CHAPTER 1

INTRODUCTION

1.1 Introduction to Computer Graphics

COMPUTER GRAPHICS is concerned with all aspects of producing pictures or images using a computer. The field began humbly almost 50 years ago, with the display of a few lines on a cathode-ray tube (CRT); 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 a virtual environment in 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.

VISUALIZATION is any technique for creating images, diagrams or animations to communicate a message.

1.2 Image Types

➢ 2D computer graphics:

2D computer graphics are the computer-based generation of digital images mostly from two-dimensional models, such as 2D geometric models, text, and digital images, and by techniques specific to them.2D computer graphics are mainly used in applications that were originally developed upon traditional printing and drawing technologies, such as typography, cartography, technical drawing, and advertising. Two-dimensional models are preferred, because they give more direct control of the image than 3D computer graphics, whose approach is more akin to photography than to typography.

There are two approaches to 2D graphics: vector and raster graphics.

- Pixel art: Pixel art is a form of digital art, created through the use of raster graphics software, where images are edited on the pixel level.
- Vector graphics: Vector graphics formats are complementary to raster graphics, which is the representation of images as an array of pixels, as it is typically used for the representation of photographic images.

➢ 3D computer graphics:

With the birth of the workstation computers (like LISP machines, paint box computers and Silicon Graphics workstations) came the 3D computer graphics.3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes0 of performing calculations and rendering 2D images.

1.3 Applications of Computer Graphics

Some of the applications of computer graphics are listed below:

- Computational biology
- Computational physics
- Computer-aided design
- Computer simulation
- Digital art
- Education
- Graphic design
- Video Games
- Virtual reality
- Web design

1.4 Introduction to OpenGL

As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

Basic OpenGL Operation

The figure shown below gives an abstract, high-level block diagram of how OpenGL processes data. In the diagram, commands enter from the left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.

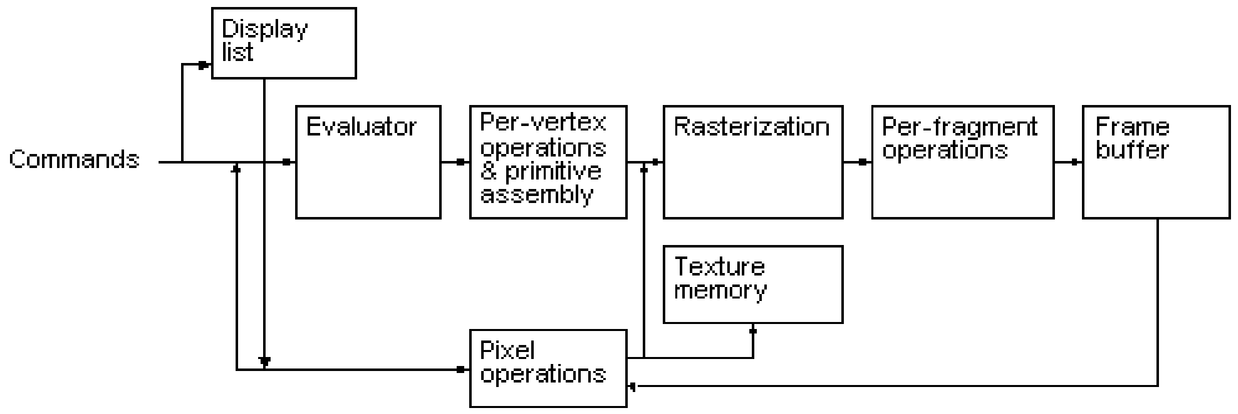


Fig 1.1: Basic OpenGL Operation.

1.5 Introduction to GLUT

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works on both Win32 PCs and X11 workstations.

GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits like Motif. GLUT is simple, easy, and small. My intent is to keep GLUT that way.

The GLUT library supports the following functionality:

- Multiple windows for OpenGL rendering.
- Call back driven event processing.
- An `idle' routine and timers.
- Utility routines to generate various solid and wire frame objects.
- Support for bitmap and stroke fonts.
- Miscellaneous window management functions.

1.6 Overview of the project

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications. This project, named DYNAMIC SORTING ALOGRITHM VISUALIZER uses OpenGL software interface and develops visuals on sorting processes. This project uses the techniques like basic geometric primitives, color functions, bit character functions and many other functions for displaying, translation and scaling of visual bars. This project emphasizes on using various functions that are available in openGL like transpose, polygon and many more.

1.7 Aim of the project

The aim of this project is to create a visualization of sorting techniques. The program sorts the randomized numbers based on the sorting technique selected by user and displays the sorting process on window.

CHAPTER 2

SOFTWARE REQUIREMENT SPECIFICATION

2.1 Software Requirements

- Operating System : Windows 98/XP or Higher
- Programming Language : C,C++
- CodeBlocks 13.17 or higher: This Software package contains the required libraries to support C++ language.
- Toolkit : GLUT Toolkit

2.2 Hardware Requirements

This package has been developed on:

- Processor : Pentium Processor
- Processor Speed : 333 MHz
- RAM : 32 MB or Higher
- Graphics Card : 512MB
- Monitor : Color
- Keyboard : Low Profile, Dispatch able Type
- I/O Parts : Monitor

CHAPTER 3

DESIGN

3.1 Initialization

Initialize the interaction with the windows. Initialize the display mode, double buffer and depth buffer. Initialize the various callback functions for drawing and redrawing arrows and target blocks, for keyboard interface. Initialize the window position and size and create the window to display the output.

3.2 Flow of control

The flow of control in the flowchart is with respect to the Dynamic Sorting Algorithm Visualizer. For any program flowchart is compulsory to understand the program. We consider the flowchart for the program for which the flow starts from the start and proceeds to the main function after which it comes to the initialization of callback functions and further it proceeds to the Splash Screen where we have to press enter key to start the menu page, next user has to choose from many options to perform desired operation using keyboard keys. Finally the flow comes to quit which is the end of the flowchart.

