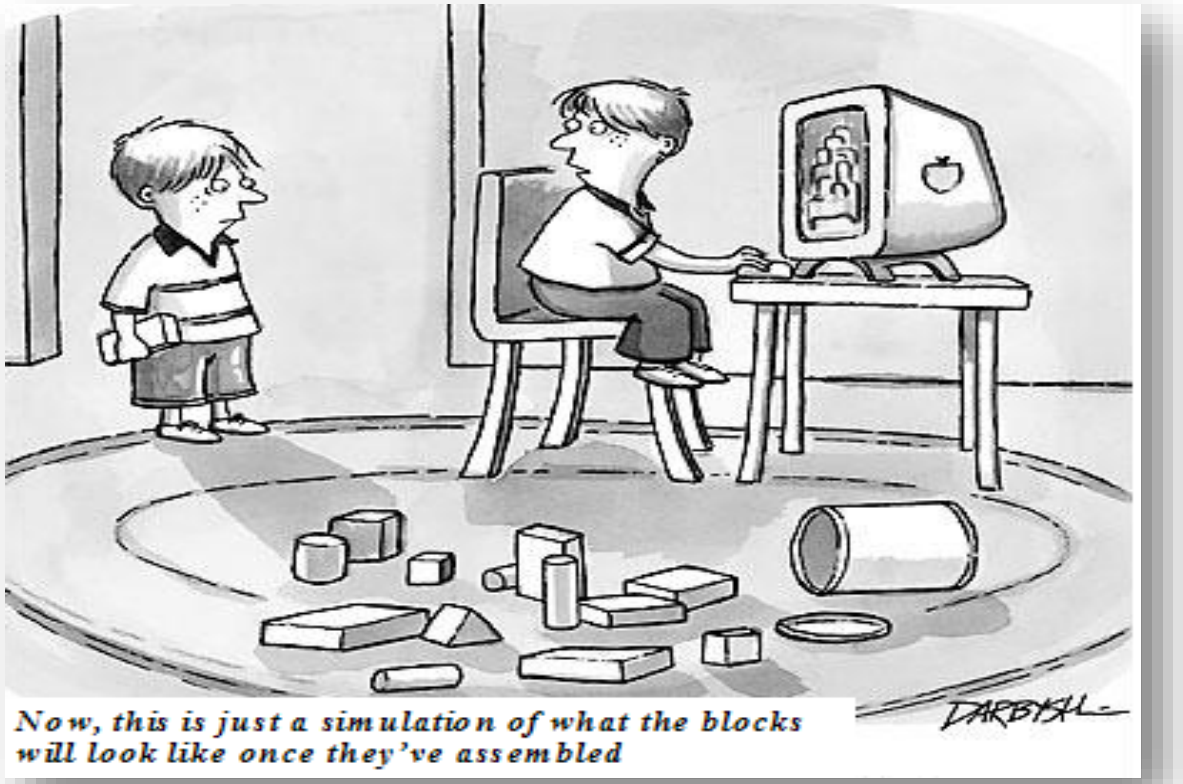
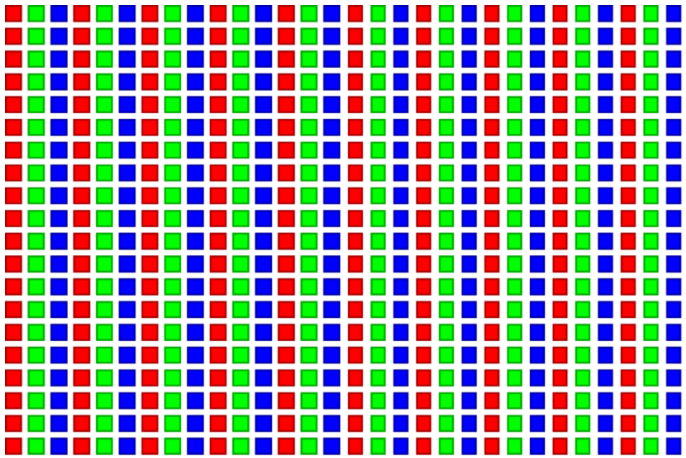
ECE569
Module 13

