Report Modelling And Simulation Complex Model 2021

Subject: spread of COVID-19 in a city and control policy

Author: Duc Dinh Anh
Email: ducda.m20ict@st.usth.edu.vn
Mobile: (+84) 982 086 651

July 2021

1 Spread of the disease

In this section, we want to discover the evolution of virus, beside we can track the steps of infectious process to see how and why virus can spread.

Model M1
Strategy

We build this model base on the SEIR strategy which following states:
- S: **Susceptible**, meaning the individual can be infected,
- E: **Exposed**, meaning that the individual has been infected, nut cannot be infected other individuals,
- I: **Infectious**, meaning that the individual has been infected and can infect other individuals,
- R: **Recovered**, meaning that the individual has recovered from disease and cannot be infected anymore.

- ***Note**: S individual will be infected by I individual. E will become I from three to seven day and I will become R from seven to ten day.

Requirement

- Implement population of species "Individuals" with an attribute for each epidemic state.
- Each individual are moving randomly by using wander skill which is built in method of GAMA.
- At each state, I individual infects one of S individual.
- Create mechanism that allow E individuals automatically change their states to I.(Following the duration)
- Create mechanism that allow I individuals automatically change their states to R.(Following the duration)
- Displaying each individual with a circle and a colour depends on their state.
- Creates 500 individuals and one infected individual in population, plot number of species in each states.

Result

```
global{
  int number_of_people <- 500;
  int number_of_infected_people <- 1;
  float dangerous_distance <- 0.5#m;
  int pandemic_duration <- 0;
  init{
    create individuals number: number_of_people;
    loop i from: 0 to: number_of_infected_people - 1{
      ask one_of(individuals){
        is_infected <- true;
        my_color <- #yellow;
        epidemic_state <- "E";
        count_date_expose <- 1;
      }
    }
  }
}
```

- In the global agent I create some attributes to calibrate the parameter of model and there are:
- number_of_people: allow me to add more individual.
- number_of_infected_people: allow me to add infected people.
- pandemic_duration: counting the date of pandemic duration.
- dangerous_distance: Is the perfect distance for virus go inside susceptible individual.

```
species individuals skills:[moving]{
  bool is_infected <- false;
  string epidemic_state <- "S";
  rgb my_color <- #blue;
  int count_date_expose <- 0;
  int count_date_infectious <- 0;
  etc.................
}
```

```
species individuals skills:[moving]{
  etc.................
  reflex move{
    do wander speed: 1.0;
  }
  //Expose to infectious
  reflex dynamicTurnBad when:(count_date_expose >= 72)
  and (count_date_expose <= 240)
  and (epidemic_state = "E"){
    write "Change bad state";
    epidemic_state <- "I";
    is_infected <- true;
    my_color <- #red;
    count_date_infectious <- count_date_expose + 1;
  }
  //Infectious to recovery
  reflex dynamicTurnGood
  when: (count_date_infectious >= 240)
  and (count_date_infectious <= 720)
  and (epidemic_state = "I"){
    write "Change good state";
    epidemic_state <- "R";
    is_infected <- false;
    my_color <- #green;
    count_date_expose <- 0;
    count_date_infectious <- 0;
  }
  reflex infect when:(epidemic_state = "I"){
    ask individuals at_distance 3.0 {
      if (self.epidemic_state = "S"){
        self.is_infected <- true;
        self.epidemic_state <- "E";
        self.my_color <- #yellow;
      }
    }
  }
  //Counter of expose duration
  // one day is 24 hour -> 3 days is 72 hour
  // one day is 24 hour -> 10 days is 240 hour
  // one day is 24 hour -> 30 days is 720 hour
  reflex increaseExposeDate when: epidemic_state = "E"{
    if(count_date_expose = 240){
      count_date_expose <- 0;
    }
    if(cycle mod 60 = 0 and cycle != 0){
      count_date_expose <- count_date_expose + 1;
    }
  }
  reflex increaseInfectiousDate when: epidemic_state = "I"{
    if(count_date_infectious = 720){
      count_date_infectious <- 0;
    }
    if(cycle mod 60 = 0 and cycle != 0){
      count_date_infectious <- count_date_infectious + 1;
    }
    pandemic_duration <- count_date_infectious;
  }
  etc.................
}
```

- In the individual agent I create some attributes to contain the value that can control the model follows my scenario:
- is_infected: separating people who is infected or not infected.
- epidemic_state: point out the state of each individual.
- my_color: color of each individual which can be decided by it's state.
- count_date_expose: counting the E duration of each individual.
- count_date_infectious: counting the I duration of each individual.
- This agent have five actions an each action have their own function base on the requirement. - "reflex infect": This action allows individual infects for susceptible individual. In this action, I will pick one Infectious individual and use this one to find the other individuals which has the same distance with the value of dangerous_distance variable.
- "reflex dynamicTurnBad" and "reflex increaseExposeDate": These two actions connect to each other, the second action counts the date when an individual stays in E state, and it will automatically change to I state when the condition of the first action satisfies, that means the state of individual will remain at E until it meets the end of the duration after that it will become I state.
- "reflex dynamicTurnGood" and "reflex increaseInfectiousDate": These two actions connect to each other, the second action counts the date when an individual stays in I state, and it will automatically change to R state when the condition of the first action satisfies, that means the state of an individual will remain at I until it meets the end of the duration after that it will become R state.

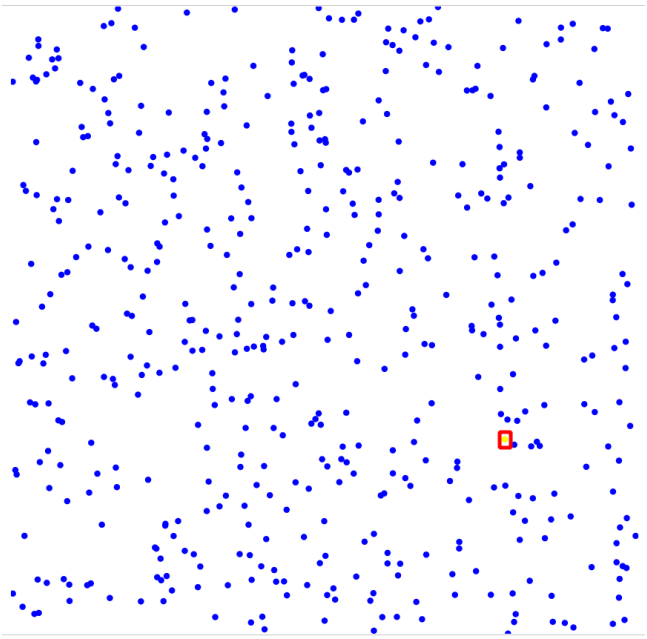

Figure 1: Simulation of model M1_1, following the requirement 1 infected individual and 500 susceptible individuals.

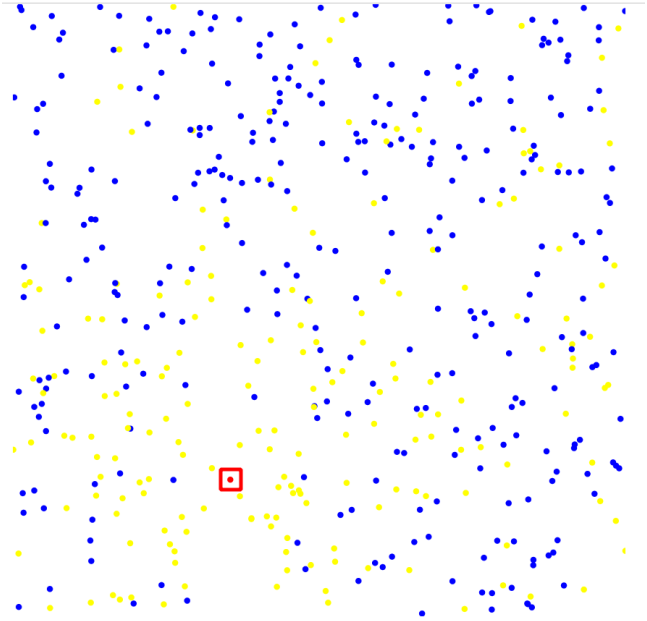

Figure 2: Simulation of model M1_1, after three days E individual turn to state I and start spreading desease to other individuals.