3.3 Flowchart

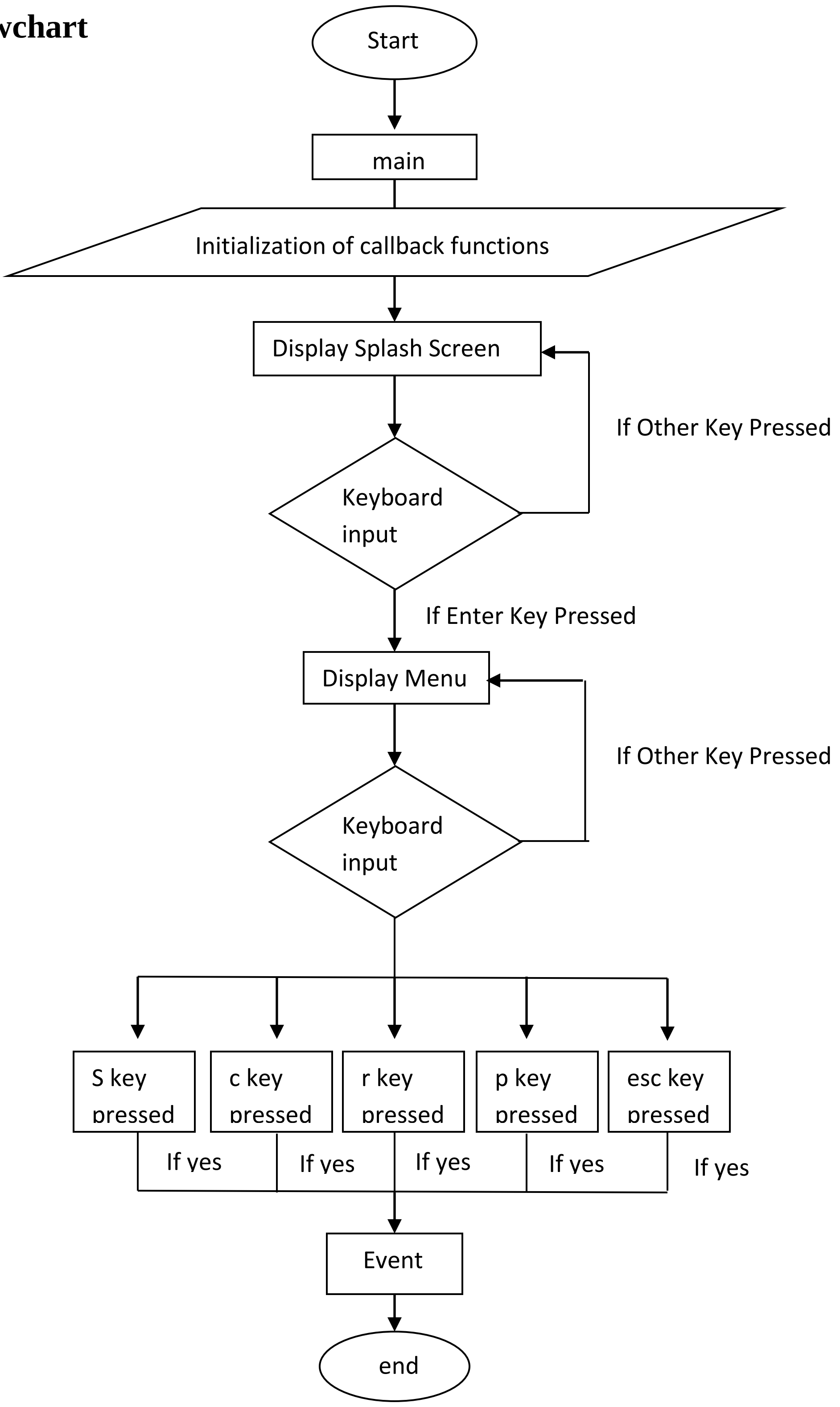


Fig 3.1: Flowchart for Dynamic Sorting Algorithm Visualizer program.

CHAPTER 4

IMPLEMENTATION

4.1 OpenGL Functions

The various functions used in implementing this project are:

- **glutInit(&argc,argv)**
    - This function is used to initialize glut.

- **glutInitDisplayMode()**
    - This function is used to initialize a display mode for the screen. It can choose one of the following constant values as it's parameters:
        - GLUT_SINGLE
        - GLUT_RGB
        - GLUT_DOUBLE

- **glutInitWindowPosition()**
    - This function is used to set the position of the top left corner of the window. It takes in 2 integer values as input parameters which are the coordinates specified for the position for the top left corner of the window.

- **glutInitWindowSize()**
    - This function is used to specify the size of the created window, inside which the scene will be generated. It takes in two integer parameters, the width and height of the window.

- **glutCreateWindow()**
    - This function creates the window with the specified dimensions and position and assigns the string parameter passed to it as the name of the window.

- **glutMainLoop()**
    - This function is called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary, any callbacks required for the program.
- **glBegin()**
    - This function is used in drawing the basic primitives like lines, points, polygons and quads. It takes in a constant value which specifies the primitive to be drawn. Some of the parameters it can have are:
        - GL_POLYGON
        - GL_LINES
        - GL_LINE_LOOP

        - GL_QUADS
        - GL_QUAD_STRIP

- **glClear()**
    - This function is used to clear the contents of the buffer passed as the parameter to this function, onto the screen.
    - It takes a constant value as a parameter like:
        - GL_CLEAR_BUFFER_BIT
        - GL_DEPTH_BUFFER_BIT

- **glClearColor()**
    - This function takes in four floating values:
        - Red
        - Green
        - Blue
        - Alpha
    - Based on RGB values, the color is decided. The Alpha value gives the degree of transparency of the color.
    - The values for RGBA will range from 0.0 to 1.0

- **glEnd()**
    - This function works with the glBegin() function. It marks the end of all the vertices to be used for drawing the primitive.
- **glVertex2x()**
    - This function is used to take in the coordinates of a vertex in a geometric primitive. For integer values, the placeholder ‘x’ is ‘i’ while for floating values, ‘x’ is’d’.
- **glutTimerFunc (unsigned int msec, void (*func)(int value), value)**
    - This function registers the timer callback func to be triggered in at least msec milliseconds. The value parameter to the timer callback will be the value parameter to glutTimerFunc.
- **glutBitmapCharacter(void *font, int character)**
    - This function renders the characters in the named bitmap font. Some of the fonts available are:
        - GLUT_BITMAP_8_BY_13
        - GLUT_BITMAP_TIMES_ROMAN_10
        - GLUT_BITMAP_HELVETICA_18

CHAPTER 5

TESTING

Test Cases

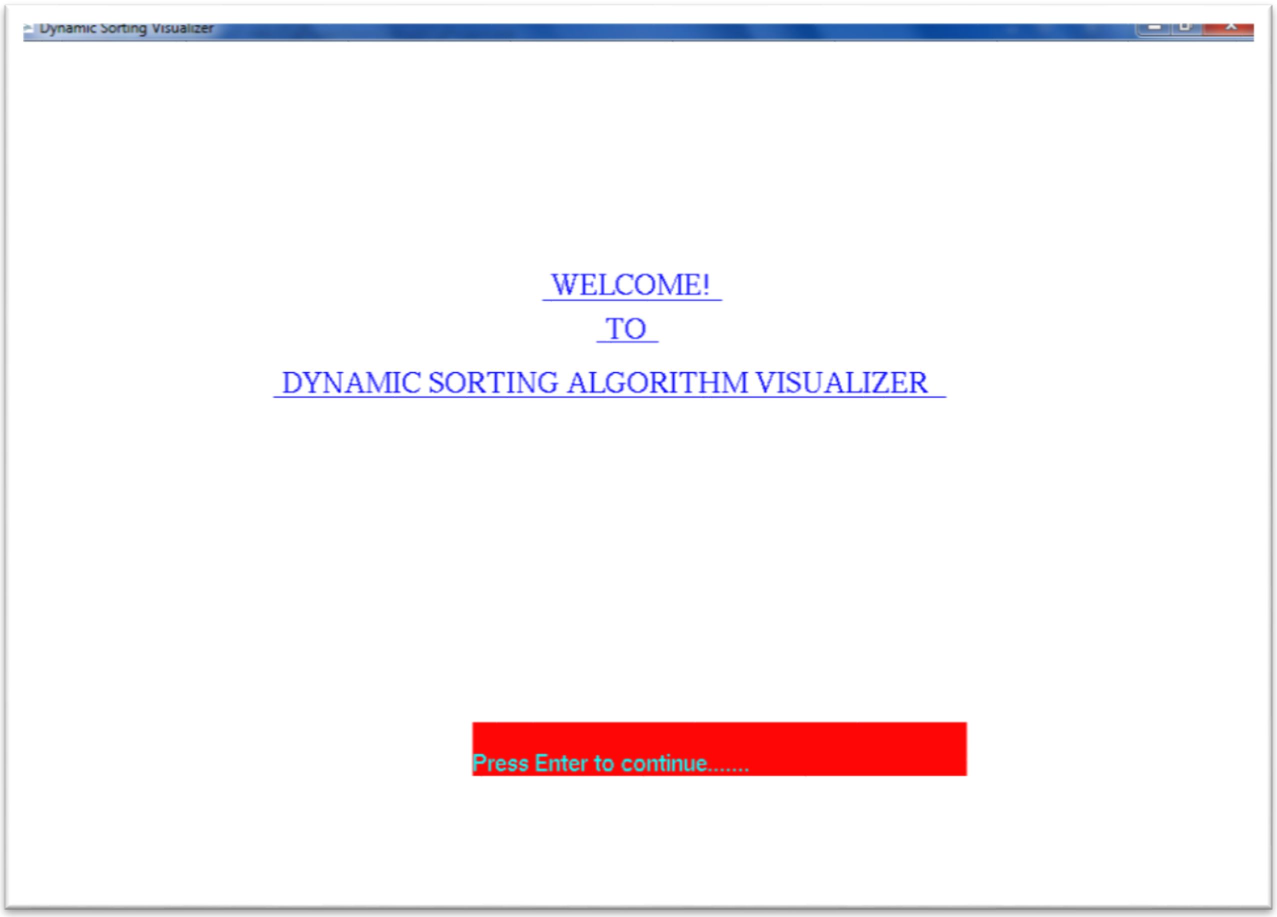
Insert for Test Case	Expected Output	Observed Output	Remarks
s KEY	It should start the sorting of the number represented as bars of varying height using selected algorithm.	It sorts the number using sorting algorithm selected.	Pass.
c KEY	It should display different sorting algorithms available	It displays different sorting algorithms.	Pass.
r KEY	It should display new randomized numbers in the form of bars of varying height.	It randomizes and displays the numbers in bars of varying height.	Pass
p KEY	It should pause the sorting process when pressed.	It stops the sorting process.	Pass
ESC KEY	It should close the program.	The program closes.	Pass.

