Asciify: The Ascii Art Generator

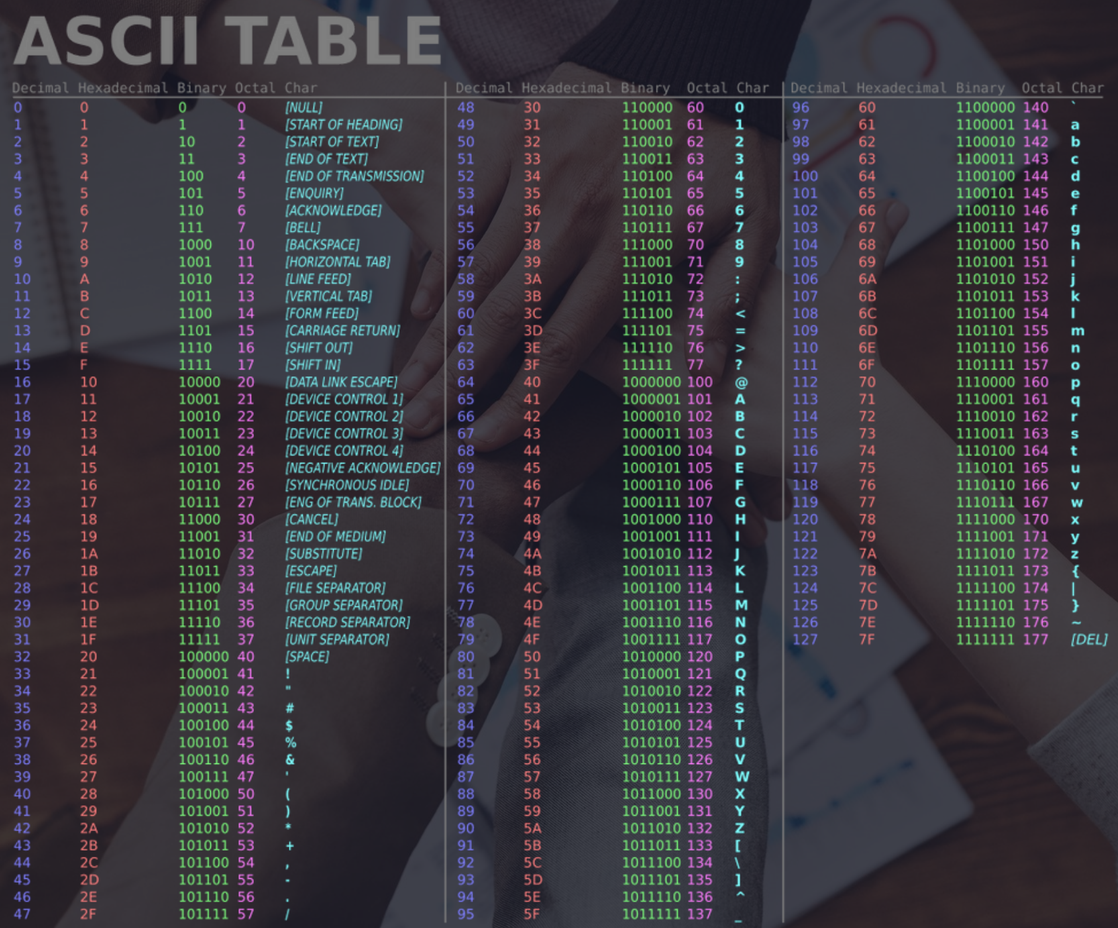
DESIGNED BY KENNY TRAN &
NHUT LUONG (FRANK)

NEXT

Table of Contents

NEXT

Part 1: What is an Ascii?