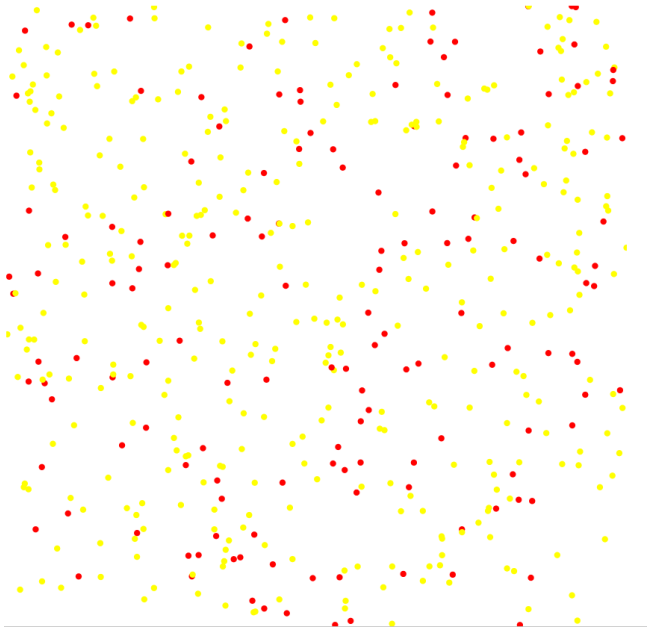

Figure 3: Simulation of model M1_1, all E individuals change to I state.

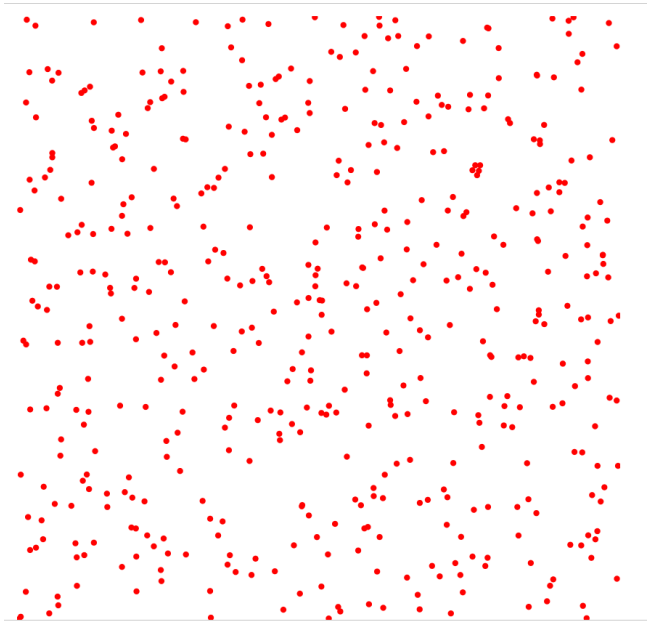

Figure 4: Simulation of model M1_1, all E individuals become infected individual.

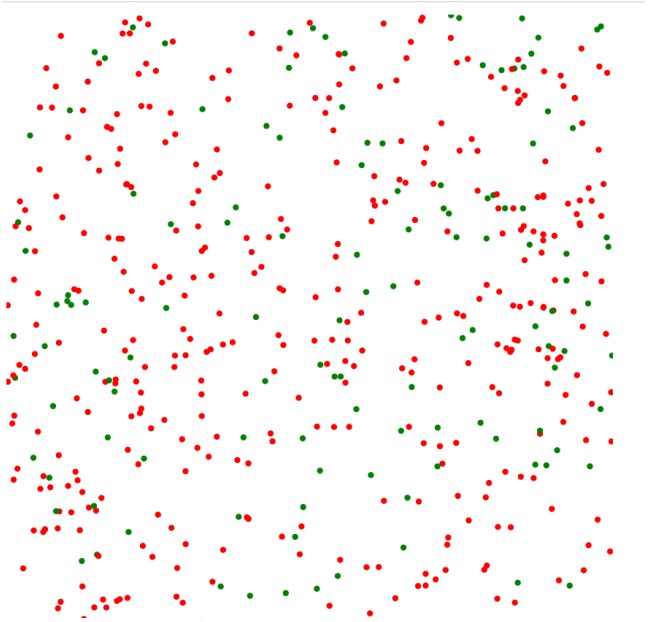

Figure 5: Simulation of model M1_1, all I individuals gradually chage to recovery individual.

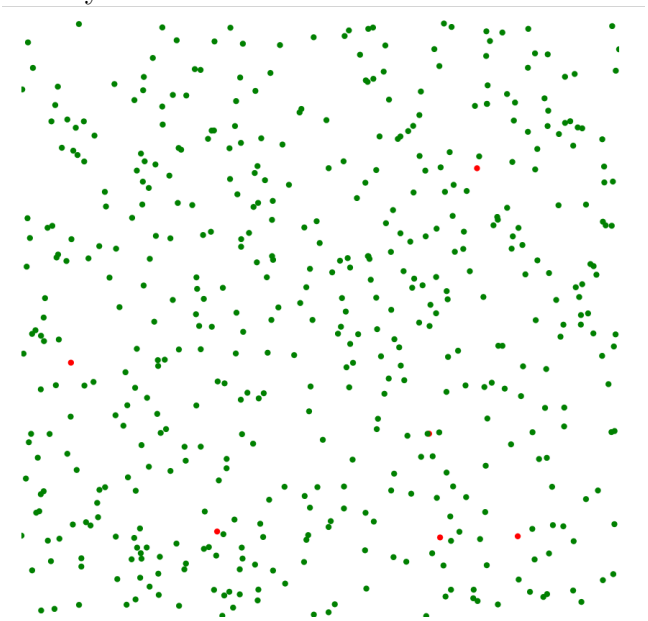

Figure 6: Simulation of model M1_1, number of individuals who have I state increase very fast.

Exploration

E1.1. Effect of randomness:
- Define a gui experiment, creating 10 experiments (with different seed values) and plot the evolution of the number of I individuals for each simulation.

```
experiment E1_1 type: gui{
  parameter "Number of people: " var: number_of_people
  min:10 max: 1000 step: 10;
  parameter "Number of infected people: "
  var: number_of_infected_people min: 1 max: 50 step: 10;
  init{
    create simulation with:[
    number_of_people::number_of_people,
    number_of_infected_people::number_of_infected_people
    ,seed::1];
    create simulation with:[
    number_of_people::number_of_people,
    number_of_infected_people::number_of_infected_people
    ,seed::2];
    etc.................
    create simulation with:[
    number_of_people::number_of_people,
    number_of_infected_people::number_of_infected_people
    ,seed::9];
    create simulation with:[
    number_of_people::number_of_people,
    number_of_infected_people::number_of_infected_people
    ,seed::10];
  }
  etc.................
}
```

```
experiment E1_1 type: gui{
  etc.................
  output{
    display E1_1{
      chart "Ch" type:series{
        datalist ["#S", "#E", "#I", "#R"]
        value: [individuals count (
        each.epidemic_state = "S"),
        individuals count (each.epidemic_state = "I"),
        individuals count (each.epidemic_state = "E"),
        individuals count (each.epidemic_state = "R")]
        color: [#blue, #yellow, #red, #green];
      }
    }
  }
}
```

- I design an experiment that can simulate ten simulations one time, and each simulation have different seed like the requirement. In this experiment I want to discover the evolution of each state base on the number of individual.

