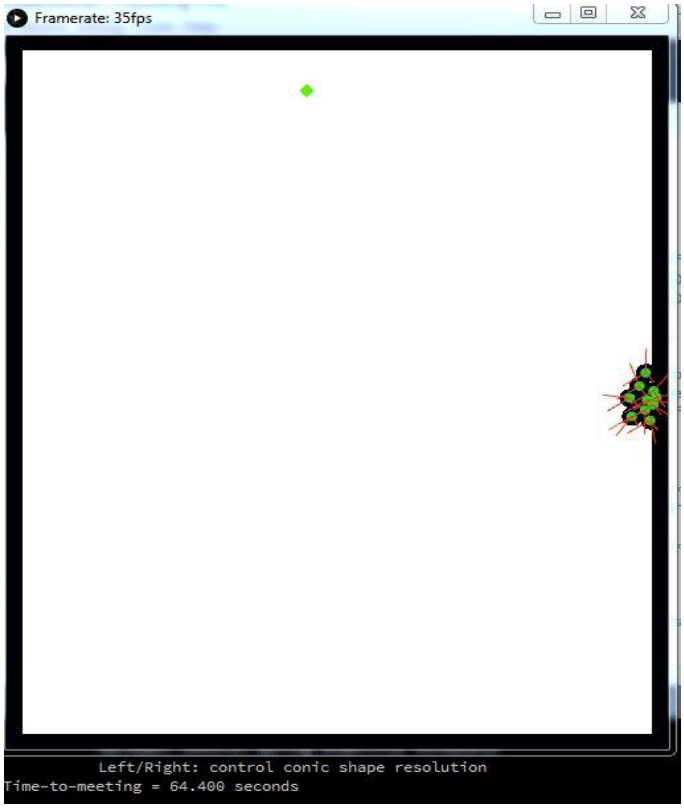
Time taken by 10 special entities to travel 400 units (in sec). Horizontal; Flocking; No crowd	
Simulation 1	51.350
Simulation 1	59.900
Simulation 1	67.600
Average	59.616

B. Entities as flock travelling in horizontal direction with no crowd and cohesion and separation forces off

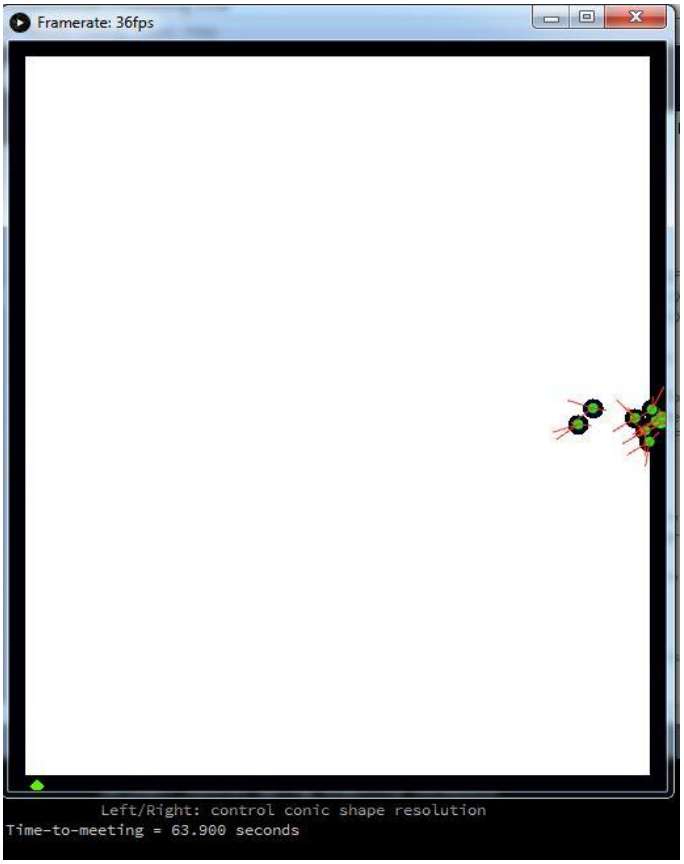
Simulation is performed for ten entities moving as a flock in the horizontal direction in the absence of crowd. The cohesion and separation forces are turned off. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.



Simulation 1



Simulation 2

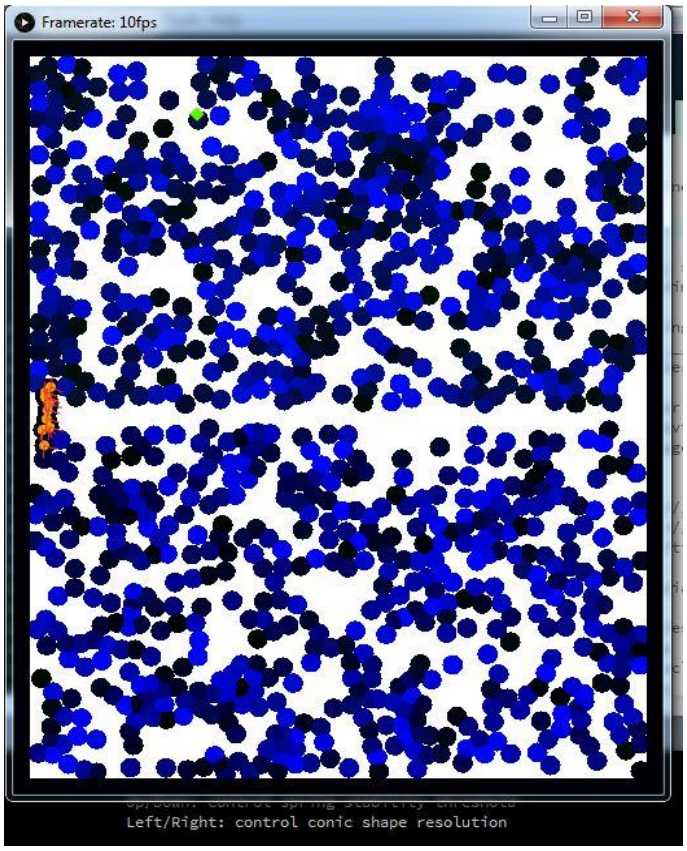


Simulation 3

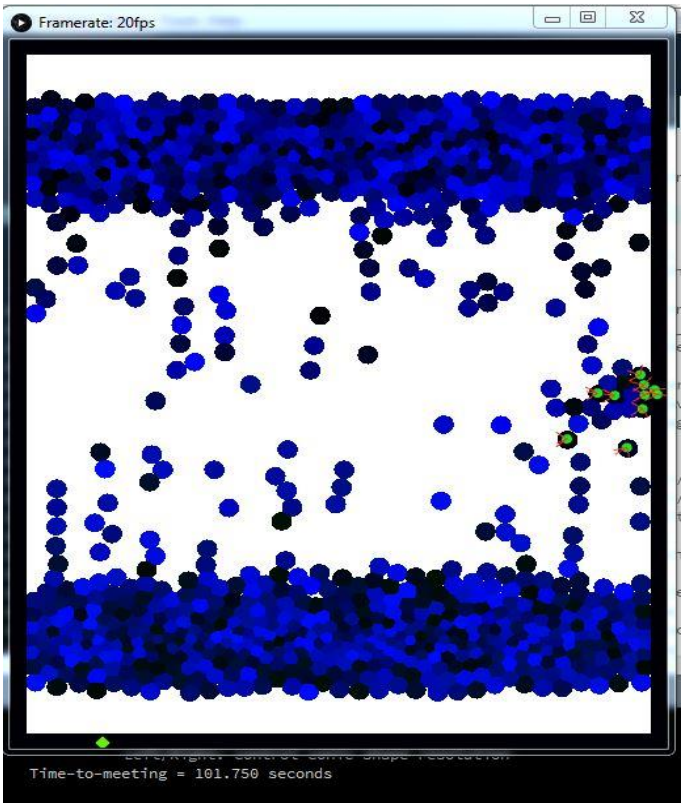
Time taken by 10 special entities to travel 400 units (in sec). Horizontal; Flocking; No crowd; Separation and Cohesion forces off	
Simulation 1	63.100
Simulation 1	64.400
Simulation 1	63.900
Average	63.8

C. Entities as flock travelling in horizontal direction in the presence of crowd

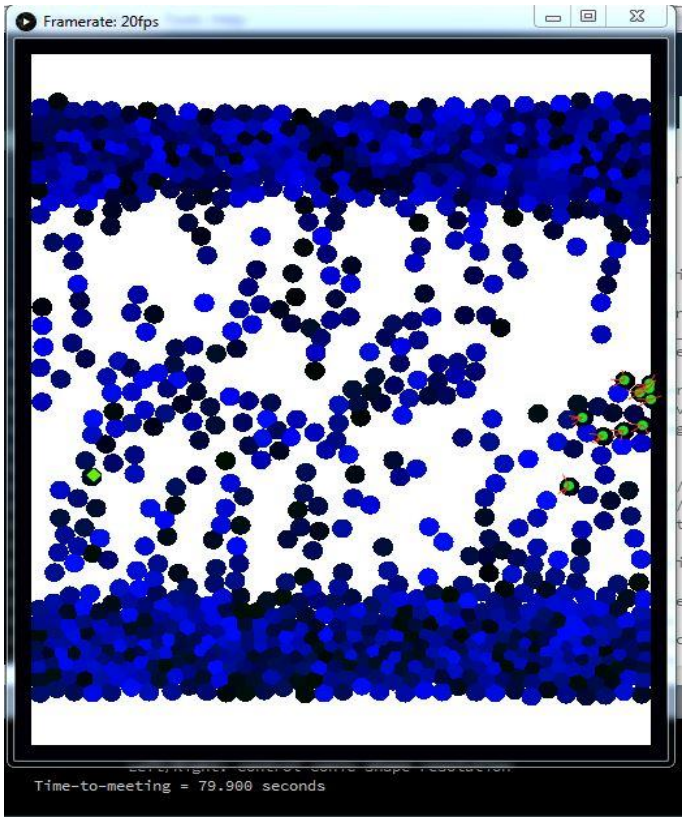
Simulation is performed for ten entities moving as a flock in the horizontal direction in the presence of 990 other entities acting as crowd and moving in bidirectional flow. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.



