VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"JNANA SANGAMA", BELAGAVI, KARNATAKA– 560018,

Project Report
On

"DYNAMIC SORTING ALGORITHM VISUALIZER"

Submitted in partial fulfillment of the requirement of
Project Work [18CSL67]
In
Computer Graphics Laboratory

By,

ABUBAKAR SIDDIQ	**[4MU18CS001]**
ADEEB AHMED	**[4MU18CS002]**
ADITYA V DHAPTE	**[4MU18CS003]**

Under the guidance of
Prof. Arpitha P M
Associate Professor
Department of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MYSURU ROYAL INSTITUTE OF TECHNOLOGY
Lakshmipura Road, Palahally Post, SR Patna, Mandya – 571606

Academic Year: 2020 – 2021

MYSURU ROYAL INSTITUTE OF TECHNOLOGY

Lakshmipura Road, Palahally Post, SR Patna, Mandya – 571606

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the mini project work entitled **"Dynamic Sorting Algorithm Visualizer"** is a bonafide work carried out by **ABUBAKAR SIDDIQ (4MU18CS001), ADEEB AHMED (4MU18CS002)** & **ADITYA V DHAPTE (4MU18CS003)** in partial fulfilment for the Computer Graphics Laboratory with mini project (18CSL67) prescribed by the Visvesvaraya Technological University, Belagavi during the year 2020-2021 for the sixth semester B.E. Computer Science and Engineering. The project report has been approved as it satisfies the academic requirements with respect to the mini project work prescribed for the sixth semester **Computer Graphics Laboratory** with Mini Project.

Signature of Guide	Signature of HOD
Prof. Arpitha P M	**Prof. Soumya B**
Assistant Professor	Assistant Professor
Dept. of CS&E	Dept. of CS&E
MRIT, Mandya	MRIT, Mandya

Name of the examiners — Signature with date

1.

2.

ACKNOWLEDGEMENT

We sincerely owe our gratitude to all the persons who helped and guided us in completing this mini project.

We are thankful to **Dr. Suresh Chandra, Principal, MRIT**, **Mandya**, for having supported us in our academic endeavours.

We are extremely thankful to **Soumya B, Head Department of Computer Science and Engineering,** for her valuable support and her timely inquiries into the progress of the work.

We express our earnest gratitude towards our guide **Asst. Prof. Arpitha P M, Department of Computer Science and Engineering,** who helped us in getting things done and was always inspirational.

We are obliged to all **teaching and non-teaching staff members** of **Department of Computer Science and Engineering, Mandya,** for the valuable information provided by them in their respective fields. We are grateful for their co-operation during the period of our project.

Lastly we thank almighty, our parents and friends for their constant encouragement without which this project would not be possible.

ABUBAKAR SIDDIQ [4MU18CS001]
ADEEB AHMED [4MU18CS002]
ADITYA V DHAPTE [4MU18CS003]

ABSTRACT

Sorting problems is to rearrange the items of the given list in Ascending Order. The underlying concept of sorting algorithm includes the comparisons, swapping of elements, and assignments. In DSAV (Dynamic Sorting Algorithm Visualizer), we are Visualizing the Bubble Sort Algorithm. In this algorithm, comparisons starts from the first two elements of the array and finding the largest item and move (or bubble) it to the top. With each subsequent iteration, find then ext largest item and bubble it up towards the top of the array. Our DSA Visualizer depicts the swapping of elements by swapping the circles (which are the items of the array in our case). This swapping process occurs for at the max of n iterations. We formulate and study a new computational model for dynamic data. In this model, the data changes gradually and the goal of an algorithm is to compute the solution to some problem on the data at each time step, under the constraint that it only has limited access to the data each time. As the data is constantly changing and the algorithm might be unaware of these changes, it cannot be expected to always output the exact right solution; we are interested in algorithms that guarantee to output an approximate solution. In particular, we focus on the fundamental problems of sorting and selection, where the true ordering of the elements changes slowly. We provide algorithms with performance close to the optimal in expectation and with high probability. Thus at the end of nth iteration, we can conclude that the array of elements are sorted in the ascending order which is the desired output of the DSAV.