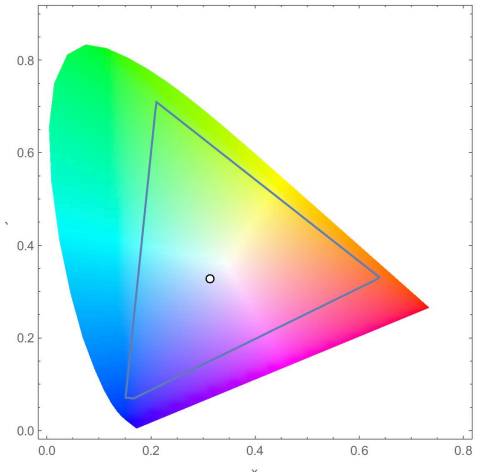


- Color space conversion

Case Study: Multi-dimensional grid kernel configurations with Color Space Conversion

- Each pixel in an image is an RGB value
- The format of an image's row is (r g b) (r g b) … (r g b)
  - RGB ranges are not distributed uniformly
- Many different color spaces, here we show the constants to convert to AdbobeRGB color space
- The vertical axis (y value) and horizontal axis (x value) show the fraction of the pixel intensity that should be allocated to G and B. The remaining fraction (1-y–x) of the pixel intensity that should be assigned to R
- The triangle contains all the representable colors in this color space





RGB to Grayscale Conversion

A grayscale digital image is an image in which the value of each pixel carries only intensity information.

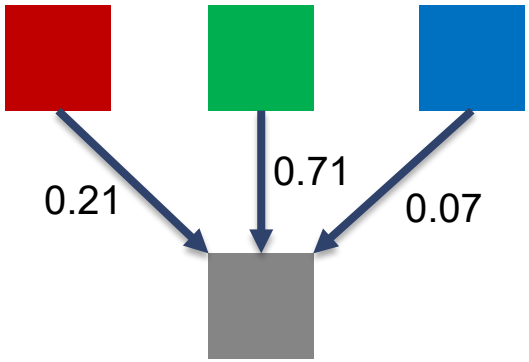
Medical Imaging

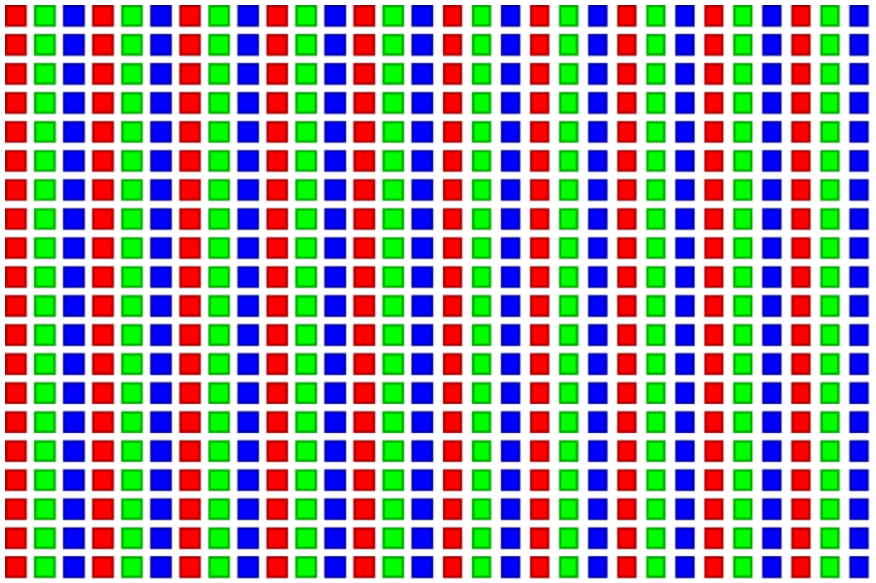
Image enhancement!

