600086 Lab Book – Daniel Bates

# Week 1 – Lab 1

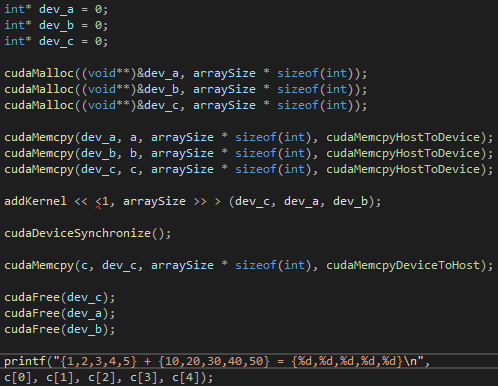
Date: 2nd Feb 2022

## Q1. Setting up CUDA

### Question:

Set up the default CUDA program. Create a solution that adds 2 arrays together on the GPU

### Solution:



### Test data:

a = {1,2,3,4,5} b = {10,20,30,40,50}

### Sample output:

{1,2,3,4,5} + {10,20,30,40,50} = {11,22,33,44,55}

### Reflection:

The process can be broken down into 5 steps

1. Allocate memory
2. Copy data to GPU
3. Perform addition
4. Copy results to CPU
5. Release GPU buffers

### Metadata:

GPU vector addition

### Further information:

N/A

## Q2. CUDA error checking

### Question:

Add CUDA error checking

### Solution:



\*Snippet own code (the whole thing takes up too much space)

### Test data:

N/A

### Sample output:

N/A

### Reflection:

Error checking seems really simple to implement this way but I think using exceptions would be a better way to do it

### Metadata:

Errors

### Further information:

N/A

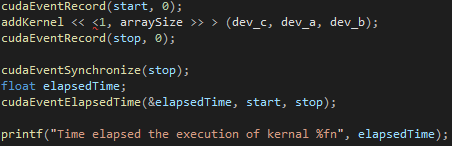
## Q3. Check time range

### Question:

Create a timer for the kernel execution

### Solution:





### Test data:

N/A

### Sample output:

N/A

### Reflection:

Pretty simple to create a timer

### Metadata:

Timer

### Further information:

N/A

# Week 2 – Lab 2

Date: 9th Feb 2022

## Q1. Understand the block and thread indices

### Question:

List the values for the built-in variables threadIdx.x and blockIdx.x corresponding to the given thread configurations used for executing the kernel addKernel( ) function on GPU

### Solution:

N/A

### Test data:



### Sample output:



### Reflection:

I think the first number in “addKernel << <1, 5 >> > (dev\_c, dev\_a, dev\_b);” represents the number of blocks and the second represents the size of the blocks. So, in this case there is 1 block that has 5 threads.

### Metadata:

Understanding threads and blocks

### Further information:

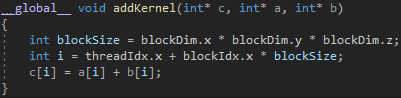
The lab sheet is a bit confusing to read to me, wasn’t really sure what was being asked of me

## Q2. Find vector addition using multiple 1D thread blocks

### Question:

For the vector addition problem considered in the CUDA template, find the solution based on the given thread configurations.

### Solution:



### Test data:



### Sample output:



### Reflection:

A 1D vector with multiple blocks has an index equal to threadIdx.x + blockIdx.x \* blockSize

### Metadata:

Multiple 1D threads

### Further information:

N/A

### Q3. Understand the thread indices for 2D blocks

### Question:

List the values for the built-in variables threadIdx.x and threadIdx.y corresponding to given thread configurations used for executing the kernel addKernel( ) function on GPU

### Solution:

N/A

### Test data:



### Sample output:



### Reflection:

Only the first index of the dim3 is currently used.

