

# Individual Project Report

——21099695d Zhoudao LU

## Problem statement:

### Topic 3: Simulation for Social Distancing

Suppose we have a type of virus that causes severe respiratory syndrome and can transmit to other people if they get close. The virus causes severe illness that is more likely in elderly patients and those with certain underlying medical conditions

Develop a C++ program for scientists to study the impact of vaccination by developing a simulator. In the simulator,  $N$  users will walk in an area of  $X$  meter \*  $X$  meter. Some users randomly walk (i.e., random direction with random step length), and others walk along straight lines. When a user reaches the boundary of the area, the user may change the walking direction to a random direction within the area and continue.

When two users move closer within  $C$  meters, the virus can transmit to another user. Initially we have  $R$  contagious users and  $V$  vaccinated users among the total of  $N$  users. We assume that vaccinated users will not be transmitted or become contagious (CLAIM: which may not be the case in reality).

In your program, it receives inputs from scientists, e.g.,  $N$ ,  $X$ ,  $C$ ,  $R$ , and  $V$ . You may add other variables for scientists, e.g., walking speed, social distancing rules, percentage of users that violates the social distancing rules, possibility of virus transmission within  $C$  meters, etc. The program receives inputs from scientists and visualizes users with different colors indicating their status (e.g., vaccinated, contagious, others). (You can use external libraries for text/background coloring for this topic) The users' locations, their statuses, and total number of users in different statuses should be updated in the simulation.

### Problems summary:

1. How to make some users randomly walk (i.e., random direction with random step length), and others walk along straight lines?
2. When a user reaches the boundary of the area, how to make the user change the walking direction to a random direction within the area and continue?
3. When two users move closer within  $C$  meters, how to make the virus can transmit to another user?
4. What other variables I want to add?

5. How to visualize users with different colors indicating their status (e.g., vaccinated, contagious, others)?
6. How to update the users' locations, their statuses, and total number of users in different statuses?

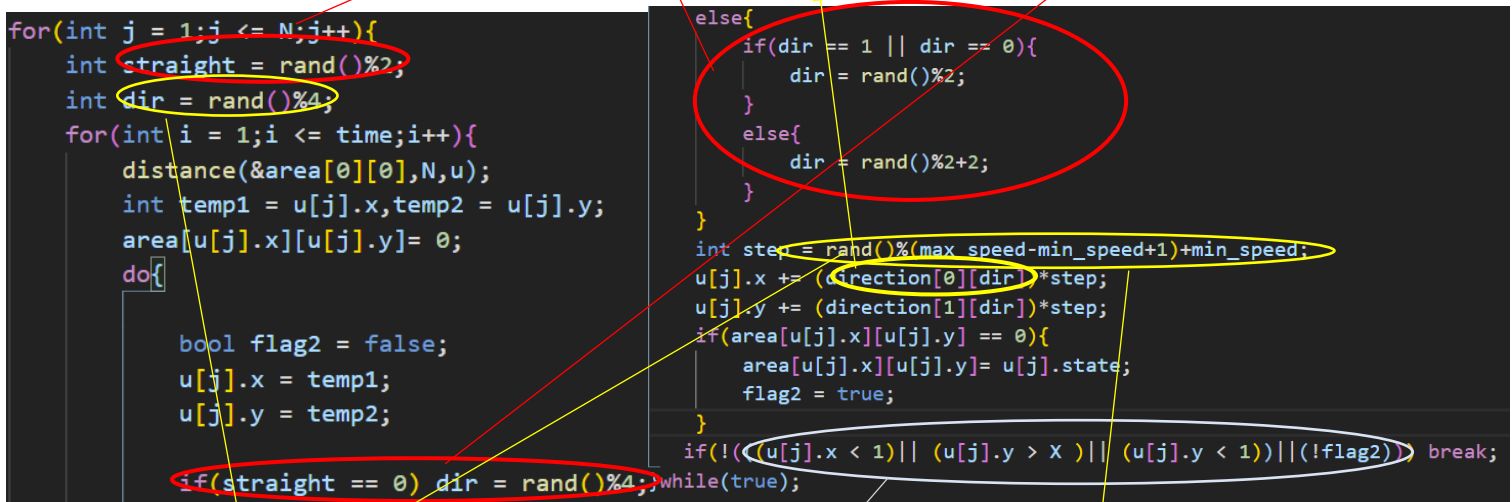
### The objectives of the program:

To set a basic simulator

### What I want to achieve?

For 1, How to make some users randomly walk (i.e., random direction with random step length), and others walk along straight lines?

