

Ascify: The Ascii Art Generator

DESIGNED BY KENNY TRAN &
NHUT LUONG (FRANK)



NEXT

Table of Contents

- PART 1: What Is An Ascii?
- PART 2: Project Goal
- PART 3: Breaking Down the Code
- PART 4: Questions and Feedback

NEXT

Part 1: What is an Ascii?

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(88	58	1011000	130	X					
41	29	101001	51)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	-					

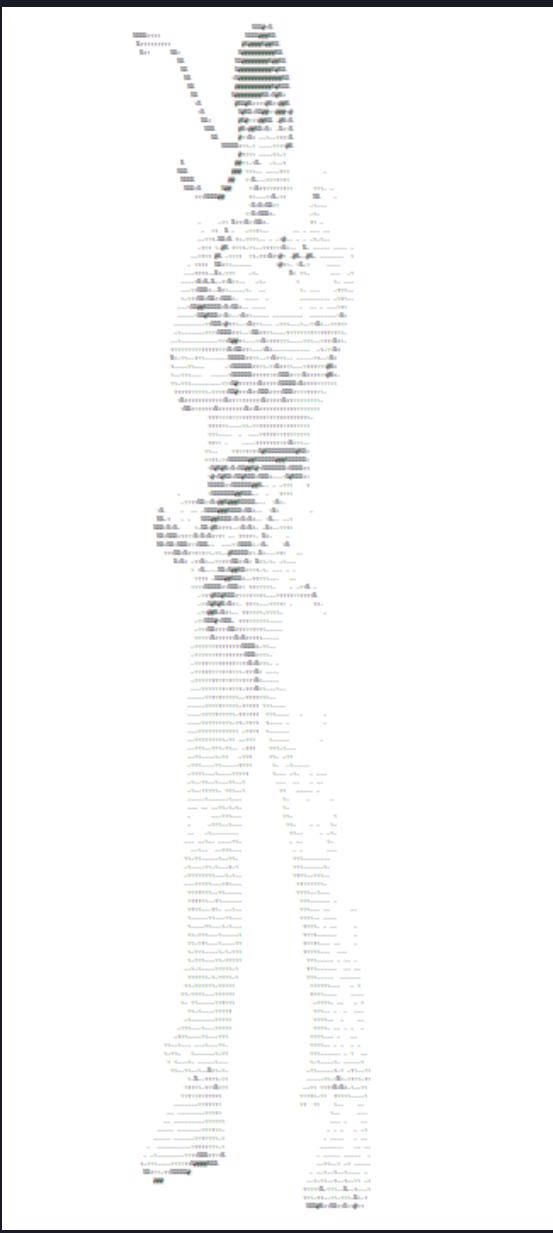
An acronym for American Standard Code for Information Interchange.

It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127

NEXT

Part 1: Ascii Art

Images that are created using the ASCII text characters



```
%%%@+%
%%%@000%
@%@@@%@%
%@@@@@@@@%
%@@@0000@0@%
%@@@@@@@@%@%
+@@@@@@@@@@%
@@@@@@@@@%@%
%@@@@@@@%+@%
@%@%+@@@+@%
%@%:++@%@%
@%+@%@%
@%+@%+@%
@:+%+ . . . :+%
%%%+::: . . . :::@%
@+::: . . . :::
@:@:.+. . . +.
@@@ :::.. . . . +:
%@@ :%..:::+:+
%@@@ :+%++:++:+:+
+++%%%@@ +:....:+.%+
:@%+%%: .
:@%+%%+.
. . . %++%:+%+.
. . . +: .
```