### Metadata:

Indices for 2D blocks

### Further information:

N/A

### Q4. Find vector addition using one 2D thread block

### Question:

For the vector addition problem considered in the CUDA template, find the solution based on the given thread configurations

### Solution:



### Test data:



### Sample output:



### Reflection:

The threads index in a 2D thread block is equal to threadIdx.x + threadIdx.y \* blockDim.x

### Metadata:

One 2D block

### Further information:

N/A

### Q5. Find vector addition using multiple 2D thread blocks

### Question:

For the vector addition problem considered in the CUDA template, find the solution based on the given thread configurations

### Solution:



### Test data:



### Sample output:



### Reflection:

The thread index for multiple 2D blocks is equal to:

(threadIdx.x + blockIdx.x \* blockDim.x) + (threadIdx.y + blockIdx.y \* blockDim.y)

### Metadata:

Multiple 2D blocks

### Further information:

While I have successfully done the 2D block section of the lab I don’t feel as though I thoroughly understand it yet. Will ask if it can be explained to me again next lab

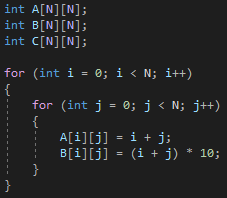
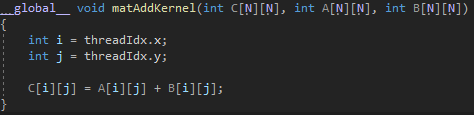
### Q6. Matrix addition

### Question:

Write a CUDA program to find the addition of two matrices

### Solution:







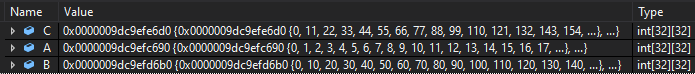


\*I’ve just included key parts that I think are important

### Test data:

N/A

### Sample output:



### Reflection:

For the cudaMalloc, cudaMemcpy etc the size is as follows:



This task was very confusing I don’t really understand the dA,dB and dC variables and stuff like that

### Metadata:

Matrix addition

### Further information:

I think this task didn’t have enough guidance on the actual lab sheet, I had to google how to do it eventually as I ran out of time in the lab and even after rewatching the lectures this week multiple times I still couldn’t understand what I was doing wrong

# Week 3 – Lab 3

Date: 16/02/2022

## Q1. Vector dot-product

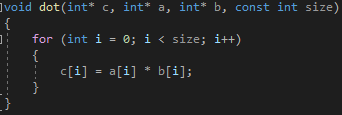
### Question:

**Part 1 –** Write a C++ program to calculate the dot-product of two vectors

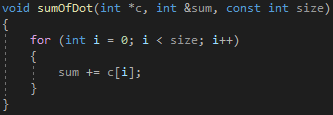
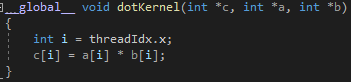
**Part 2 –** Write a C++ program to calculate the dot-product of two vectors on the GPU & calculate the sum of the dot-product of the two vectors on the CPU

### Solution:

**Part 1 –**

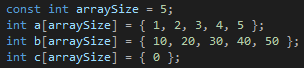
****

**Part 2 –**

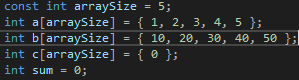
****

### Test data:

**Part 1 –**

****

**Part 2 –**



### Sample output:

**Part 1 –**

****

**Part 2 –**

****

### Reflection:

This is pretty much the same as lab 1

### Metadata:

Dot-product

### Further information:

N/A

## Q2. Vector dot-product using unified memory

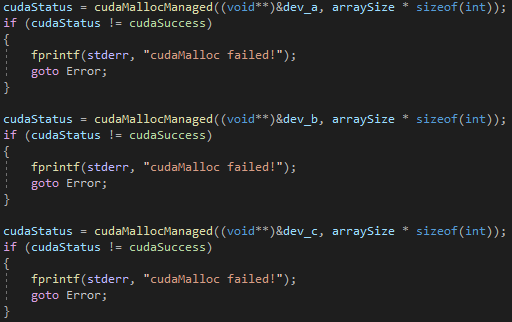
### Question:

**Part 1 –** Create a CUDA program that calculates the vector dot-product using managed memory

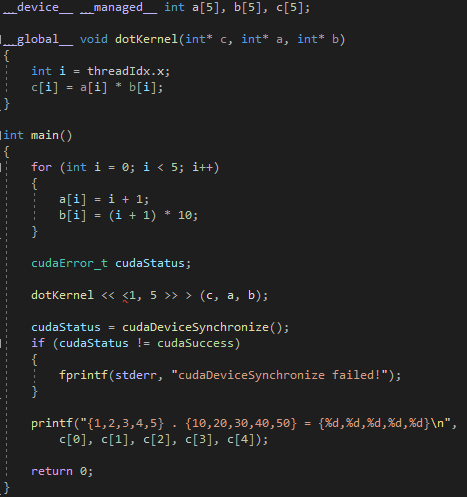
**Part 2 –** Create a CUDA program that calculates the vector dot-product using GPU-declared \_\_managed\_\_ memory