Fig 5.1: Testing

CHAPTER 6

RESULTS

The program starts with a splash screen which greets the user. The Screen displays the title of our project. The screen prompts the user to press Enter key to continue to the next screen. This screen is used to enhance the look and feel of the application to make it visually appealing.



WELCOME!
TO
DYNAMIC SORTING ALGORITHM VISUALIZER

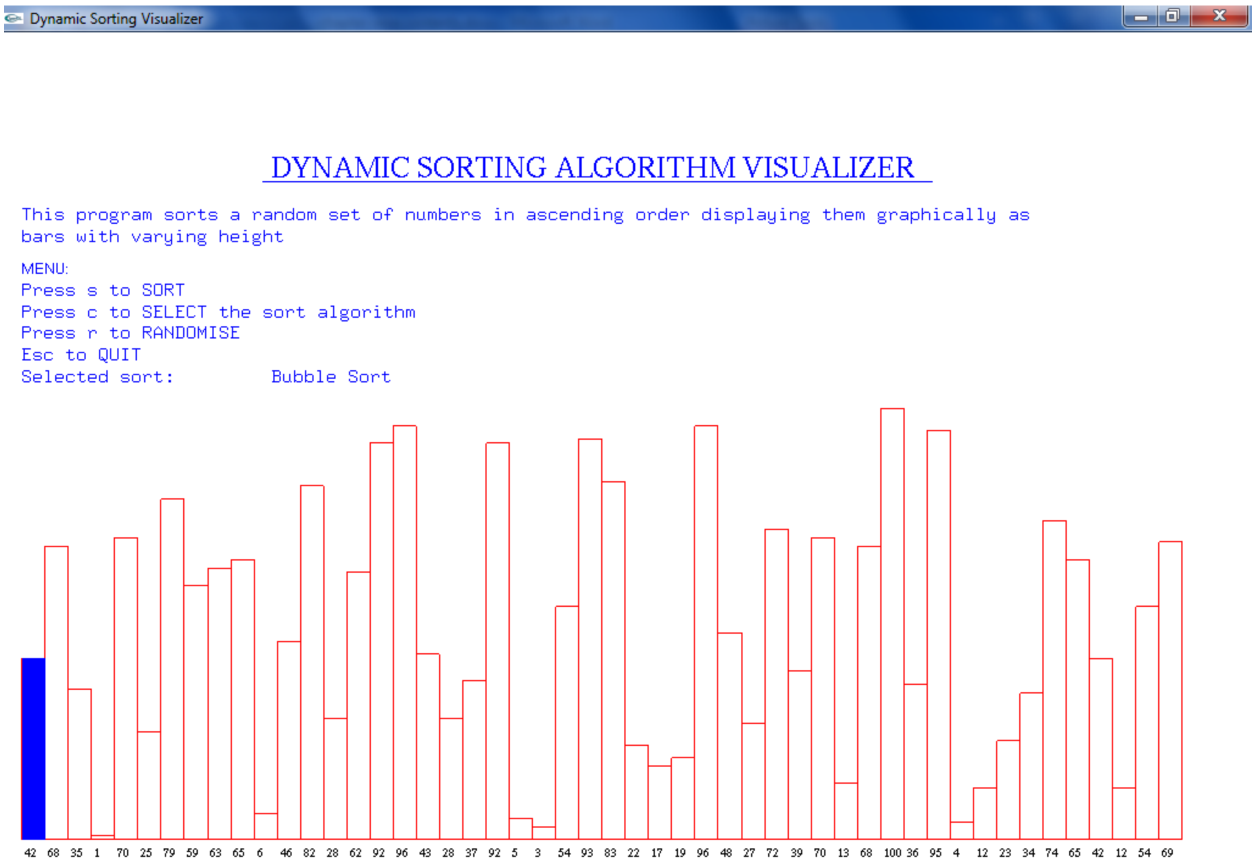
Press Enter to continue.......

Fig 6.1: Display of the Splash Screen

This is the first scene which appears when the program is executed. The display prompts the user to press Enter key. The content is displayed with the help of the following function:

1. front(): This function displays the Greetings to user and also displays the name of the program.

2. bitmap_output(): This is a user defined function which helps to display the text on screen at desired position.