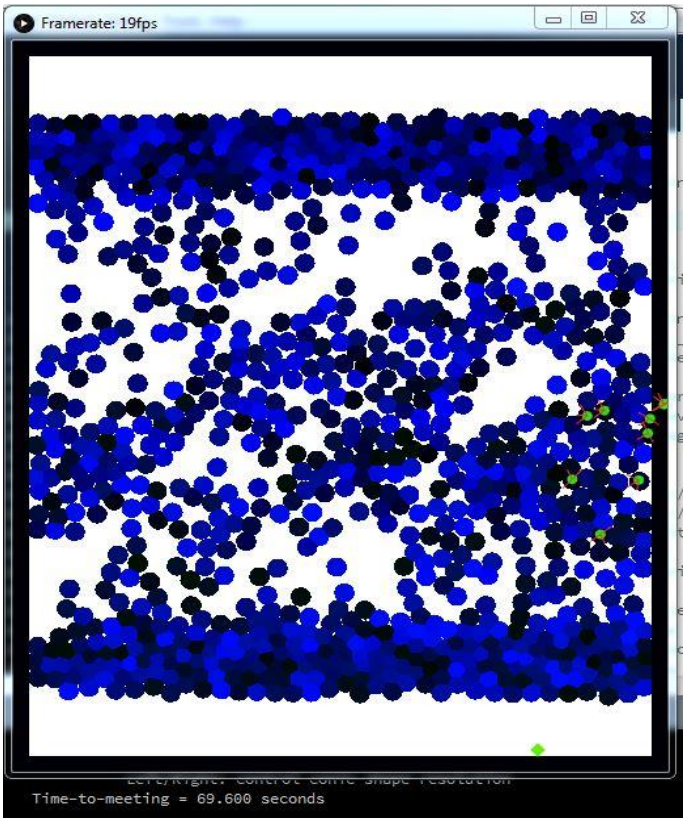
Start



Simulation 2



Simulation 1

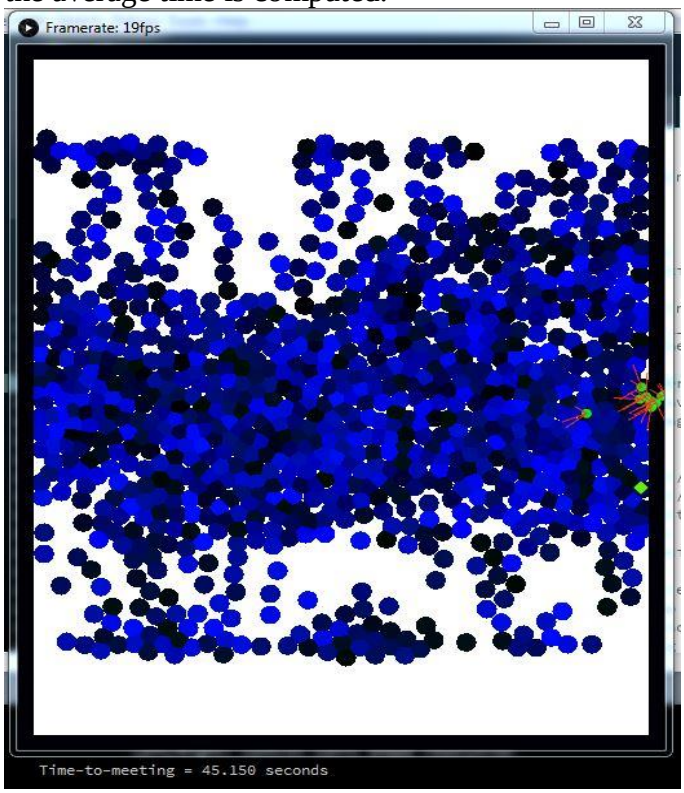


Simulation 3

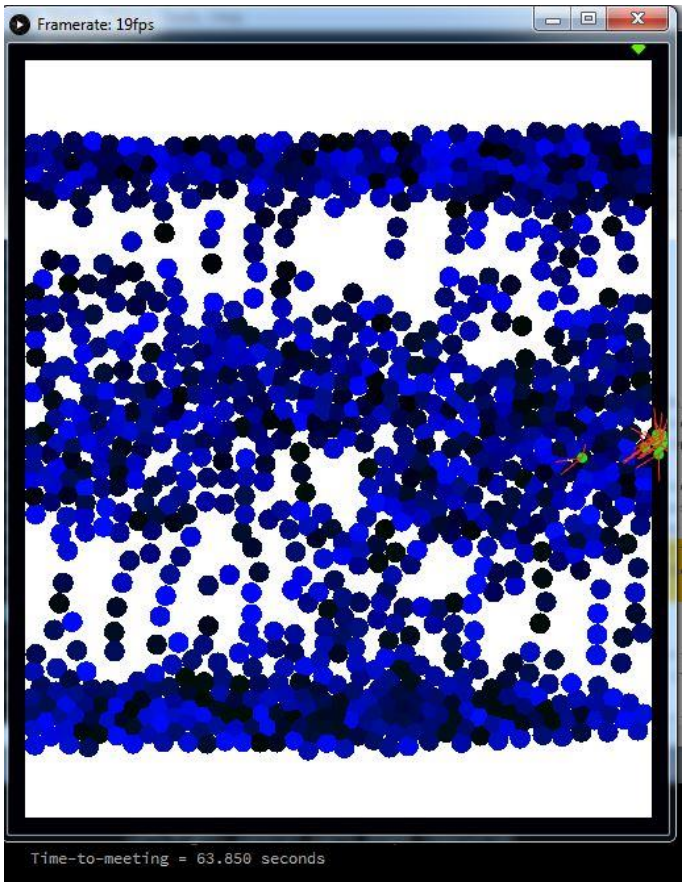
Time taken by 10 special entities to travel 400 units (in sec). Horizontal; Flocking; With crowd (990 other entities in bidirectional flow)	
Simulation 1	79.900
Simulation 1	101.750
Simulation 1	69.600
Average	83.75

D. Entities as flock travelling in horizontal direction in the presence of crowd with cohesion and separation off

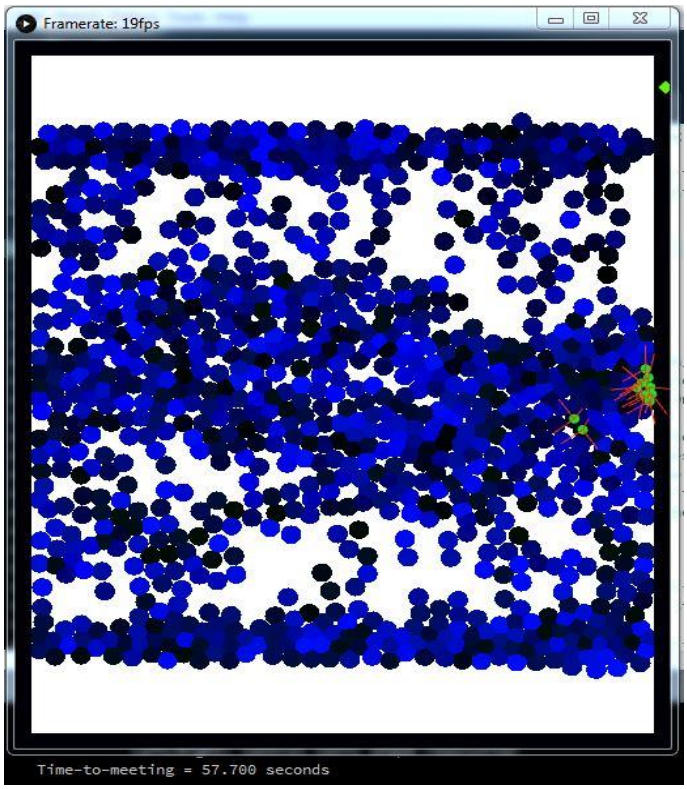
Simulation is performed for ten entities moving as a flock in the horizontal direction in the presence of 990 other entities acting as crowd and moving in bidirectional flow. The cohesion and separation forces are turned off. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.



