Homework Assignment 010 (2020)

LED Display and Animation

Create a Win application project named as **<yourID><yourName>Ass010Two**LEDAnimation. Remember to rename your Form1.cs file with a meaningful class name, say LEDDisplay. We will create a 16x32 LED display to cycle and shrink/expending your own lighting pattern, which is set up interactively. In addition, the application can record and animate the user defined pictures; e.g., animation of Taiwanese Green Shorty (the pedestrian traffic lights).

The LED display consists of a grid of 16x32 LED bubbles, constructed as a two dimensional array of objects of PictureBox UI class. Make sure they are instantiated and initialized with a dimmed image when the form is loaded. Prepare two Bitmap objects for displaying a lighted LED and a dimmed one. The two image files are embedded as project resources.

Subscribe the click event of each LED object of the PictureBox to let us toggle its lighting status, either lighted up or dimmed. A single event handler (function or method) is shared by the click events of all of the PictureBox objects.

We will exercise bitwise operations to control the lit of an LED with a bit value. When the bit value is 1, the corresponding LED is lighted; otherwise, it is dimmed. Each row of 32 LEDs is controlled by a 0-1 code of int type. Therefore, the picture of the 16-row display (an animation frame) is controlled by an array of 32 0-1 elements; i.e., int[] aFrame. An animation clip is therefore recorded as a number of int[] objects (frames), which can be repeatedly played.

User can click on the LED object to toggle its lit condition to compose a picture. The composed picture can be saved as a frame of an animation clip and added to a list. We use the template list class List<> to dynamically and interactively add to or remove a frame from the list. Note that a picture frame is an element of type int[]. Therefore the type of the list is List< int[]>.

Once several pictures are dynamically created and added to the list, user can click on an animation button to repeatedly play the recorded pictures as an animation clip. Therefore, an animation timer is required and repeatedly advance the picture frame one after the other, until the user stop it. The recorded pictures can be interactively modified and saved as a file. The file should be a text file with the following format:

<number of Frames(n)>

<Row0 of Frame0> <Row1 of Frame0> … <Row15 of Frame0>

<Row0 of Frame1> <Row1 of Frame1> … <Row15 of Frame1>

…

<Row0 of Frame(n-1)> <Row1 of Frame(n-1)> … <Row15 of Frame(n-1)>

The following is an example:

8

0 3072 7680 3072 512 896 832 3872 4512 384 320 544 1040 6152 8 24

…

To play a saved animation clip, file open function should be provided to read in the contents and dynamically reconstruct the list of pictures (a frame represented by 16 int[] objects).

Therefore, the data fields required are:

```
PictureBox[,] LEDs;  // 16x32 LED
Bitmap onBmp, offBmp; // Bitmaps for On/Off
List<int[]> frames;  // Animation frames; dynamic array
int currentIdx = 0; // Animation index
int[] originalPicture; // Saved from screen
int[] changedPicture;  // cycling or shrinking
int shift;
int delta; // -1 shrinking +1 expanding
bool cycling = false;
```

The following methods are suggested for conducting frame recording, displaying, and filing:

```
// Scan images of LEDs to constitute an array or int
int[] recordAFrame()
{
    int[] aFrame = new int[16];
    for (int r = 0; r < 16; r++  )
    {
        // Read LED images of row r to compose the int value
       ...
    }
    return aFrame;
}


// Display the LED screen images for a given array of int
void showAFrame( int [] frame )
{
    for (int r = 0; r < 16; r++)
    {
       // Set LED on or off for row r
       ...
    }
}

// Save the list of frames to a file
void saveAllFramesToAFile( string filePath )
{
    StreamWriter sw = new StreamWriter(filePath);
    sw.WriteLine(frames.Count);
    foreach( int[] frame in frames )
    {
       // A frame consists of 16 int values
       ...
    }
    sw.Close();
}

// Open a file to read in frames for animation
void readAllFramesFromAFile( string filePath )
```

```
{
    StreamReader sr = new StreamReader(filePath);
    int num = Convert.ToInt32(sr.ReadLine());
    frames.Clear();
    for( int i = 0; i < num; i++ )
    {
        // read in frame i; a line consists of 16 int values
         // frame i is an array of 16 int values
    }
    sr.Close();
}
```

In addition to the source code, you are asked to construct at least two animation files of your own design. The two files must be submitted to assignment site.