TABLE OF CONTENTS

CONTENTS **PAGE NO.**

Chapter 1

INTRODUCTION

1.1 COMPUTER GRAPHICS

Computer graphics is concerned with all aspects of producing pictures or images using a computer. The field has begun humbly almost 50 years ago, with a display of few lines on a cathode ray tube, now we can create images by computer that are indistinguishable from photographs of real objects. We routinely train pilots with simulated airplanes, generating graphical displays of virtual environment of real time. Feature length movies made entirely by computer have been successful, both critically and financially, Massive multiplayer games can involve tens of thousands of concurrent participants.

Application of Computer Graphics:

The development of computer graphics has been driven both by needs of the user community and by advancement in technology. The applications of computer graphics of computer graphics are many and varied, we can divide them into four categories:

1. Display of information
2. Design
3. Simulation and Animation
4. User interface

1.2 OpenGL

OpenGL was originally developed by Silicon Graphics, Inc. (SGI) as a multi-purpose, platform-independent graphics API. Since 1992, the development of OpenGL has-been overseen by the OpenGL Architecture Review Board (ARB), which is made up of major graphics vendors and other industry leaders, currently consisting of 3DLabs, ATI, Dell, Evans & Sutherland, Hewlett-Packard, IBM, Intel, Matrox, NVIDIA, SGI, Sun Microsystems, and Silicon Graphics. The Role of ARB is to establish and maintain the OpenGL specifications, which dictates which features must be included when one is developing an OpenGL distribution. OpenGL is the premier environment for developing portable, interactive2D and 3D graphics applications. OpenGL intentionally provides only low-level rendering routines, allowing the programmer

a great deal of control and flexibility. The provided routines can easily be used to build high-level rendering and modelling libraries, and in fact, the OpenGL Utility Library (GLU), which is included in most OpenGL distributions, does exactly that. Note that OpenGL is just a graphics library; Unlike DirectX, it does not include support for sound, input, networking, or anything else not directly related to graphics.

OpenGL is a collection of several hundred functions providing access to all of the features offered by our graphics hardware. Internally, it acts as a state machine – a collection of states that tell OpenGL what to do and that are changed in a very well-defined manner. Using API, we can set various aspects of the state machine, including such things as the current colour, lighting, blending et al. When rendering, everything drawn is affected by the current settings of the state machine. At the core of OpenGL is the rendering pipeline.

There are many libraries available that build upon and around OpenGL to add support and functionality beyond the low-level rendering support that it excels at. Most important amongst them are GLU, GLUT and SDL.GLU uses only GL functions but contains code for creating common objects and simplifying viewing. GLUT provides support for any form of functionality related to windowing, menus or input. SDL stands for Simple Direct Media Layer is a cross-platform multimedia library, including support for audio, input, 2D graphics.

1.3 Project Description

DSA (DYNAMIC SORTING ALGORITHM):

Dynamic Sorting Algorithm Visualizer basically implements one of the best sorting algorithms, (i.e) Bubble Sort. This Visualizer makes use of the fact that OpenGL provides the low-level rendering routines allowing the programmer a great deal of control and flexibility. DSAV also make use of the prominent library of OpenGL (i.e.) GLUT. DSAV is broadly classified into 3 major steps. They are as follows:

1. Initial Representation:

In the initial representation phase, DSAV generates a random sequence of numbers, which marks the beginning of this phase. This random sequence of numbers will be the input to our sorting algorithm. Based on this input we draw the scaled circles using

GL_TRIANGLE_FAN of OpenGL with suitable radius and place them on the main window for sorting. Each circle is separated from each other by a constant distance.

2. Swapping Process:

The Basic step behind a sorting algorithm is to swap two key elements. Here in our case is to swap the circles. Key elements are the circles that are undergoing the swapping process in the current iteration. The basic step of algorithm is comparison, through which we get the key elements for swapping. Once the key circles are known, we note down the X- coordinate of both the circles. Also, we initialize a flag by setting it to TRUE (1). Swapping circles is achieved by the following process:

- Decrease the centre of circle in right till its centre is greater than left circle.
- Increase the centre of circle in left till its centre is less than the right circle.