Simulation 1



Simulation 2

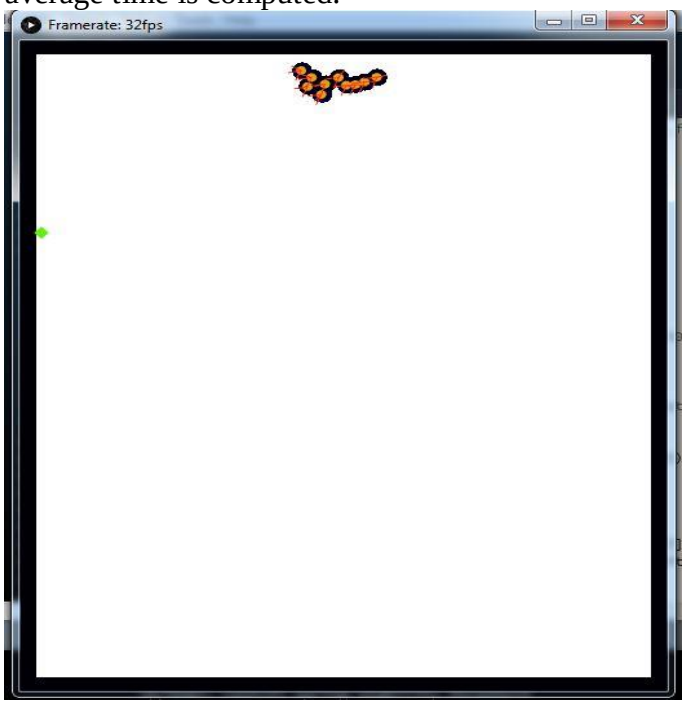


Simulation 3

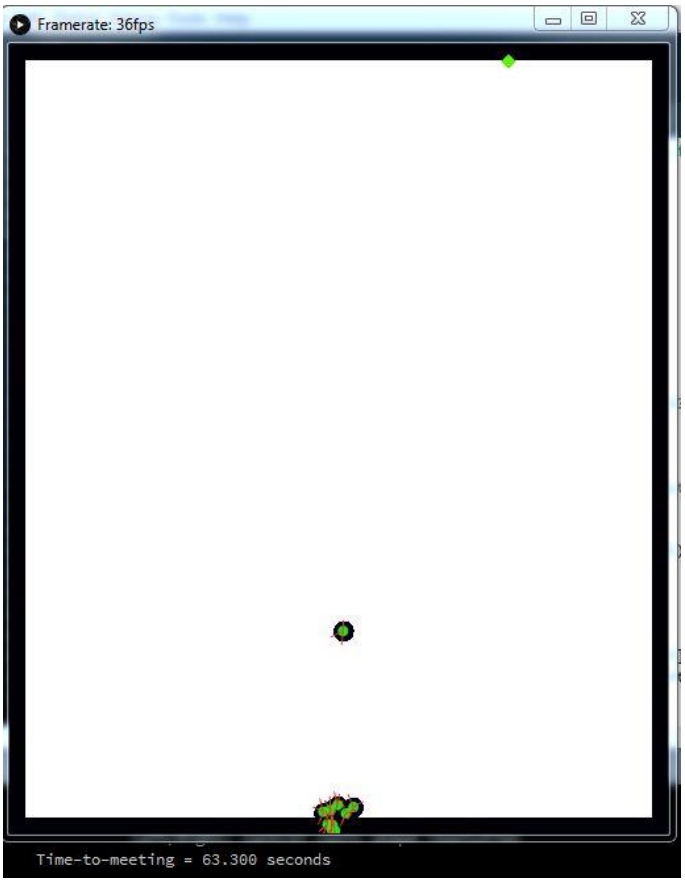
Time taken by 10 special entities to travel 400 units (in sec). Horizontal; Flocking; With crowd (990 other entities in bidirectional flow); Separation and Cohesion forces off	
Simulation 1	45.150
Simulation 1	63.850
Simulation 1	57.700
Average	55.56

E. Entities as flock travelling in vertical direction with no crowd

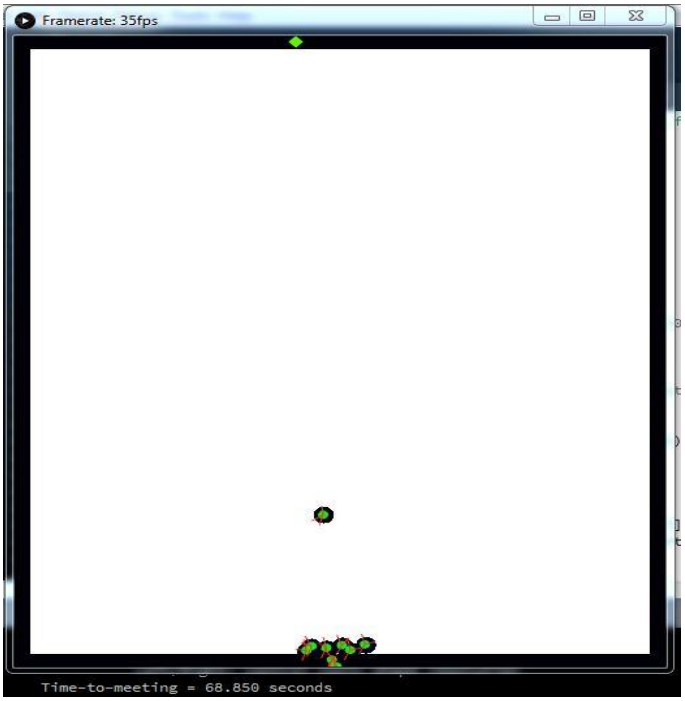
Simulation is performed for ten entities moving as a flock in the vertical direction in the absence of any crowd. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.



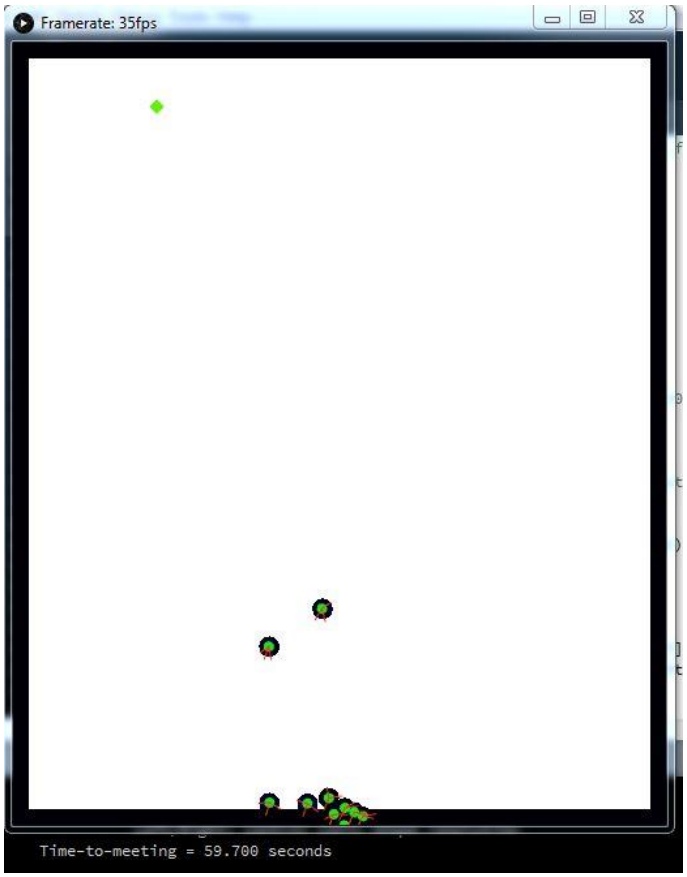
Start



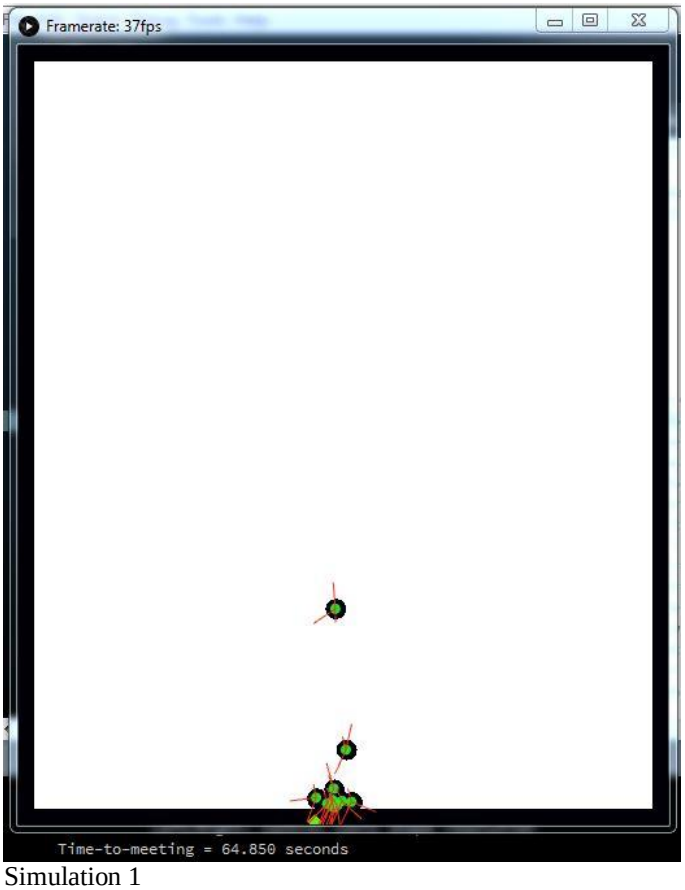
Simulation 1



Simulation 2



Simulation 3



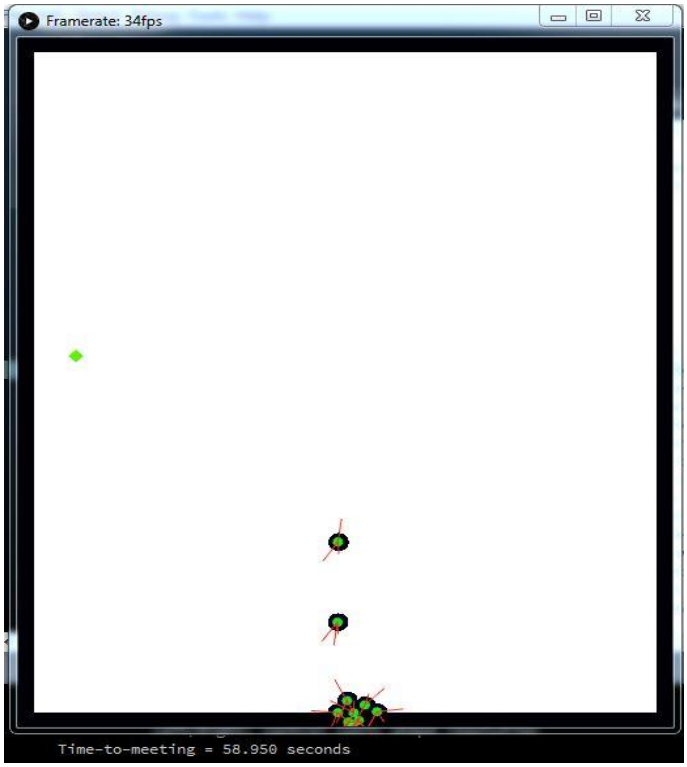
Simulation 1

Time taken by 10 special entities to travel 400 units (in sec). Vertical: Flocking: No crowd	
Simulation 1	63.300
Simulation 1	68.850
Simulation 1	59.700
Average	63.95