I set a variable named "straight", if "straight" equals to 0, the user randomly walks; If "straight" equals to 1, the user walks along straight lines. "direction [][]" is a direction array, by



```

for(int j = 1; j <= N; j++){
    int straight = rand()%2;
    int dir = rand()%4;
    for(int i = 1; i <= time; i++){
        distance(&area[0][0], N, u);
        int temp1 = u[j].x, temp2 = u[j].y;
        area[u[j].x][u[j].y] = 0;
        do{
            bool flag2 = false;
            u[j].x = temp1;
            u[j].y = temp2;
            if(straight == 0) dir = rand()%4;
            else{
                if(dir == 1 || dir == 0){
                    dir = rand()%2;
                }
                else{
                    dir = rand()%2+2;
                }
            }
            int step = rand()%(max_speed-min_speed+1)+min_speed;
            u[j].x += (direction[0][dir])*step;
            u[j].y += (direction[1][dir])*step;
            if(area[u[j].x][u[j].y] == 0){
                area[u[j].x][u[j].y] = u[j].state;
                flag2 = true;
            }
        } while(!((u[j].x < 1) || (u[j].y > X) || (u[j].y < 1) || (!flag2)));
        break;
    }
}

```

The image shows a C++ code snippet for a user movement simulator. Annotations include red circles around `int straight = rand()%2;`, `int dir = rand()%4;`, and `if(straight == 0) dir = rand()%4;`. A red circle also highlights the direction selection logic in the `else` block. Yellow circles highlight `int step = rand()%(max_speed-min_speed+1)+min_speed;` and the boundary check `if(!((u[j].x < 1) || (u[j].y > X) || (u[j].y < 1) || (!flag2)))`. Yellow arrows point from the text explanations to these specific code lines.

get the "dir" from 0 to 3, the user who are randomly walk can walk in a random direction; By get the "step" from "min\_speed" to "max\_speed", the users who are randomly walk can walk in random step length.

For 2, when a user reaches the boundary of the area, how to make the user change the walking direction to a random direction within the area and continue?

It the same as the question 1, if the users reach the boundary, they still can walk in a random direction and just can't cross the lines. I run this codes in every minute which I set, so I don't specially write a program to make the users turn when they meet the boundary.

For 3, when two users move closer within C meters, how to make the virus can transmit to another user?

I make a function named distance(int\*,int,user []), if "distance1" is less than or equal to C, and one of the two sides was infected and the other was neither infected nor vaccinated, the other side will be contagious. Of course, if the scientists choose my additional functions, maybe the users have another possibility to be infected.

```
for(int i = 1; i <= n; i++){
    for(int j = i+1; j <= n; j++){
        //to compute the distance
        distance1 = sqrt(pow(u[i].x-u[j].x,2)+pow(u[i].y-u[j].y,2));
        if((distance1 <= C)&&(u[i].state != 2)&&(u[j].state != 2)){
            if((u[i].state == 1&&u[j].state == 3)|| (u[j].state == 1&&u[i].state == 3)){
                // some of the parameters are used for the other functions
                if(u[i].mask == true&& u[j].mask==true) p = p2;
                else if(u[i].mask == true || u[j].mask==true) p = p1;
                p = p*1000000;
                int k = rand()%999999+1;
                if(k <= p){
                    u[j].state = 1;
                    u[i].state = 1;
                    flag = false;
                }
            }
        }
    }
}
```

For 4, what other variables I want to add?

I add three additional functions, walking speed, possibility of virus transmission within C meters and the number of users wearing masks.