ASCII TABLE

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

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

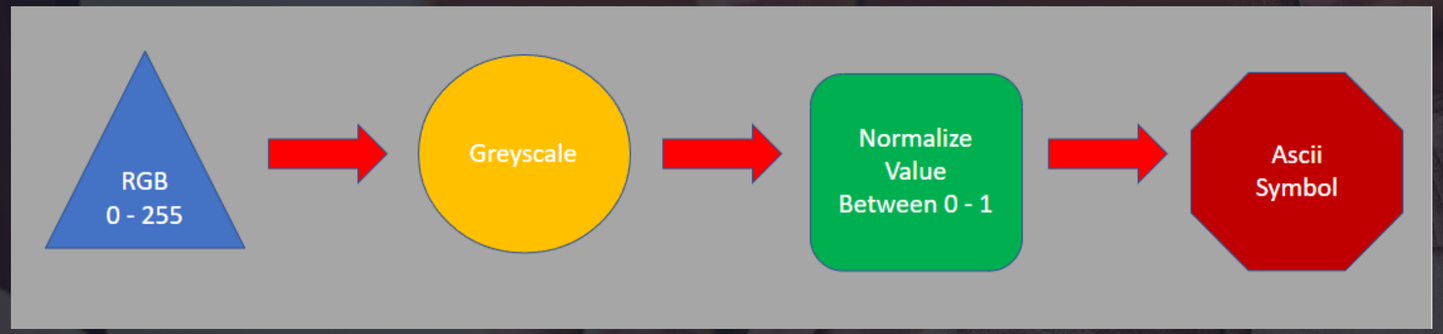
NEXT

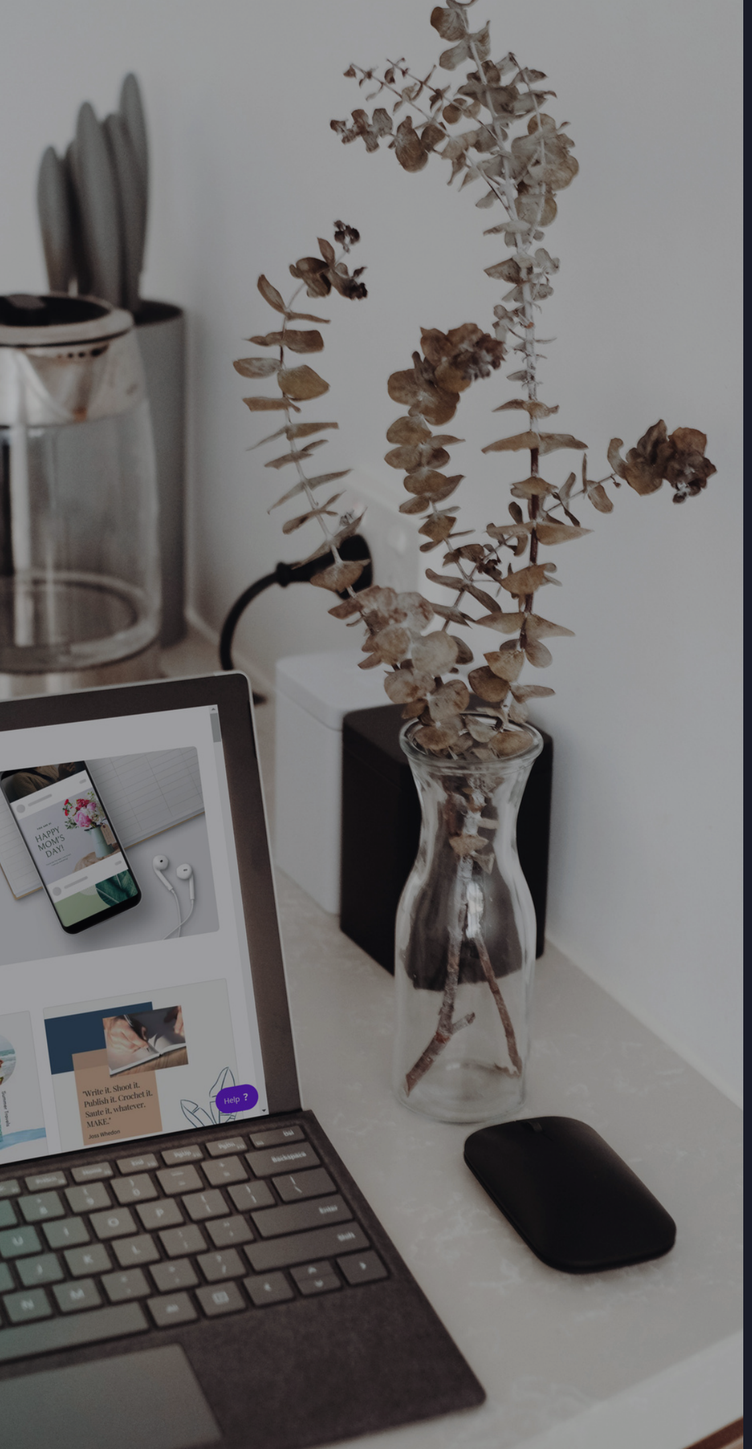
PART 2: WHAT IS OUR GOAL?