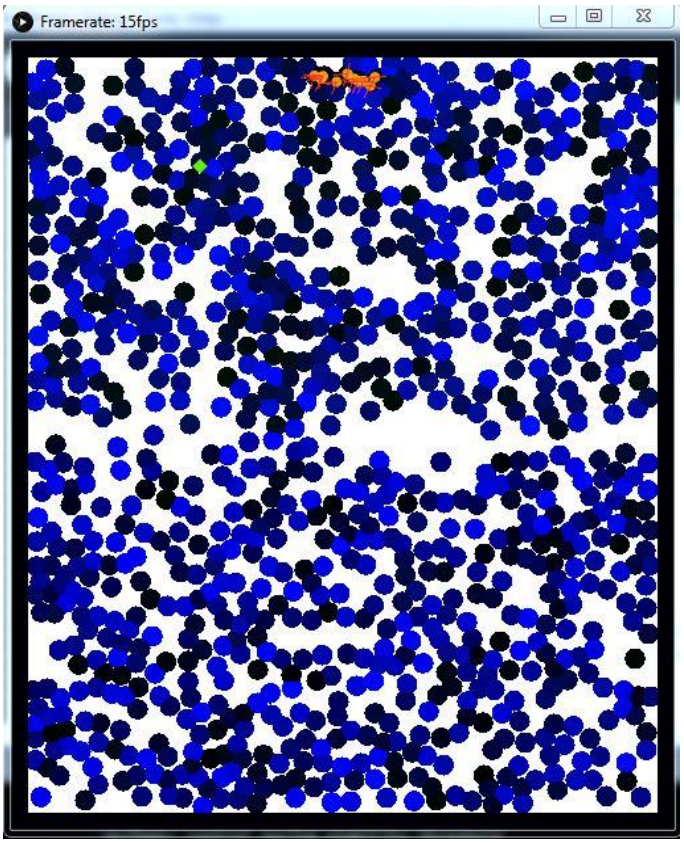
F. Entities as flock travelling in vertical direction with no crowd and cohesion and separation forces off

Simulation is performed for ten entities moving as a flock in the vertical direction in the absence of crowd. The cohesion and separation forces are turned off. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.

Simulation 2



Simulation 3

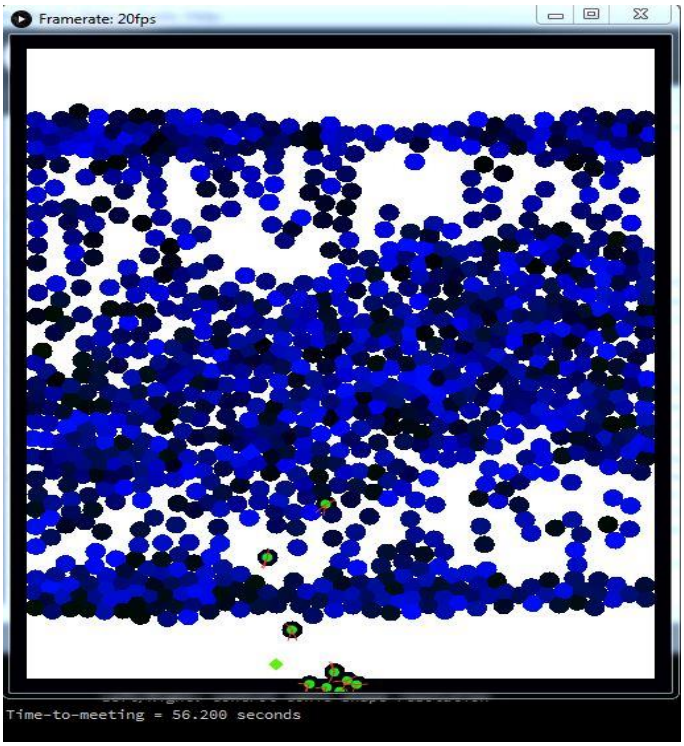


Start

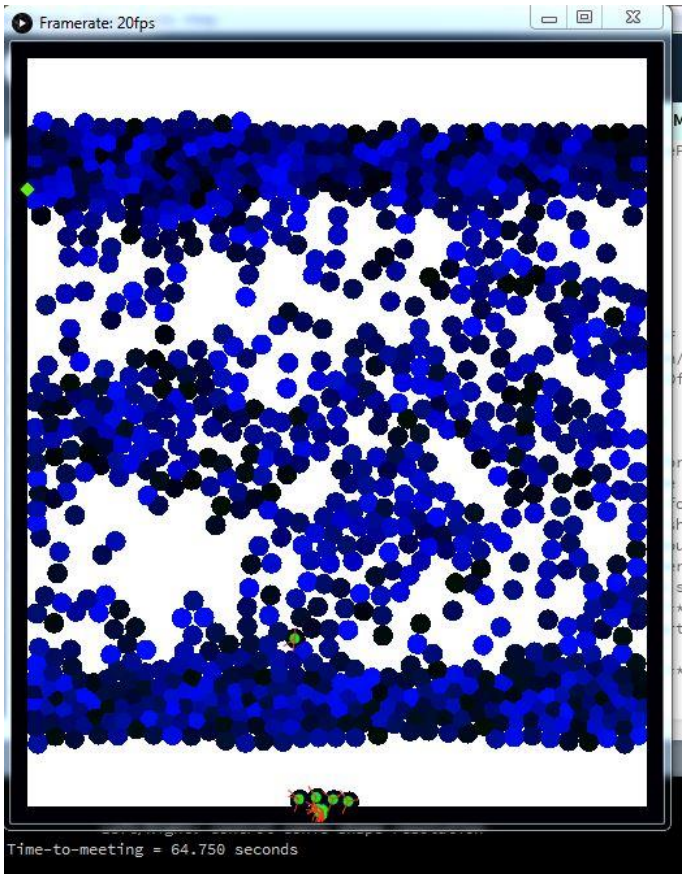
Time taken by 10 special entities to travel 400 units (in sec). Vertical; Flocking; No crowd; Separation and Cohesion forces off	
Simulation 1	64.850
Simulation 1	48.800
Simulation 1	58.950
Average	57.53

G. Entities as flock travelling in vertical direction in the presence of crowd

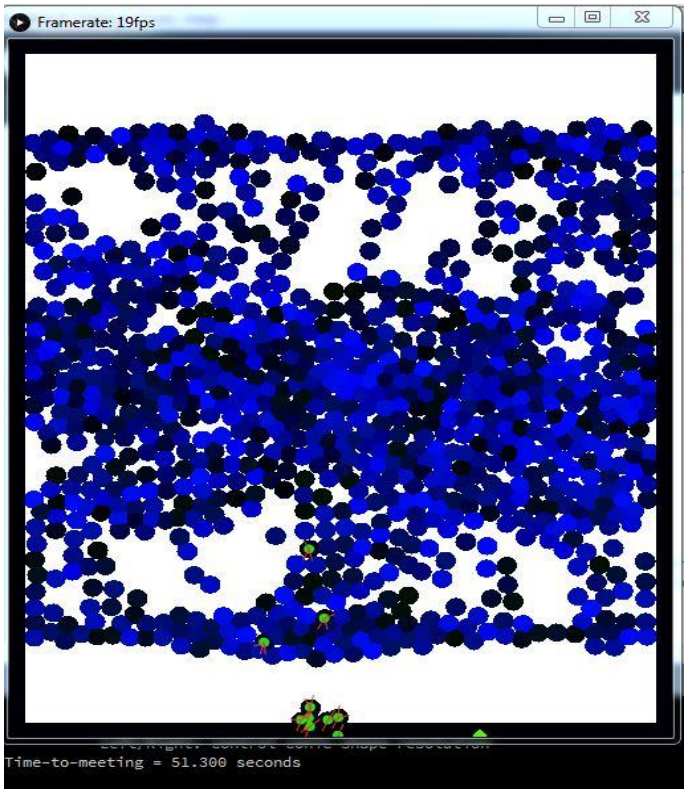
Simulation is performed for ten entities moving as a flock in the vertical direction in the presence of 990 other entities acting as crowd and moving in bidirectional flow. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.



