2022 IM^{2}C Problem D (Winter)

-MetaVerse Hong Kong-

IMMC 22823005

Summary

Talking about “MetaVerse” is just like talking about “the Internet” in the 1970s, whilst synchronizing real-time image with MetaVerse could be lots of computing and storage resources for the computer. In order to offer the user a real life experience and leave them with a want to try again feeling, we have developed our own building blocks based on how we think would be the most comfortable for both the users and the computer to achieve a pleasurable corporation.

In the first part of our model, we search some of the other MetaVerse games that already exists such as Minecraft and Roblox. We made a hypothesis and we discussed how to design our building blocks. We found two possible ways, which are mostly different to one another. The first way is to use little cubes as our building blocks, which might easily construct every kinds of buildings, but it is quite confusing when the computer need to make a man. Therefore, our second method is to use the whole building and man as our building blocks. This means that we need to find out every varieties of distinctive objects and that is quite a big number. As a result, we sort out a final plan which integrate out first and second plans. We use small cubes to build up buildings just like how the bricks are set up; at the same time, we see the smaller objects, such as human and cars, as individuals.

For the calculation of computing and storage resources, we read information from the external data to get what object is in a one cubic millimeter three-dimensional grid. Then we judge what type of basic component the object in the block is and match it with the known basic component. Then, it is determined whether it is a known foundation member without considering the color. After judgment, fill in the code number of basic components and record it. If the square is empty, it is recorded as - 1. If the above two conditions are not met, it shall be determined as the basic component with color and recorded. Then we define two arrays. One is used to store the known basic components, the location, type and color of the basic components. The other is used to store the occupied three-dimensional grid. Our core idea is to divide Hong Kong into blocks with a size of 1mm, scan them separately, judge what basic components are contained in each block and fill in three-dimensional blocks. If we don't find the basic construction after judgment, we will use several empty square fast or colored basic components. Next, judge whether it is occupied by the array, and mark the occupied three-dimensional box. Then look for the center point of people, vehicles and plant basic components, and then read the coordinates of the center point to judge whether the point is occupied. If it is not occupied, fill it in. If it is occupied, do not fill it in.

Lastly, our group estimate the storage and computing resources that the computer need to maintain the real-time Hong Kong in a MetaVerse. We leave extra space for the computer so that there will be enough for future addition of building blocks. Moreover, we choose the time precision of synchronization of 1 second in order to keep the date changing.

Content

1. Problem Analysis

1.1 Problem Introduction

• *Background:*

MetaVerse has been popularized recently and it is a user friendly virtual world where it could possibly achieve an overall pleasure-able experience by allowing users to create their own world. On the other hand, this requires quite a lot computing and storage resource to maintain the virtual world. Therefore, our group's goal is to create a virtual Hong Kong with our own building blocks[1] and to calculate how much resource it takes for the computer to accomplish this task.

1.2 Hypothesis and Preparatory Assumptions

• *Hypothesis:*

Minecraft usually requires 1-2 GB of storage without any constructions in it and it will require up to approximately 20 GB after storing user's data. Similarly, Roblox needs about 1 GB to run smoothly while it will increase its memory space after user build something in it. As a result, our group take a rough hypothesis that our MetaVerse will be about 1 GB with only the basic building blocks and it will take about 150 GB to maintain a real-time virtual Hong Kong.

• *Assumptions of How to build our Building Blocks:*

i. The first idea is that we could see Hong Kong as a cuboid with side lengths of the longest diameter from one side to another. Then we would divide the cuboid into little cubes which have a small side length of about 1 decimeter. Our next step would be input all of the possible things in Hong Kong into the computer, such as rocks, people, air, so that the computer could recognize these after it scans the whole Hong Kong.

However, this will lead to the inevitable chaos for the computer, because there might be more than one substance in those little cubes, and the computer is not able to figure out which to output in our virtual Hong Kong.

ii. Then we think of setting our building blocks as a whole building, car and man, but in different kinds. For example, we could make high buildings and lower ones and we could make men and women. When the computer receive how the real-time Hong Kong looks like, it could just choose which building blocks to output and it won't necessarily worry about the combination of various building blocks.