3. Final Rendering:

Once the all the iterations are completed, we get the sorted array of elements. Thus, DSA Visualizer depicts this phase by presenting the circles in the increasing order of radius. Then we provide the menu system to user, which compromises of Randomizing, Sorting and Quit states. Based on the input of user the respective actions are carried out. Randomize: To generate the new random sequence. Sort: To employ the bubble sort algorithm to the newly generated sequence. Quit: To quit from the DSAV.

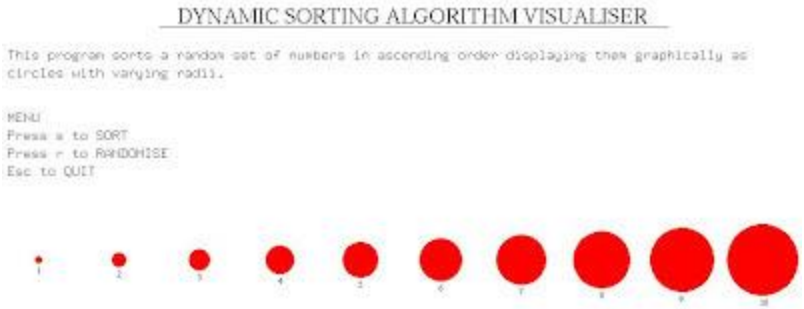
Dynamic Sorting Algorithm

```
/* Bubble sort code */
#include <stdio.h>
int main()
{
  int array[100], n, c, d, swap;
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter %d integers\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);
  for (c = 0 ; c < n – 1; c++)
  {
    for (d = 0 ; d < n – c – 1; d++)
    {
      if (array[d] > array[d+1]) /* For decreasing order use '<' instead of '>' */
      {
        swap       = array[d];
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
  printf("Sorted list in ascending order:\n");
  for (c = 0; c < n; c++)
     printf("%d\n", array[c]);
  return 0;
}
```

Types of Sorting Algorithms:

- Quick Sort.
- Bubble Sort.
- Merge Sort.
- Insertion Sort.
- Selection Sort.
- Heap Sort.
- Radix Sort.
- Bucket Sort.

DYNAMIC SORTING ALGORITHM VISUALISER:



Sorting can be defined as arranging list, number, or any times in a systematic way in a certain order of sequence having comparable properties. In simple word Sorting is the process to rearrange the items of a given list in Ascending Order/Descending Order.

Previously in our Computer Graphics Project blog we have discussed about Bucket sort program a sorting program. In that Computer Graphics Project program, the user uses to provide two input arrays to store and show the sorting process and result, while in this project the sorting is show dynamically with Circles representing the input list to sort.

The underlying concept of sorting algorithm includes the comparisons, swapping of elements, and assignments. The DSAV (Dynamic Sorting Algorithm Visualizer), Visualizes the Bubble Sort Algorithm. In this algorithm, comparisons start from the first two elements of the array and finding the largest item and move (or bubble) it to the top. With each subsequent iteration, find the next largest item and bubble it up towards the top of the array.

This DSA Visualizer depicts the swapping of elements by swapping the circles (which are the items of the array in our case), for each item different radii of circle are generated according to the value of the item. This swapping process occurs for at the max of n iterations.

Thus, at the *end of nth iteration,* the array of elements is sorted in the ascending order which is the desired output of the DSAV. Below is the logic or say short algorithm of the Dynamic Sorting Algorithm Visualizer Computer Graphics Project.

Logical flow of DSA:

If not in the process of swapping of 2 circles, then only get 2 new circles to swap.

While the counter_i < 10

While counter_j <9

If the a[counter_j] > a[counter_j]

Swap 2 circles

Once exchanged goto swap

Increment counter_j

Increment counter_i

Swap:

Print which circles are getting swapped.