### Solution:

**Part 1 –**

****

**Part 2 –**

****

### Test data:

N/A

### Sample output:

**Part 1 –**

****

**Part 2 –**

****

### Reflection:

This way of doing things seems to cut out a lot of code compared to the methods that have been previously used in the labs

### Metadata:

Unified memory

### Further information:

N/A

## Q3. Vector dot-product using shared Memory

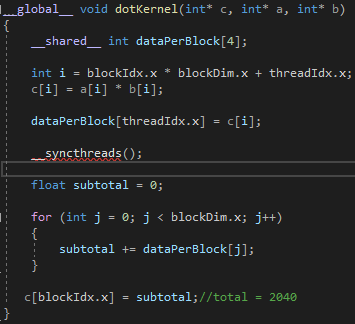
### Question:

**Part 1 –** Analyse the process given in the lab sheet and identify areas where thread execution needs to be synchronized by calling CUDA function: \_\_syncthreads();

**Part 2 –** Consider different thread configurations, for example, <<>>, <<>>, <<>> and observe if the given program can calculate the vector dot-product correctly. If not, analyse the issues and consider how to fix them.

### Solution:

**Part 1 –**

****

**Part 2 –** N/A

### Test data:

**Part 1 –** N/A

**Part 2 –**

****

****

****

### Sample output:

**Part 1 –**



**Part 2 –**







### Reflection:

**Part 1 –** I don’t really understand what it is doing I get the same results with and without \_\_syncthreads()

**Part 2 –** The program only gets the right sum with the 2,4 layout. This could fixed by making the number of blocks and how many values of the c vector added together as the same.

### Metadata:

Shared memory

### Further information:

N/A

# Week 4 – Lab 4

Date: 23/02/2022

## Q1. Create an OpenGL-CUDA program based on a CUDA SDK sample

### Question:

Set up the project and make sure it compiles

### Solution:

N/A

### Test data:

N/A

### Sample output:

### Reflection:

The hidden files need to made visible in the file explorer

### Metadata:

CUDA SDK

### Further information:

N/A

## Q2. Understand pixel colour

### Question:

Edit the line “d\_output[i] = make\_uchar4(c \* 0xff, c \* 0xff, c \* 0xff, 0);” in various way to understand pixel colouring works

### Solution:

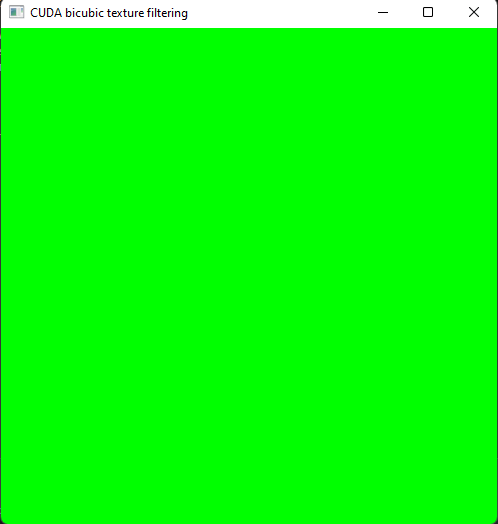
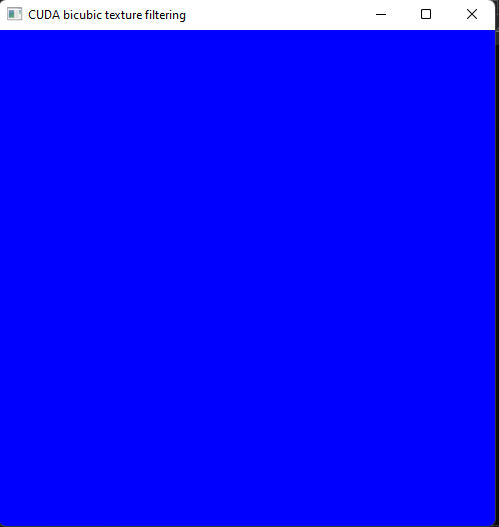
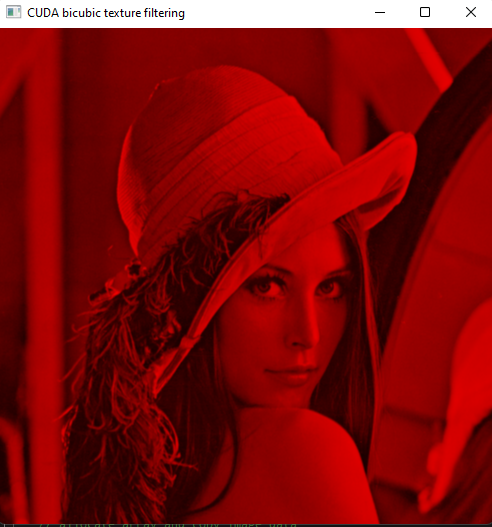
N/A