Simulation 1



Simulation 2

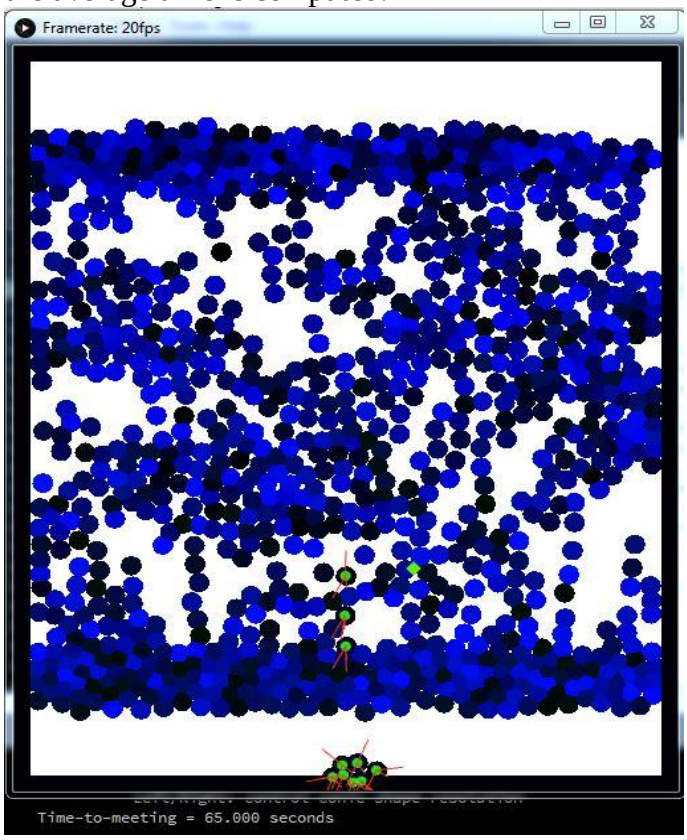


Simulation 3

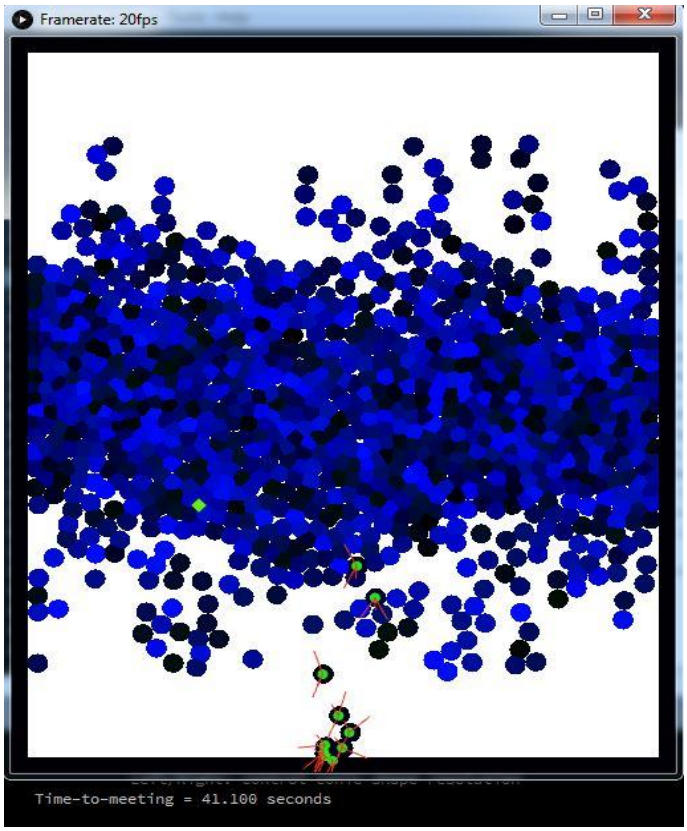
Time taken by 10 special entities to travel 400 units (in sec). Vertical; Flocking; With crowd (990 other entities in bidirectional flow)	
Simulation 1	56.200
Simulation 1	64.750
Simulation 1	51.300
Average	57.41

H. Entities as flock travelling in vertical direction in the presence of crowd with cohesion and separation off

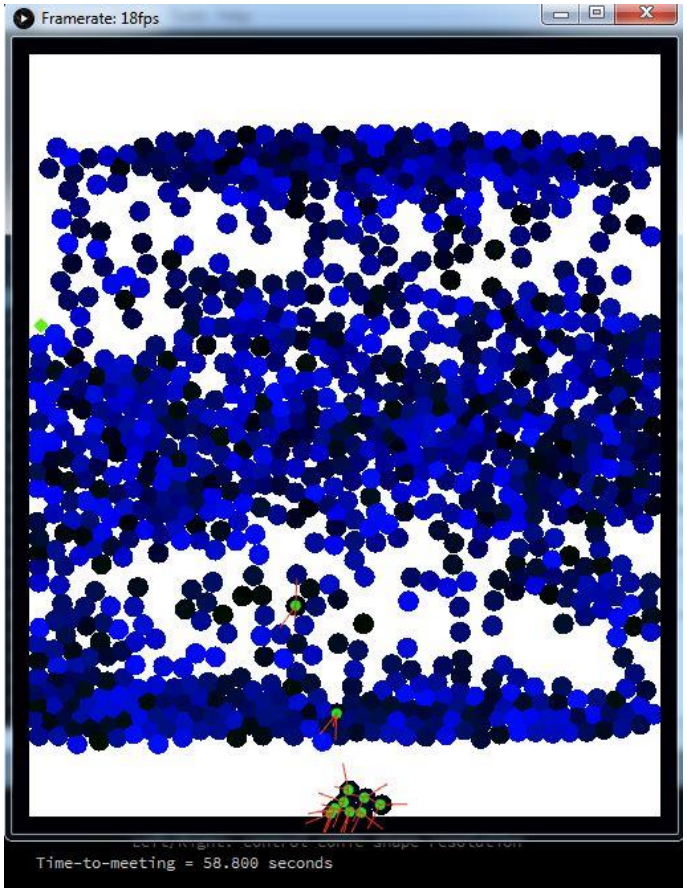
Simulation is performed for ten entities moving as a flock in the vertical direction in the presence of 990 other entities acting as crowd and moving in bidirectional flow. The cohesion and separation forces are turned off. Time taken by the flock to travel 400 units is measured in three simulations and the average time is computed.



Simulation 1



Simulation 2

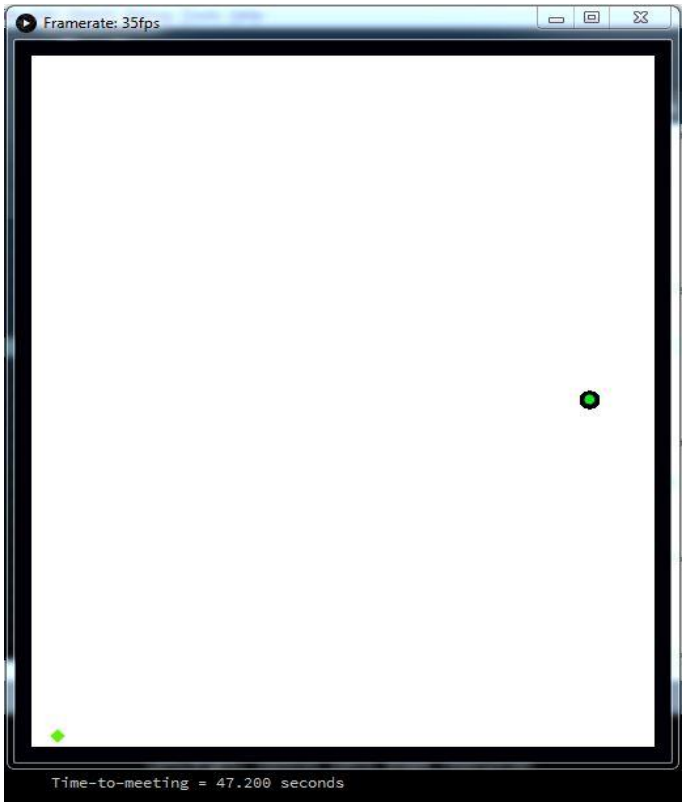


Simulation 3

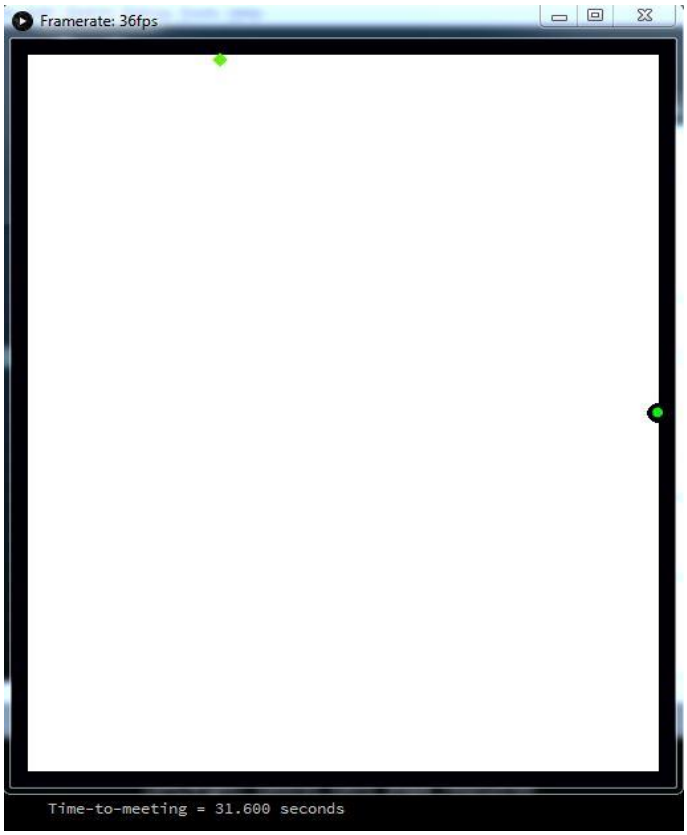
Time taken by 10 special entities to travel 400 units (in sec). Vertical; Flocking; With crowd (990 other entities in bidirectional flow); Separation and Cohesion forces off	
Simulation 1	65.0
Simulation 1	41.100
Simulation 1	58.800
Average	54.96