We go on Google Map to see how Hong Kong actually looks like, it turns out that the apartments are not in the same shape even if they are in a same block. We would have to design lots of categories of buildings and it is not idealistic.

iii. As a result, we decide to separate our calculation and design. For buildings and land in Hong Kong, we are going to build our building blocks of cubes for the computer to construct the buildings. We are going to design cubes of different materials such as water and iron. On the other hand, we are going to make building blocks of different kinds of people, cars and those smaller components of Hong Kong which is much more difficult for the computer to build by itself.

1.3 Overview of Timeline

Based on the two tasks of designing and estimating, our team decided to complete the following goal with chronological order:

1. design the building blocks of the building in Hong Kong
2. design the building blocks of other components in Hong Kong
3. calculate how much calculating and storage resource are required for each of the building blocks
4. calculate how many building blocks are there in Hong Kong for each
5. sum up all the resources needed for every building blocks

2. Building Blocks

2.1 Basic Building Blocks

We decided to use little cubes with side lengths of 1cm as our building blocks for all of the buildings in Hong Kong, so that it will be easier for the computer when there are more than 1 substance exist in one cube. We decided the basic cubes into two parts, natural and man-made.

Natural Building Blocks include:

1. water
2. soil
3. stone
4. sand

Man-made Building Blocks include:

1. gold
2. copper
3. alloy
4. granite
5. glass

6. concrete
7. wood
8. asphalt
9. plastic
10. marble
11. **COLOR BLOCK - to control and change color**

2.2 Calculation/Procedures

Building blocks are designed as different kinds of small regular cube. The whole virtual Hong Kong will be built by countless and colorful small cube.

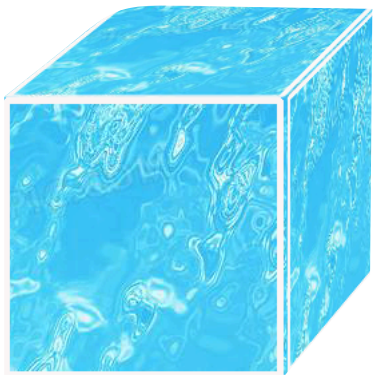
Advantage:

1) The virtual Hong Kong will be very like the real one. Cubes can build every kind of buildings in Hong Kong, even they have special and strange structure.
2) Users can create anything they can imagine. This allowing them to fully apply their creativity in building a virtual world they want

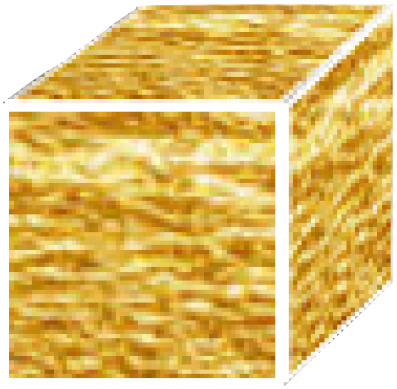
Disadvantage:

1) This method needs huge estimate the computing resources, because it needs countless small cubes to create the virtual Hong Kong.
2) Although it can make the virtual Hong Kong very realistic, the small size of the cube will take user a lot of effort and time to build a high building or any big stuffs.

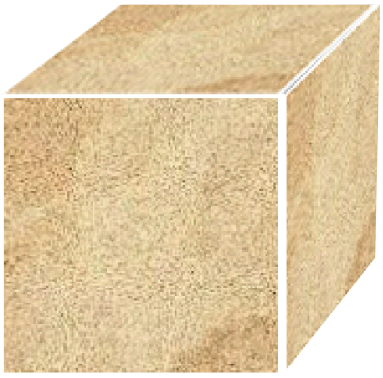
2.3 Examples and Graphs



-water cube-



-gold cube-



-sand cube-

3. Car Blocks

3.1 Basic Car Blocks

1. SUV
2. Subway
3. bigger bus
4. smaller bus
5. Carriage
6. Truck

3.2 Car Block's Volume

Vehicle	length(m)	width(m)	height(m)	volume(mm^3)
SUV	2.5	2	2	8×10^9
Subway	23	4	3	2.76×10^{11}
Bus(big)	12	2	4.5	1.08×10^{11}
Bus(small)	10	2	3.5	7.0×10^{10}
Carriage	2.5	2	1.7	7.8×10^9
Truck	8	3	4	9.6×10^{10}

#The length, width and height here are on the surface and independent of the volume calculation results#