### Test data:

### 

### 

### Sample output:

### Reflection:

For some reason colours are ordered BGR instead of RGB which is what I’m used to

### Metadata:

Pixel colours

### Further information:

N/A

## Q3. Image transformation

### Question:

**Part 1 –** Translate the image

**Part 2 –** Scale the image

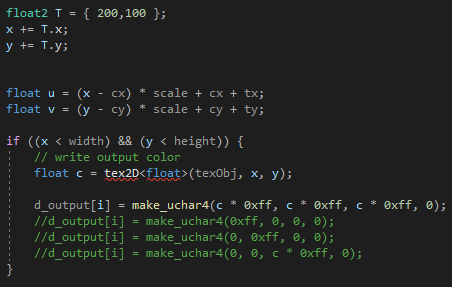
**Part 3 –** Rotate the image

**Part 4 –** Scale by position

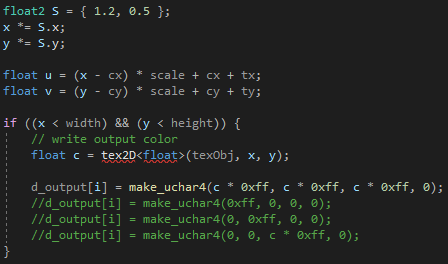
**Part 5 –** Rotate by image centre

### Solution:

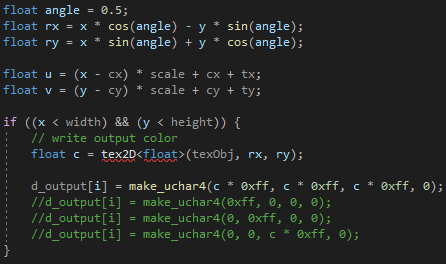
**Part 1 –**

****

**Part 2 –**

****

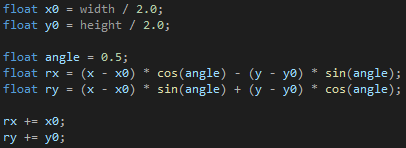
**Part 3 –**



**Part 4 –**



**Part 5 –**

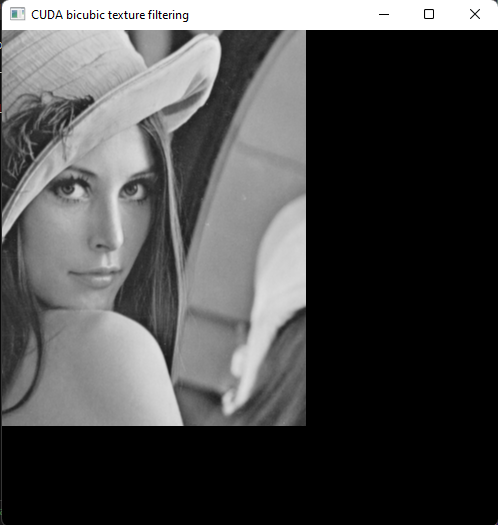


### Test data:

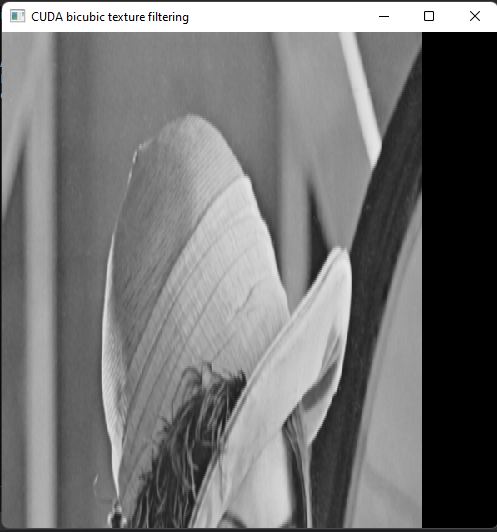
