**Sorting algorithm visualization H09**

IWASAKI, Keita 20684826 [kiwasakiaa@connect.ust.hk](mailto:kiwasakiaa@connect.ust.hk)

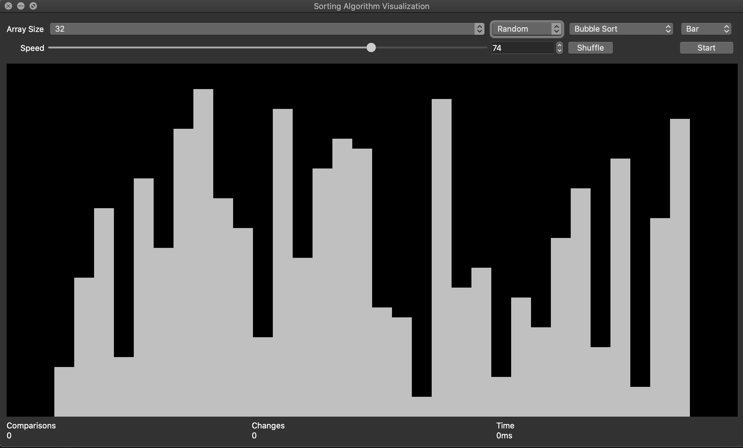
LEE, Young Kyu 20219190 [ykleeac@connect.ust.hk](mailto:ykleeac@connect.ust.hk)

**Features incorporated**

* **Various shuffle types**

Users can select how to shuffle elements before the sorting.

* + Random sort

****

* + Reverse sort

**A picture containing screenshot

Description automatically generated**

* + Almost sorted

**A screenshot of a cell phone

Description automatically generated**

* **Shape of the elements**
  + Bar

A close up of a device

Description automatically generated

* + Star

A screenshot of a computer screen

Description automatically generated

* **Information about each sorting**

1. Number of comparisons
2. Number of changes
3. Execution time

A screenshot of a computer screen

Description automatically generated

* Color bars

Moving bars are colored differently

A picture containing music

Description automatically generated

* **Number of the elements**

**Users can select from the following options**

2, 4, 6, 8, 16, 32, 64, 256, 512, 1024

* **Speed of the sorting**

Users can select an integer between 0~100

* **Sorting algorithm**

Users can select a sorting algorithm from the following

1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Merge Sort
5. Heap Sort
6. Quick Sort
7. Radix Sort (Base 2)
8. Radix Sort(Base 10)
9. Counting Sort
10. Gnome Sort
11. Cocktail Sort
12. Comb Sort
13. Odd Even Sort

**OOP design**

There are three classes defined in our projects: MainWindow, Paint and Sorting.

Paint class’s main work is drawing the elements. Sorting class is responsible for the sorting logic, all the sorting algorithms are written in this class.MainWindow is the class to work as a mediator between UI and other classes. It receives the signal from UI components and notifies to other classes accordingly and vice versa.

**Implementation**

**Paint**

A screenshot of a cell phone

Description automatically generated

Paint class is the derived class of `QWidget` class and it is a class to draw elements on the palette. It controls width, space and color of each bar. It is the member variable of MainWindow class.

**Member variables explanation**

* bool animation : true when elements are moving, false when not
* int penWidth : the width of each element, it changes according to the number of elements
* int spacing : space between each element, default is 0
* int\* numbers : array of elements
* int\* indices : array of indices to be painted in different color
* int size : size of numbers array, use this when we paint all the elements by iterating through numbers array by for-loop
* int sizeColorIndices : size of indices array, use this for iteration
* QPalette palette : A QPalette makes the UI easily configurable and easier to keep color consistent. It specifies the background color of the painting zone
* QPen pen : QPen is the class to draw lines. Each element is drawn by pen
* QColor lineColor : color of each element
* QColor backgroundColor : background color of the palette.
* QString paintType : The type of element(Bar or Star)
* QStringList paintTypes : A Qt style array of string it contains “Bar” and “Star” user can select this from the dropmenu in ui
* std::vector<QColor> colors : A vector of QColor, which stores the possible colors of each element. (red or green)
* QMediaPlayer\* sortingsound : A sound for each element movement

**Member functions explanation**

* Paint(QWidget \*parent = *nullptr*)

A constructor for Paint class. It sets the background color of the palette(painting area). It also sets the initial values of member variables `animation`, `Penwith`, `spacing` and `pen`. Finally, it initializes `sorting sound` using the binaries of the sound stored in the `Resources` folder.