I. Single entity *travelling in horizontal direction with no crowd*

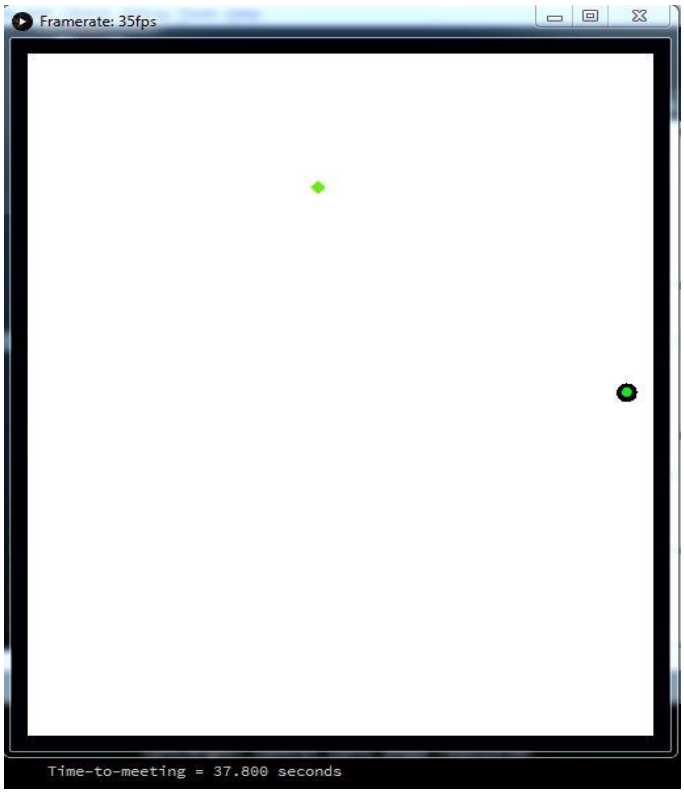
Simulation is performed for a single entity in the horizontal direction in the absence of any crowd. Time taken by the entity to travel 400 units is measured in three simulations and the average time is computed



Simulation 1



Simulation 2

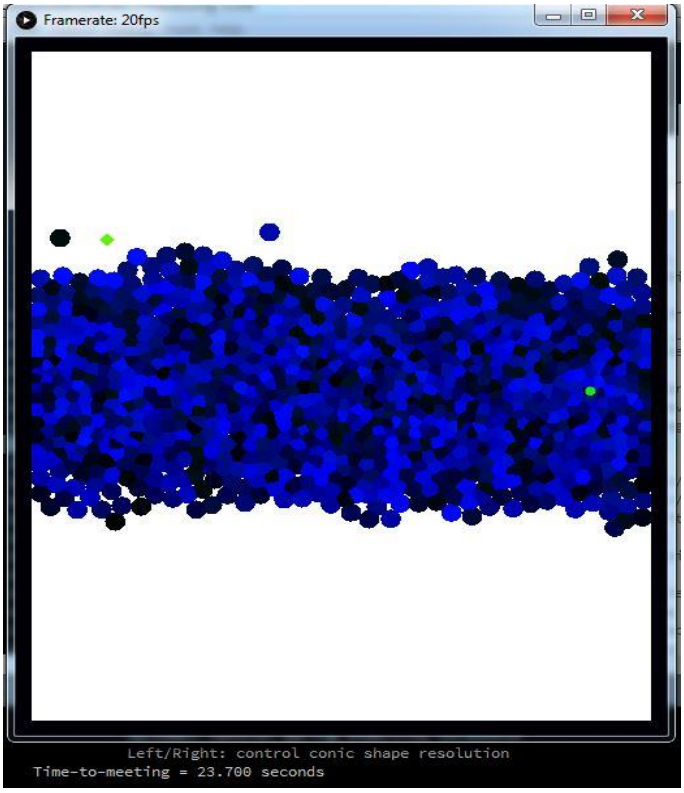


Simulation 3

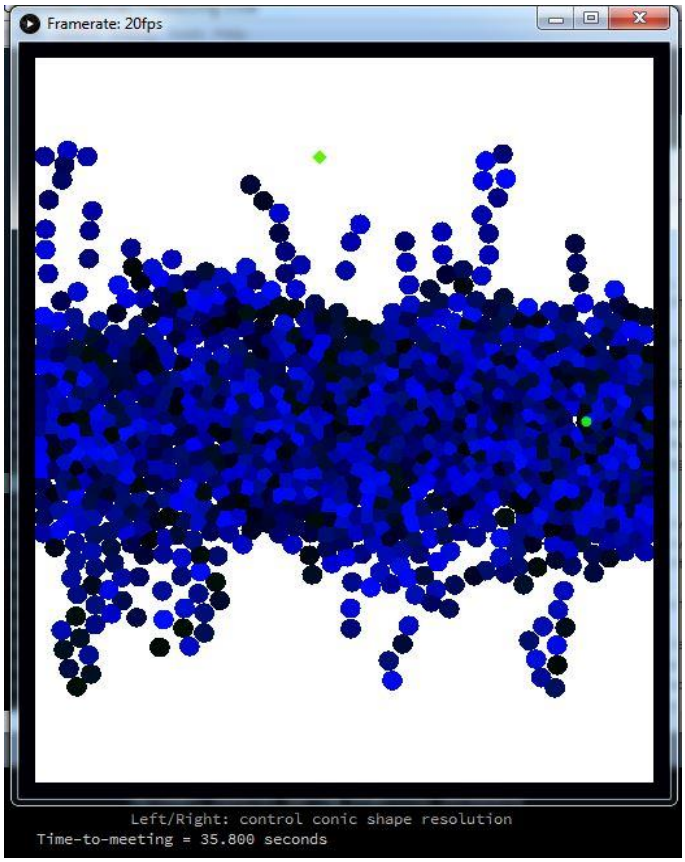
Time taken by 1 special entity to travel 400 units (in sec). Horizontal; No crowd;	
Simulation 1	47.200
Simulation 1	31.600
Simulation 1	37.800
Average	38.86

J. Single entity travelling in horizontal direction in the presence of crowd

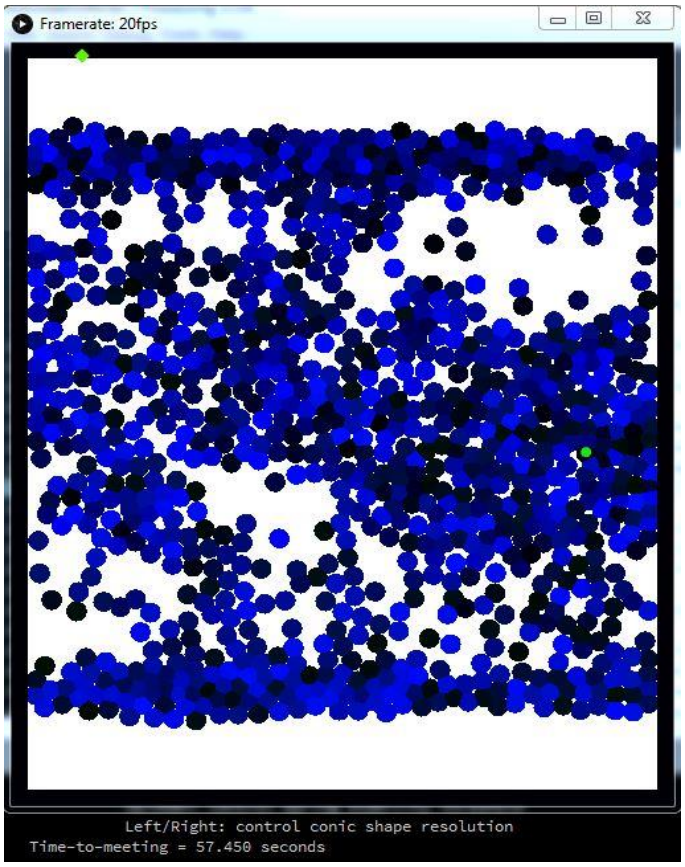
Simulation is performed for a single entity moving in the horizontal direction in the presence of 990 other entities acting as crowd and moving in bidirectional flow. Time taken by the entity to travel 400 units is measured in three simulations and the average time is computed.