3.3 Calculations/Procedures

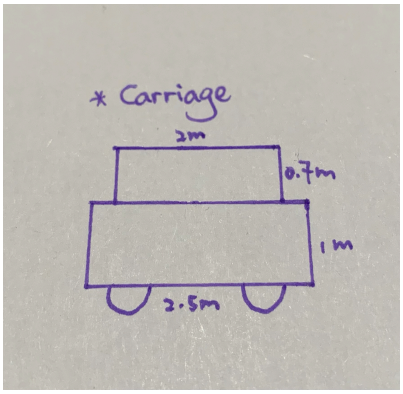
In the beginning, we also want to use similar blocks in Minecraft to build a car. Because we think this can restore the authenticity of a car to the greatest extent. However, we find that doing so will greatly waste our storage resources and computing resources. Therefore, we decided to design the basic components separately for people and common vehicles on the basis of the original basic components. By inquiring about the traffic situation in Hong Kong and relevant documentaries, we finally determined the following modes of transportation: car, bus, truck, SUV and subway. Then we selected the most common models in Hong Kong in each category for data search and field measurement and then combined the two into our basic model.

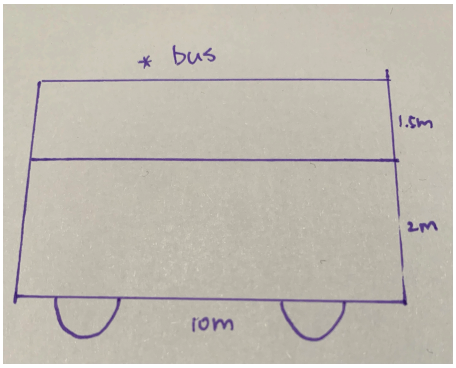
We decided use different shape to build up our car models, it is easy and direct for computer to figure out which car is going to be. Firstly, we use two rectangles to build up the carriage, a small one on the top and a larger under. Secondly, we use two rectangles with same length but different width to represent a single-decker bus. Similarly, we use two larger rectangle to represent double-

decker bus, because in Hong Kong, there are 50% of single-decker bus and 50% of double-decker bus. Then, we use several same rectangle with cables between to represent subway. We use two totally different-size rectangle with a left-right position to represent SUV while we use a long and thick rectangle to be truck.

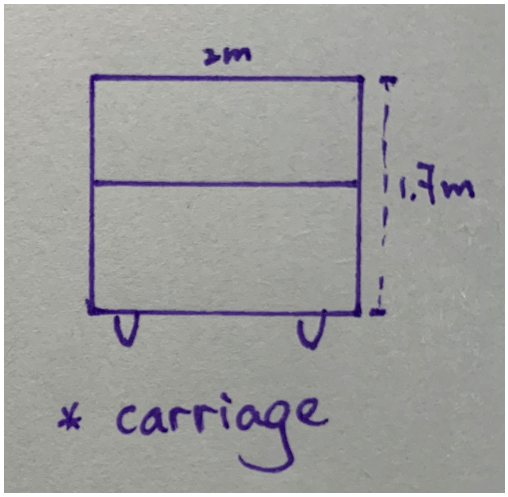
Lastly, we change those 2-dimensional cars into 3-dimension, we search up the average width of every kind of cars. As a result, we use the length, width and height and times it altogether in order to get the volume of each car.