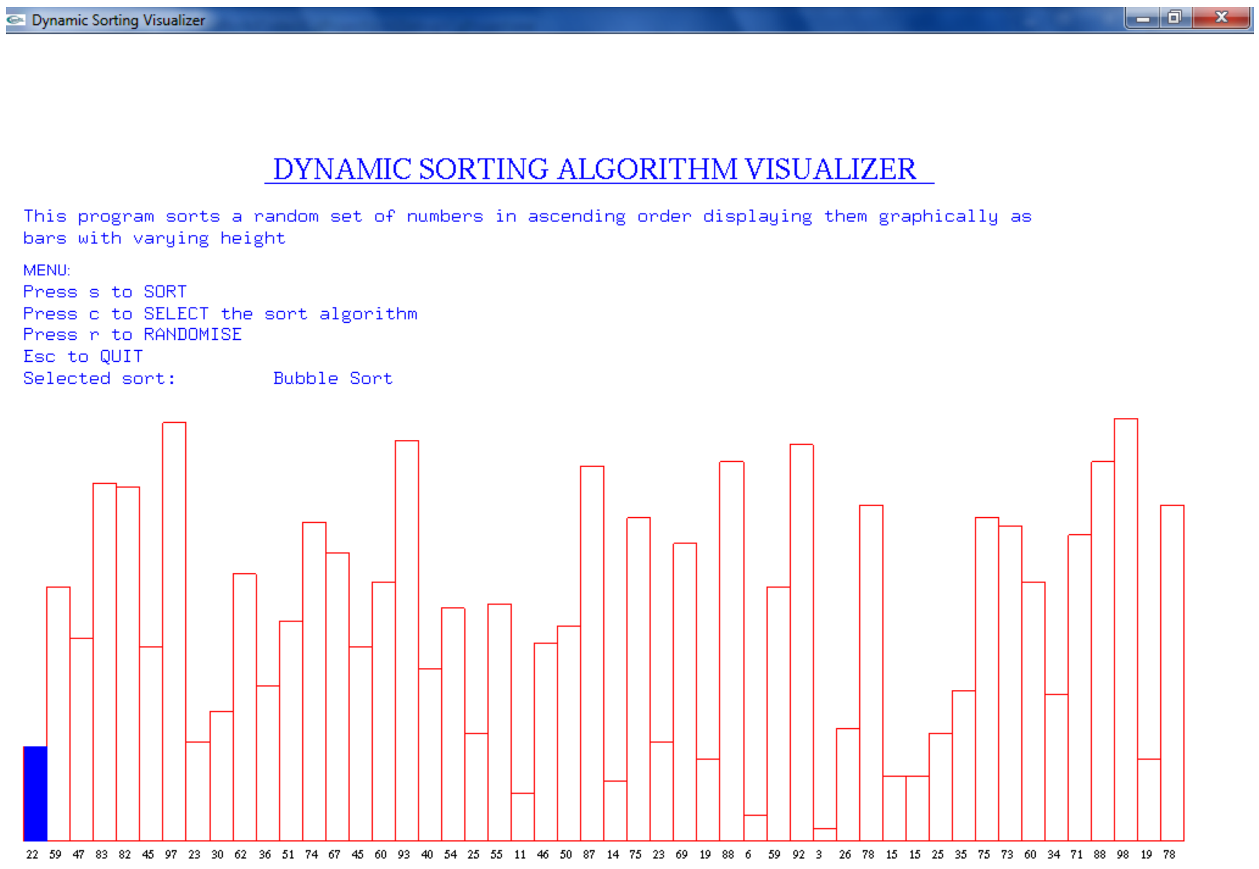
DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:          Bubble Sort
```

Fig 6.2: Display of the Menu Screen

This is the screen which appears when the user presses Enter key on Splash Screen. The action menu is displayed with the help of the following functions

1. display_text(): This function displays the whole menu and also displays the randomized numbers in the form of bars of varying height.
2. bitmap_output(): This is a user defined function which helps to display the text on screen at desired position.
3. int_str(): This is a user defined function which converts the character value to integer value.

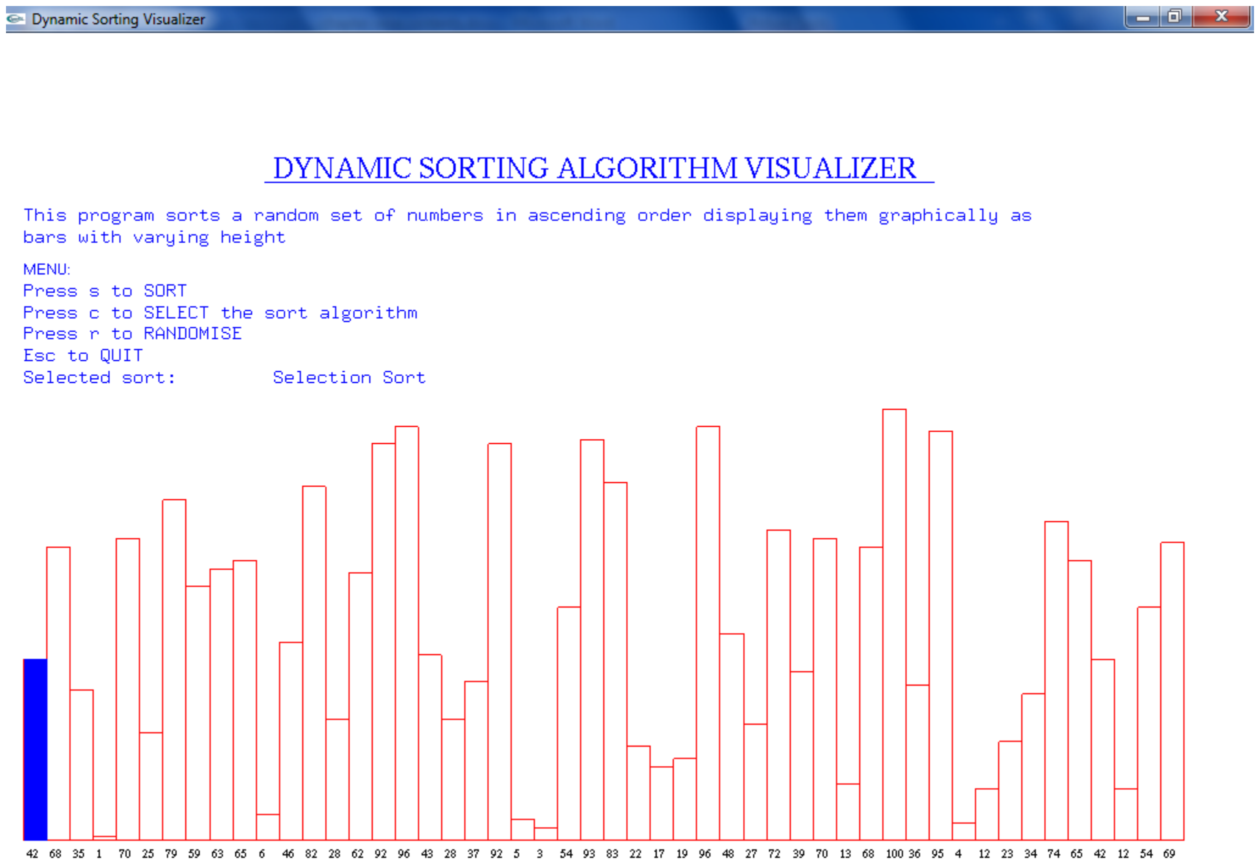


DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:        Bubble Sort
```

Fig 6.3: Sort Algorithms

On pressing the “c” key the user can change the sort algorithm which he wants to use for performing the sort on randomized numbers.

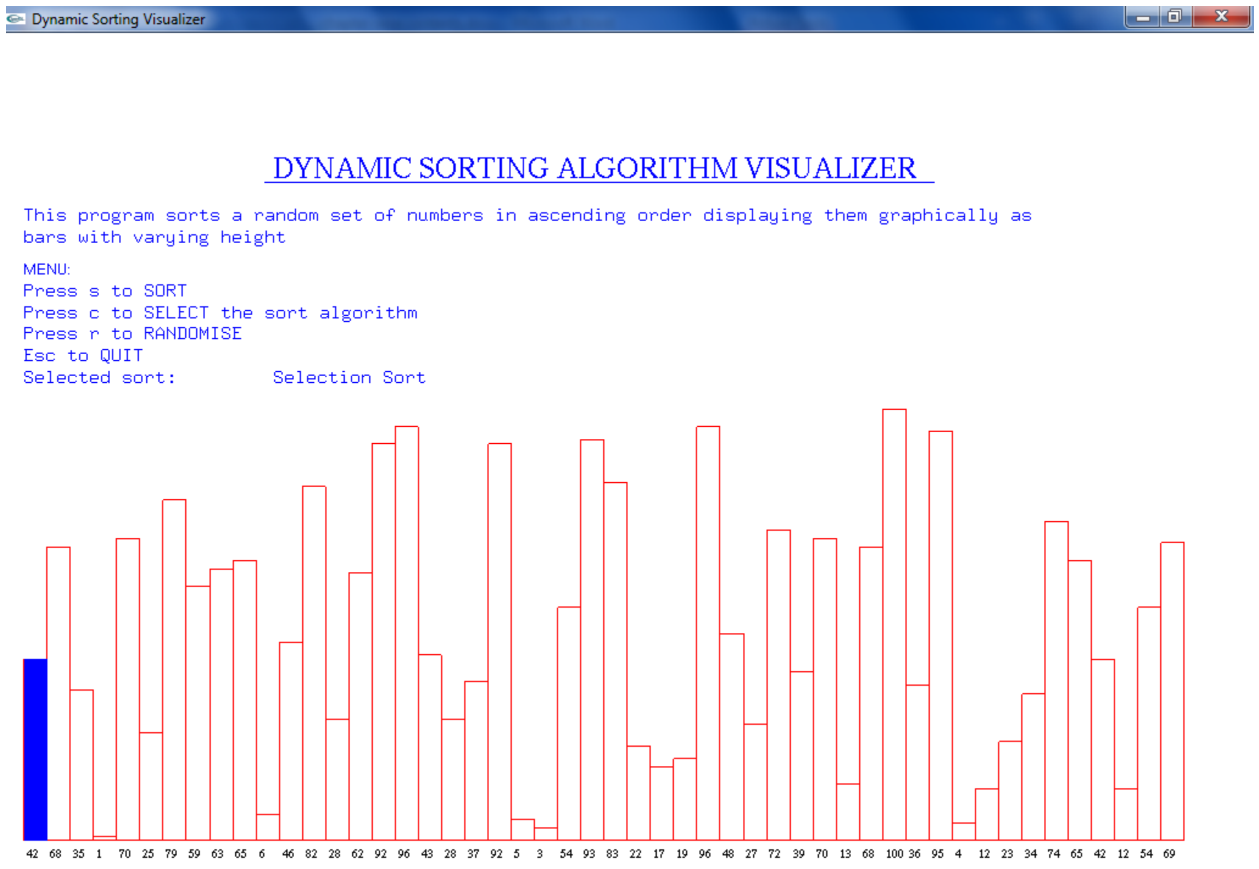


DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height
MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:         Selection Sort