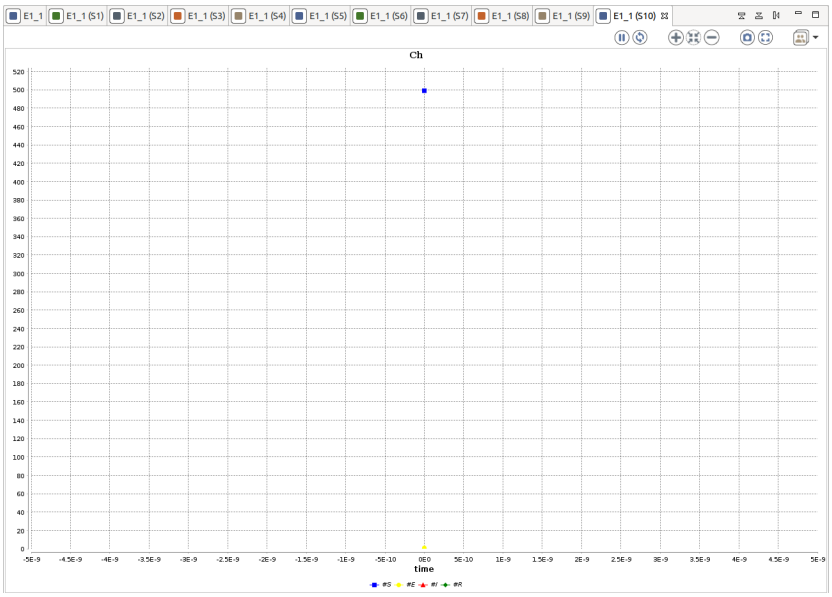

Figure 7: Exploration E1_1, ten experiments.

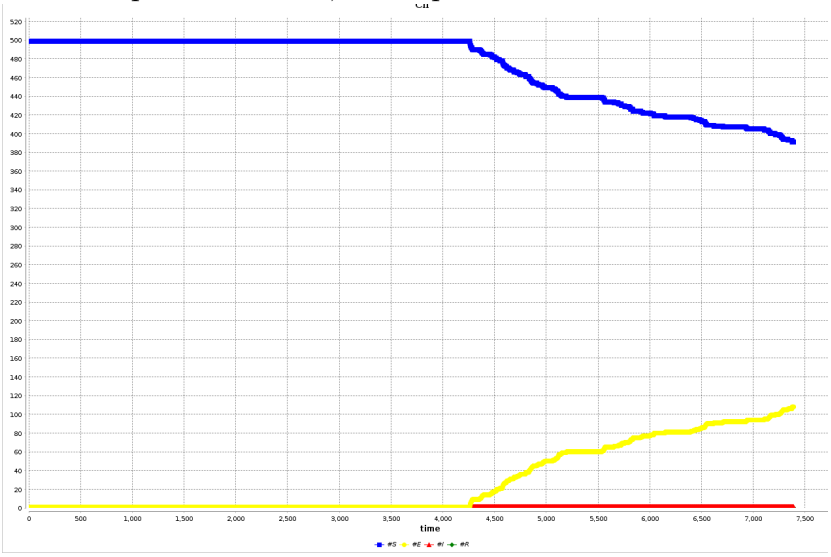

Figure 8: Exploration E1_1, E cases increase, S cases decrease.

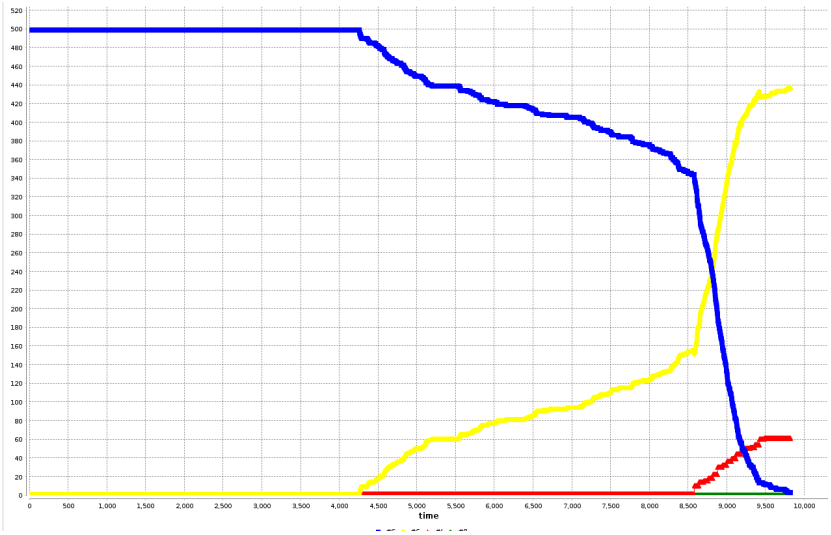

Figure 9: Exploration E1_1, I cases appear.

E1.2. Impact of the number of individuals.
Define a gui experiment, creating 11 experiments, with the same seed value but with a number of individuals from 200 to 2000 (with a step of 200) and plot the evolution of the number of I individuals.

```
experiment E1_2 type: gui{
  parameter "Number of people: "
  var: number_of_people min:200 max: 2000 step: 200;
  parameter "Number of infected people: "
  var: number_of_infected_people min: 1 max: 50 step: 10;

  init{
    create simulation with:[
    number_of_people::number_of_people,
    number_of_infected_people::number_of_infected_people,
    seed::seed];
    create simulation with:[
    number_of_people::number_of_people,
    number_of_infected_people::number_of_infected_people,
    seed::seed];
    create simulation with:[
    number_of_people::number_of_people,
    number_of_infected_people::number_of_infected_people,
    seed::seed];
```

```
      create simulation with:[
      number_of_people::number_of_people,
      number_of_infected_people::number_of_infected_people,
      seed::seed];
      create simulation with:[
      number_of_people::number_of_people,
      number_of_infected_people::number_of_infected_people,
      seed::seed];
      create simulation with:[
      number_of_people::number_of_people,
      number_of_infected_people::number_of_infected_people,
      seed::seed];
      create simulation with:[
      number_of_people::number_of_people,
      number_of_infected_people::number_of_infected_people,
      seed::seed];
      create simulation with:[
      number_of_people::number_of_people,
      number_of_infected_people::number_of_infected_people,
      seed::seed];
      create simulation with:[
      number_of_people::number_of_people,
      number_of_infected_people::number_of_infected_people,
      seed::seed];
      create simulation with:[
      number_of_people::number_of_people,
      number_of_infected_people::number_of_infected_people,
      seed::seed];
    }
    output{
      display E1_2{
        chart "Ch" type:series{
          datalist ["#S", "#E", "#I", "#R"]
          value: [
          individuals count (each.epidemic_state = "S"),
          individuals count (each.epidemic_state = "E"),
          individuals count (each.epidemic_state = "I"),
          individuals count (each.epidemic_state = "R")]
          color: [#blue, #yellow, #red, #green];
        }
      }

    }
}
```

- I create 11 experiments which use the same seed and each experiment shows the number of people in four states, throw these experiment I can track the increase of people in four states, follow the instruction of requirement I create 200 to 2000 individuals to simulate the increase of people in each state and the increase step of number of people is 200.

Figure 10: Exploration E1_2, eleven experiments use the same seed with the number of people from 200 to 2000.

E1.3. Impact of the number of individuals. Define 2 indicators: - the cycle of the maximum number of infected peoples. - The duration of the epidemic (when the number of Exposed and Infected is 0). Define a batch experiment plotting the evolution of these indicators in function of the number of peoples. Save these values in a csv file.

```
experiment E1_3 type:
batch until: (
individuals count(each.epidemic_state="I") = 0)
and (individuals count(each.epidemic_state="E") = 0){
  parameter "Number of people: "
  var: number_of_people min:200 max: 2000 step: 200;
  parameter "Number of infected people: "
  var: number_of_infected_people init:10 min: 10 max: 10;
  init {
    save ["Number of people","duration"]
    to: "data.csv" type: "csv" rewrite: true
    header: false;
  }
  reflex saving{
    ask simulations{
    save [self.number_of_people, self.pandemic_duration]
    to: "data.csv" type: "csv" rewrite: false;
    }
  }
}
```

2 Spread in a city with a heterogeneous population

- In this model, we will simulate the process of spreading desease in real environment. Therefore, we need to create the diversity of individual's attribute which bases on the attribute of human for example age, gender, actvity etc.

Model M2

Requirement

- Import shape file or randomly generate shape files (use open street map). - Building should have a type among: Home, office, school, shops, coffee, restaurant, park. - Can generate building randomly in GAMA at inisialisation. - Display the buildings with their own colour which bases on their type.