Call swap circles function again with counter_j and counter_j+1 values.

Mark the end of the function sort.

Chapter 2

AIMS & OBJECTIVE OF THE PROJECT

2.1 AIM

In computer graphics sort is popular for its capability to detect an error in almost-sorted arrays and fix it with just linear complexity. Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.

- if somebody needs some data, one can find it easily in the quickest time.
- Memory consumed by the data is minimum.
- Memory fragmentation is minimum when data is manipulated(add/remove/update).

2.1 OBJECTIVE

The goal is to design an algorithm that, at any point in time, tracks the top few elements of the underlying total order. Since sorting can often reduce the complexity of a problem, it is an important algorithm in Computer Science. A sorting algorithm is an algorithm that put elements of a list in a certain order. The most frequently used orders are numerical order and lexicographical order. Computer graphics applications require visibility sorting for correctly rendering transparent objects and efficiently exploiting acceleration features.

Sorting algorithms can be divided into two categories: data-driven ones and data-independent ones. More formally, the output of any sorting algorithm must satisfy two conditions:

1. The output is in nondecreasing order (each element is no smaller than the previous element according to the desired total order);
2. The output is a permutation (a reordering, yet retaining all of the original elements) of the input.

Chapter 3

REQUIREMENT ANALYSIS

3.1 Hardware Requirement

Processor:	1.5 GHz processor or faster, Intel Pentium
RAM:	256 MB
Hard Disk:	50 GB free disk space
Backup Media:	Pen drive/CD-ROM

3.2 Software Requirement

OS:	WINDOWS XP (MIN) Windows XP
Languages/Tools Used:	OpenGL, C, C++, Object Oriented Concepts.
Editor:	Notepad++
Platform / IDE:	Code blocks

Chapter 4

SYSTEM DESIGN

4.1 FLOW CHART

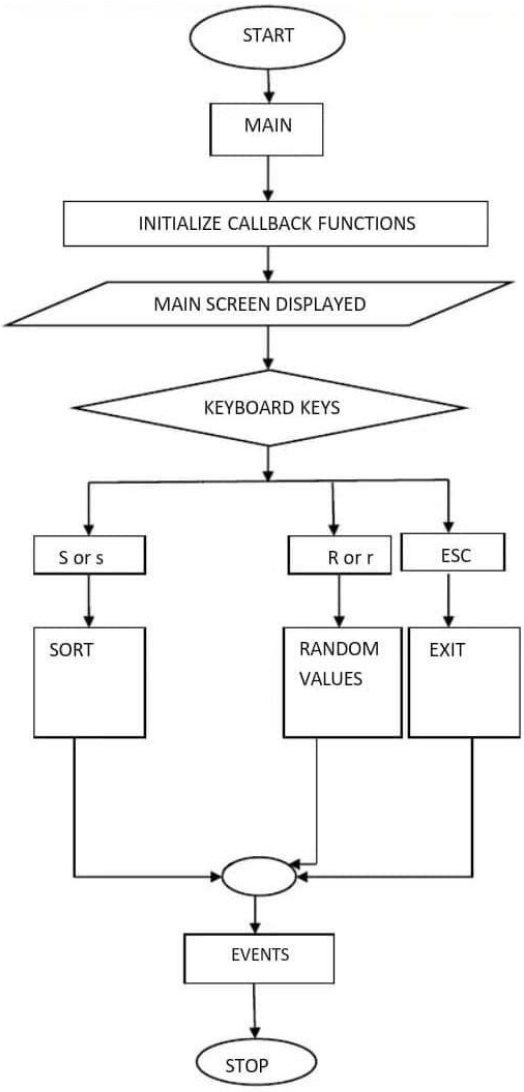


Fig: Flow Chart of DSAV

Description of Flow Chart :

Graphical view of sorting algorithm. glutKeyboardFunc is used for keyboard functions. Key 's' used to sort. Key 'c' used to select the sort algorithm. Key 'r' is used to randomize. Esc key is used to exit.

Sorting is the process to rearrange the items of a given list in ascending order/ descending order.