42 68 35 1 70 25 79 59 63 65 6 46 82 28 62 92 96 43 28 37 92 5 3 54 93 83 22 17 19 96 48 27 72 39 70 13 68 100 36 95 4 12 23 34 74 65 42 12 54 69
```

Fig 6.4: Randomize the number bars

On pressing the “r” key the user can randomize the number that are to be used by algorithm for sorting. On pressing this key changes the height of the bars representing the numbers.



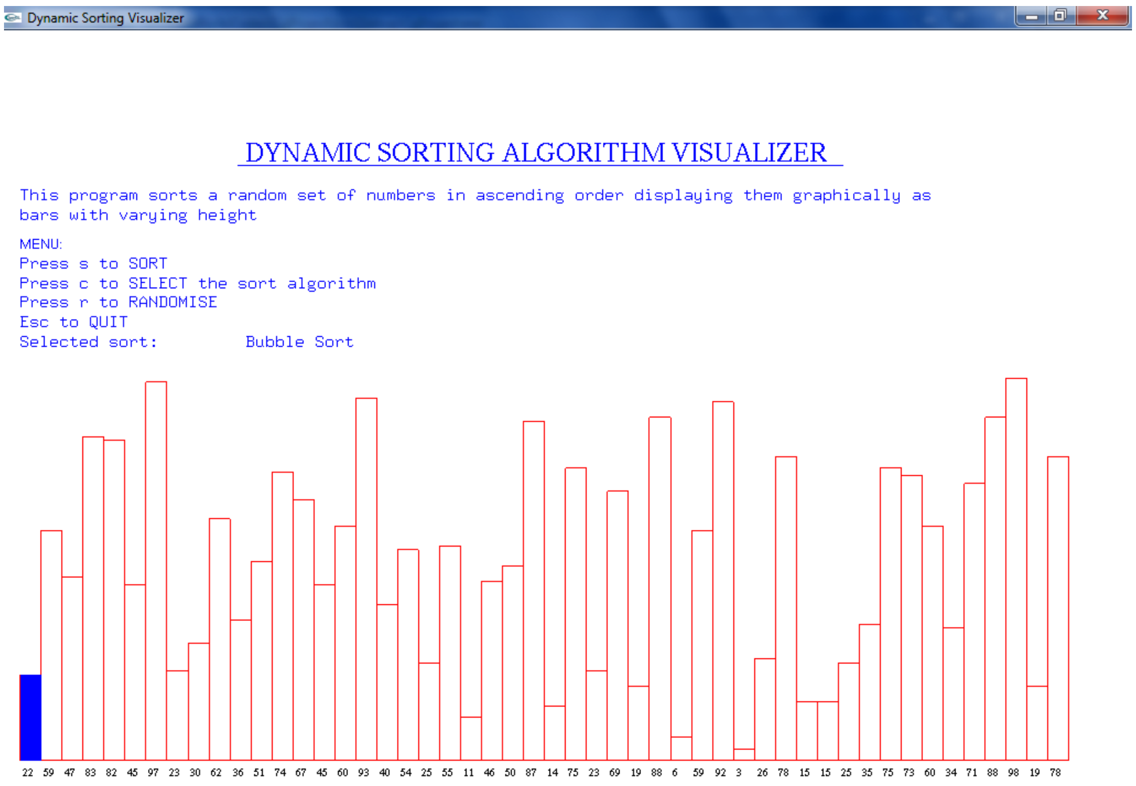
Dynamic Sorting Visualizer

DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:          Selection Sort
```

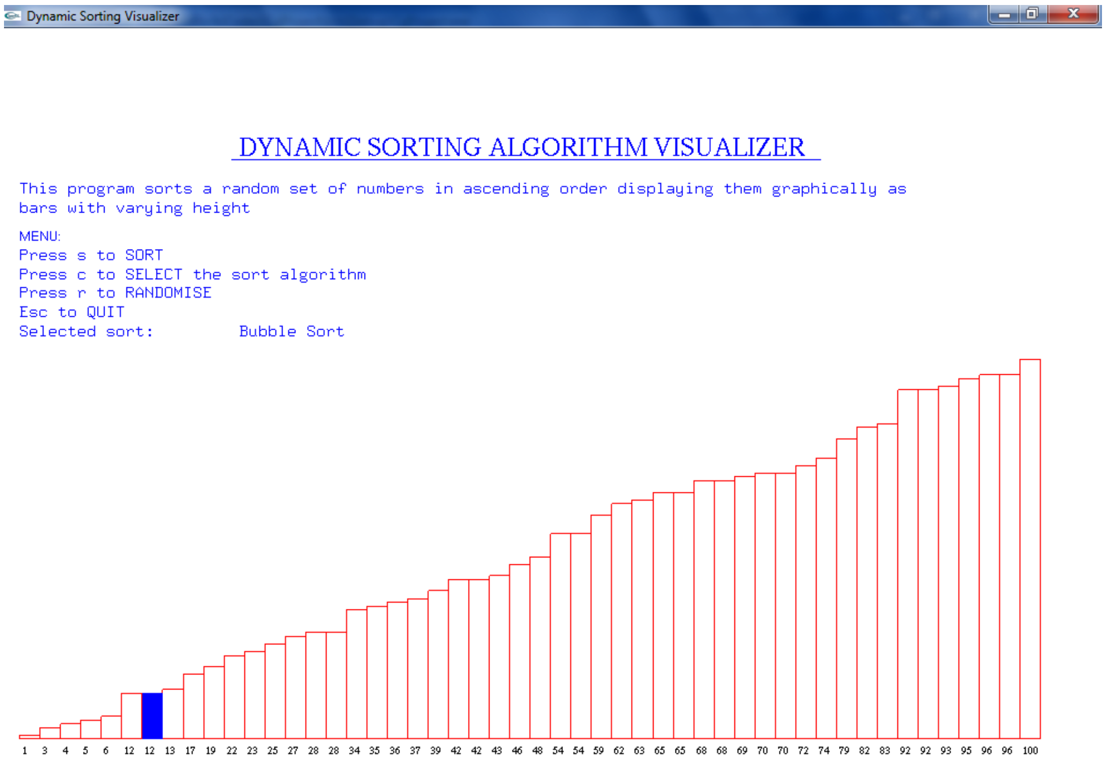
Fig 6.5: Sorting

On pressing the “s” key, the user can perform the sorting on the numbers using selected sorting algorithm. The sorting process is visualized as the numbers are shifted from one position to other.



