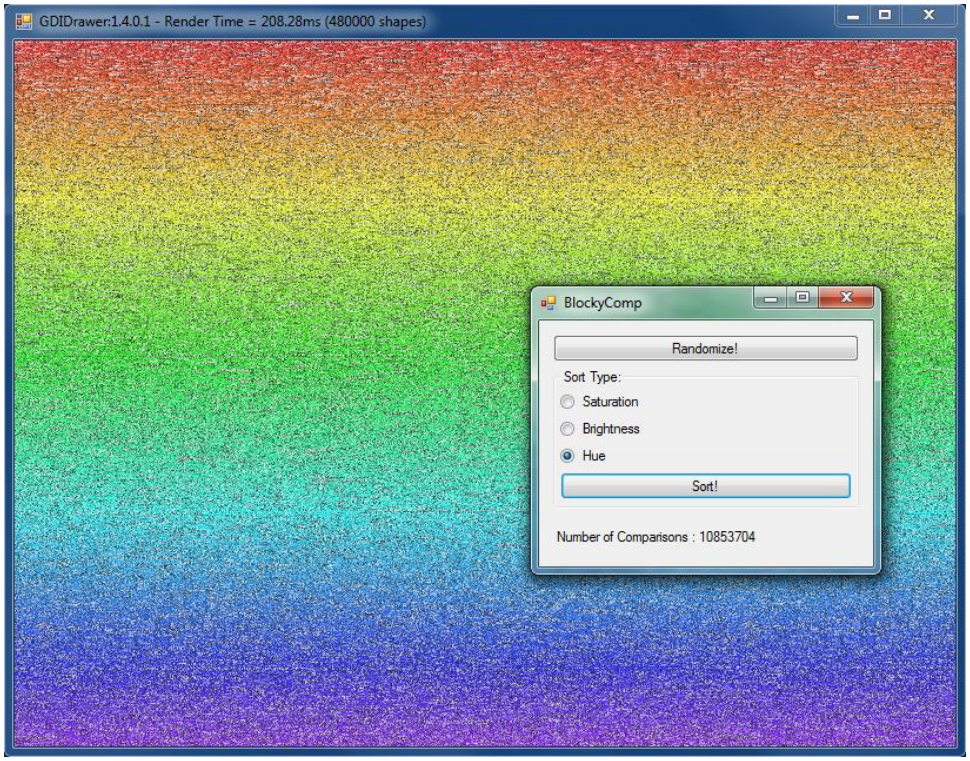
CMPE2300 ICA#05 – BlockyComp

In this ICA, you will implement the `IComparable` interface to sort some colors.



Create a Block class that implements `IComparable`.

This class will contain:

- a `public` constant `int` for the block size (10) [This will be changed at compile time]
- a `Color` field implemented as an automatic `public set`, `private get` property.
- a `public` enumeration for the sorting style (see above)
- a `public static` instance of the sorting style enumeration
- a `public static int` to hold the number of times the `CompareTo` method has been called

Add a constructor that accepts a color to initialize the color field.

Add a render method that will accept an X position, Y position, and a `CDrawer`. Render a rectangle at the specified position, for the block size, in the appropriate color.

Complete the `CompareTo` method. Using the static sort type, determine the correct algorithm to use to generate the return value. Note: The `Color` type has built-in support for generating saturation, brightness, and hue values.

In the main form add:

- a private `List<Block>`
- a private static `Random`
- a private static `CDrawer`
- private constant `int`s for the number of rows and number of columns (based on block size from `Block`)

Add a Render method. This method will clear the drawer. If there are [cols * rows] elements in the list, you will render each block in the list. You must provide coordinates to the Render method, so use a nested for loop construct to provide the coordinates (based on the row and column count).

For the 'Randomize!' button handler, clear the list and add exactly enough blocks to entirely fill the drawer window. The blocks will have colors with RGB components randomized 32-255 (use the utility in GDIDrawer namespace for this). Render.

For the 'Sort' button handler, sort the blocks using the `Sort` method of your list. Render. Show the number of comparisons that were made when the sorting is complete. NOTE: Clear the sort count prior to sorting so the sort count is always the number of comparisons for this sorting operation.

Depending on how you set up your code, you may need to look at the radio buttons to set the sort type prior to sorting.

Oh yes, also, be cool – don't block the UI.