Simulation 1



Simulation 2

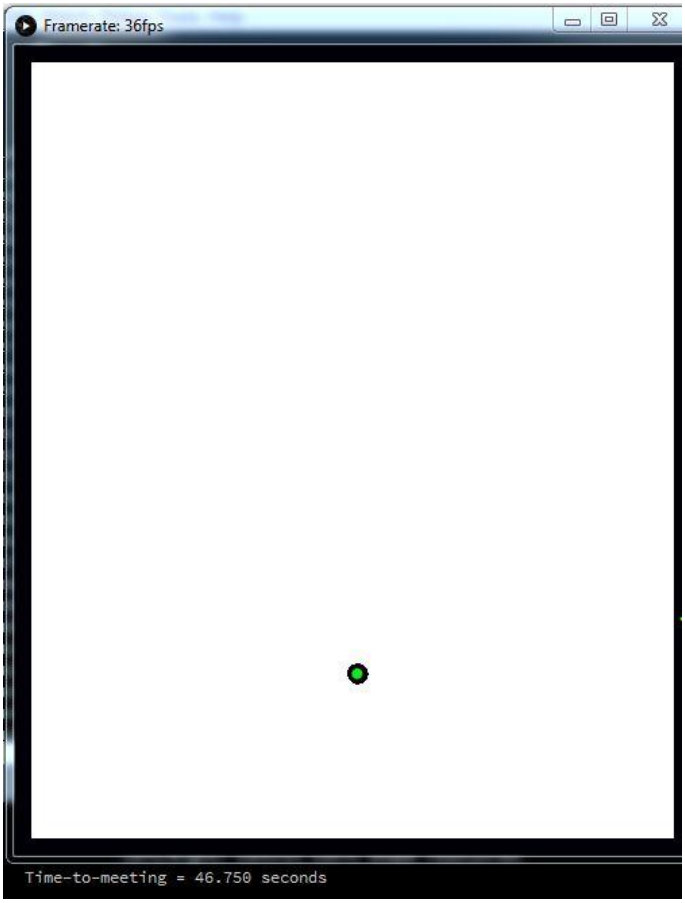


Simulation 3

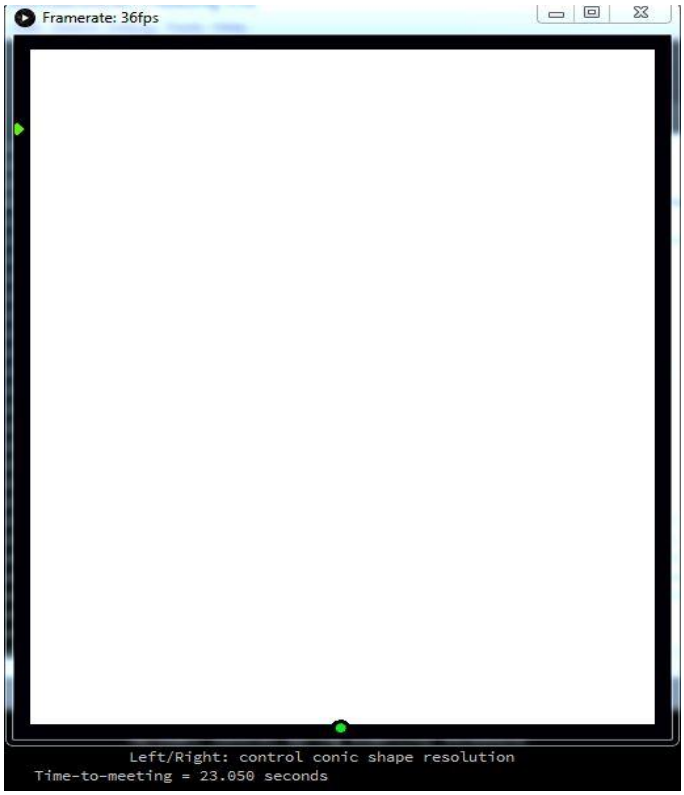
Time taken by 1 special entity to travel 400 units (in sec). Horizontal; With crowd;	
Simulation 1	23.700
Simulation 1	35.800
Simulation 1	57.450
Average	38.98

K. Single entity travelling in vertical direction with no crowd

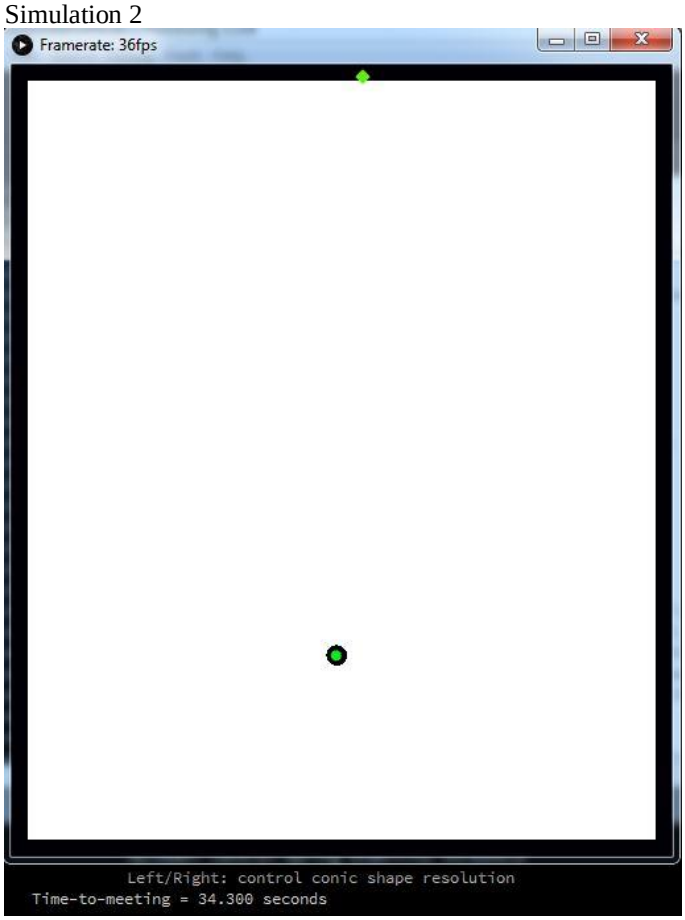
Simulation is performed for a single entity in the vertical direction in the absence of any crowd. Time taken by the entity to travel 400 units is measured in three simulations and the average time is computed.



Simulation 1



Simulation 2

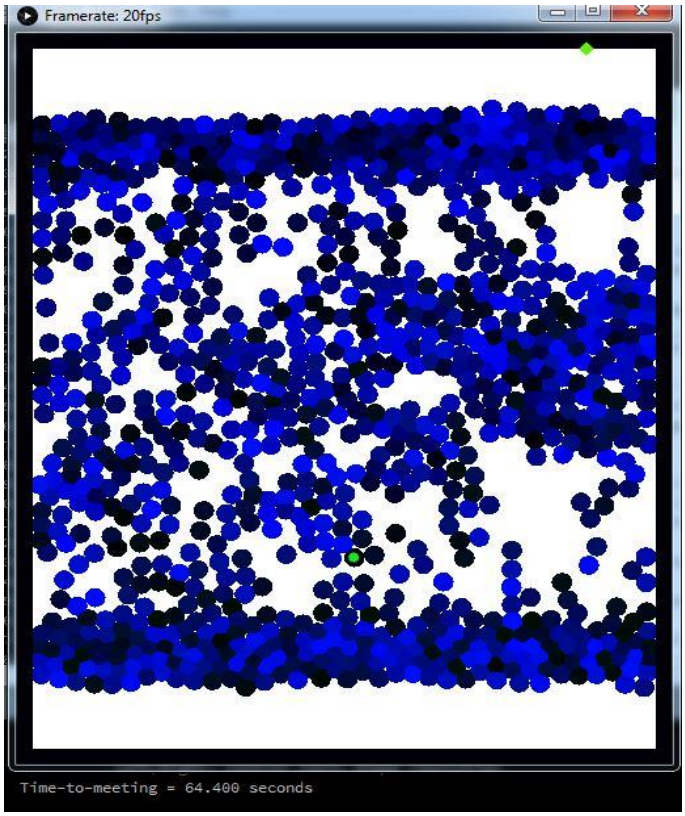


Simulation 3

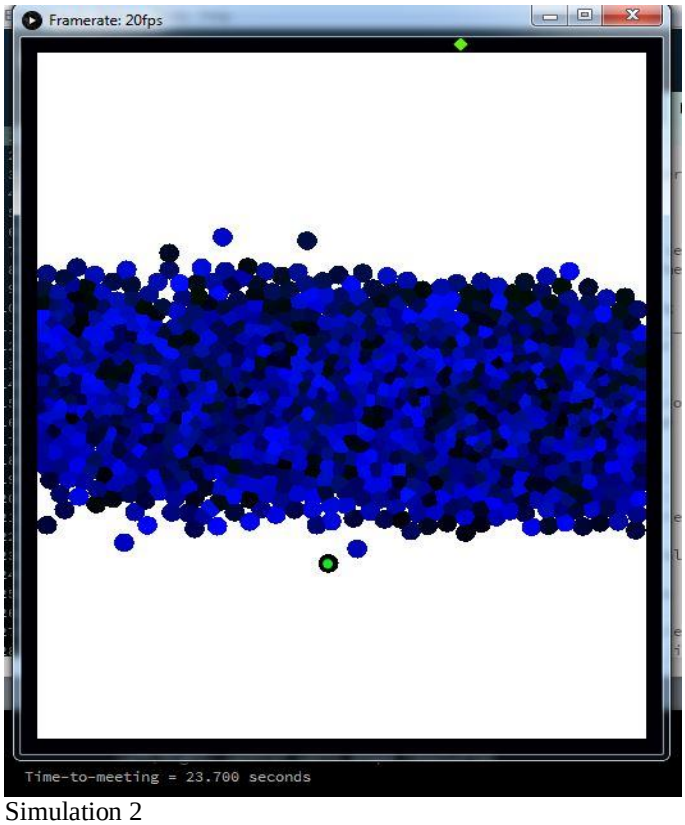
Time taken by 1 special entity to travel 400 units (in sec). Vertical; No crowd;	
Simulation 1	46.750
Simulation 1	23.050
Simulation 1	34.300
Average	34.7