Different types of sorting used:
1) Bubble Sort,
2) Ripple Sort,
3) Insertion Sort.

Bubble Sort: Simple sorting algorithm that repeatedly steps through the list, compares adjacent pairs and swaps them if they are in wrong order. The pass through the list is repeated until the list is sorted.

Worst Complexity: n^2
Best Complexity: n

Ripple Sort: Also known as cocktail sort, is a variation of bubble sort that is both a stable sorting algorithm and a comparison sort.
Worst Complexity: n°2
Best Complexity: n
Insertion Sort: simple sorting algorithm that builds the final sorted array one item at a time. Worst Complexity: n^2. Best Complexity: n

4.2 Description of OpenGL functions

Glut library functions used are as follows:

- **main ():** Execution of any program always starts with main function irrespective of where it is written in a program.

- **glutInit (&argc, char** argv):** This function will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.

- **glutInitDisplayMode(unsigned int mode):** The initial display mode is used when creating top-level windows, sub windows and overlays to determine the OpenGL display mode for the to be created window or overlay.

- **glutCreateWindow(char *title):** This function creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name.

- **glutInitWindowSize(int width, int height):** This function allows you to request initial dimensions for future windows.

- **glutInitWindowPosition(int x, int y):** This function allows us to to request an initial position for future windows.

- **GlutCreateMenu(void (*func)(int value)):** This function defines the call back that has to be called when a menu item is selected. This function has one parameter, the value. This function returns int, and is the menu identifier.

- **glutAttachMenu(int button):** This function attaches the current menu to a certain (mouse) event, one can let a menu listen to a specified mouse button, button can be one of the following: GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON and GLUT_RIGHT_BUTTON.

- **glutMainLoop():** This function enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any call-backs that have been registered.

- **glBegin(enum) and glEnd():** These functions delimit the vertices that define a primitive or a group of like primitives. glBegin accepts a single argument that specifies in which often ways the vertices are interpreted.

- **glColor3f(R, G,B):** This function is used to pick a colour of specified R, G, B value.

- **glVertex2f(x,y):** This function is used to draw a vertex at a location x,y.

- **glClearColor(R,G,B,A):** This function specifies the red, green, blue and alpha values used by glClear to clear colour buffers. Values specified by glClearColor are clamped to the range 0 to

- **gluOrtho2D(Gldouble left,Gldouble right,Gldouble bottom,Gldouble top):** This function sets up a two-dimensional orthographic viewing region. This is equivalent to calling glOrtho with near = -1 and far = 1.

4.3 Description of user-defined functions

Various user-defined functions used are:

- **void quit ():** This function is used to terminate the program.
- **void texture ():** Used to initialize pointer variable for successful code execution.
- **void display(void):** It is a user-defined display function.
- **void myinit():** This function is used to set the OpenGL state variables dealing with viewing and attributes.

Chapter 5

IMPLEMENTAION

5.1 Circle_draw() code

```
void circle_draw(circle c)
{
        float i;
        glBegin(GL_TRIANGLE_FAN);
        glVertex2f(c.x, c.y);
        for (i=0;i<360;i+=1)
                glVertex2f(c.x + sin(i) * c.r, c.y + cos(i) * c.r);
        glEnd();
        int x = c.x-2;
        int y = c.y-(c.r+10);
        int rad = c.r / 4;
        char r[3] = "";
        int_str(rad,r);
        glColor3f(0.0,0.0,0.0);
        bitmap_output(x, y, r, GLUT_BITMAP_TIMES_ROMAN_10);
}
```