Result

```
global{
  shape_file buildings0_shape_file <-
  shape_file("../includes/buildings.shp");
  shape_file clean_roads0_shape_file <-
  shape_file("../includes/clean_roads.shp");
  geometry shape <- envelope(clean_roads0_shape_file);
  graph road_graph;
  int number_of_inhabitant <- 150;
  float step <- 1#mn;
  date starting_date <- date([2021,1,2,0,0,0]);
  int pandemic_duration <- 0;
  int number_people_infected <- 30;
  init{
    create inhabitants number: number_of_inhabitant{
      //Get random age
      rdAge <- rnd(0,2);
      my_color <- ageColorList[rdAge];
      if(my_color = #aqua){
        is_adult <- true;
      }
      else if(my_color = #lawngreen){
        is_kid <- true;
      }
      else if(my_color = #mediumorchid){
        is_old <- true;
      }
      if(is_adult = true){
        my_age <- rnd(22, 55);
      }
      else if(is_kid = true){
        my_age <- rnd(1, 21);
      }
      else if(is_old = true){
        my_age <- rnd(56, 100);
      }
```

```
    //Random gender
    rdGender <- rnd(0,1);
    my_gender <- genderChoiceList[rdGender];
    if(my_gender = "male"){
      is_male <- true;
    }
    else if(my_gender = "female"){
      is_female <- true;
    }
  }
  // create infected people
  loop i from: 0 to: number_people_infected - 1{
    ask one_of(inhabitants){
      write "i:" + i;
      self.is_susceptible_state <- false;
      self.is_exposed_state <- true;
      self.my_color <- #gold;
      self.epidemic_state <- "E";
    }
  }
  create buildings from: buildings0_shape_file{
    var <-
    [#darkgrey,#yellow,#green,#brown,#orange,#blue,#purple];
    rdIndex <- rnd(0,6);
    my_color <- var[rdIndex];
    if(my_color = #darkgrey){
      is_building <- true;
      if(have_people = false){
        int number_of_child <- rnd(0,3);
        int number_of_male_adult <- 1;
        int number_of_female_adult <- 1;
        int number_of_grand_father <- 1;
        int number_og_grand_mother <- 1;
        loop index from: 0 to: number_of_male_adult{
          ask one_of(inhabitants where(each.is_adult = true
          and each.my_gender="male")){
            if(self.location != myself.location
            and self.have_home = false){
              self.location <- any_location_in(myself);
              self.have_home <- true;
              self.houseLocation <- location;
              break;
            }
          }
        }
```

```
        loop index from: 0 to: number_of_female_adult{
          ask one_of(inhabitants where(each.is_adult = true
          and each.my_gender="female")){
            if(self.location != myself.location
            and self.have_home = false){
              self.location <- any_location_in(myself);
              self.have_home <- true;
              self.houseLocation <- location;
              break;
            }
          }
        }
        loop index from: 0 to: number_of_grand_father{
          ask one_of(inhabitants where(each.is_old = true
          and each.my_gender="male")){
            if(self.location != myself.location
            and self.have_home = false){
              self.location <- any_location_in(myself);
              self.have_home <- true;
              self.houseLocation <- location;
              break;
            }
          }
        }
        loop index from: 0 to: number_og_grand_mother{
          ask one_of(inhabitants where(each.is_old = true
          and each.my_gender="female")){
            if(self.location != myself.location
            and self.have_home = false){
              self.location <- any_location_in(myself);
              self.have_home <- true;
              self.houseLocation <- location;
              break;
            }
          }
        }
        loop index from: 0 to: number_of_child{
          ask one_of(inhabitants where(each.is_kid = true )){
            if(self.location != myself.location
            and self.have_home = false){
              self.location <- any_location_in(myself);
              self.have_home <- true;
              self.houseLocation <- location;
              break;
            }
          }
        }
        nb_people <- number_of_child + number_of_male_adult + number_of_female_
        number_of_grand_father + number_og_grand_mother;
      }
    }
```

```
      else if(my_color = #yellow){
        is_workspace <- true;
      }
      else if(my_color = #green){
        is_park <- true;
      }
      else if(my_color = #brown){
        is_coffeeshop <- true;
      }
      else if(my_color = #orange){
        is_restaurant <- true;
      }
      else if(my_color = #blue){
        is_school <- true;
      }
      else if(my_color = #purple){
        is_gorcery <- true;
      }
    }
    create roads from: clean_roads0_shape_file;
    do init_other_location();
    road_graph <- as_edge_graph(roads);
  }
  action init_other_location{
    buildings workspace;
    buildings park;
    buildings coffeeshop;
    buildings restaurant;
    buildings school;
    buildings gorcery;
    loop i from: 0 to: number_of_inhabitant {
    ask one_of(inhabitants){
```

```
            workspace <- one_of(buildings
            where(each.is_workspace = true));
            park <- one_of(buildings
            where(each.is_park = true));
            coffeeshop <- one_of(buildings
            where(each.is_coffeeshop = true));
            restaurant <- one_of(buildings
            where(each.is_restaurant = true));
            school <- one_of(buildings
            where(each.is_school = true));
            gorcery <- one_of(buildings
            where(each.is_gorcery = true));

            workLocation <- any_location_in(workspace);
            parkLocation <- any_location_in(park);
            coffeeLocation <- any_location_in(coffeeshop);
            restaurantLocation <- any_location_in(restaurant);
            schoolLocation <- any_location_in(school);
            gorceryLocation <- any_location_in(gorcery);
          }
        }
    }
    reflex dd{
      write current_date;
      write current_date.hour;
      write "---------------";
    }
}
```

- "create inhabitants number_of_inhabitant", "create buildings from: buildings0_shape_file", "create roads from: clean_roads0_shape_file;": in global initialization I create three species which are inhabitants or individuals, buildings, roads. In inhabitans I add some attributes like age, gender. To create infected people I use a loop. Beside, I create the colour for each individual base on their age. the shape files are imported to both species, with road I use default information in the shape file but I need to modify and add some attributes of builfdings. After creating all inhabitans, I put inhabitant in the building and each building have four to seven people, buildings also have colour base on their type.