Task of each Thread

```
For each pixel (r g b) at (I, J) do:
    grayPixel[I,J] = 0.21*r + 0.71*g + 0.07*b
```

- just a dot product <[r,g,b],[0.21,0.71,0.07]> with the constants being specific to input RGB space



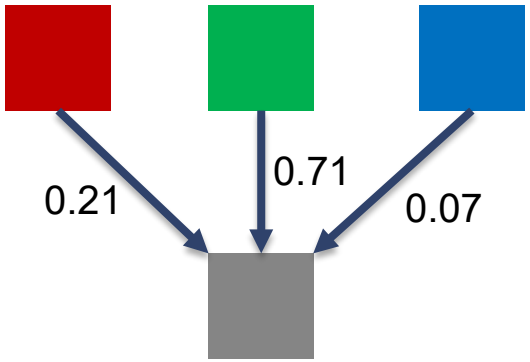


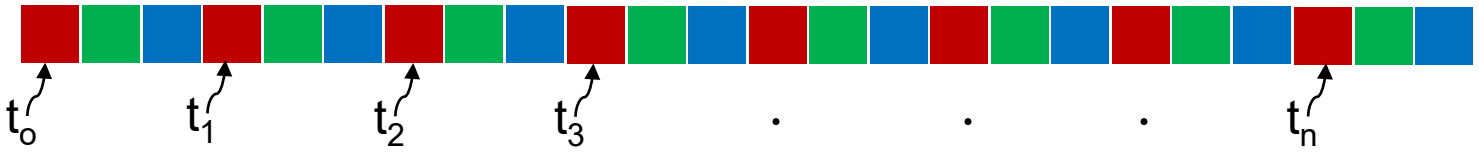
Assume 1D array with RGB components

- **Modify**

```
index = threadIdx.x + blockIdx.x * blockDim.x;
```

  – Such that subsequent threads are mapped to subsequent “r” components in the rgb array.