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

NEXT

Part 3: Asciify Process

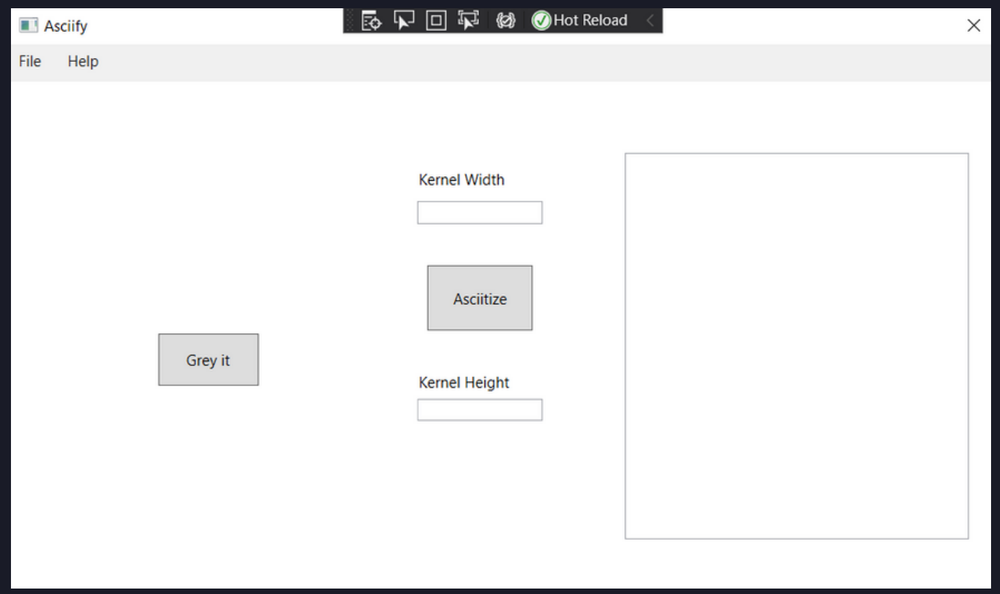