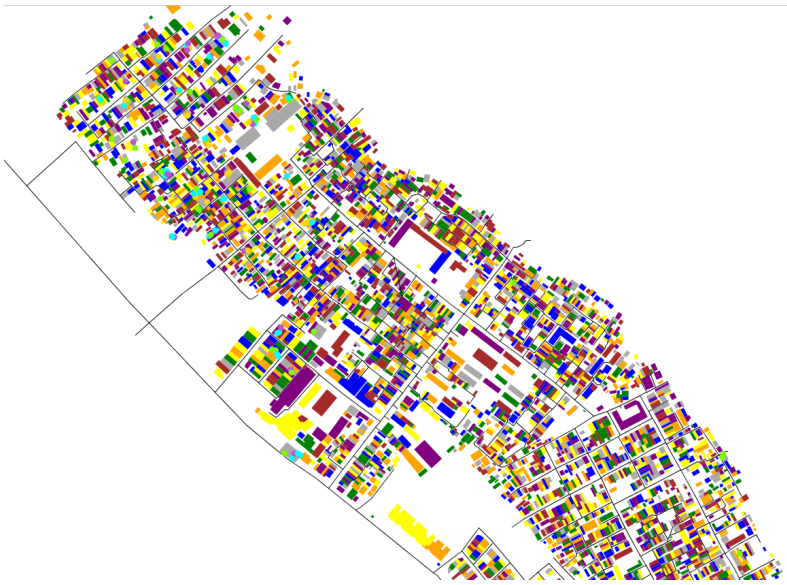

Figure 11: Exploration M2_1, the city after importing the shape files, each building have their own colour base on their type, inhabitants are small circle with plenty of colout to distinguish the age of each inhabitant.

```
species inhabitants skills:[moving]{
    list ageColorList <- [#aqua,#lawngreen,#mediumorchid];
    list genderChoiceList <- ["male","female"];
  rgb my_color <- #blue;
  int rdAge;
  bool is_adult;
  bool is_kid;
  bool is_old;
  int my_age;
  int rdGender;
  string my_gender;
  bool is_male;
  bool is_female;
  bool have_home <- false;
  bool is_susceptible_state <- true;
  bool is_infected_state <- false;
  bool is_exposed_state <- false;
  bool is_recovery_state <- false;
  string epidemic_state;
  point houseLocation;
  point workLocation;
  point coffeeLocation;
```

```
point schoolLocation;
point parkLocation;
point restaurantLocation;
point gorceryLocation;
point target <- nil;
int count_date_expose <- 0;
int count_date_infectious <- 0;
// Adult schedule
reflex goToCoffeShop when: (is_adult = true
and current_date.hour = 6){
  target <- any_location_in(coffeeLocation);
  do goto target: target on: road_graph;
}
reflex goToWork when: (is_adult = true
and current_date.hour = 7){
  target <- any_location_in(workLocation);
  do goto target: target on: road_graph;
}
reflex adultGoShopping when: (is_adult = true
and current_date.hour = 17){
  target <- gorceryLocation;
  do goto target: target on: road_graph;
}
reflex goRestaurant when: (is_adult = true
and current_date.hour = 19){
  target <- restaurantLocation;
  do goto target: target on: road_graph;
}
//Kid schedule
reflex goToSchool when: (is_kid = true
and current_date.hour = 7){
  target <- schoolLocation;
  do goto target: target on: road_graph;
  if(location = target){
    target <- houseLocation;
  }
}
reflex kidGoToPark when: (is_kid = true
and current_date.hour = 16){
  target <- parkLocation;
  do goto target: target on: road_graph;
  if(location = target){
    target <- houseLocation;
  }
}
```

```
//Retire people/ old people
reflex oldPlpGoToPark when: (is_old = true
and (current_date.hour = 5 or current_date.hour = 16)){
  target <- parkLocation;
  do goto target: target on: road_graph;
}
reflex oldPlpGoShopping when: (is_old = true
and current_date.hour = 7){
  target <- gorceryLocation;
  do goto target: target on: road_graph;
}
reflex oldPlpGoHone when: (is_old = true
and current_date.hour = 10){
  target <- houseLocation;
  do goto target: target on: road_graph;
}
//Infected process
// one day is 24 hour -> 3 days is 72 hour
// one day is 24 hour -> 10 days is 240 hour
// one day is 24 hour -> 30 days is 720 hour
reflex dynamicTurnBad when:
(count_date_expose >= 72)
and (count_date_expose <= 240)
and (epidemic_state = "E")
and (is_exposed_state = true){
  write "Change bad state";
  epidemic_state <- "I";
  is_infected_state <- true;
  is_exposed_state <- false;
  is_susceptible_state <- false;
  my_color <- #red;
  count_date_infectious <- count_date_expose + 1;
}
reflex dynamicTurnGood when:
(count_date_infectious >= 240)
and (count_date_infectious <= 720)
and (epidemic_state = "I")
and (is_infected_state = true){
  write "Change good state";
  epidemic_state <- "R";
  is_infected_state <- false;
  is_recovery_state <- true;
  my_color <- #lightblue;
  count_date_expose <- 0;
  count_date_infectious <- 0;
}
```

```
  reflex infect when:(epidemic_state = "I"){
    buildings my_location <-
    first(buildings overlapping(self));
    list<inhabitants> list_colleague <-
    (inhabitants overlapping(my_location))
    where(each.is_susceptible_state = true);
    loop person over: list_colleague{
      ask person{
        person.is_susceptible_state <- false;
        person.is_exposed_state <- true;
        epidemic_state <- "E";
        person.my_color <- #gold;
      }
    }
  }
  reflex increaseExposeDate when:
  epidemic_state = "E"{
    if(count_date_expose = 240){
      count_date_expose <- 0;
    }
    if(cycle mod 60 = 0 and cycle != 0){
      count_date_expose <- count_date_expose + 1;
    }
  }
  reflex increaseInfectiousDate when:
  epidemic_state = "I"{
    if(count_date_infectious = 720){
      count_date_infectious <- 0;
    }
    if(cycle mod 60 = 0 and cycle != 0){
      count_date_infectious <- count_date_infectious + 1;
    }
    pandemic_duration <- count_date_infectious;
  }
  // End of the day
  reflex goHone when:
  (target != nil and current_date.hour = 18)
  or (is_adult = true and current_date.hour = 21){
    target <- houseLocation;
    do goto target: target on: road_graph;
    if(location = target){
      target <- nil;
    }
  }
  aspect goem{
    draw circle(4) color: my_color;
  }
}
```

- Each inhabitants have their own calendar so they can move every location that is chose in initalization section. I still use four action to describe the process of desease and they are "dynamicTurnBad", "dynamicTurnGood", "increaseExposeDate", "increaseInfectiousDate"; The different here is the action infect rather than infecting outside, in this model the virus will be transmitted inside a building which they live. Therefore I need to modify this action, first my model will find all the inhabitans who is susceptible inhabitant by using overlapping built-in function, second the process of transmitting desease start, all the inhabitants in the buiding which infected inhabitants stay will be infected.

Figure 12: Exploration M2_1 + M2_2, some individuals with purple colour are moving outside.

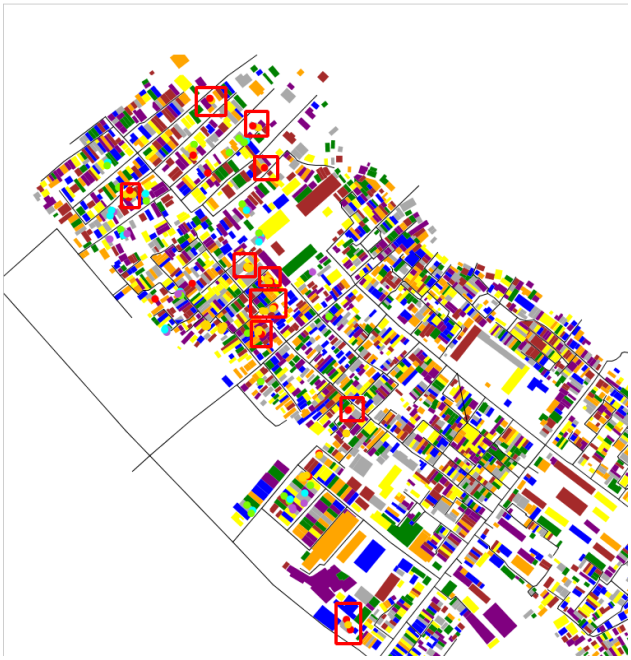

Figure 13: Exploration M2_1 + M2_2, all inhabitants stay with the infected inhabitants become exposed inhabitants and they will become infected inhabitant after they reach the E duration(From 3 to 10 days).

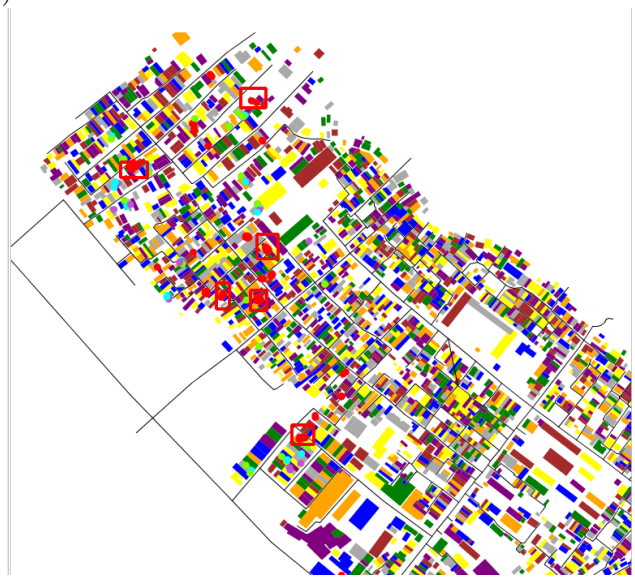

Figure 14: Exploration M2_1 + M2_2, all exposed inhabitants become infected inhabitants.