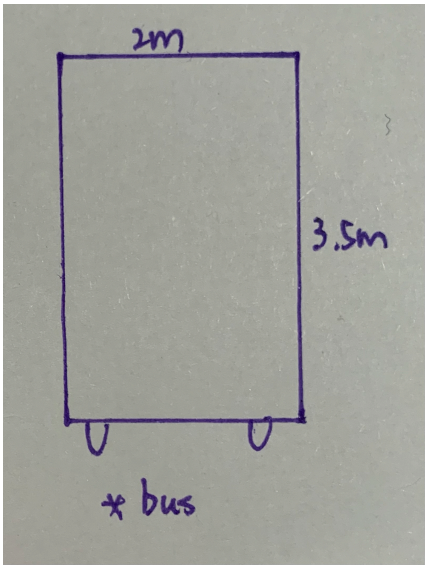
3.4 Examples and Graphs





 # side views





 # front views

4. Human Blocks

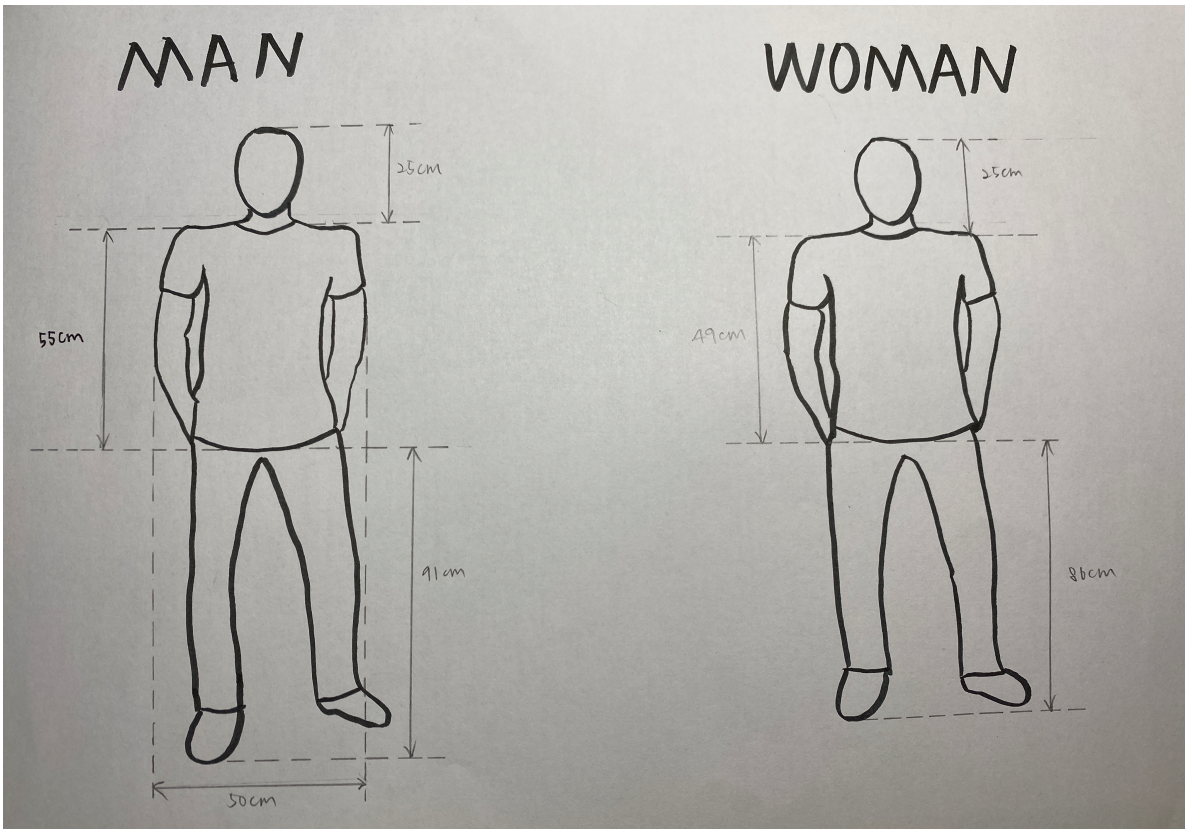
4.1 Basic Human Blocks

1. Man
2. Woman

4.2 Human Block's Volume

Human	Height(cm)	Width(cm)	Thickness(cm)	Volume(mm^3)
Man	171	50	30	1.33×10^8
Woman	160	50	30	1.17×10^8

3.3 Calculations/Procedures



We divided human's body into three parts to calculate: head, body and legs. For both male and female, their head and neck have a volume of $\frac{4}{3}\cdot\pi\cdot9.1^2\cdot8\cdot\frac{1}{2}+\frac{1}{3}\cdot\pi\cdot9.1^2\cdot13$ $=2.515\times10^6\,mm^3$. At the same time, man's body has a volume of $55\cdot50\cdot30=8.25\times10^7\,mm^3$ while

woman's body has a volume of approximately $49 \cdot 50 \cdot 30 = 7.35\times10^7$ mm^3. Moreover, man's leg has a volume of 4.785×10^7 mm^3 while woman's leg has a volume of 4.07×10^7 mm^3.