5.2 Swap_circle() code

```
void swap_circles(circle *cc1,circle *cc2)
{
        if (swapping == 0)
        {
                initial_x1 = cc1->x;
                initial_x2 = cc2->x;
                        swapping = 1;
                printf("%f - %f\n",cc1->r,cc2->r);
        }
if (initial_x1 <= cc2->x)
                cc2->x -= 1.0;
        if (initial_x2 >= cc1->x)
                cc1->x += 1.0;
        printf("one %f - %f\n",initial_x1,cc2->x);
        printf("two %f - %f\n",initial_x2,cc1->x);
        if (abs(initial_x1-cc2->x) < 0.3 && abs(initial_x2-cc1->x) < 0.3)
        {
                swapping = 0;
                int temp = cc1->x;
                cc1->x = cc2->x;
                cc2->x = temp;
                temp = cc1->y;
```

```
                cc1->y = cc2->y;

                cc2->y = temp;

                temp = cc1->r;

                cc1->r = cc2->r;

                cc2->r = temp;

        }

}
```

5.3 String() code

```
void int_str(int rad,char r[])

{

        switch(rad)

        {

                case 1 : strcpy(r, "1"); break;

                case 2 : strcpy(r, "2"); break;

                case 3 : strcpy(r, "3"); break;

                case 4 : strcpy(r, "4"); break;

                case 5 : strcpy(r, "5"); break;

                case 6 : strcpy(r, "6"); break;

                case 7 : strcpy(r, "7"); break;

                case 8 : strcpy(r, "8"); break;

                case 9 : strcpy(r, "9"); break;

                case 10 : strcpy(r, "10"); break;
```

```
        }

}
```

5.4 Keyboard() code

```
void keyboard (unsigned char key, int x, int y)
{
        if(key==13)
                k=1;
        if (k==1)
        {
        switch (key)
        {
                case 27 : exit (0); //27 is the ascii code for the ESC key
                case 's' : sorting = 1; break;
                case 'r' : initialise(); break;
        }
        }
}
```

5.5 Init() code

```
void init(void)
{
                glClearColor(1.0,1.0,1.0,0.0);
                glMatrixMode(GL_PROJECTION);
                gluOrtho2D(0.0,900.0,0.0,600.0);
}
```