Part 3: Starting Out

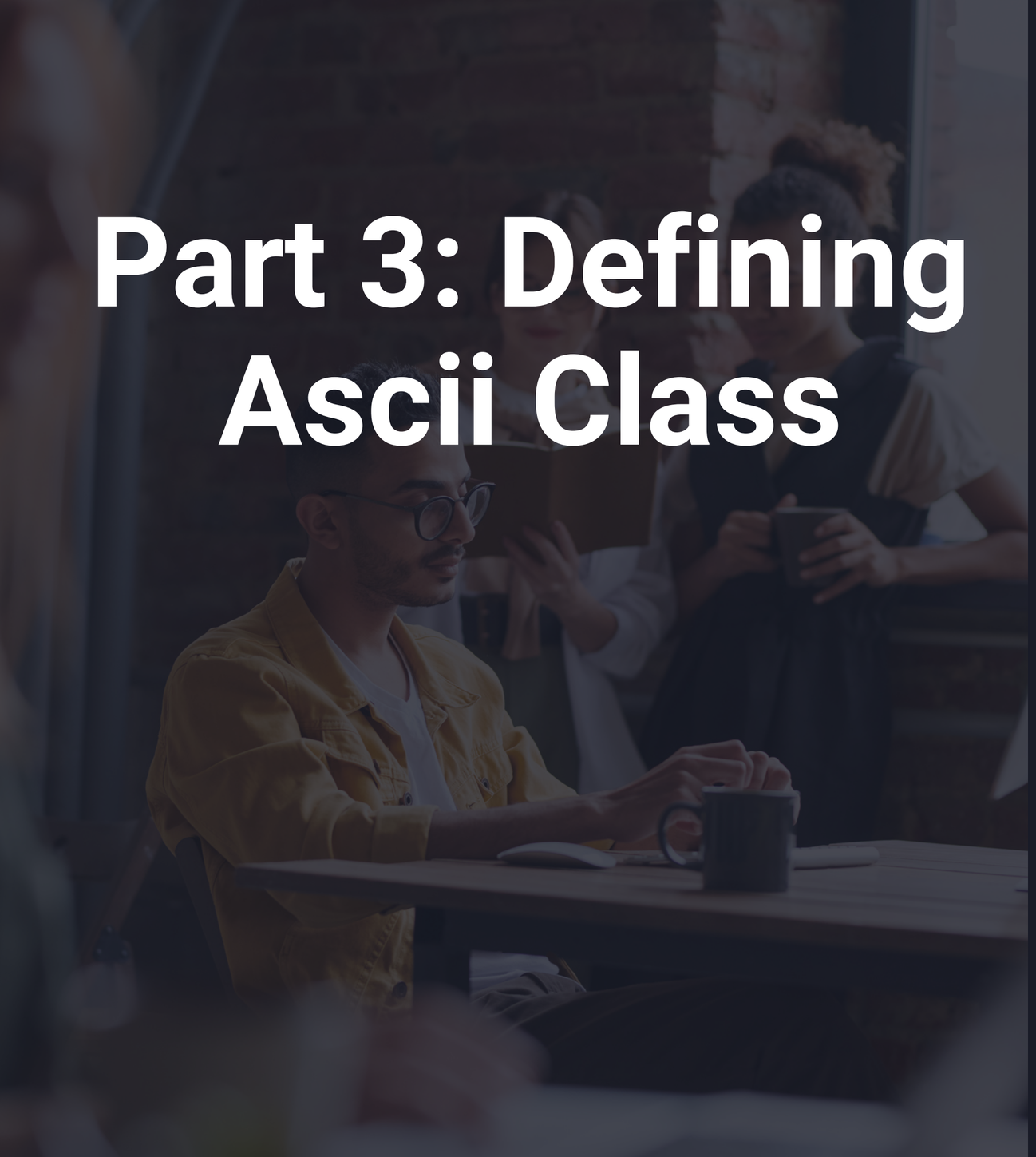
HOW DO WE GET THERE?