NEXT



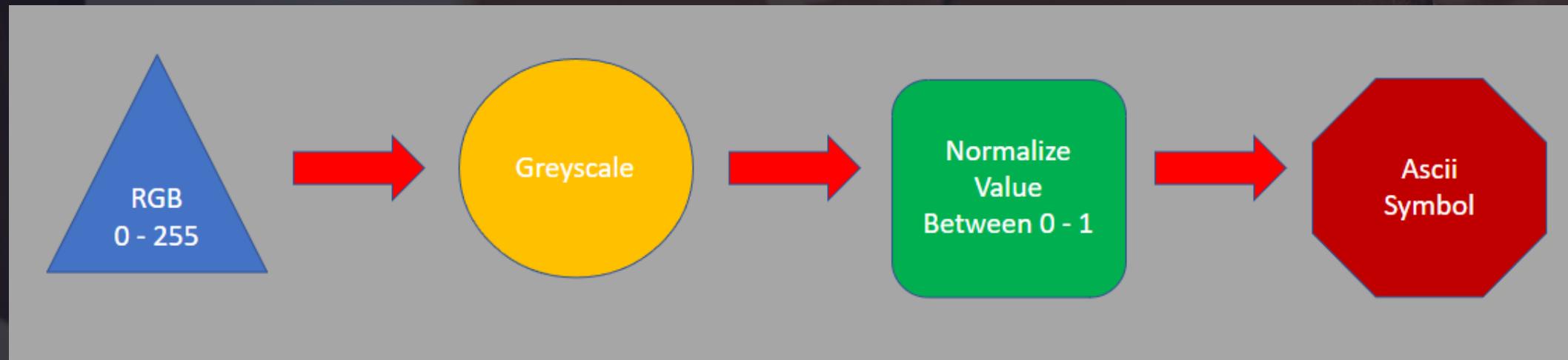


PART 2: WHAT IS OUR GOAL?

To create a functional program that converts images to "asciitized" art.

[NEXT](#)