**For (1) Walking speed**, the program will remind the scientists to enter the minimum and maximum speed, if the minimum speed is larger than the maximum, the program will ask the scientists to enter again. But because the side length of the area, if the total number and the maximum speed is too large at the same time, the system terminates abnormally with no output. This is one of the drawbacks, I will try to solve it if I have free time later.

```
cout << "Reminder: If the maximum speed is too high, the simulator may break down." << endl;
do{cout << "Please enter the minimum speed(meter/minute): ";
cin >> min_speed;
cout << "Please enter the maximum speed(meter/minute): ";
cin >> max_speed;
```

**For (2) possibility of virus transmission within C meters**, the program will remind the scientists to enter the possibility, if the possibility is not between 0 and 1, the program will ask the scientists to enter a possibility again. Then for any users who should have been directly infected, the infection possibility may become p.

```
cout << "Please enter the possibility of virus transmission within " << C << " meters(a decimal from 0 to 1, not a fraction): ";
cin >> p;
if(p <= 1 && p >= 0 ){
    break;
}
cout << "Sorry! Your possibility is not between 0 and 1, please enter the data again." << endl;
```

For (3) the number of users wearing masks, the scientists need to enter the total number of users wearing masks, the number of contagious users wearing masks, the number of users who are neither vaccinated nor contagious wearing masks and two possibilities. The details will be introduced in “User manual”. For any users who should have been directly infected, the infection possibility may become  $p$ ,  $p_1$ , or  $p_2$  up to different situations.

```
cout << "There are now " << N << " users. " << endl;
cout << "Please enter the total number of users wearing masks: ";
cin >> T;
if(T > N){
    do{
        cout << endl;
        cout << "Sorry! This number is too large, please enter the number of contagious users wearing masks again: ";
        cin >> T;
    }while(T > N);
}

cout << "Please enter the possibility of virus transmission if one side takes a mask(a decimal from 0 to 1): ";
cin >> p1;

cout << "Please enter the possibility of virus transmission if both side take a mask(a decimal from 0 to 1): ";
cin >> p2;
if(p2 <= 1 && p2 >= 0 ){
    break;
}
cout << "Sorry! Your possibility is not between 0 and 1, please enter the data again." << endl;
cout << endl;
```

If the number of users is too large or the possibilities are not between 0 to 1, the program will remind the scientists to enter again.

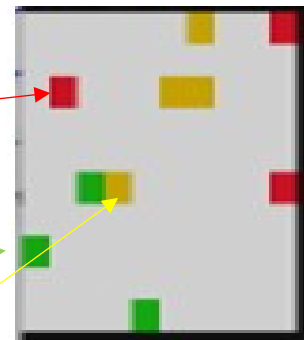
For 5, How to visualize users with different colors indicating their status (e.g., vaccinated, contagious, others), I use a library named “Windows.h” for text/background coloring. I write a function named “SetColorAndBackground(int,int)”. The details of this function will be introduced in “Program design”.

```
for(int i = 1;i <= X;i++){
    for(int j = 1;j <= X;j++){
        if(area[i][j] == 1){
            SetColorAndBackground(7,4);
            cout << " ";
        }
        else if(area[i][j] == 2){
            SetColorAndBackground(7,2);
            cout << " ";
        }
        else if((area[i][j] == 3)){
            SetColorAndBackground(7,6);
            cout << " ";
        }
    }
}
```

red

green

yellow



For 6, how to update the users' locations, their statuses, and total number of users in different statuses. This is the most difficult problem. Because I am a new hand of C++, and I don't have the confidence to control "time.h", and the function is up to our design, so I decided not to use the operation which I am not familiar with. And because of the random steps of the users, status updated per second may not very intuitive. By these reasons, I think about another way.

```
cout << "After how long time(minute(s)) do you want to see the change: ";  
cin >> time;
```

The scientists can enter a number to observe the changes of graphics or data after "time" minute(s). After the display, the scientists can continue to enter "time" to see the difference.

### Program design:

My program has one struct and six functions excepting main ().

"user" is a struct to store the state of different users. For "state," if "state" equals 1, the user is contagious. If it equals 2, the user is vaccinated, and 3 indicates the user is neither infectious nor vaccinated.