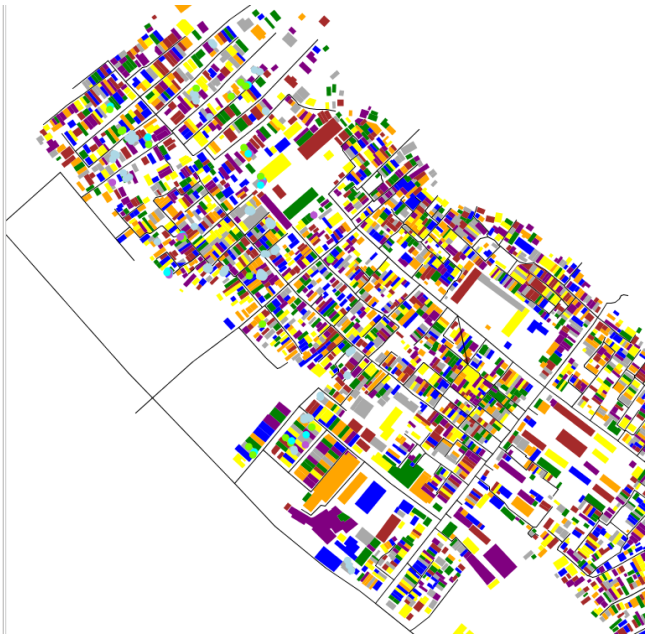

Figure 15: Exploration M2_1 + M2_2, all infected inhabitants become recovery inhabitants after they reach the I duration(From 10 to 30 days).

E2.1. Effect of randomness:

- Following the requirement, I need to create an experiment which is as same as E1.1. In this experiment, I can track the increase of people in each state and also the speed of the process of spreading dease in real environment.

```
experiment E2_1 type: gui{
  parameter "Number of people: "
  var: number_of_inhabitant min:10 max: 1000 step: 10;
  parameter "Number of infected people: "
  var: number_people_infected min: 5 max: 50 step: 10;

  init{
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::1];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::2];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::3];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::4];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::5];
```

```
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::6];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::7];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::8];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::9];
    create simulation with:[
    number_of_inhabitant::number_of_inhabitant,
    number_people_infected::number_people_infected,
    seed::10];
  }
  output{
    display E1_1{
      chart "Ch" type:series{
        datalist ["#S", "#E", "#I", "#R"]
        value: [inhabitants count (
        each.epidemic_state = "S"),
        inhabitants count (each.epidemic_state = "E"),
        inhabitants count (each.epidemic_state = "I"),
        inhabitants count (each.epidemic_state = "R")]
        color: [#blue, #gold, #red, #lightblue];
      }
    }

  }
}
```

Model M2_3

Requirement

- Building contains a household. (Actually I do this model first and I reuse it to the M2_1 and M2_2 so you can see the initialization for household in both model)
- Randomly choosing the role of each people in the household for example: father, mother, kid, grand mother, grand father.
- Creating an agenda for each inhabitant who is living in the same house. (Follow the calendar of requirement).

Result

```
species inhabitants skills:[moving]{
  reflex infect when:(epidemic_state = "I"){
    ask inhabitants at_distance 3.0 {
      if (self.epidemic_state = "S"){
        self.is_susceptible_state <- false;
        self.is_exposed_state <- true;
        self.epidemic_state <- "E";
        self.my_color <- #gold;
      }
    }
  }
  experiment M2_3{
  output{
    display map{
      species buildings aspect:goem;
      species roads aspect: goem;
      species inhabitants aspect: goem;
    }
    monitor "nb susceptible people"
    value: inhabitants count(
    each.is_susceptible_state = true);
    monitor "nb exposed people"
    value: inhabitants count(
    each.is_exposed_state = true);
    monitor "nb infected people"
    value: inhabitants count(
    each.is_infected_state = true);
    }
  }
}
```

- I still use the source code of model M2_1 and M2_2 and the diffent is I add the new action which is infect action which allow the I inhabitans infect the S inhabitants when they have the same range of distance.

Exploration

E2.2. Population characteristics.
- Define a gui experiment, plotting (with histograms for example) the ratio male-female, the population distribution in terms of age and in terms on number of people.

```
experiment E2_2{
  output{
    display Population_gender {
      chart "Population gender" type: histogram {
        datalist ["Male", "Female"]
        value:[length(inhabitants where(each.my_gender="male")),
        length(inhabitants where(each.my_gender="female"))];
      }
    }
    display Population_by_age {
      chart "Population by age" type: histogram {
        datalist ["Childen", "Adults", "Old people"]
        value:[
           length(inhabitants where(3 <= each.my_age
           and each.my_age <= 21)),
           length(inhabitants where(22 <= each.my_age
           and each.my_age <= 55)),
           length(inhabitants where(55 <= each.my_age
           and each.my_age <= 100))
          ];
      }
    }
    monitor "nb susceptible people"
    value: inhabitants
    count(each.is_susceptible_state = true);
    monitor "nb exposed people"
    value: inhabitants
    count(each.is_exposed_state = true);
    monitor "nb infected people"
    value: inhabitants
    count(each.is_infected_state = true);
  }
}
```

- In this exploration, the system will create gender histogram and also the age histogram. The gender histogram allows me to see how many female and male inhabitant in our model and the age histogram shows the number of people in each group of age and there are adult, old, kid.

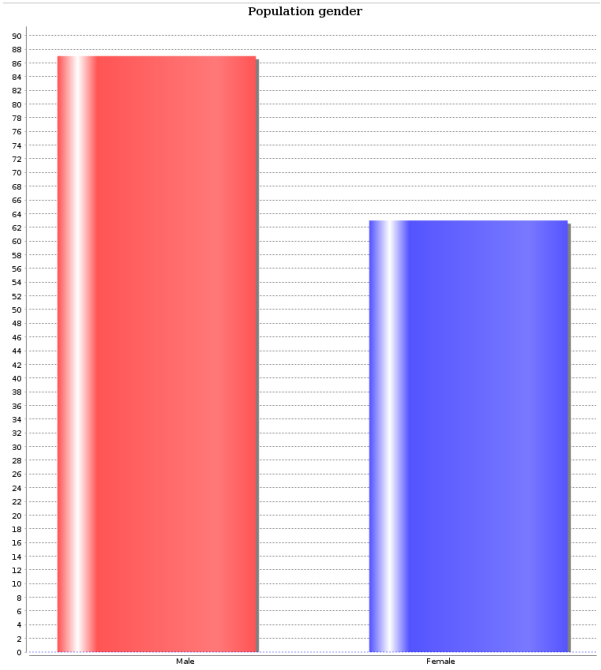

Figure 16: Exploration E2_2, graph of gender.

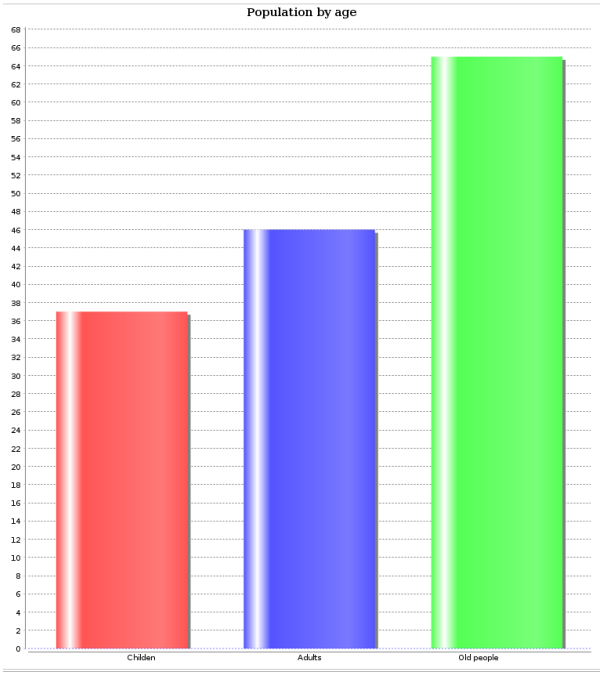

Figure 17: Exploration E2_2, graph of age.