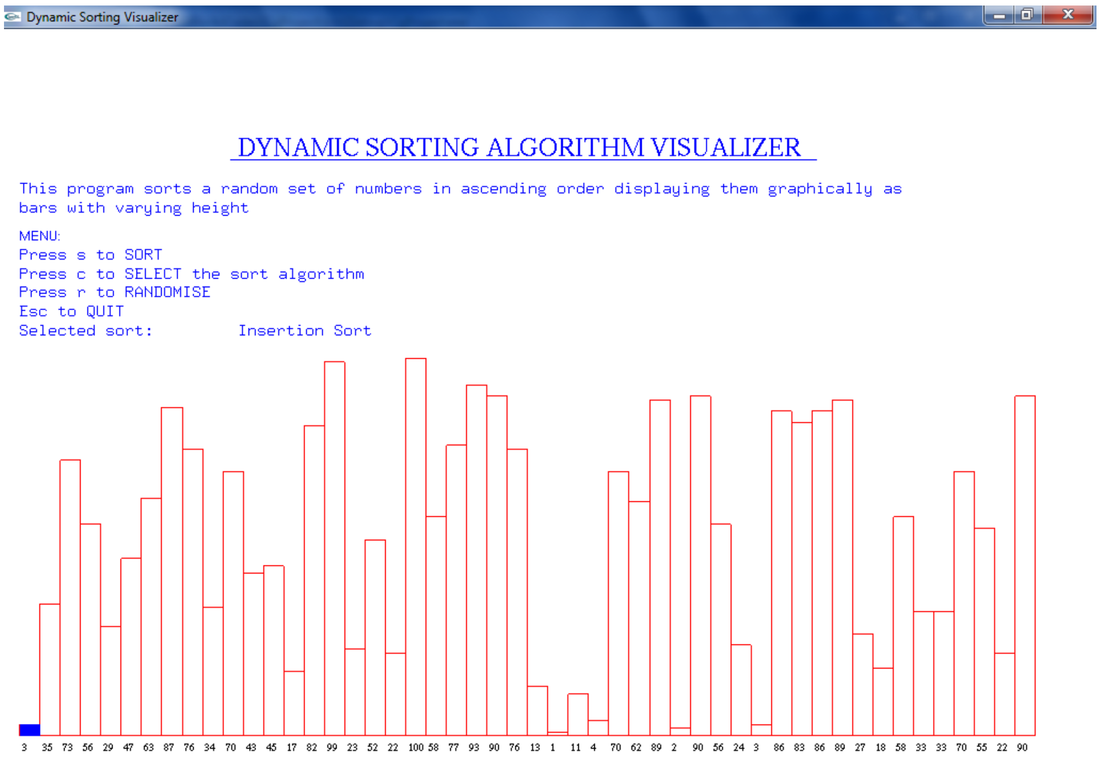
```
DYNAMIC SORTING ALGORITHM VISUALIZER

This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:         Bubble Sort

22 59 47 83 82 45 97 23 30 62 36 51 74 67 45 60 93 40 54 25 55 11 46 50 87 14 75 23 69 19 88 6 59 92 3 26 78 15 15 25 35 75 73 60 34 71 88 98 19 78
```

Fig 6.6: Bubble Sort Input



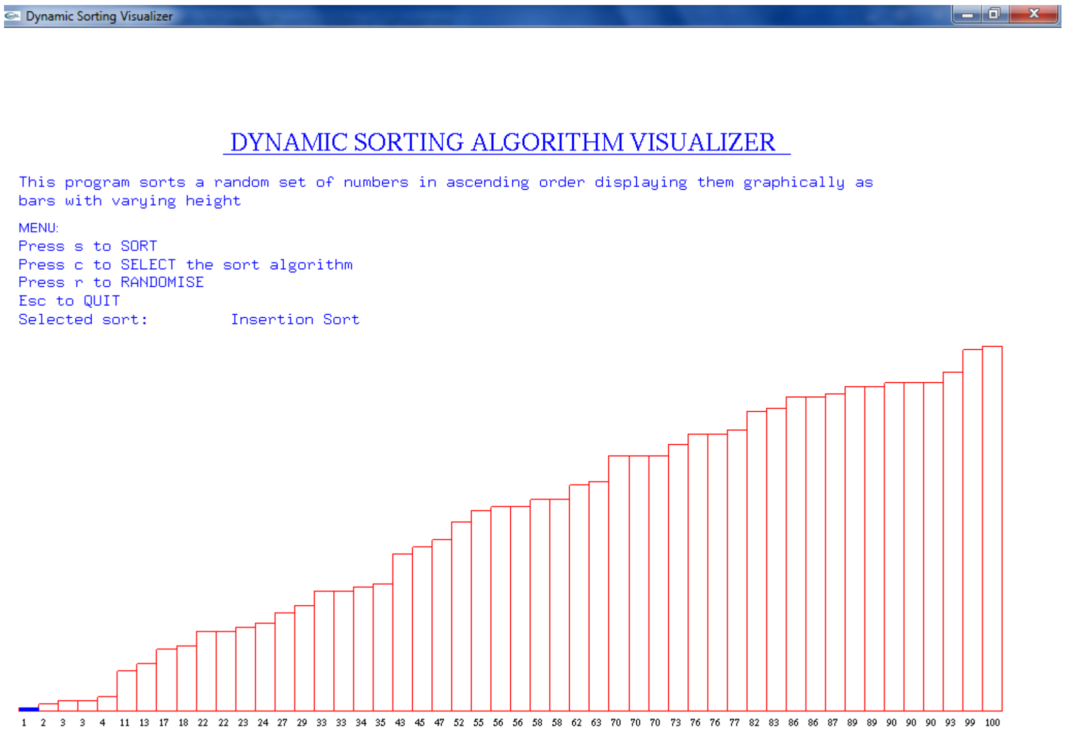
```
DYNAMIC SORTING ALGORITHM VISUALIZER

This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:         Bubble Sort

1 3 4 5 6 12 12 13 17 19 22 23 25 27 28 28 34 35 36 37 39 42 42 43 46 48 54 54 59 62 63 65 65 68 68 69 70 70 72 74 79 82 83 92 92 93 95 96 96 100
```

Fig 6.7: Bubble Sort Output



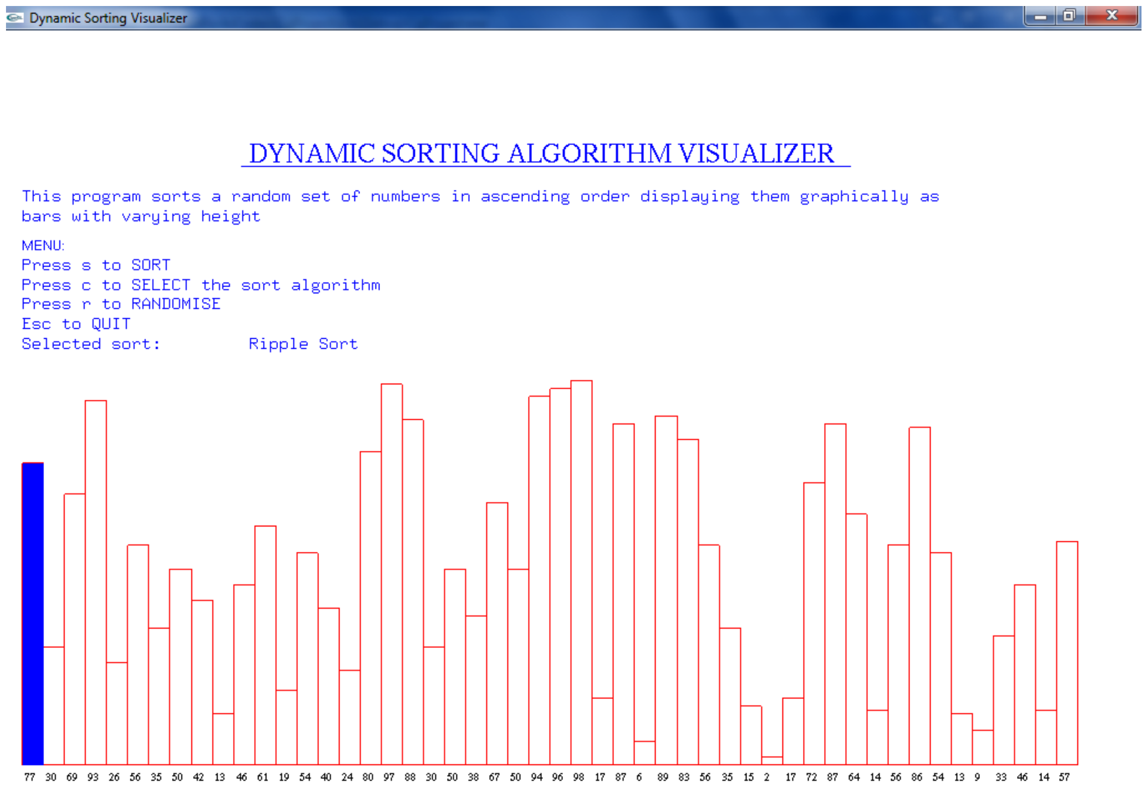
```
Dynamic Sorting Visualizer

DYNAMIC SORTING ALGORITHM VISUALIZER

This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:         Insertion Sort

3 35 73 56 29 47 63 87 76 34 70 43 45 17 82 99 23 52 22 100 58 77 93 90 76 13 1 11 4 70 62 89 2 90 56 24 3 86 83 86 89 27 18 58 33 33 70 55 22 90
```