5. Other Blocks

Tree Blocks:

Categories	Volume(mm^3)
Grass	1
Tree	2.8×10^8
Towering Tree	5.7×10^8
Haystack	7.2×10^9

6. Storage Resource

6.1 Calculate Method

Transportation: 8+276+108+70+7.8+96=565.8 (m^3)=565800000000 (mm^3)
People: (1.33+1.17)*10^8=2.5*10^8 (mm^3)
Botany: 1+2.8*10^8+5.7*10^8 + 7.2*10^9=8050000001 (mm^3)

Total: 565800000000+2.5*10^8**+8050000001=5.741***10^{11} **(mm^3)**

6.2 Estimate Result

We regard a millimeter three-dimensional block as a 16 * 16 lattice depiction. Because we have a total of 5.741*10^{11} cubic millimeters of three-dimensional blocks that need to be composed of lattice, the total number of lattices is 9185600000000*9185600000000. The storage space required to store such a large lattice is:
9185600000000*9185600000000/8=**143497.830062EB**

7. Possible Pseudocode

//blue part is the definition

```
#include <iostream>
#include <stirng>

using namespace std;

string Select_categorize(categorize)  //calculate times = 1 + 1 + 32 + 2 + 1 = 37
{
   string Read_type; //calculate times = 1
   //Definition: Used to read Hong Kong's data externally

   cin>> Read_type; //calculate times=1
   //read substances inside a cube

   //determine which substance it is

   for (int i = 0; i < 16; i++) //calculate times= 16 * 2
   {
       if (categorize[i] == Read_type && Read_type != "none") //Determine whether it is a building block (except for the color block) and the building block is not empty
      {
         return i; //return the number of the building block
      }
   }

   if (Read_type == "none") //calculate times= 2
   {
      return -1; //If the box is empty, return -1 as to keep a record
   }

   return -2; //If it doesn't match, it will be the base color block and return -2 as to keep a record
   //calculate time = 1

}


int main()
{

   int HongKong[50000000 , 70000000 , 1000000, 3]; //calculate times = 1
    //"Hongkong" represents a three-dimensional image of the entire Hong Kong. The first three are the three-dimensional coordinates x, y, and z. The last digit is used to store the type and color (RBG) of the object filled in the box.

   int space_hongkong[50000000 , 70000000 , 1000000]; //calculate times = 1
   //"space_hongkong" is used to determine whether the block is occupied or not, if it is occupied, it will be displayed as 1
```

```
string categorize[16] = {"water","soil","stone","sand","gold","copper","alloy","granite","glass","concrete","wood","fibre","marble","asphalt","plastic","none"}; //calculate times = 1
   // Type of building block (except color blocks)


   //we will do three loops to cover every square in the range of Hong Kong
   for (int i = 0; i < 50000000; i++;) //calculate times = 55 * 50000000 * 70000000 * 1000000
   {
      for (int j = 0; j < 70000000; j++)
      {
         for (int k = 0; k < 1000000; k++)
         {
            int Number = Select_categorize(categorize); //calculate times = 1 + 37
            //Determine the type of building block in the cube

            HongKong[i, j, k, 0] = Number; //calculate times = 1
            //A three-dimensional image of Hong Kong recording the types of building blocks

               if (Number == -2) //Determine whether the block type is inconsistent with the building block (except for the color block) and is not empty //calculate times = 1 If so, the block is a color block
            {
               cin >> R_color >> B_color >> G_color; //calculate times = 3
               //read color in RGB format

               HongKong[i, j, k, 1] = R_color; //calculate times = 1
               HongKong[i, j, k, 2] = B_color; //calculate times = 1
               HongKong[i, j, k, 3] = G_color; //calculate times = 1
               //record color in the 3-dimensional Hong Kong


               space_hongkong[i, j, k] = 1; //calculate times = 1
               //Record the space in the three-dimensional image as 1 (occupied)
            }
            else
            {
               if (Number == -1) //If the cube is empty //calculate times= 1
               {
                  HongKong[i, j, k, 1] = -1; //calculate times= 1
                  HongKong[i, j, k, 2] = -1; //calculate times= 1
                  HongKong[i, j, k, 3] = -1; //calculate times= 1
                  //marked as -1 for keeping record
               }
                  else //If neither match, then the block is in the basic building block (except the color block)
               {
                  HongKong[i, j, k, 1] = -2; //calculate times = 1
```

```
                HongKong[i, j, k, 2] = -2; //calculate times = 1
                HongKong[i, j, k, 3] = -2; //calculate times = 1
                //marked as -2 for keeping record


                space_hongkong[i, j, k] = 1; //calculate times = 1
                //Record the space in the three-dimensional image as 1 (occupied)
            }

        }
    }
  }
}


string object[12] = {"man", "woman", "haystack", "little tree", "middle tree", "big tree", "limousine", "truck", "bus", "double-decker bus", "SUV", "underground"}; //calculate times = 1
  //type of building block

  int xyz_object[12, 100]; //calculate times = 1
  //Definition: Record the 3D details of the building block

  for (int i = 0; i < 12; i++) //calculate times = (3 +  3 * 1 * 10^10) * 12
  {
    int j = 0; //calculate times = 1
    int data = -1; //calculate times = 1
    //Define data to read the 3D details of the object

    cin >> data; //calculate times = 1
    //Read 3D details

    while (data != -1) //Store 3D details in an array
            //calculate times = 3 * 1 * 10^10 (The average volume occupied by the base block is 1*10^10mm^3)
    {
      xyz_object[i][j] = data; //calculate times = 1
      cin >> data; //calculate times = 1
      j++; //calculate times = 1
    }
  }


  cin >> type; //Read the type of the base block  //calculate times = 1



  while (type != "") //Determine whether the base block has been read //calculate times =  (2 + 6 *
```

```
2）* number of base blocks
   {
      int Number; //calculate times = 1
      int x,y,z; //calculate times = 1
      //define the number and coordinates of the basic blocks

        for (int i = 0; i < 12; i++) //calculate times = 6 * 2 (Take the middle value according to the principle of normal distribution)
      {
        if (type == object[i]) //Determine the type of the base blocks  //calculate times = 1
        {
           Number = i + 100; //Plus 100 for easy differentiation from base blocks  //calculate times = 1
           break;
        }
      }

      cin >> x >> y >> z; //calculate times = 3
      //Read the 3D coordinates of the center point of the base blocks

        if (space_hongkong[x][y][z] != 1)    //Determine if the center point is occupied   //calculate times = 1
      {
        space_hongkong[x][y][z] = 1; //calculate times = 1
        //Record the space in the three-dimensional image as 1 (occupied)
        HongKong[x][y][z] = Number; //calculate times = 1
        //Record the number of the base object into the 3D space of Hong Kong
      }

      cin >> type; //calculate times = 1
      //read next object
   }

   output(HongKong); //convert code into pixel and output virtual Hong Kong
```