E2.3. Epidemic plotting. Define an experiment to plot epidemic spread.

```
experiment E2_3{
  output{
    display Epidemic_plotting {
      chart "States of the agents"
      type: series style: line {
        datalist ["#S", "#E", "#I", "#R"]
        value: [inhabitants
        count (each.is_susceptible_state = true),
        inhabitants
        count (each.is_exposed_state = true),
        inhabitants
        count (each.is_infected_state = true),
        inhabitants
        count (each.is_recovery_state = true)]
        color: [#blue, #gold, #red, #lightblue];
      }
    }
    monitor "nb susceptible people"
    value: inhabitants
    count(each.is_susceptible_state = true);
    monitor "nb exposed people"
    value: inhabitants
    count(each.is_exposed_state = true);
    monitor "nb infected people"
    value: inhabitants
    count(each.is_infected_state = true);
  }
}
```

- To discover the epidemic I create an experiment which can count all the case in each state.

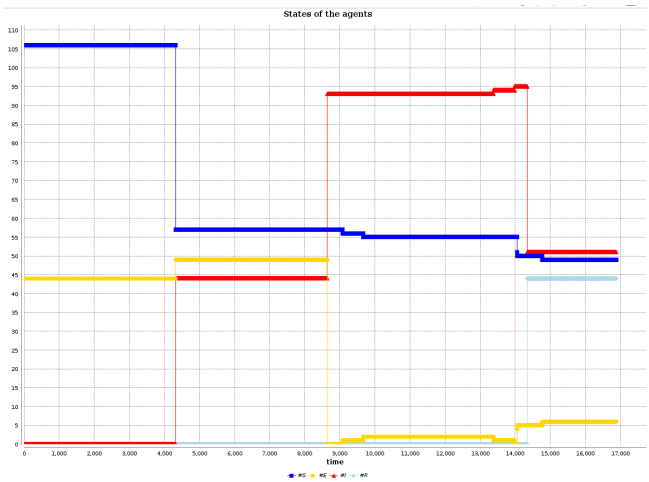

Figure 18: Exploration E2_3, number of peoples in each state.

Model M2.4

Requirement

- Model can simulate the life circle of virus in the environment.
- People can be infected even they don't contact with infected people.
- Modify building which can load virus. The number of virus increase when infected people are in building and decrease over time.

Result

- In this model, I add one species and it is virus. Virus can life in the environment few days and it can infect people. rather than move function I use wander to describe the complex when viruses move around. To count the day of virus's life circle I create an action which count the day in viruses life circle. I also modify buildings species to make sure that can become the virus load.

```
species virus skills:[moving]{
  rgb my_color <- #red;
  bool is_live <- true;
  int count_life_time <- 0;
  reflex move{
    do wander speed: 0.005;
  }
  reflex infect when: is_live = true{
    ask inhabitants at_distance 3.0 {
      if (self.epidemic_state = "S"){
        self.is_susceptible_state <- true;
        self.is_exposed_state <- true;
        self.epidemic_state <- "E";
        self.my_color <- #gold;
      }
    }
  }
  // virus can life 3 days on the environment
  reflex death when: count_life_time = 72{
    is_live <- false;
    my_color <- #black;
    do die;
  }
  reflex count_life_time when: is_live = true{
    if(count_life_time = 72){
      count_life_time <- 0;
    }
    if(cycle mod 60 = 0){
      count_life_time <- count_life_time + 1;
    }
  }
  aspect goem{
    draw triangle(4) color: my_color;
  }
}
species buildings{
  list var;
  int rdIndex;
  rgb my_color;
  bool is_building <- false;
  bool is_workspace <- false;
  bool is_park <- false;
  bool is_coffeeshop <- false;
```

```
bool is_restaurant <- false;
bool is_school <- false;
bool is_gorcery <- false;
bool have_people <- flip(0.5);
int nb_people <- 0;
int nb_virus <- 0;
rgb old_color;
bool has_virus <- false;
//bool test <- true;
int count_life_time <- 0;
reflex virus_load{
  list<inhabitants> list_of_people
  <- (inhabitants overlapping(self));
  loop person over: list_of_people{
    if(person.is_infected_state = true){
      //myself.old_color <- myself.my_color;
      //write "Detect infected people";
      has_virus <- true;
      nb_virus <- nb_virus + 1;
      //test <- false;
    }
  }
}

reflex infect when: has_virus = true{
  list<inhabitants> list_of_people
  <- (inhabitants overlapping(self));
  loop person over: list_of_people{
    if(person.is_susceptible_state = true){
      person.is_susceptible_state <- false;
      person.is_exposed_state <- true;
      person.epidemic_state <- "E";
      person.my_color <- #gold;
    }
  }
}

reflex decrease_virus when: has_virus = true
and count_life_time = 24{
  nb_virus <- max(0, nb_virus - 1);
}
reflex death when: count_life_time = 48{
  nb_virus <- 0;
}
```

```
  reflex count_life_time when: has_virus = true{
    if(count_life_time = 48){
      count_life_time <- 0;
    }
    if(cycle mod 60 = 0){
      count_life_time <- count_life_time + 1;
    }
  }
  reflex color_building {
    if (has_virus) {
      my_color <- #black;
    }
    if (self.nb_virus = 0) {
      has_virus <- false;

      if (is_building = true) {
        my_color <- #darkgrey;
      }
      if (is_workspace = true) {
        my_color <- #yellow;
      }
      if (is_park = true) {
        my_color <- #green;
      }
      if (is_coffeeshop = true) {
        my_color <- #brown;
      }
      if (is_restaurant = true) {
        my_color <- #orange;
      }
      if (is_school = true) {
        my_color <- #blue;
      }
      if (is_gorcery = true) {
        my_color <- #purple;
      }
    }
  }
  //rgb my_color;
  aspect goem{
    draw shape color: my_color;
  }
}
```

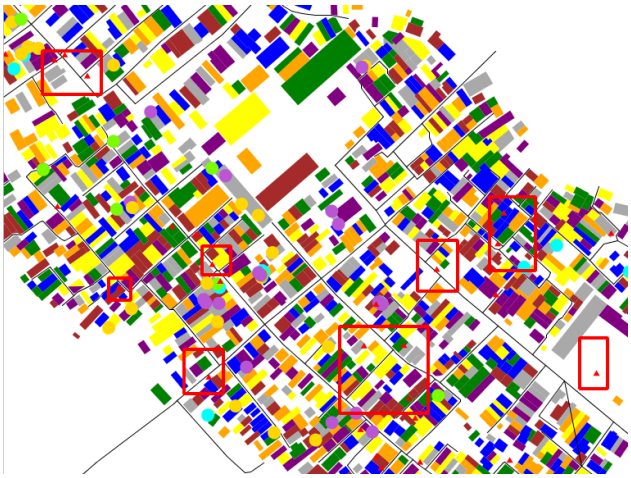

Figure 19: Exploration M2_4, a triangle is the virus in the environment.

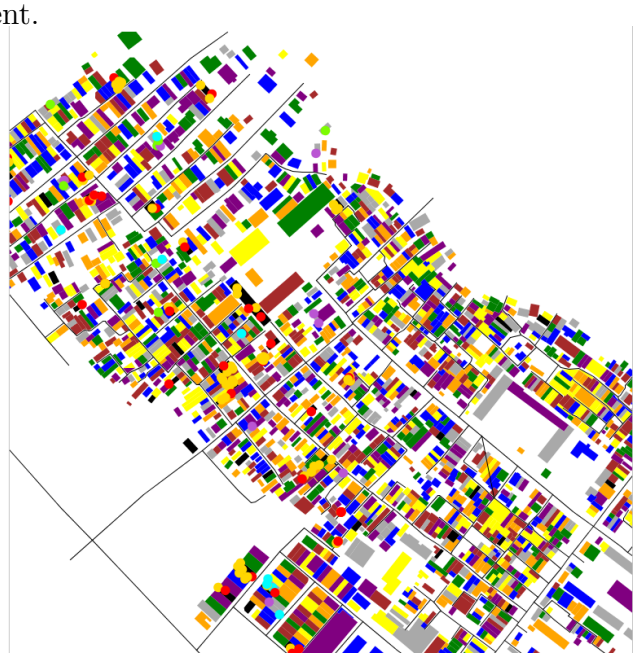

Figure 20: Exploration M2_4, E inhabitants become I inhabitants and the places where become black colour that means infected inhabitant went to these places.

