CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

Project ini dapat memvisualisasikan data pada data structure hash table menjadi grafis 2D secara dinamis dengan inputan yang ditentukan oleh user. Data yang di inputkan akan masuk ke data structure hash table, di proses dan mendapatkan nilai yang kemudian dikirim ke user interface. Pada code user interface nilai yang dihasilkan dari data structure digunakan untuk draw dan menentukan koordinat dari visualisasi.

5.1.1 Set Size of Array

(16) The user input will go to int ts, (18) ts for set size of TABLE_SIZE.
(19) HashTableList is a list of arrays, each array of TABLE_SIZE will have a list.
(21) Number of arrays will be loaded according to TABLE_SIZE with null initial value.
5.1.2 Insert Value

```
public void insertvalue(int value)
{
    int hash = (value % TABLE_SIZE);
    if (table[hash] == null)
        table[hash] = new HashTableList(value);
    else
    {
        HashTableList entry = table[hash];
        while (entry.next != null)
            entry = entry.next;
        //while (entry.value == value)
        //entry.value = value;
        if (entry.value == value)
            entry.value = value;
        else
            entry.next = new HashTableList(value);
        size++;
    }
}
```

Illustration 5.2: Insert Value

(39) The user input will go to int value, (41) the value will modulo with TABLE_SIZE to get location from value, (42) If the initial array is empty then the value will be printed on array list 0, (46) If the initial array is not null then the array will have a list, (47) A loop will be performed to check the location until it gets an empty location and then the data will be printed.
5.1.3 Get Location of Value

```java
public int[] getLoc(int value) {
    int hash = (value % TABLE_SIZE);
    int[] result = {0, 0}; // 00
    int[] notFound = {100, 100};
    int count = 0;
    if (table[hash] != null) {
        HashTableList entry = table[hash];
        result[0] = hash;
        while (entry != null) {
            if (entry.value == value) {
                result[1] = count;
                // Log.i("index2", String.valueOf(count));
                break;
            } else
                count++;
            entry = entry.next;
        }
        return result;
    } else return notFound;
}
```

Illustration 5.3: Get Location of Value

(64) Each input value will get the location, the value of the input will go to
the int value. (65) value will be modulo with TABLE_SIZE to determine the draw
location. (69) If table [modulo result] is not null then table will create list. (71)
result [0] is the result to determine the position of x, derived from value modulo
with TABLE_SIZE. (72) if next list in each array is not null then count will
increase. (74) result [1] comes from the count value. (81) if the process is
complete then result is {hash, count}, result is function to get coordinate location
x, y at the time of data in draw.
5.1.4 Search and Get Value Location

(86) Each input value will get the location, the value of the input will go to
the int value. (87) value will be dimodulo with TABLE_SIZE to determine the
location of draw. (91) If table [module result] is not null then array will have list.
(97) if the data from the list is equal to value then result x is the result of modulo
and result y is count. (103) If the data from the list is not equal to the value count
will increase and (104) will go to the next entry. (106) If data is found it will
return to result with result {hash, count} or coordinate x, y. (88) if the data is not
found then the result will default to {1000,1000} which means that x will be in the
1000 and y positions at 1000. This value is unlikely to be used in the simulation
because the visualized data are limited in coordinates {12.4 }.
5.1.5 Delete Value

(112) The value that will be removed by the user will go to int value. (113) value modulo with TABLE_SIZE to determine the coordinates of the value location. (114) if table [result modulo] not null then will go to next process. (120) if the entry value is equal to value delete then check back (122) if the entry is the same as the [modulo] table then (123) the [modulo result] table will be replaced with the value after it. (125) If the entry value is not equal to the delete value then delEntry.next will become null.
5.1.6 Draw Rectangle based Array Size

(112) The value that will be removed by the user will go to int value. (113) value modulo with TABLE_SIZE to determine the coordinates of the value location. (114) if table [result modulo] not null then will go to next process. (120) if the entry value is equal to value delete then check back (122) if the entry is the same as the [modulo] table then (123) the [modulo result] table will be replaced with the value after it. (125) If the entry value is not equal to the delete value then delEntry.next will become null.