Chapter 6

SNAPSHOTS

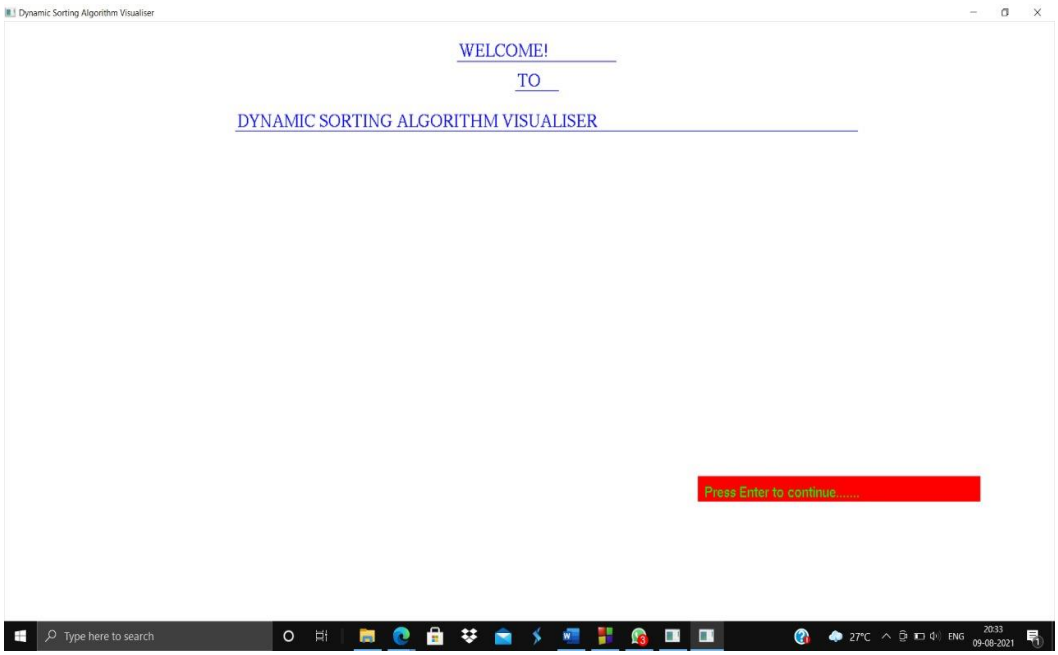


Fig 6.1 : Main Screen of DSAV

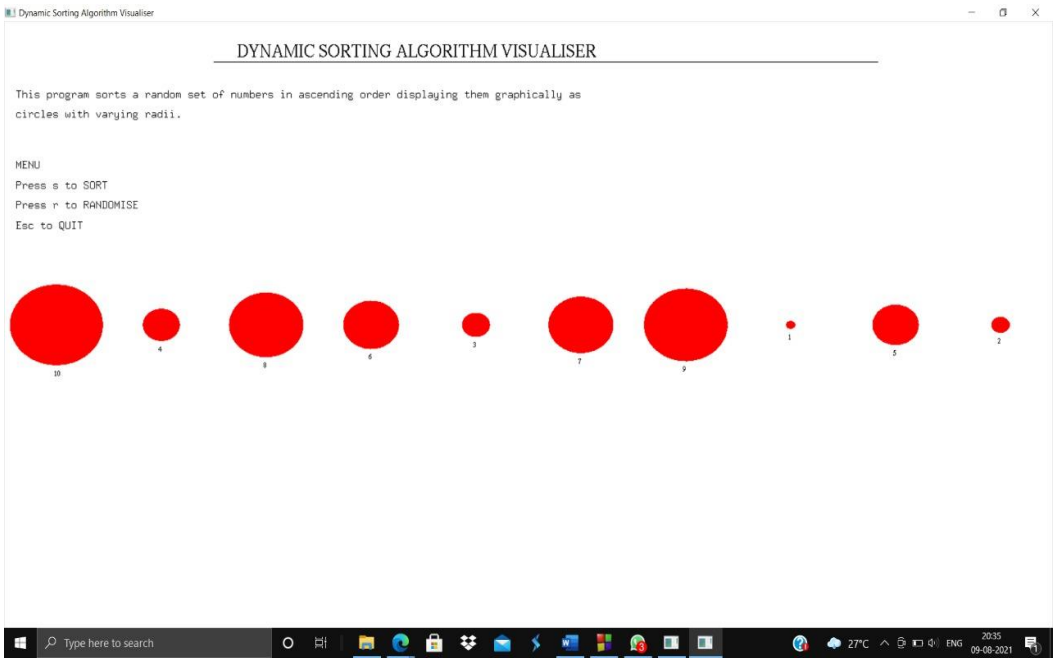


Fig 6.2 : Distorted elements set on the screen of DSAV

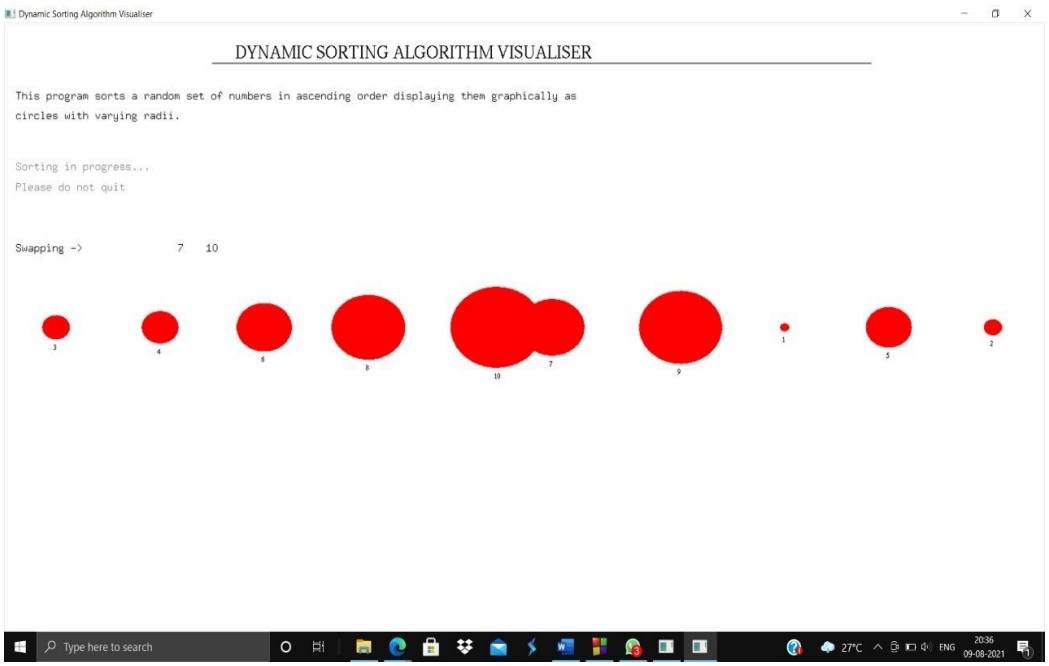


Fig 6.3 : Swapping of circle in process

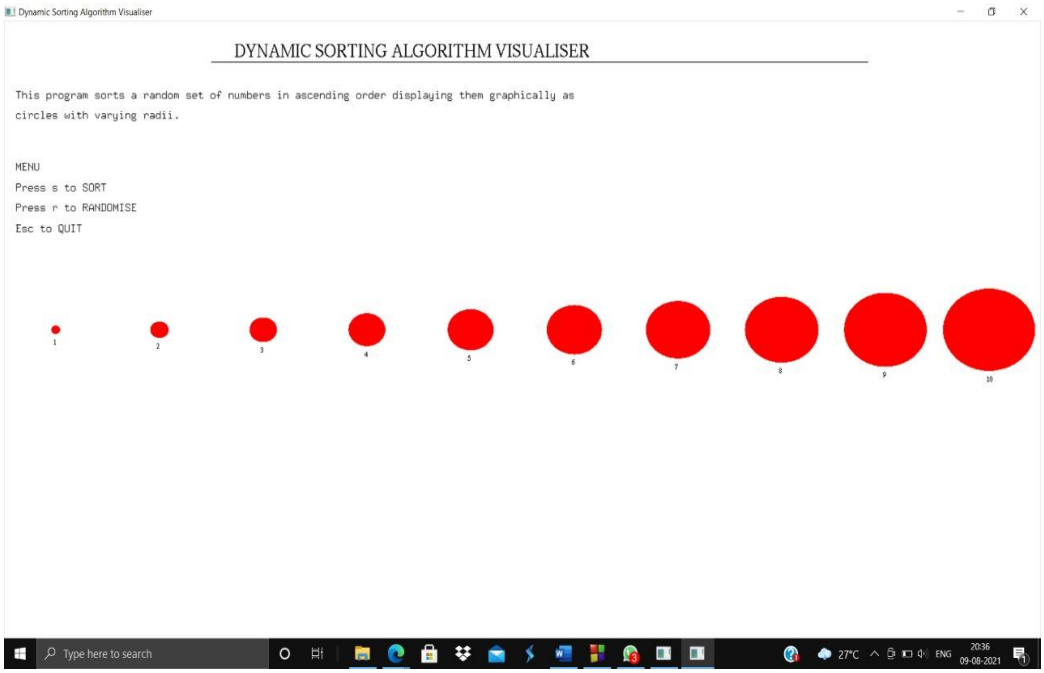


Fig 6.4 : Sorted elements set of DSAV

CONCLUSION

This graphic application provides a easier tool in technical academic teaching to logically visualize the concept of DSA. Where the users can view the DYNAMIC SORTING ALGORITHM with a smarter approach of both learning & practicing hands-on. Visual aids are said to be more effective than verbal explanations alone. This way we have tested our application to provide a smart solution for better understanding of the Dynamic Sorting Algorithm.

Bibliography

Text Books:

Interactive Computer Graphics
A Top-Down Approach Using OpenGL
- Edward Angel.

Weblinks:

1. http://www.videotutorialsrock.com/
2. http://www.youtube.com/
3. http://nehe.gamedev.net/
4. https://www.openglprojects.in/