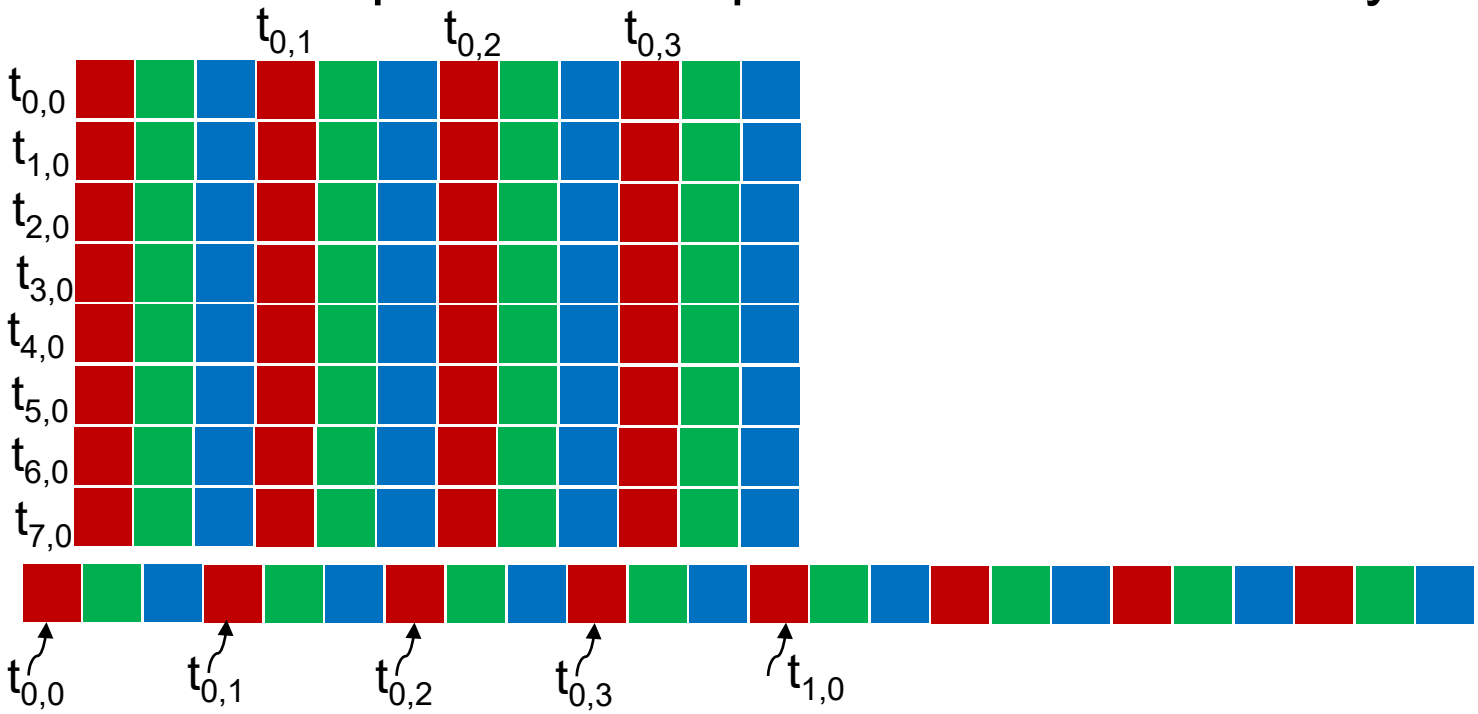


2D image (width,height) in RGB format

- **Modify**

```
Col = threadIdx.x + blockIdx.x * blockDim.x;
Row = threadIdx.y + blockIdx.y * blockDim.y;
Index = Row * width + Col;
```

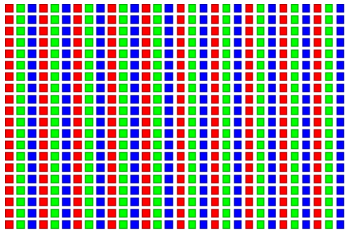
  – Such that subsequent threads are mapped to subsequent “r” components in the 2D array.



RGB to Grayscale Conversion Code

```
#define CHANNELS 3 // we have 3 channels corresponding to RGB
// The input image is encoded as unsigned characters [0, 255]
__global__ void colorConvert(unsigned char * grayImage,
                             unsigned char * rgbImage,
                             int width, int height) {
 int x = threadIdx.x + blockIdx.x * blockDim.x;  // x for column
 int y = threadIdx.y + blockIdx.y * blockDim.y;  // y for row

 if (x < width && y < height) {
```



```
 }
}
```

RGB to Grayscale Conversion Code

```
#define CHANNELS 3 // we have 3 channels corresponding to RGB
// The input image is encoded as unsigned characters [0, 255]
__global__ void colorConvert(unsigned char * grayImage,
                             unsigned char * rgbImage,
                             int width, int height) {
 int x = threadIdx.x + blockIdx.x * blockDim.x;
 int y = threadIdx.y + blockIdx.y * blockDim.y;

 if (x < width && y < height) {
   // get 1D coordinate for the grayscale image
   int grayOffset = y*width + x;
   // one can think of the RGB image having
   // CHANNEL times columns than the gray scale image
   int rgbOffset = grayOffset*CHANNELS;
   unsigned char r =  rgbImage[rgbOffset     ]; // red value for pixel
   unsigned char g = rgbImage[rgbOffset + 1]; // green value for pixel
   unsigned char b = rgbImage[rgbOffset + 2]; // blue value for pixel
   // perform the rescaling and store it
   // We multiply by floating point constants
   grayImage[grayOffset] = 0.21f*r + 0.71f*g + 0.07f*b;
 }
}
```

Next

- **Image blurring**
  - Slightly more challenging since we need to use neighboring pixels.