Part 3: Asciiify Process

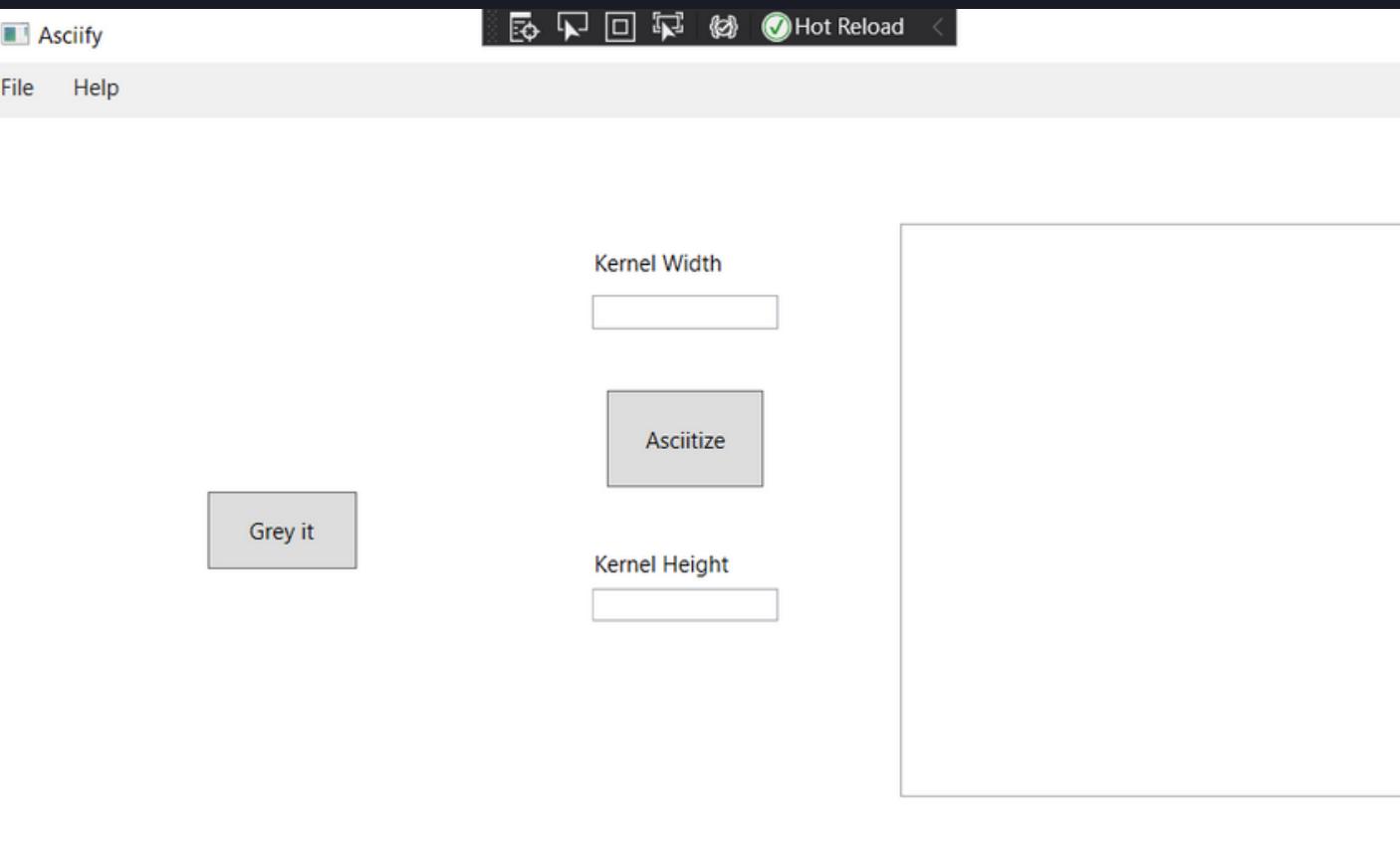




Part 3: Starting Out

HOW DO WE GET THERE?

USER INTERFACE
(EARLY PHASE)



BITMAP
CREATOR
CLASS

```
6  public class BitmapMaker {  
7      //THE BITMAP'S SIZE.  
8      private int _width;  
9      private int _height;  
10     //THE PIXEL ARRAY.  
11     private byte[] _pixelData;  
12     //THE NUMBER OF BYTES PER ROW.  
13     private int _stride;  
14  
15     PROPERTIES  
16  
17     CONSTRUCTORS  
18  
19     PUBLIC METHODS  
20  
21     PRIVATE METHODS  
22  
23 } //end class
```

Part 3: Defining Ascii Class

```
8     class BitmapAscii {
9         //Fields
10        private int _kernelHeight;
11        private int _kernelWidth;
12        private BitmapMaker _imgPath;
13
14        #region PROPERTIES
15        1 reference
16        public int KernelHeight{
17            get{ return _kernelHeight; }
18            }//end property
19
20        1 reference
21        public int KernelWidth {
22            get { return _kernelWidth; }
23            }//end property
24
25        1 reference
26        public BitmapMaker ImgPath {
27            get { return _imgPath; }
28            }//end property
29        #endregion
30
31        #region CONSTRUCTOR
32        1 reference
33        public BitmapAscii(BitmapMaker imgPath, int kernelWidth, int kernelHeight)
34            _kernelWidth = kernelWidth;
35            _kernelHeight = kernelHeight;
36            _imgPath = imgPath;
37            }//end constructor
```

ASCII CLASS: CONVERT LOADED
TO ASCII ART

NEXT

Part 3: Run "Grey It" and "Asciitize"

```
44     private void Button_ClickGreyTheImage(object sender, RoutedEventArgs e) {
45         if (filepath != string.Empty) {
46             //new instance of bitmap
47             BitmapMaker greyBitmap = new BitmapMaker(filepath);
48
49             //new instance of bitmap ascii
50             BitmapAscii tempBitmap = new BitmapAscii();
51
52             //set greybitmap to our bitmapascii convert to grey scale method
53             greyBitmap = tempBitmap.ConvertToGreyScale(filepath);
54
55             //set greyimage.source to makebitmap method
56             greyImage.Source = greyBitmap.MakeBitmap();
57         } else {
58             MessageBox.Show("Must upload image to convert to greyscale");
59         }
60     }
61 }
```

```
68     private string Asciitize(BitmapMaker imgSource, int kernelWidth, int kernelHeight) {
69         //Contains the ascii text version of the picture
70         string finalAsciiString = ...String.Empty;
71         double normalizedVal;
72         string asciiCharacter;
73
74         for (int y = 0; y < imgSource.Height; y += kernelHeight) {
75             for (int x = 0; x < imgSource.Width; x += kernelWidth) {
76                 //Get current pixel color
77                 Color pixelColor = imgSource.GetPixelColor(x, y);
78
79                 //Get avg pixel value and return normalized value
80                 normalizedVal = AverageColor(pixelColor);
81
82                 //Convert normalized value to char symbol
83                 asciiCharacter = GreyToString(normalizedVal);
84
85                 //Add symbol to ascii set
86                 finalAsciiString += asciiCharacter;
87             }
88         }
89         //Creates new line when reaching end of width
90         finalAsciiString += "\n";
91     }
92
93     return finalAsciiString;
94 }
```