"x" and "y" are used to store the users' coordinates, "mask" is for my additional function. If the users wear a mask, it will change from false to true.

```
struct user{  
    int state;  
    int x,y;  
    bool mask;  
};
```

"BacicalInput()" is a function to remind the scientists to enter the basic information of the simulator. At first, I make a menu to let the scientists make a choice: graph display, data display or both. I will explain these in detail in "User manual". And then, the scientists should enter the basic information, like N (the number of users), X (the side length of the area which the users can move), C (the distance(meters) when two users move closer, the virus can transmit to another user), R (the number of contagious users) and V (the number of vaccinated users).

```
What function do you want?  
(1) Graph display  
(2) Data display  
(3) Both
```

By the ways, for some of the input of numbers, if the number is too large, I set up a mechanism to remind the scientists to re-enter.

SetColorAndBackground(int,int) is a function to set different color of the graph display. This first parameter decides the color of the background, the second one decides the color of the

```
void SetColorAndBackground(int ForgC,int BackC){  
    WORD wColor = ((BackC & 0x0F) << 4)+(ForgC & 0x0F);  
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),wColor);  
    return;  
}
```

written words. If the background color is red, the users are contagious; If the background color is green, the users are vaccinated; If the background color is yellow, the users are neither

infectious nor vaccinated.

DataOutput() is a function to display the related data, if the scientists choose 2 or 3 when the program asks “What function do you want”, the number of users, the number of contagious users and the number of vaccinated users will be printed out on the screen.

distance(int\* ,int,user []) is a function to make the users infect when they are too close to the contagious users. The first parameter is to read the array of the area where users can walk; The second parameter decides the distance(meters) when two users move closer, the virus can transmit to another user; The third one is to read the array of the struct “users”.

By the way, if the users are too close, I consider the transmission of the virus may need some time, so the users may not be contagious at the same time. The program will update the statement of one user next minute.

LatterInput() is a function to ask if the user want to continue after each cycle. The details will be introduced in “User manual”.

```
What do you want to do next?
(1) Observing the next change of the graph or the data
(2) Abandoning this simulation

Please enter your choice (1 or 2):
```

AddInput(user []) is a function to enter the data of the additional functions. If the scientists want to add other variables, they can choose ‘y’ and choose which features the scientists want to consider. The details were be introduced in “The objectives of the program” or will be introduced in “User manual”.

### Structure of the program:

I show the idea of main () here.

At first, the program call the function “BacicalInput()”, to set the basic variables N, X, C, R, V and the area((line38~43)), like picture (1).

```
char choice;
choice = BasicInput();
srand(time(0));
user u[N+5];
const int length = X+5;
int area[length][length] = {0};
```

(1)

```
for(int i = 1; i <= N; i++){
    u[i].mask = false;
    u[i].x = rand()%X+1;
    u[i].y = rand()%X+1;
    if(area[u[i].x][u[i].y] != 0){
        i--;
        continue;
    }
    if(i <= R){
        u[i].state = 1;
        area[u[i].x][u[i].y] = 1;
    }
}
```

(2) part of the codes

On the premise of basic variables, the program initializes users' statuses and locations. Picture (2) is part of the codes. u[i].x, u[i].y is used to initialize the users' location. u[i].state is for the statement of the users(line46~68).

And then, the program call the function “AddInput(user [])” for the additional functions(line71).

```
AddInput(u);
```

Next, the program will make the first presentation according to what the scientists want, graph, data, or both. Part of the codes are shown below. (Integrated: line75~107)

```
if(choice != '2'){
    cout << endl;
    cout << "The initialization grph is as follows: " << endl;
    for(int i = 1; i <= X; i++){
        for(int j = 1; j <= X; j++){
            if(area[i][j] == 1){
                SetColorAndBackground(7,4);
                cout << " ";
            }
            else if(area[i][j] == 2){
                SetColorAndBackground(7,2);
                cout << " ";
            }
            else if((area[i][j] == 3)){
                SetColorAndBackground(7,6);
                cout << " ";
            }
        }
    }
}
```

```
if(choice != '1'){
    cout << endl;
    cout << "The initialization data is as follows: " << endl;
    DataOutput();
}
```

(3)

After these operations, the program will ask whether the scientists want to see the next change. The scientists can enter a number to observe the changes of graphics or data after “time” minute(s). After the display, the scientists can continue to enter” time” to see the difference. Part of the codes are shown below. (Integrated: line110~164)