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
11      //THE PIXEL ARRAY.
12      private byte[] _pixelData;
13
14      //THE NUMBER OF BYTES PER ROW.
15      private int _stride;
16
17      PROPERTIES
29
30      CONSTRUCTORS
53
54      PUBLIC METHODS
197
198     PRIVATE METHODS
246
247 }//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
        1 reference
15      public int KernelHeight{
16          get{ return _kernelHeight; }
17      }//end property
18
        1 reference
19      public int KernelWidth {
20          get { return _kernelWidth; }
21      }//end property
22
        1 reference
23      public BitmapMaker ImgPath {
24          get { return _imgPath; }
25      }//end property
26      #endregion
27
28      #region CONSTRUCTOR
        1 reference
29      public BitmapAscii(BitmapMaker imgPath, int kernelWidth, int kernelHeight)
30          _kernelWidth = kernelWidth;
31          _kernelHeight = kernelHeight;
32          _imgPath = imgPath;
33      }//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          }//end else
60      }//end event
```

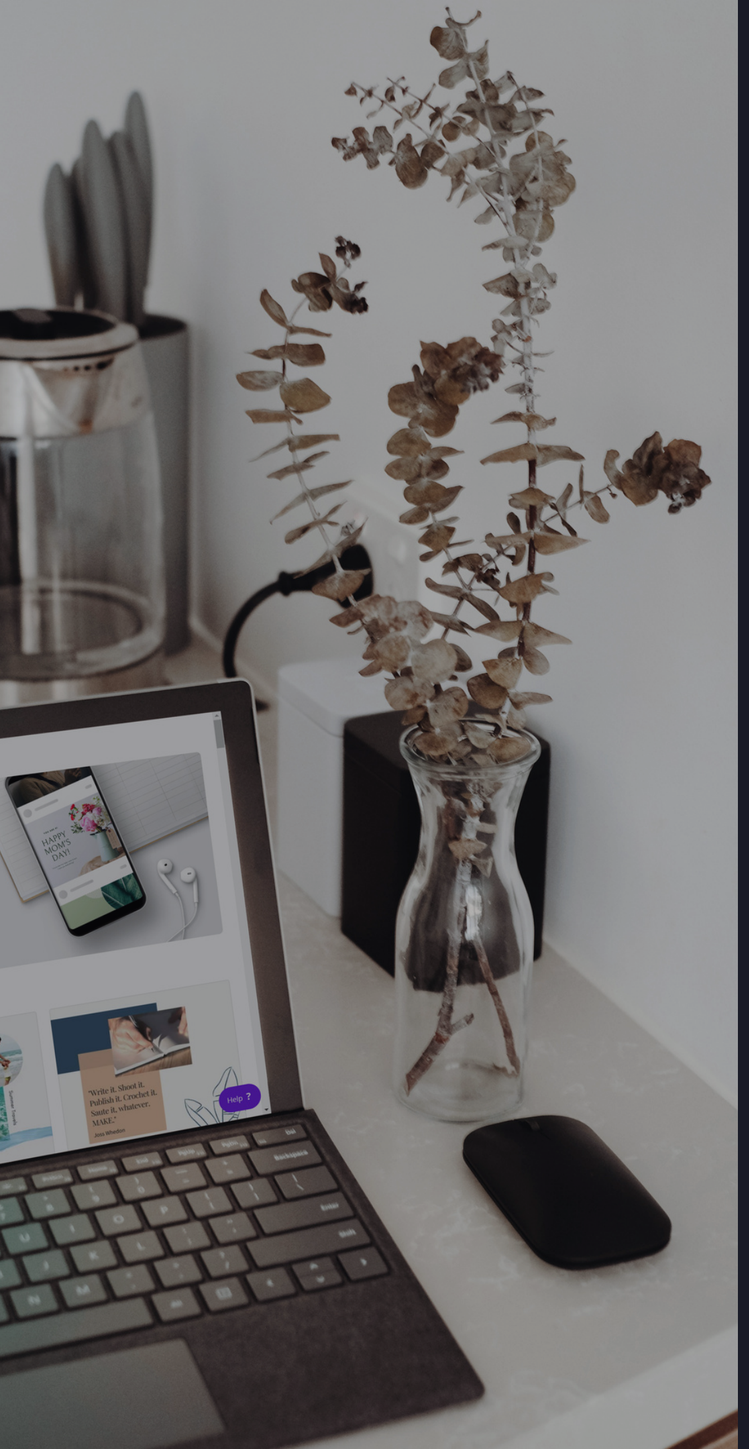
```
68      private string Asciitize(BitmapMaker imgSource, int kernelWidth, int kernelHeight) {
69          //Contains the ascii text version of the picture
70          string finalAsciiString = String.Empty;
71          double normalizedVal;
72          string asciiCharacter;
73
74          //Convert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

RIGHT: ASCII CONVERSION

LEFT: GREYSCALE CONVERSION

Part 3: Greyscale Conversion

```
41          public BitmapMaker ConvertToGreyScale(string imageFilePath) {
42              BitmapMaker greyBitmap = new BitmapMaker(imageFilePath);
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
```