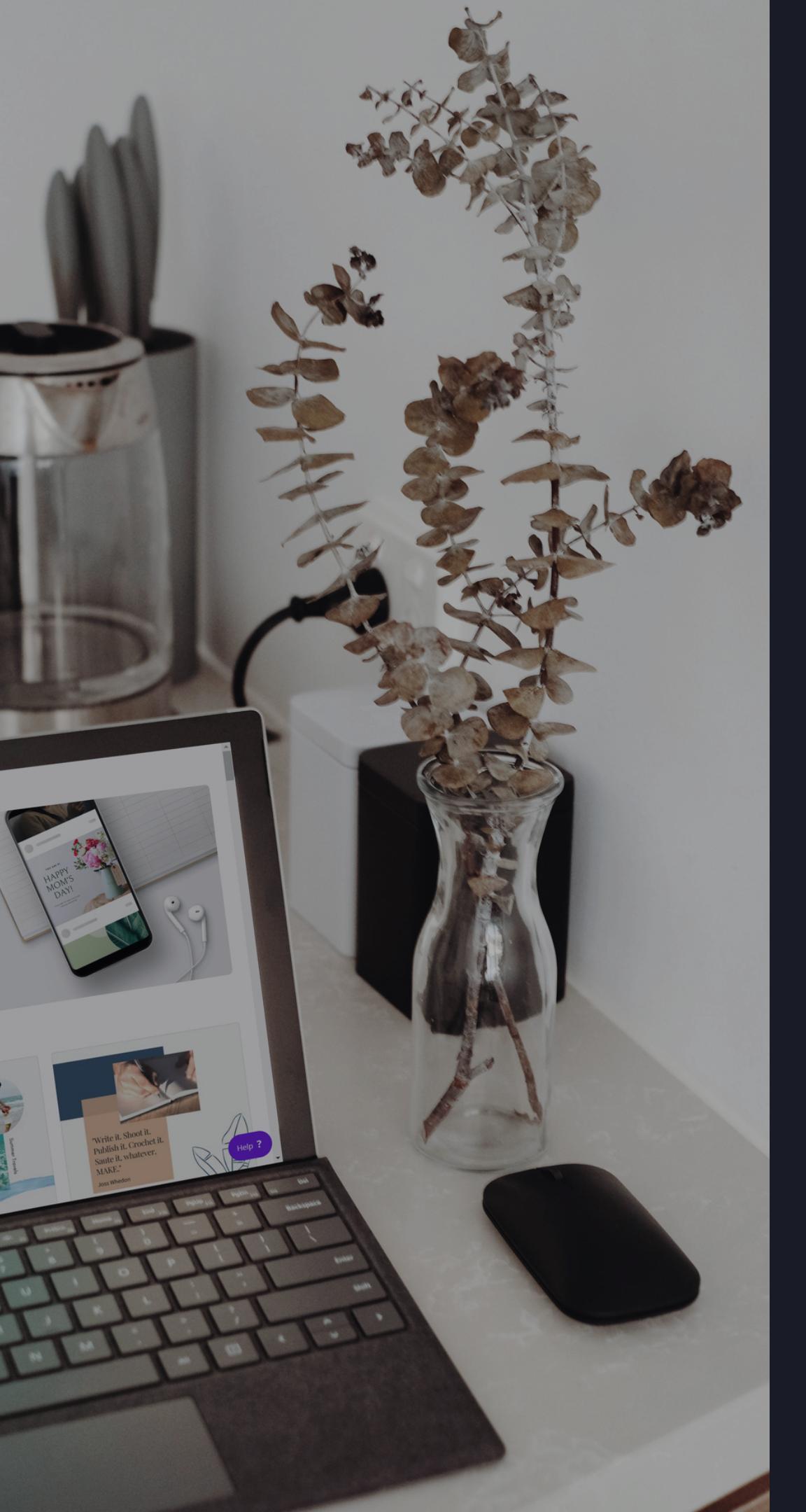
RIGHT: ASCII CONVERSION

LEFT: GREYSCALE CONVERSION

Part 3: Greyscale Conversion

```
41  public BitmapMaker ConvertToGreyScale(string imagePath) {  
42      BitmapMaker greyBitmap = new BitmapMaker(imagePath);  
43      double luminosity; //avg rgb value of luminosity  
44      Color currentColor;  
45  
46      //Loop through each pixel to convert orginal rgb to grey value  
47      for (int y = 0; y < greyBitmap.Height; y++) {  
48          for (int x = 0; x < greyBitmap.Width; x++) {  
49              //Get pixel color  
50              currentColor = greyBitmap.GetPixelColor(x, y);  
51  
52              //Pass color values to get rgb's avg through luminosity method  
53              luminosity = AveragePixel(currentColor);  
54  
55              //Set greyscale pixel  