**By the way, if the “max\_speed”, “time” or the side length are too large, after entering “time”, the program may terminate suddenly. If this situation happens, just enter some smaller data to try.**

```
choice2 = LatterInput();
if(choice2 == '1'){
    cout << endl;
    cout << "The users randomly walk, or walk along straight lines (The speed we set is between 1 and 6 meter every minute, you can see the change after time minute(s)).";
    cout << endl;
    cout << "After how long time(minute(s)) do you want to see the change: ";
    cin >> time;
    bool flag2 = false;
```

Because the situation is updated, the number of contagious users is still changed. So, we need to count the number again. the codes are shown below (line167~173).

```
R = 0;
for(int i = 1; i <= N; i++){
    if(u[i].state == 1){
        R++;
        area[u[i].x][u[i].y] = 1;
    }
}
```

```
if(R+V == N){
    cout << "Oh no! Anyone who hasn't been vaccinated has been infected." << endl;
    cout << endl;
    cout << " Welcome to use our simulator next time!!! " << endl;
    break;
}
```

(4)

After that, the screen will show the new status. The codes are like the picture (3).(line177~216)

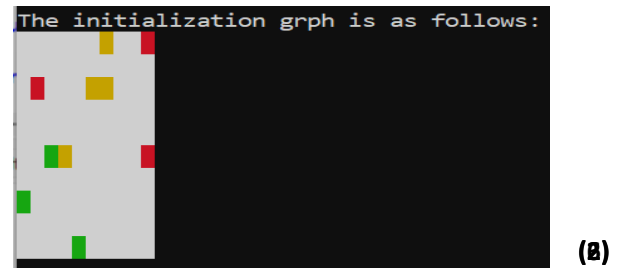
If all the users who are not vaccinated are contagious, I think the program is not necessary to continue, so I terminate the program, like picture (4).(line219~225)

### User manual:

```
What function do you want?
(1) Graph display
(2) Data display
(3) Both

Please enter your choice (1,2 or 3):  (5)
```

```
The initialization grph is as follows:
  (8)
```



## 1. User can make a choice at the beginning

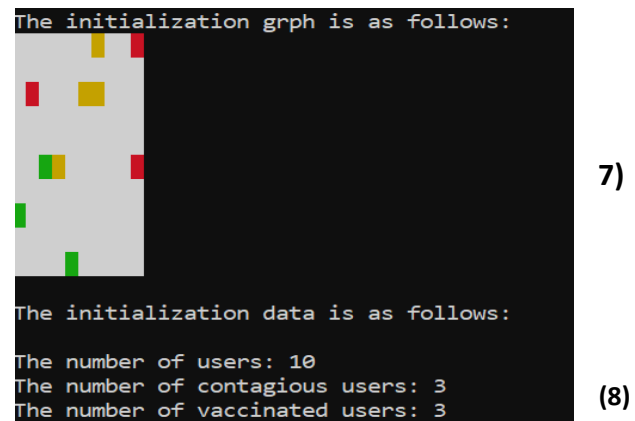
If you choose 1, only the graph can be displayed, like picture (6); If you choose 2, only the data can be displayed, like picture (7); If you choose 3, both will be displayed, like picture (8). If you don't enter 1, 2 or 3, the program will ask you to enter again, like picture (9).

```
The initialization data is as follows:
The number of users: 10
The number of contagious users: 3
The number of vaccinated users: 3 (
```

```
Please enter your choice (1,2 or 3): 4
Error! Please enter your choice (1,2 or 3) again: s
Error! Please enter your choice (1,2 or 3) again:  (
```

```
The initialization grph is as follows:
  (7)

The initialization data is as follows:
The number of users: 10
The number of contagious users: 3
The number of vaccinated users: 3 (8)
```



9)

## 2. Then, user can enter the basic information

```
input panel:
Please enter the number of users: 100

Reminder: because of the limitation of screen size, the input value of the "distance(meters) when two users
move closer, the virus can transmit to another user" should no more than 700

Please enter the side length(meters) of the area which the users can move: 100
Please enter the distance(meters) when two users move closer, the virus can transmit to another user: 10
Please enter the number of contagious users: 20
Please enter the number of vaccinated users: 30
```

If the side length is too large, this program will break down. So, to prevent this situation



happening, there is a limitation of the input range.