Exploration

E2.4. comparing the disease spread with and without environmental transmission.

- I want to compare this model with the M2_3 model because they are almost the same excepts the environment spred of M2_4. I use the same experiment with M2_3 which are E2_3.
- The speed and also the number of people in exploration of M2_4 is clearly higher than the number in exploration of M2_5, the reason is in the model M2_4 we have the feature that allows virus can be transmitted in the environment so it enhances the number of exposed inhabitant very fast.

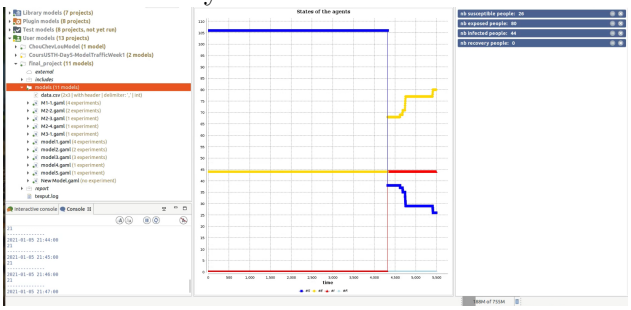

Figure 21: Exploration E2_3, the date is displayed on the console, it shows that the number of exposed habitants starts increasing from the fifth day.

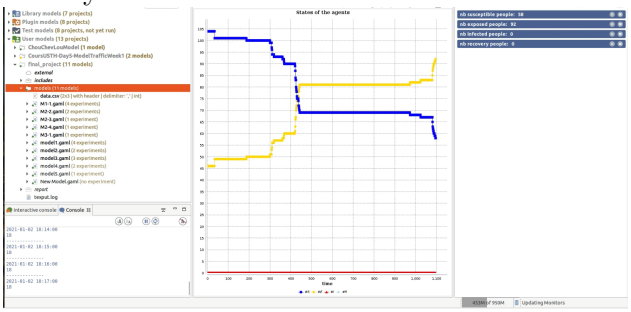

Figure 22: Exploration E2_4, the date is displayed on the console, it shows that the number of exposed habitants starts increasing from the second day, it also increases very fast because of environmental transmission feature.

3 Public health policy

- Create two new species: the LocalAuthority and the Policy.
- LocalAuthority decides the policy.
- The policy can be trigger by time, infected inhabitants appear
etc.
- The policy is listed in the requirement.
RESULT

```
global{
  etc..................
  int number_of_policy <- 4;
  int policy_index <- 0;
  list list_of_policy <- ["lockdown","lockage","lockschool","wearmask"];
  string law <- "freedom";
  float infect_prop <- 0.0;
  int count_policy_date <- 0;
  create policies number: number_of_policy{
    policy <- list_of_policy[policy_index];
    policy_index <- policy_index + 1;
  }
  create local_authority number: 1;
  etc..................
}
species local_authority{
  reflex checkPolicyDate when: law != "freedom"{
    // one day is 24 hour -> 3 days is 72 hour
    // one day is 24 hour -> 10 days is 240 hour
    // one day is 24 hour -> 30 days is 720 hour
    if(count_policy_date >= 72){// for testing: 3 days for one policy 72 hours
      law <- "freedom"; // Change to freedom state
    }
  }
  reflex searchPolicy
  when: law = "freedom"
  and inhabitants count(each.is_infected_state = true) > 1{
  //Check, can we free all inhabitants ?
    ask one_of(policies){
      law <- self.policy;
    }
    if (law = "wearmask"){
    //The propability of disease transmit when inhabitant wears mask
      infect_prop <- 0.5;
    }
    else{
      infect_prop <- 0.0;
    }
  }
  reflex increasePolicyDuration when: law != "freedom"{// count law's duration
```

```
    if(count_policy_date = 72){
      count_policy_date <- 0;
    }
    if(cycle mod 60 = 0 and cycle != 0){
      count_policy_date <- count_policy_date + 1;
      write("count_policy_date: "+count_policy_date);
    }
  }
}

species policies{
  string policy;
}

experiment M3_1{
  output{
    display map{
      species buildings aspect:goem;
      species roads aspect: goem;
      species inhabitants aspect: goem;
      species virus aspect: goem;
    }
    monitor "nb susceptible people"
    value: inhabitants count(each.is_susceptible_state = true);
    monitor "nb exposed people"
    value: inhabitants count(each.is_exposed_state = true);
    monitor "nb infected people"
    value: inhabitants count(each.is_infected_state = true);
    monitor "nb recovery people"
    value: inhabitants count(each.is_recovery_state = true);
  }
}
```

- I add two new species which are local_authority and policies, polices contain all the laws in the list_of_policy. Beside, the local_authority choose the suitable law, the decision of this species is based on the number of infected case when the number of infected case is higher than one local_authority will promulgate the law, to control the date of the law I create the action which start counting the date when the law is executed. I also modify the action of inhabitants species so that they can execute action base on the global variable law.

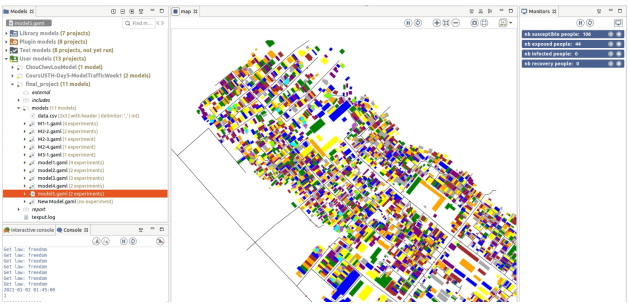

Figure 23: Exploration M3_1, all inhabitants get freedom law.

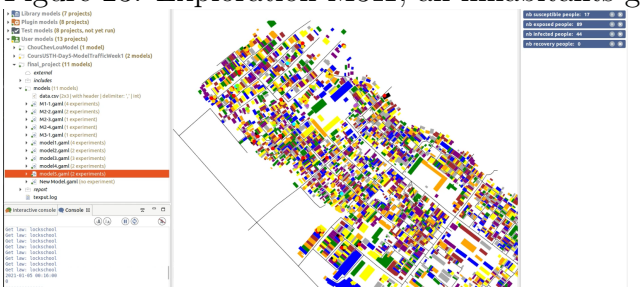

Figure 24: Exploration M3_1, all inhabitants get lockschool law when it finds out the infected case.

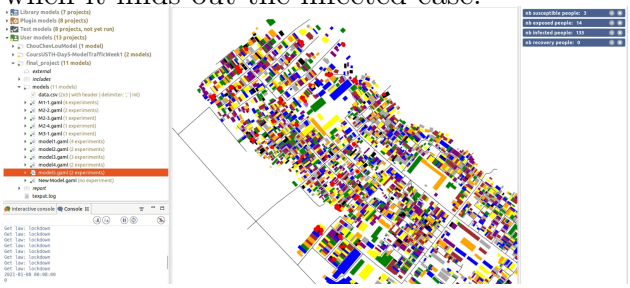

Figure 25: Exploration M3_1, after three days local_authority will consider to change the law bases on the desease situation, if model still have infected cases, it will change the law, and when infected cases don't exist, it will promulgate the freedom law so inhabitants can move around again.

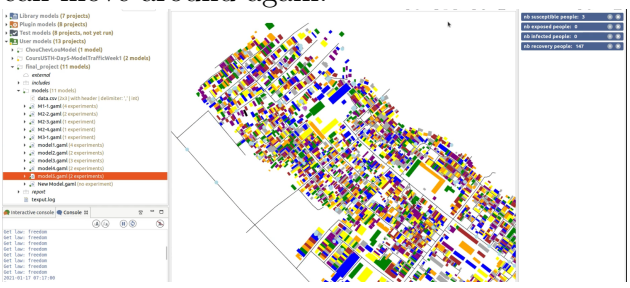

Figure 26: Exploration M3_1, freedom again !.