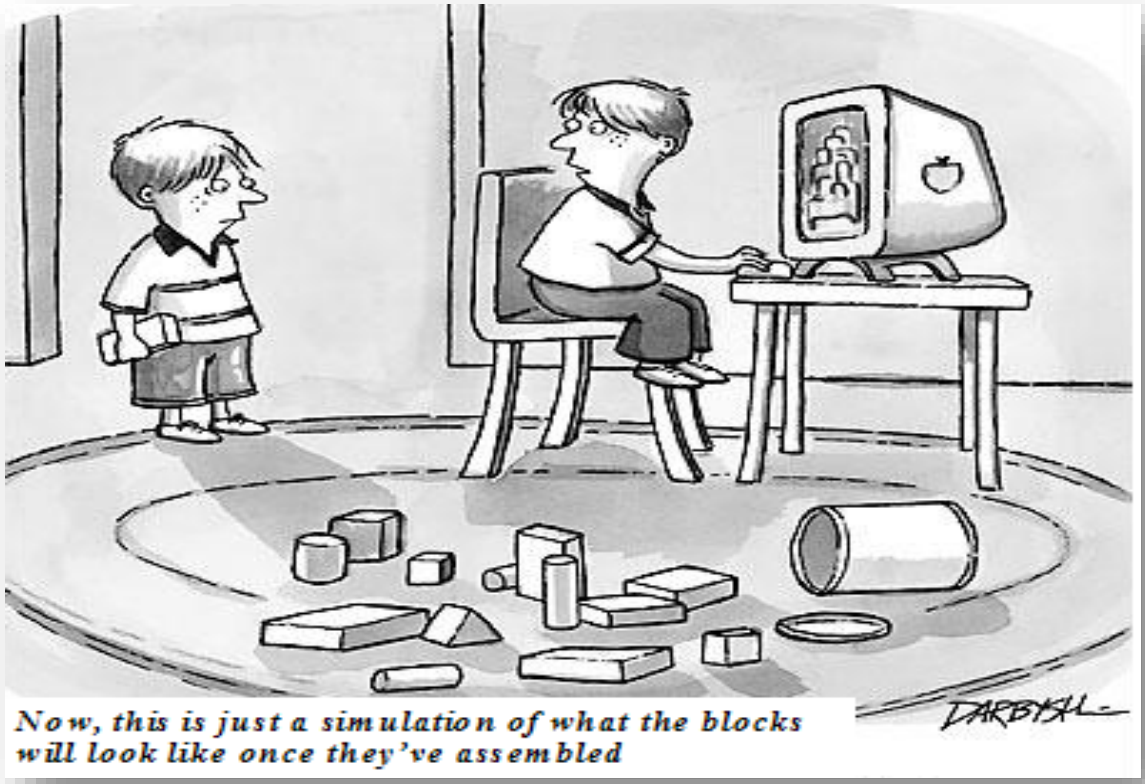
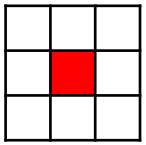
ECE569
Module 14



- Image blurring

Image Blurring

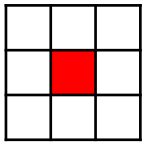
- Big picture impression



2D Kernel

```
__global__
 void blurKernel(unsigned char * in, unsigned char * out, int w, int h)
 {



}
```

2x**BLUR_SIZE** x 2xBLUR_SIZE



BLUR_SIZE: pixel distance
1 for 3x3 neighborhood

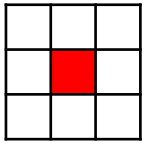
Need to map threads to data

-We will eventually use Row*Width+Column expression

-If (**i,j**) indicates the center pixel , BLUR_SIZE = 1, we visit (**i-1,j-1**) (top left corner) to (**i+1,j+1**) (bottom right corner)

```
__global__
 void blurKernel(unsigned char * in, unsigned char * out, int w, int h) {

      // write the boundary condition

 }
```

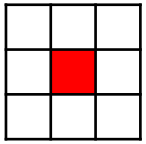
2x**BLUR_SIZE** x 2xBLUR_SIZE



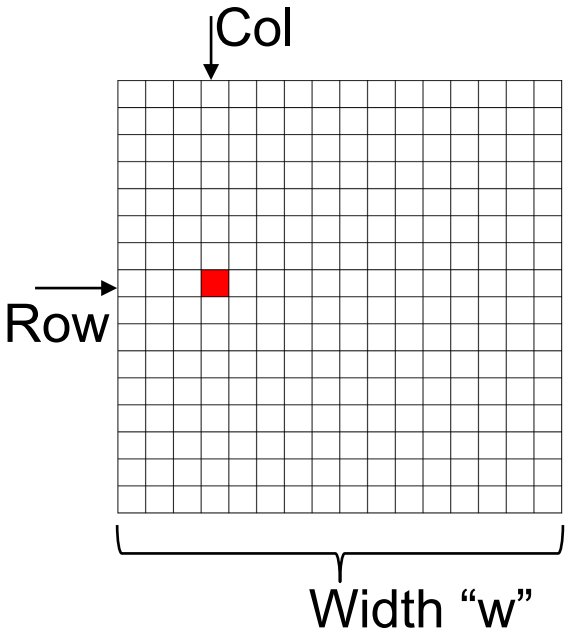
BLUR_SIZE: pixel distance
1 for 3x3 neighborhood

```
__global__
 void blurKernel(unsigned char * in, unsigned char * out, int w, int h) {
    int Col  = blockIdx.x * blockDim.x + threadIdx.x;
    int Row  = blockIdx.y * blockDim.y + threadIdx.y;
    if (Col < w && Row < h) {

       // write the nested for loop and its body

    }
 }
```