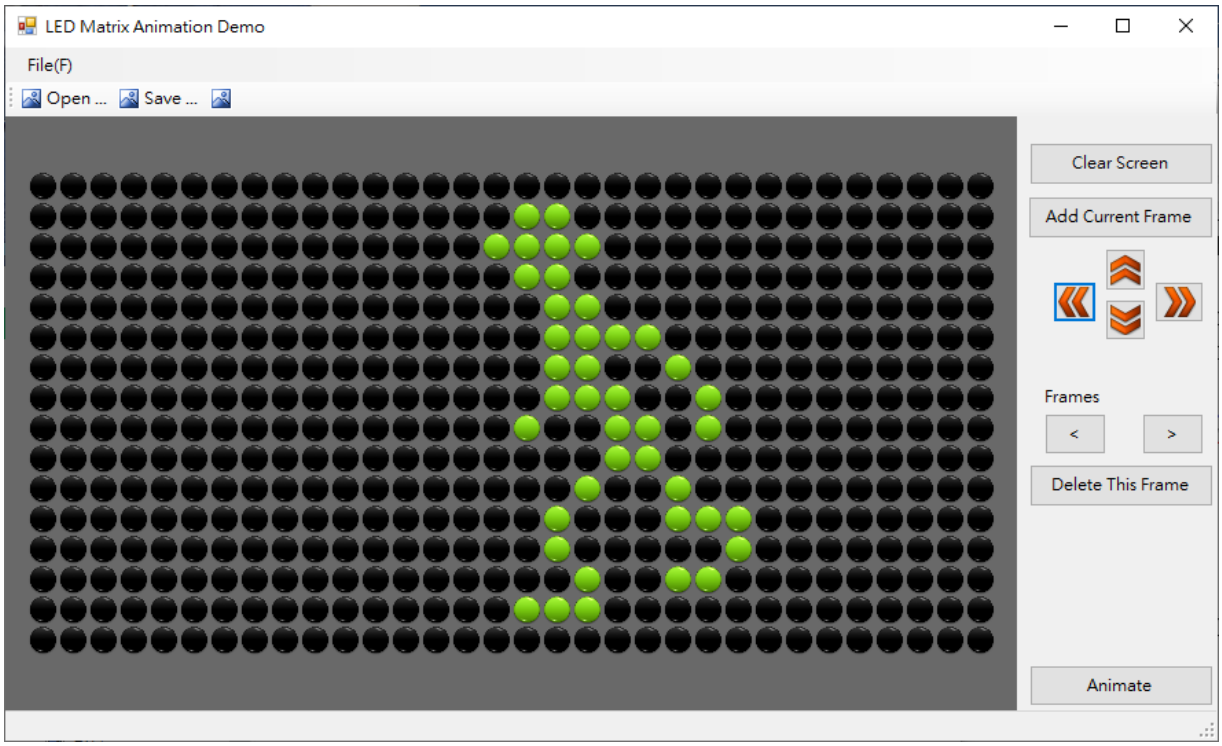
Extra work for this assignment is to provide the cycling and shrink functions on a static picture. At first, save the current picture and make a copy for modifying in cycling (panning) or shrinking operation. You should instantiate another timer object for the operations of cycling and shrinking/expanding. The shrinking/expanding logic is to recalculate the coordinates (row and column indices) of lit-LEDs and turn them on. Before that, turn all LEDs off first. Notice that the coordinates of the shrinking/expanding center is (x, y)= (15, 7.5). Possible implementation code is:

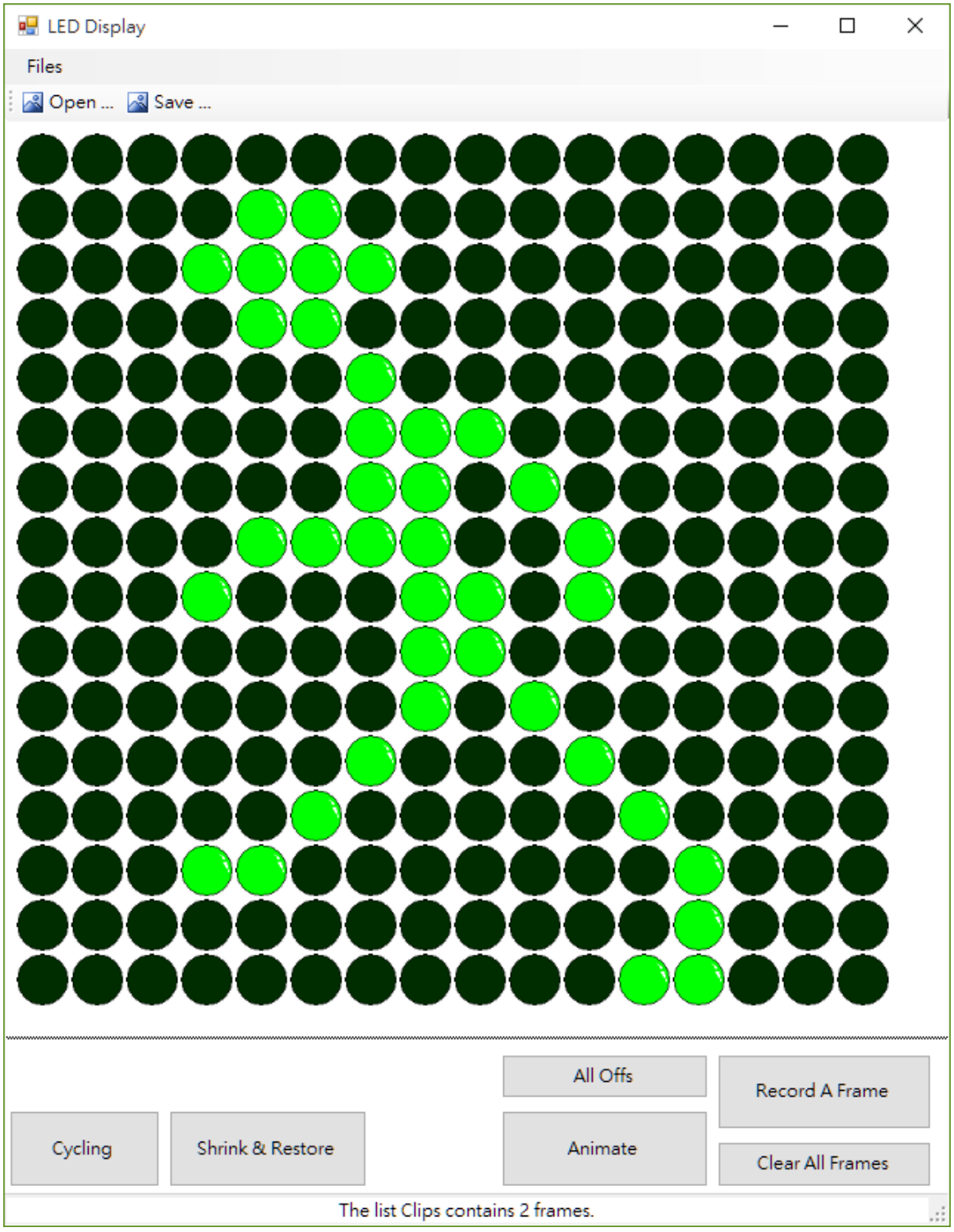
```
double scale = count / 8.0;
int newr, newc;
newr =(int)Math.Round( 7.5 + ( r - 7.5 ) * scale );

newc =(int)Math.Round( 15 + ( c - 15 ) * scale );
```

Appendix:

Snapshot of the Demo App




LED Display
Files
Open ...
Save ...
All Offs
Record A Frame
Cycling
Shrink & Restore
Animate
Clear All Frames
The list Clips contains 2 frames.