Fig 6.8: Insertion Sort Input



```
Dynamic Sorting Visualizer

DYNAMIC SORTING ALGORITHM VISUALIZER

This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:         Insertion Sort

1 2 3 3 4 11 13 17 18 22 22 23 24 27 29 33 33 34 35 43 45 47 52 55 56 56 58 58 62 63 70 70 70 73 76 76 77 82 83 86 86 87 89 89 90 90 90 93 99 100
```

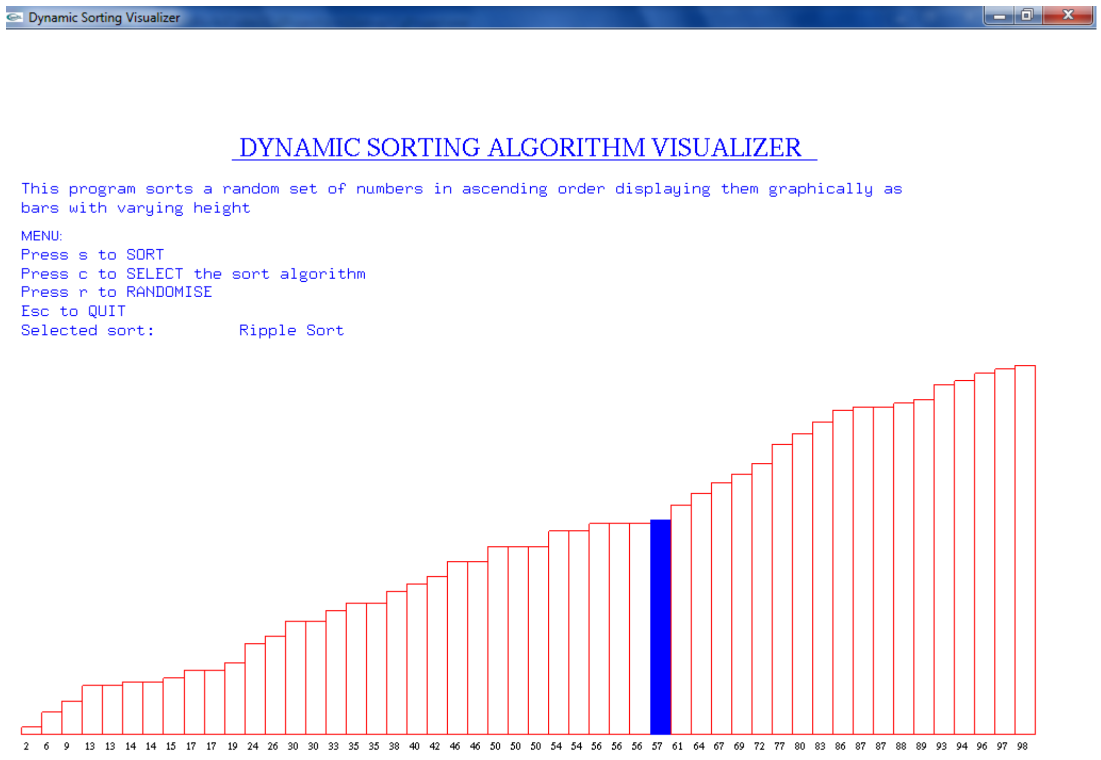
Fig 6.9: Insertion Sort Output



DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height
MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:        Ripple Sort
```

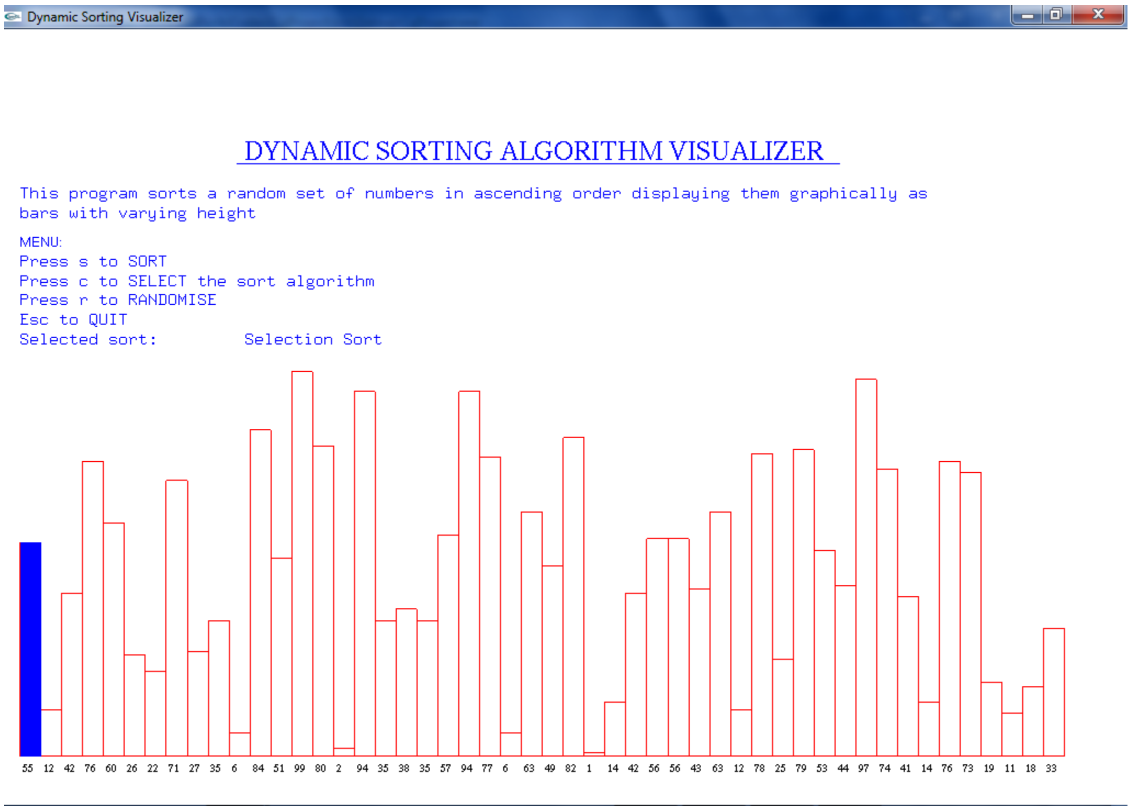
Fig 6.10: Ripple Sort Input



DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height
MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:        Ripple Sort
```