56              greyBitmap.SetPixel(x, y, Color.FromArgb(currentColor.A, (byte)luminosity, (byte)luminosity, (byte)luminosity));  
57          } //end for  
58      } //end for  
59  
60      return greyBitmap;  
61  } //end method
```





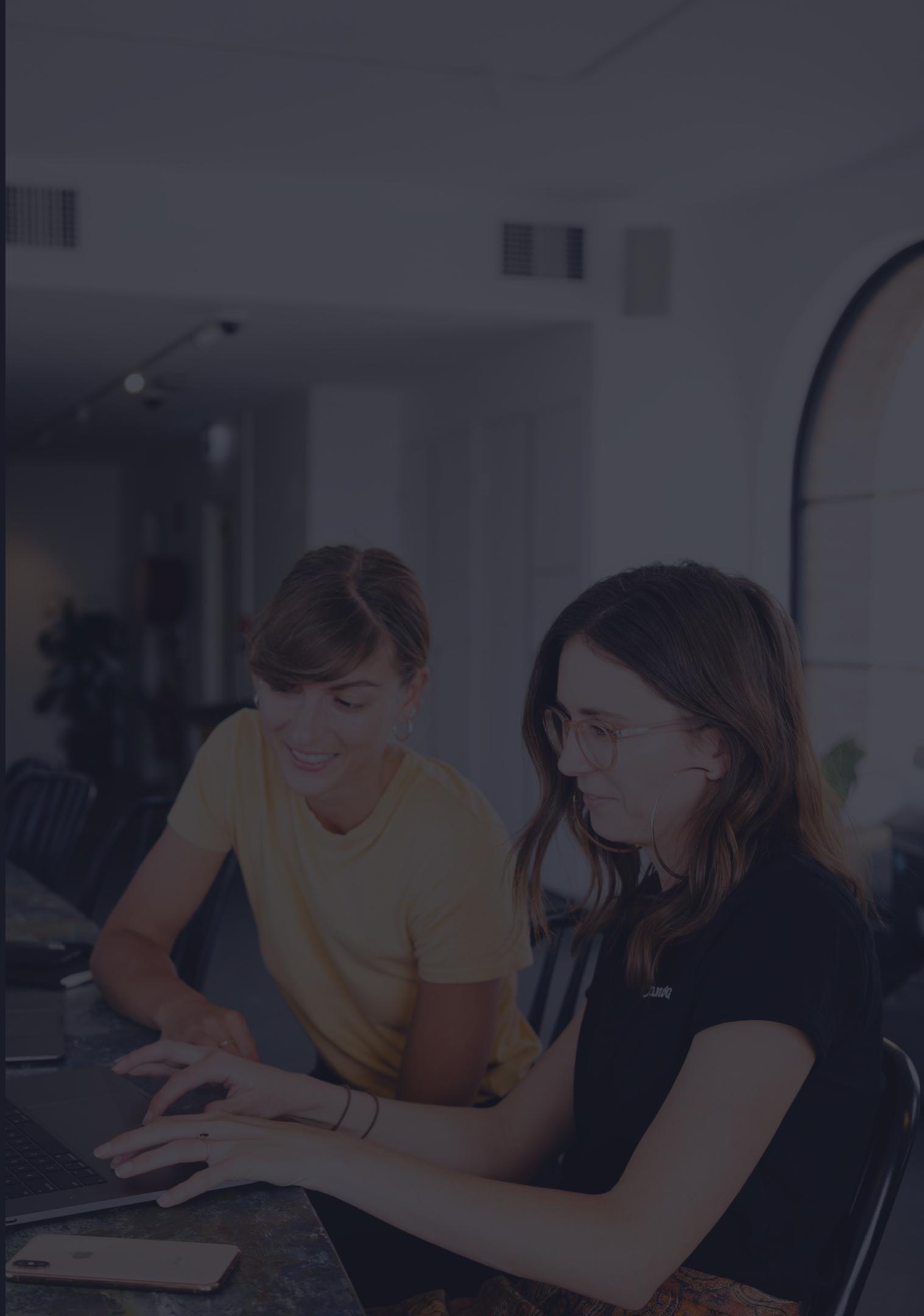
Part 3: Average Pixel and Normalized Value

```
96      private double AveragePixel(Color rgb) {  
97          //Luminosity algorithm (the perceived brightness of a colour)  
98          //Green is weighted more for perception  
99          return ((.21 * rgb.R) + (.72 * rgb.G) + (.07 * rgb.B));  
100         }//end method  
101  
102     private double AverageColor(Color pixelColors) {  
103         double normalizedKernelValue = 0.0;  
104         //Returns normalized value from the avg pixel  
105         normalizedKernelValue = (AveragePixel(pixelColors)) / 255;  
106  
107         return normalizedKernelValue;  
108     }//end method
```