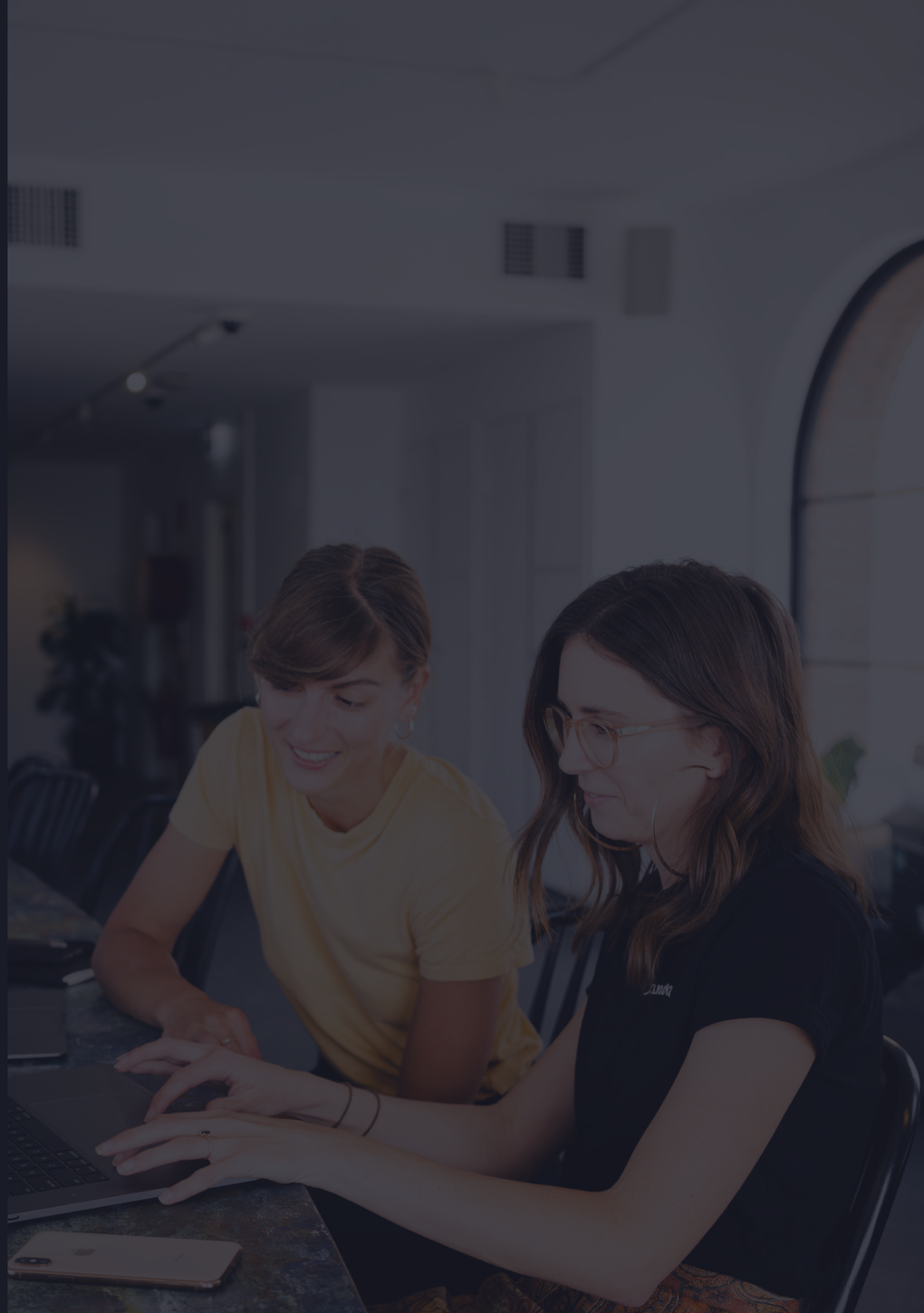
Part 3: Average Pixel and Normalized Value

```
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
        1 reference
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
```



Questions and Feedback


Thank you!