2x**BLUR_SIZE** x 2xBLUR_SIZE



BLUR_SIZE: pixel distance
1 for 3x3 neighborhood



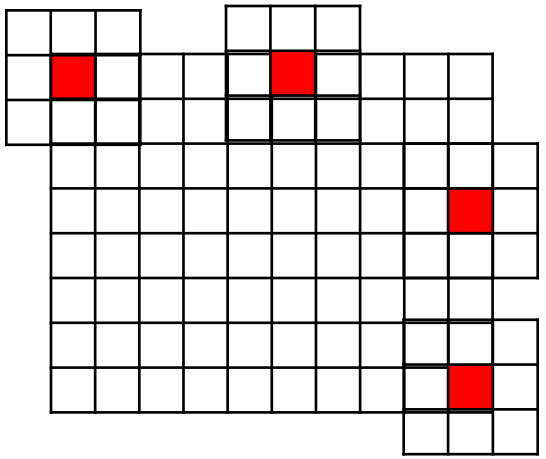
```
__global__
 void blurKernel(unsigned char * in, unsigned char * out, int w, int h) {
    int Col  = blockIdx.x * blockDim.x + threadIdx.x;
    int Row  = blockIdx.y * blockDim.y + threadIdx.y;
    if (Col < w && Row < h) {
    int pixVal=0;
    int pixels=0;
       // Get the average of the surrounding 2xBLUR_SIZE x 2xBLUR_SIZE box
       for(int blurRow = -BLUR_SIZE; blurRow < BLUR_SIZE+1; ++blurRow) {
          for(int blurCol = -BLUR_SIZE; blurCol < BLUR_SIZE+1; ++blurCol) {

                pixVal += in[ (Row+blurRow) * w + (Col+blurCol) ];
                pixels++; // Keep track of number of pixels in the accumulated total

          }
       }
       // Write our new pixel value out

    }
 }
```

```
__global__
 void blurKernel(unsigned char * in, unsigned char * out, int w, int h) {
    int Col  = blockIdx.x * blockDim.x + threadIdx.x;
    int Row  = blockIdx.y * blockDim.y + threadIdx.y;
    if (Col < w && Row < h) {
    int pixVal=0;
    int pixels=0;
       // Get the average of the surrounding 2xBLUR_SIZE x 2xBLUR_SIZE box
       for(int blurRow = -BLUR_SIZE; blurRow < BLUR_SIZE+1; ++blurRow) {
          for(int blurCol = -BLUR_SIZE; blurCol < BLUR_SIZE+1; ++blurCol) {

                pixVal += in[ (Row+blurRow) * w + (Col+blurCol) ];
                pixels++; // Keep track of number of pixels in the accumulated total

          }
       }
       // Write our new pixel value out
       out[Row * w + Col] = (unsigned char)(pixVal / pixels);

 }
```

This white space must be left for something. What did we miss?

```
__global__
 void blurKernel(unsigned char * in, unsigned char * out, int w, int h) {
    int Col  = blockIdx.x * blockDim.x + threadIdx.x;
    int Row  = blockIdx.y * blockDim.y + threadIdx.x;
    if (Col < w && Row < h) {
    int pixVal=0;
    int pixels=0;
       // Get the average of the surrounding 2xBLUR_SIZE x 2xBLUR_SIZE box
       for(int blurRow = -BLUR_SIZE; blurRow < BLUR_SIZE+1; ++blurRow) {
          for(int blurCol = -BLUR_SIZE; blurCol < BLUR_SIZE+1; ++blurCol) {

                pixVal += in[ (Row+blurRow) * w + (Col+blurCol) ];
                pixels++; // Keep track of number of pixels in the accumulated total

          }
       }
       // Write our new pixel value out
       out[Row * w + Col] = (unsigned char)(pixVal / pixels);

 }
```

Corner and edge cases need to be checked



```
__global__
 void blurKernel(unsigned char * in, unsigned char * out, int w, int h) {
    int Col  = blockIdx.x * blockDim.x + threadIdx.x;
    int Row  = blockIdx.y * blockDim.y + threadIdx.y;
    if (Col < w && Row < h) {
       int pixVal = 0;
       int pixels = 0;
       // Get the average of the surrounding 2xBLUR_SIZE x 2xBLUR_SIZE box
       for(int blurRow = -BLUR_SIZE; blurRow < BLUR_SIZE+1; ++blurRow) {
          for(int blurCol = -BLUR_SIZE; blurCol < BLUR_SIZE+1; ++blurCol) {
             int curRow = Row + blurRow;
             int curCol = Col + blurCol;
             // Verify we have a valid image pixel
             if(curRow > -1 && curRow < h && curCol > -1 && curCol < w) {
                pixVal += in[curRow * w + curCol];
                pixels++; // Keep track of number of pixels in the accumulated total
             }
          }
       }
       // Write our new pixel value out
       out[Row * w + Col] = (unsigned char)(pixVal / pixels);
    }
 }
```

Next

- **CUDA Programming Model**