**All parts –** N/A

### Sample output:

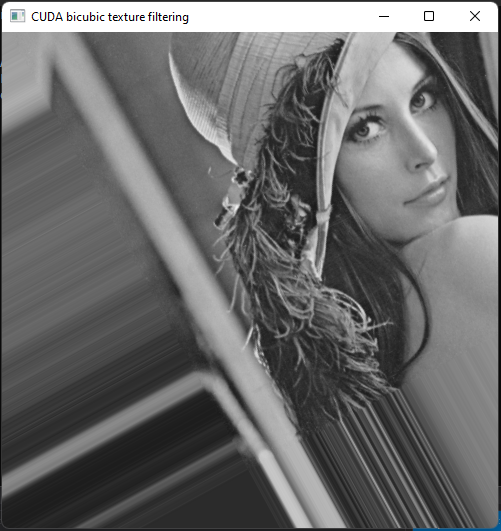
**Part 1 –**

****

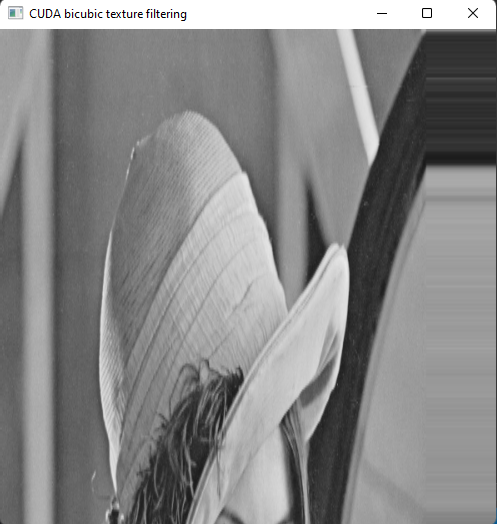
**Part 2 –**

****

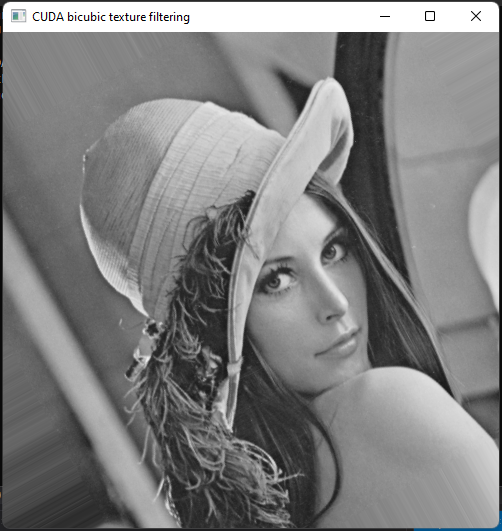
**Part 3 –**

****

**Part 4 –**

****

**Part 5 –**

****

### Reflection:

These were all pretty straight forward apart for the rotating by image centre

### Metadata:

Transformations

### Further information:

N/A

## Q4. Image smoothing

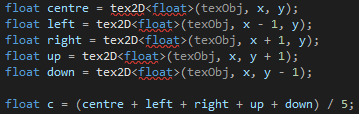
### Question:

**Part 1 –** Implement image smoothing

**Part 2 –** Modify thread configurations and observe performance

### Solution:

**Part 1 –**

****

### Test data:

**Part 1 –** N/A

### Sample output:

**Part 1 –**



**Part 2 –**

|  |  |
| --- | --- |
| **blocksize** | **Time** |
| 16x16 | 0.365568n |
| 32x32 | 0.437248n |
| 24x24 | 0.390144n |