5.1.7 Draw Value with Location

Earlier from getLoc process get result {hash, count} or {x, y}. (596) hashinput_toTableY is the result of the hash or x coordinate of the getLoc process on the hashtable data structure. (597) hashinput_toTableX is the result of
coordinate count or y of the getLoc process in the hashtable data structure. (599 - 609) The above looping function for determining the distance between rectangles with the main parameters is the coordinates x, y.

5.1.8 Animated Rectangle

(292) int count = 2037 is countdown timer. (296) there are 2 variables ie countdown timer and divider. If 2037 and divider 90 then the result is 22.633 or about 2 second. (300) height is a variable to determine the position of the rectangle shift, in about 2 seconds the rectangle will shift as much as 7 pixels multiplied by the x coordinates. For example if the coordinates x = 1 then in 2 seconds the rectangle will shift as much as 7 x 1 (y coordinates) x 22,633 = 158,431. The search animation will shift from pixel 0 to pixel 158,431. Another example if the coordinates x = 2 then in 2 seconds rectangle will shift as much as 7 x 2 x 22,633 = 316,862.
5.2 Testing

Testing is done to determine whether the application has reached the desired goal and weaknesses that have not been resolved in this project so that the future can be done refinement.

![Genymotion Emulator](image1.png)

Illustration 5.8: Genymotion Emulator

Testing done on xiaomi with android version 5.1.1, samsung galaxy s7 with android version 7.1.2, emulator genymotion google galaxy nexus with android version 4.4.2 and galaxy nexus 5 with android version 6.0.0.

![Smartphone Test](image2.png)

Illustration 5.9: Smartphone Test
From the test on the device the array size, insert, search and delete functions in the process by the hashtable data structure have got the appropriate result.
The drawRect process for the insert process is neatly arranged according to the coordinates of the data structure hash table. The shortcomings that still exist in this application is the maximum number of inputan array as many as 12 columns, and list as many as 4 columns.
### Table 5.1: Testing Insert

<table>
<thead>
<tr>
<th>No.</th>
<th>Input Value</th>
<th>Size of Array</th>
<th>Modulo</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>24</td>
<td>12</td>
<td>0</td>
<td>TRUE</td>
</tr>
<tr>
<td>2.</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td>TRUE</td>
</tr>
<tr>
<td>3.</td>
<td>2</td>
<td>12</td>
<td>2</td>
<td>TRUE</td>
</tr>
<tr>
<td>4.</td>
<td>4</td>
<td>12</td>
<td>4</td>
<td>TRUE</td>
</tr>
<tr>
<td>5.</td>
<td>7</td>
<td>12</td>
<td>7</td>
<td>TRUE</td>
</tr>
<tr>
<td>6.</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>TRUE</td>
</tr>
<tr>
<td>7.</td>
<td>10</td>
<td>12</td>
<td>10</td>
<td>TRUE</td>
</tr>
<tr>
<td>8.</td>
<td>11</td>
<td>12</td>
<td>11</td>
<td>TRUE</td>
</tr>
<tr>
<td>9.</td>
<td>13</td>
<td>12</td>
<td>1</td>
<td>TRUE</td>
</tr>
<tr>
<td>10.</td>
<td>14</td>
<td>12</td>
<td>2</td>
<td>TRUE</td>
</tr>
<tr>
<td>11.</td>
<td>16</td>
<td>12</td>
<td>4</td>
<td>TRUE</td>
</tr>
<tr>
<td>12.</td>
<td>19</td>
<td>12</td>
<td>7</td>
<td>TRUE</td>
</tr>
<tr>
<td>13.</td>
<td>26</td>
<td>12</td>
<td>2</td>
<td>TRUE</td>
</tr>
<tr>
<td>14.</td>
<td>28</td>
<td>12</td>
<td>4</td>
<td>TRUE</td>
</tr>
<tr>
<td>15.</td>
<td>38</td>
<td>12</td>
<td>2</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

In the experiments conducted on 15 data, it can be concluded that the data insert process runs well on the data structure and visualization can be displayed precisely according to the coordinates of each value.
The process of shifting the searching animation has not been maximal, because it is determined by the countdown timer sometimes the rectangle shift goes beyond the coordinates and sometimes has not reached the coordinates, it happens because of the performance of the device. On android devices that have low specifications this will often happen.