L. Single entity travelling in vertical direction in the presence of crowd

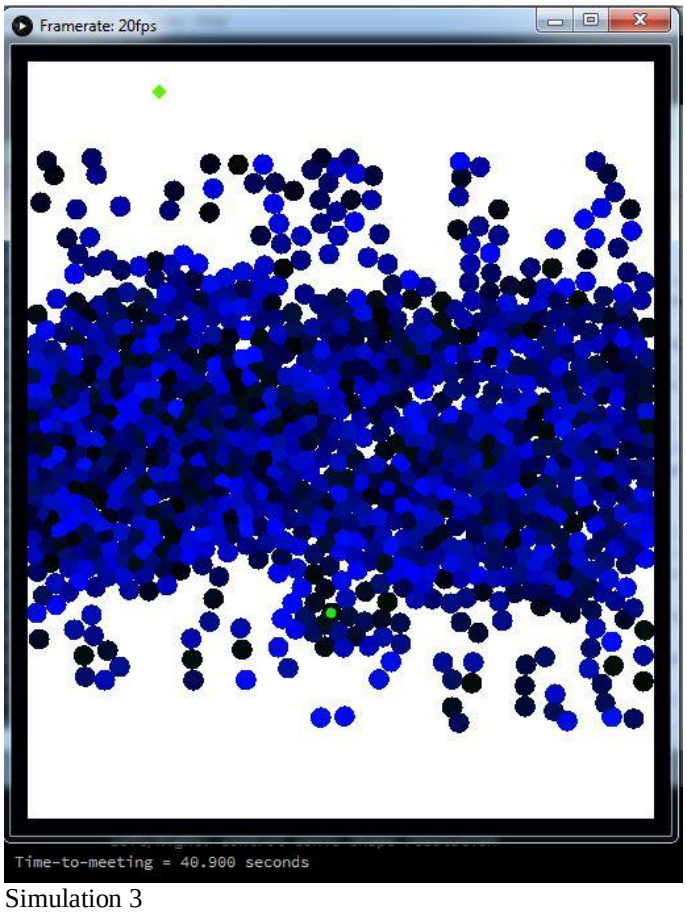
Simulation is performed for a single entity moving in the vertical direction in the presence of 990 other entities acting as crowd and moving in bidirectional flow. Time taken by the entity to travel 400 units is measured in three simulations and the average time is computed.



Simulation 1



Simulation 2



Simulation 3

Time taken by 1 special entities to travel 400 units (in sec). Vertical; With crowd;	
Simulation 1	64.400
Simulation 1	23.700
Simulation 1	40.900
Average	43.0

M. Comparison of all cases

The simulation results of all the cases are shown above. Here, we compare the simulation results of all twelve cases discussed. The table below compare the simulation results of all the cases.

Case No	Description	Average time (in sec)
A	Time taken by 10 special entities to travel 400 units (in sec). Horizontal: Flocking; No crowd;	59.616
B	Time taken by 10 special entities to travel 400 units (in sec). Horizontal; Flocking; No crowd; Separation and Cohesion forces off;	63.8
C	Time taken by 10 special entities to travel 400 units (in sec). Horizontal; Flocking; With crowd (990 other entities in bidirectional flow);	83.75
D	Time taken by 10 special entities to travel 400 units (in sec). Horizontal; Flocking; With crowd (990 other entities in bidirectional flow) Separation and Cohesion forces off;	55.56
E	Time taken by 10 special entities to travel 400 units (in sec). Vertical; Flocking; No crowd;	63.95
F	Time taken by 10 special entities to travel 400 units (in sec). Vertical; Flocking; No crowd; Separation and Cohesion forces off	57.53
G	Time taken by 10 special entities to travel 400 units (in sec). Vertical; Flocking; With crowd (990 other entities in bidirectional flow)	57.41
H	Time taken by 10 special entities to travel 400 units (in sec).	54.96

	Vertical; Flocking; With crowd (990 other entities in bidirectional flow); Separation and Cohesion forces off	
I	Time taken by 1 special entities to travel 400 units (in sec). Horizontal; No crowd;	38.86
J	Time taken by 1 special entities to travel 400 units (in sec). Horizontal; With crowd;	38.98
K	Time taken by 1 special entity to travel 400 units (in sec). Vertical; No crowd;	34.7
L	Time taken by 1 special entities to travel 400 units (in sec). Vertical; With crowd;	43.0

Table: Comparison of twelve models discussed

The comparison can be visualized effectively with the help of a graph. The graph below shows the comparison of all the cases discussed.

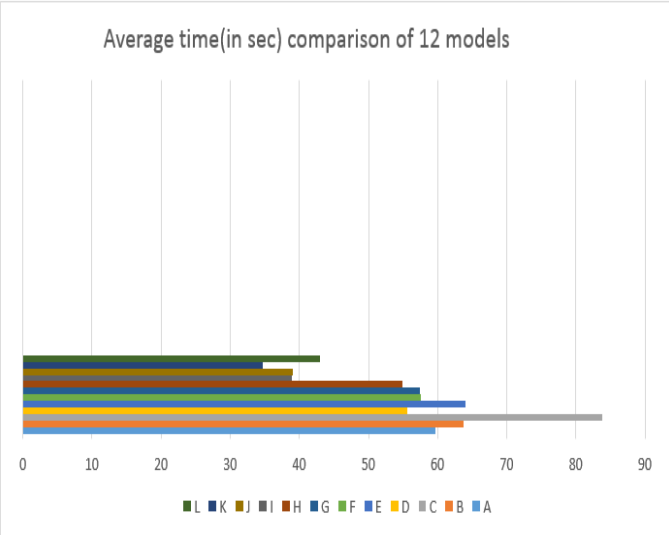


Fig 4. Graph showing the result compariosn of twelve models

N. Result comparison findings

The simulations performed above gives us the understanding of the flocking behavior of the crowds under various conditions. The findings regarding the time taken by the entities to travel a distance of400 units are discussed below:

- Time taken by the flock to travel horizontally under the influence of all three steering forces is slower than when the cohesion and separation forces are turned off
    - The result on the other hand is different when the same is performed for vertical travel. In this case, the time taken under the influence of all three steering forces is greater than when the cohesion and separation forces are turned off
- The time taken by the flock to travel horizontally in the presence of crowd is more than that of without crowd. This is understandable as the crowd is moving perpendicular to the motion of the flock
    - But this changes when the cohesion and separation forces are turned off. In this case, the flock takes less time when passing through the perpendicular crowd. This is an interesting result as we would normally expect the flock to take more time in the presence of crowd. One possible explanation is that in the absence of the two forces, the entities can travel pseudo individually and can make their own way through the crowd
- We can see in the vertical motion that the time taken by the flock is less in presence of crowd in both cases when all the three force are present as well as when the cohesion and separation forces are less. It can be understood possibly because the crowd initially gives the push to the flock while travelling in the vertical direction
- The time taken by the flock is less when the separation and cohesion forces are turned in the presence as well as in the absence of crowd while travelling in the vertical direction.
- It is interesting to note that in the respective four cases of crowd movement in the horizontal and vertical direction, the time taken by the flock is less while travelling in the vertical direction in three cases.
    - But, in flock movement in absence of crowd with all three steering forces, the time taken while travelling

in the vertical direction is more than that of time taken while travelling in horizontal direction. This result can be given more thought and can be looked into for further experimentation

- Time taken by a single entity, as expected, is less than that of flock movement in all the cases that can be compared.
- One interesting result is that the time taken by the single entity is almost same in the presence as well as the absence of crowd, while travelling in the horizontal direction. This result may change if more simulations are introduces
- In the vertical directing, the time taken in the absence of crowd is less than that of when the crowd is introduced
    - This is important to note as this is opposite to that of the case when the flock moves in the vertical direction. This again shows that the crowd push plays an important role while movement of flock in the vertical direction

V. Future scope of work

The flocking behavior has been studied for various cases only for on set of values for cohesion force, separation and alignment force. These values can be changed and the effect on the results can be observed. Various values can be experimented and a particular set of values on a range of values can be found that are best suited for modelling flocking behavior.

Also, the twelve simulation scenarios presented can be extended to get more data to better understand the flocking behavior under various conditions. Possible new scenarios includes various combinations with steering forces on or off. The flock can also be made to travel diagonally or in other directions to extract more data and to understand the flocking behavior of people in more depth.

VI. Conclusion

In this paper, flocking behavior of crowds is introduced following three steering forces namely cohesion force, separation force and alignment force. Various scenarios are modelled and simulated to understand how flock react under various conditions. The data is extracted and compared to beater understand the behavior. The scope of the work can be increased to better understand the behavior and to get in depth knowledge. Various possible improvements are discussed that can be used to extend the current model. The provided model along with new scenarios and functionality can be very useful in understanding the behavior of crowd and can help in better crowd controlling and avoiding any possible fatalities and loss.

References

[1] Hesham, Omar, and Gabriel Wainer. "Centroidal particles for interactive crowd Simulation." Proceedings of the Summer Computer Simulation Conference. Society for Computer Simulation International, 2016.

[2] Reynolds, C. W. (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.

[3] Chattaraj, U., Seyfried, A. & Chakroborty, P. Comparison of Pedestrian Fundamental Diagram across Cultures. *Adv. Complex Syst.* 12, 393–405 (2009).