If the number is too large, like the sum of the contagious and vaccinated users is larger than the total number, you should enter the data again, please be careful with your input.

```
Sorry! the sum of contagious and vaccinated users are bigger than the number of users!  
Please enter the data again.
```

But if you enter a letter, this program will break down, and we haven't set up an error correction mechanism, which requires users' attention.

### 3. Next, user can consider whether other functions are required

If you enter 'n' or 'N', the program will just consider the basic information. If you enter 'y' or 'Y', the screen shows the choices like picture (6).

```
Do you want to add other variables(y or n)? y  
  
(1) Walking speed  
(2) Possibility of virus transmission within 10 meter  
(3) The number of users wearing masks  
.....  
We are still developing other functions  
  
What variables do you want(If you want to choose more than one answer, like choosing (1), (2) and (3), you should input 123 )? 
```

(10)

If you choose 1, you can enter the minimum and maximum speed, like picture (11). If the minimum speed is larger than the maximum, the program will ask you to enter again. And as mentioned in "The objectives of the program", if the total number and the maximum speed is too large at the same time, the system terminates abnormally with no output. This requires users' attention.

```
Reminder: If the maximum speed is too high, the simulator may break down.  
Please enter the minimum speed(meter/minute): 1  
Please enter the maximum speed(meter/minute): 10
```

(11)

If you choose 2, you can enter the possibility of virus transmission within C meters. But be careful, if you enter a number which is not between 0 and 1, the program will ask you to enter again.

And please enter a decimal, not a fraction.

```
Please enter the possibility of virus transmission within 20 meters(a decimal from 0 to 1, not a fraction): 1.2  
Sorry! Your possibility is not between 0 and 1, please enter the data again.  
  
Please enter the possibility of virus transmission within 20 meters(a decimal from 0 to 1, not a fraction): 0.4
```

(12)

If you choose 3, you can enter the number of users, contagious users and users who are neither vaccinated nor contagious, if the number is too large, the program will ask you to enter again. And then, you need to enter two possibilities like this picture. The requirement is the same as the situation that you choose 2. Please enter a decimal, not a fraction.

```

There are now 100 users.
Please enter the total number of users wearing masks: 50

There are now 20 contagious users.
Please enter the number of contagious users wearing masks: 10

There are now 60 users who are neither vaccinated nor contagious.
Please enter the number of users who are neither vaccinated nor contagious wearing masks: 30

Please enter the possibility of virus transmission if one side takes a mask(a decimal from 0 to 1, not a fraction): 0.4
Please enter the possibility of virus transmission if both side take a mask(a decimal from 0 to 1, not a fraction): 0.2

```

(13)

#### 4. After the first display, the user can decide whether to see the updated data or not

If you enter other numbers, the program will remind the user to enter again.

If you enter 2, the program will terminate.

If you enter 1, you can enter the time of “After how long time do you want to see the change”, and then you can see the updated displays. **If the program terminates, don't worried, maybe the data you input is too large, just try again.**

```

What do you want to do next?
(1) Observing the next change of the graph or the data
(2) Abandoning this simulation

Please enter your choice (1 or 2): 3

Sorry! Please enter your choice (1 or 2) again: 1

The users randomly walk, or walk along straight lines (The speed we set is between 1 and 6 meter every minute, you can set it by yourself )

After how long time(minute(s)) do you want to see the change: 10

```

#### 5.The user can keep updating the data until quitting

If you want to update the data, keep choosing 1 until all the users who are not vaccinated are contagious. Then the program will terminate by itself. If you want to make it stop earlier, just type 2.

```

What do you want to do next?
(1) Observing the next change of the graph or the data
(2) Abandoning this simulation

Please enter your choice (1 or 2): 1

The users randomly walk, or walk along straight lines (The speed we set is between 1 and 6 meter every minute, you can set it by yourself )

After how long time(minute(s)) do you want to see the change: 

```