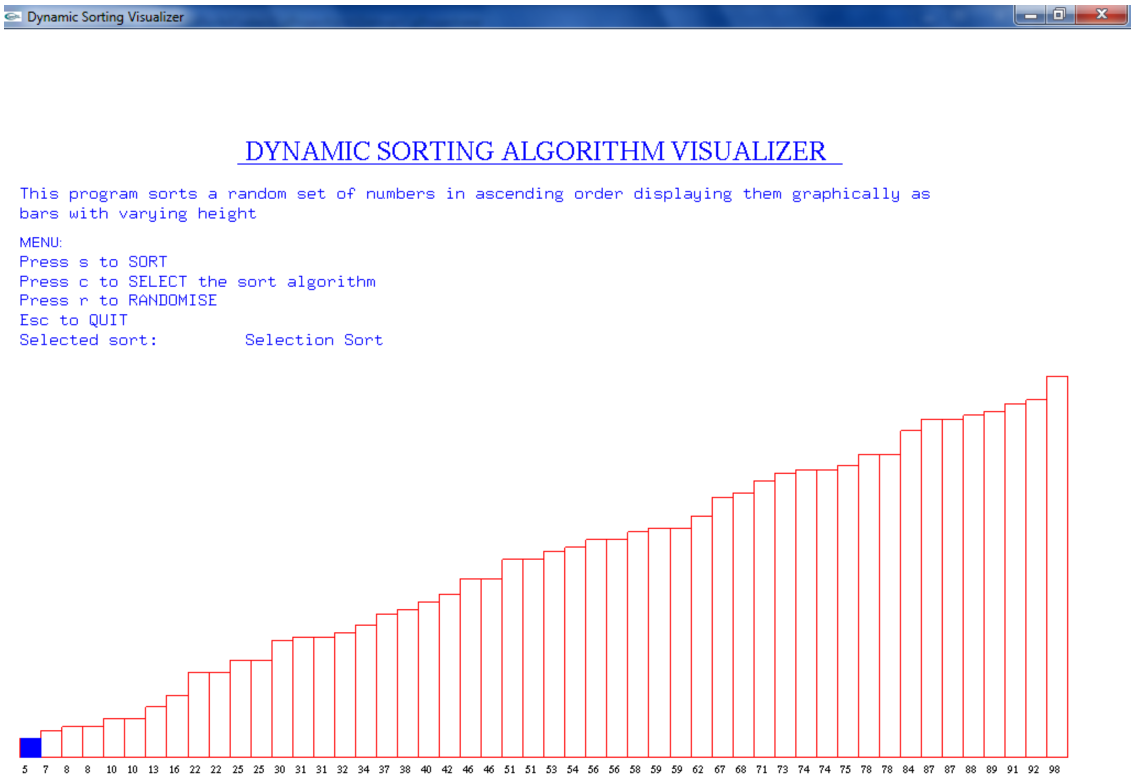
Fig 6.11: Ripple Sort Output



DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height
MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:          Selection Sort
```

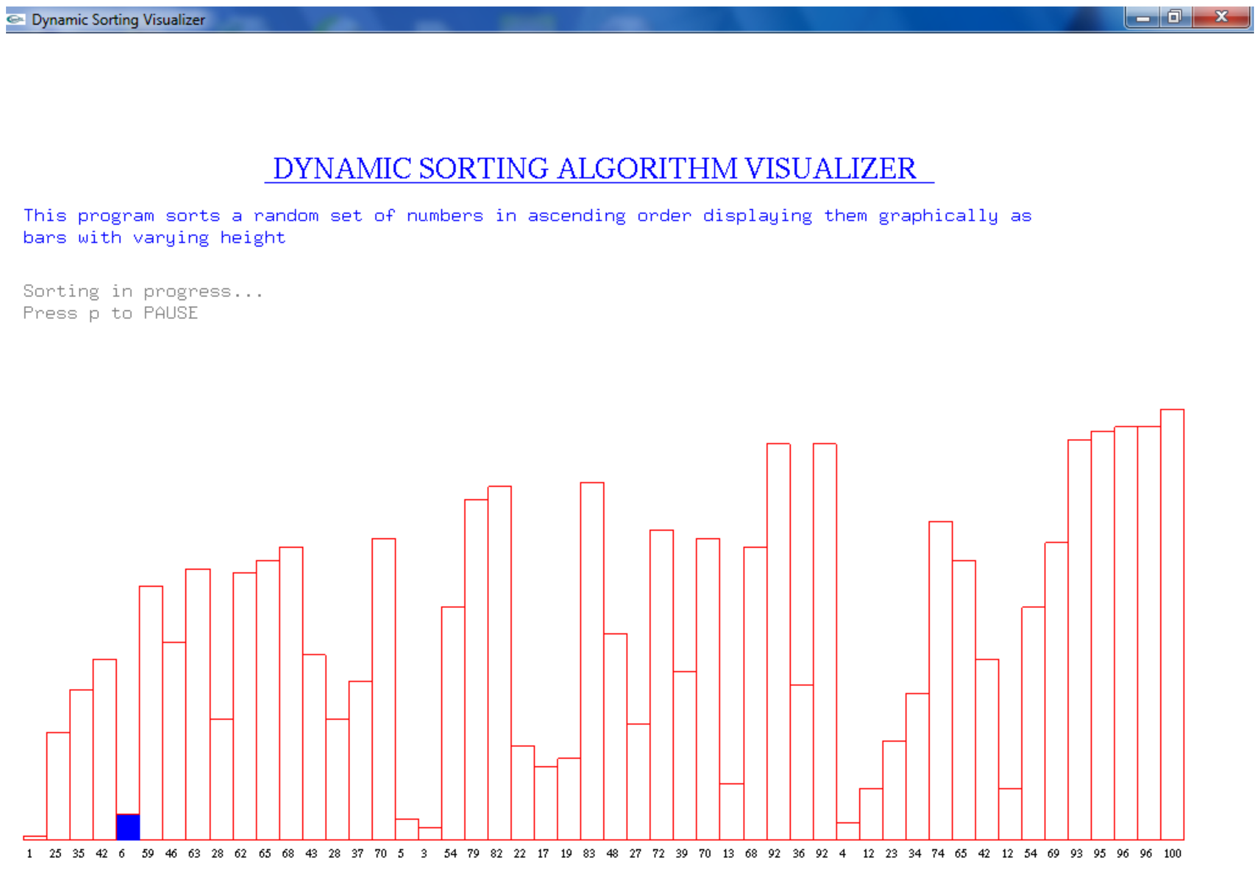
Fig 6.12: Selection Sort Input



DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height
MENU:
Press s to SORT
Press c to SELECT the sort algorithm
Press r to RANDOMISE
Esc to QUIT
Selected sort:          Selection Sort
```

Fig 6.13: Selection Sort Output



DYNAMIC SORTING ALGORITHM VISUALIZER

```
This program sorts a random set of numbers in ascending order displaying them graphically as
bars with varying height

Sorting in progress...
Press p to PAUSE
```

Fig 6.14: Pause

On pressing the “p” key the sorting process is halted. The user gets the main menu to select from different options. The sorting process can be resumed by pressing “s” Key.

The user can quit the program by pressing “ESC” Key.

Conclusion

We are able to visualize the Sorting Algorithms with the help of OpenGL Programming using glut in C/C++. The program used the various functions provided by OpenGL to create a 2D visuals. With the help of keyboard, the user can select from various options available on the window. Every aspiring engineering student wants to understand the working of sorting algorithms in detail as it helps them in solving the real world problems. We are able to visualize the Sorting Algorithms with the help of OpenGL Programming using glut in C/C++. This helps in visualize the working of various algorithms and to help understand the inner working process in algorithm.

References

[1]. Edward Angel *“Interactive Computer Graphics”* Pearson Education, 5th Edition.2008
[2]. Donald Hearn & Pauline baker *“Computer Graphics with OpenGL”* 3rd Edition. 2011
[3].www.opengl.org
[4]. www.freeglut.sourcefourge.net
[5]. www.github.com
[6]. www.openglcg.blogspot.in