Part 3: Produce Ascii Image

```
63     public override string ToString() {
64         string finalAsciiString = Asciitize(ImgPath, KernelWidth, KernelHeight);
65         return finalAsciiString;
66     } //end method
```

```
110    private string GreyToString(double normalizedKernelValue) {
111        //Normalized Values to Char Symbols
112        string asciiString = string.Empty;
113
114        if (normalizedKernelValue == 0) {
115            //Fixes the whitespace from turning black
116            //rgb.A was never changed in averagepixel() so it remained the same in terms of normalized scale
117            asciiString = " ";
118        } else if (normalizedKernelValue < 0.16) {
119            asciiString = "@";
120        } else if (normalizedKernelValue >= 0.16 && normalizedKernelValue < 0.33) {
121            asciiString = "%";
122        } else if (normalizedKernelValue >= 0.33 && normalizedKernelValue < 0.48) {
123            asciiString = "+";
124        } else if (normalizedKernelValue >= 0.48 && normalizedKernelValue < 0.64) {
125            asciiString = ":";
126        } else if (normalizedKernelValue >= 0.64 && normalizedKernelValue < 0.8) {
127            asciiString = ".";
128        } else if (normalizedKernelValue >= 0.8 && normalizedKernelValue < 1) {
129            asciiString = " ";
130        } //end if
131
132        return asciiString;
133    } //end method
```



A group of diverse people in a modern office setting, smiling and shaking hands, with a large window in the background.

Questions and Feedback

A group of diverse people are gathered in an office setting, smiling and shaking hands. In the foreground, a man in a light-colored sweater and a woman in a brown turtleneck are shaking hands. Behind them, two other women are engaged in conversation. The background features a window with a grid pattern. The overall atmosphere is professional and positive.

Thank you!

Our Team



ANDREW STAMPS

Scrum Master



KENNY TRAN

Junior Full Stack Developer



NHUT LUONG (FRANK)

Junior Full Stack Developer