8. Computing Resource

8.*1 Calculate Method*

Total Calculate times = （2 + 6 * 2）* amount of basic objects + (3 + 3 * 1 * 10^10) * 12 + 55 * 50000000 * 70000000 * 1000000 + 6

Amount of basic objects= cars + people + trees ≈ 5.34×10^{12}

8.*2 Estimate Result*
Computing Resource (Total Calculate times) ≈ **1.93×10^{23} times per second**

9. Review and Prospect

We mix our two original methods to ensure the trueness of the virtual Hong Kong and try our best to decrease the estimate the computing and storage resources. e made the models of cars and people and created a lot of cubes whose sides are 1mm that have different colors and texture to make things which have different and complex structures. What's more, we allow users to fully apply their creativity in building a virtual world they want.

However, we could not get the accurate number of how many buildings, cars or trees there are in Hong Kong, so that we could only make a rough guess based on how big is Hong Kong and multiply the average coverage of the entire world.

We could possibly make more investigation and develop a perfect way to scan and output Hong Kong with a smaller computing and storage resources for the computer.

10. References

香 港 的 女 性 及 男 性 主要統計數字 *Women and Men in Hong Kong Key Statistics 2021* 年版 *2021 Edition* 香港特別行政區 政府統計處 *Census and Statistics Department Hong Kong Special Administrative Region*. (n.d.).*https://www.censtatd.gov.hk/en/data/stat_report/product/B1130303/att/B11303032021AN21B0100.pdf*

(2022). Istem.info. http://istem.info/web/paper_preview.php?applyid=6275