These are setter functions, which simply set the value except for the function `setPenWidth(int)`. `setPen(const QPen)` and `setBrush(const QBrush)` are using pass by reference in order to save memory.

* void setPenWidth(int width) : If `width` is greater than the MAX\_PEN\_WIDTH it sets the value of constant variable MAX\_PEN\_WIDTH otherwise sets the value of `width`.
* void setPen(*const* QPen &pen)
* void setBrush(*const* QBrush &brush)
* void setSpacing(int space)
* void setPaintType(QString option)
* void setLineColor(QColor color)
* void setAnimation(bool anim)
* void setPaintData(int\*, int\*, int, int) : this function sets member variables `numbers`, `colorIndices`, `size` and `sizeColorIndices` which are essential to draw elements.
* QStringList getPaintTypes()

It simply returns a QStringList of paintTypes which are `Bar` and `Star`.

* void reset()

It resets the member variable `linecolor` to constant variable `DEFAULT\_LINE\_COLOR` which is Qt::lightGray.

* void *paintEvent*(QPaintEvent \*event)

A paint event is a request to repaint all or part of a widget. It mainly happens if repaint() or update() is invoked. This is the core function to draw elements. First, we initialize a local variable of QPainter class called painter. Next, we initialize an int variable called `colorIdx` with initial value 0. It also initialize an int variable `space`, which is a space to set the elements in the middle of the palette. It iterates through the `numbers` array and check if the index matches with any of the value in colorIndices it becomes a colored bar. It draws each bar with a function `painter.drawLine(int,int,int,int)` for a bar and a function `painter.drawPoint(int,int)` for a star. Finally, it checks the state of `sortingsound`. If it is in ` QMediaPlayer::PlayingState`, it set the position of `sortingsound` to 0. In `QMediaPlayer::StoppedState`, it simply play `sortingsound`.

**MainWindow**

A screenshot of a cell phone

Description automatically generated

MainWindow class is the derived class of `QMainWindow` class and it is the central component of this application. It interacts with all the user-defined classes and connects logic parts and UI parts.

**Member variables explanation**

* Ui::MainWindow \*ui : A default variable provided by Qt. Access `mainwindow.ui` via this variable and control UI components.
* Sorting\* sorting : An instance variable of Sorting class. To work on the sorting algorithm is the main roll.
* Paint paint : An instance variable of Paint class. This class is used to visualize the elements.
* QTime timer : A timer for measuring execution time of sorting.
* isSorting : true when elements have been sorting, false when not
* QMediaPlayer\* completionsound : A sound for the completion of sorting

**Member functions explanation**

* MainWindow(QWidget)

A constructor of MainWindow class. To set up the UI components, connect signals and slots and initialize `completionsound` are the main task for this function. It calls `createArray()` function at the end.

* ~MainWindow()

A destructor of MainWindow class. It simply deletes `completionsound` and `ui` variables.

* void onNumbersChanged(int\*, int, int\*, int)

Signal for this slot function is `Sorting::changed()` and it handles the array of numbers and colorIndices with the size of each array. It gets the number of comparisons and changed and use those value to update the UI components `labelComparisons` and `labelChanges`. It calls `paint.setPaintData()` and set the member variables of Paint instance and call `paint.update()` to trigger `paintEvent()`.

These slot functions are connected to a signal function `currentTextChanged(QString)` of UI components `comboBoxAlgorithm`, `comboBoxShuffle` or `comboBoxPaintType`. When user changes the option in comboBox this slot function is called. It calls the appropriate setter function of paint class and set the new value.

* void onChangeAlgorithm(QString)
* void onChangeShuffle(QString)
* void onChangePaintType(QString)

Other slot functions

* void onSortingFinished() : This slot function is connected to signal function `Sorting::done()`. This function is called when sorting is finished. It sets text of ui component `labelTime`, change the color of elements to green, update the buttons and make the `completionsound` at the end. The logic of making the sound is same as `sortingsound`.
* void onNumberOfSizeChange(QString) : This slot function is to deal with the changes in array size. It is connected to UI component `comboBoxArraySize`. It sets the proper value of penWidth and update the UI.
* void on\_buttonStart\_pressed() : this is called when `buttonStart` is pressed. If `isSorting` is false it sets the `isSorting` true and change the button accordingly. It starts the sorting animation and the timer. If `isSorting` is true it sets the `isSortin` false and it stops sorting(thread) calling `terminate()` which is a function of QThread.
* void on\_buttonShuffle\_pressed() : this is called when buttonShuffle is pressed. It reset the line color of elements and call `Sorting::createArray`.