### Reflection:

**Part 1 –** This was pretty straight forward

**Part 2 –** The threads failed to execute when I tried 64x64 configuration

### Metadata:

Smooth

### Further information:

N/A

# Week 5 – Lab 5

Date: 02/03/2022

## Q1. A simple matrix multiplication program in CUDA using one thread block

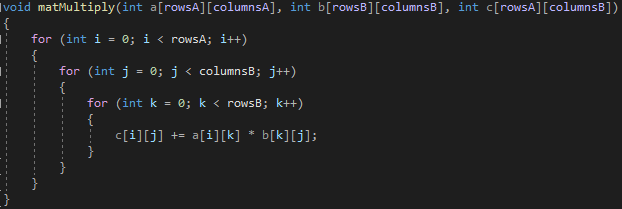
### Question:

**Part 1 –** Write a matrix multiplication function in c++

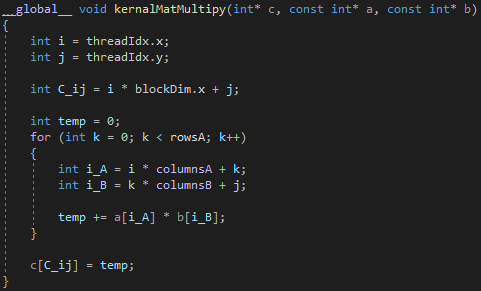
**Part 2 –** Write a matrix multiplication function in CUDA using 1 thread block

### Solution:

**Part 1 –**

****

**Part 2 –**



### Test data:

A = (1,2,3,4,5,6,7,8,9) B = (10,11,12,13,14,15,16,17,18)

### Sample output:

C = (84,90,96,201,216,231,318,342,366)

### Reflection:

I don’t really understand why you wouldn’t use 2D arrays

### Metadata:

Matrix Multiply

### Further information:

N/A

## Q2. Compare the performance of the CUDA solution against the CPU solution

### Question:

**Part 1 –** 8x8

**Part 2 –** 32x32

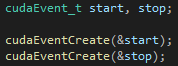
**Part 3 –** 256x256

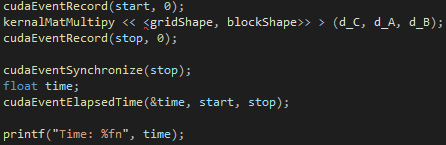
**Part 4 –** 512x512

**Part 5 –** 1024x1024

### Solution:

**All Parts –**

****

****

### Test data:

N/A

### Sample output:

|  |  |
| --- | --- |
| **Part** | **Time** |
| 1 – CPU | 0.007168n |
| 1 – GPU | 0.036864n |
| 2 – CPU | 0.006144n |
| 2 – GPU | 0.059392n |
| 3 – CPU | 0.004192n |
| 3 – GPU | 0.008224n |
| 4 – CPU | Stack overflow |
| 4 – GPU | Stack overflow |
| 5 – CPU | Stack overflow |
| 5 – GPU | Stack overflow |

### Reflection:

My timings are really weird and inconsistent but I have no clue why. I had similar issues with a rust lab session.

### Metadata:

Performance

### Further information:

N/A

# Week 6 – Lab 6

Date: 09/03/2022

## Q1. Set up a virtual canvas and draw on it an image in CUDA

### Question:

Set up the project then make the image green

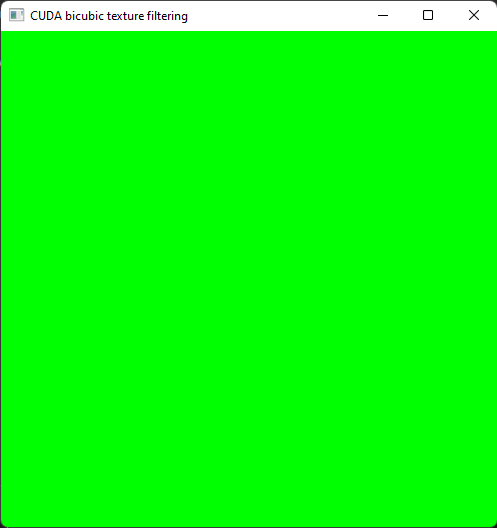
### Solution:



### Test data:

N/A

### Sample output:



### Reflection:

This was really straight forward

### Metadata:

Set up

### Further information:

N/A

## Q2. Drawing a checkboard in CUDA

### Question:

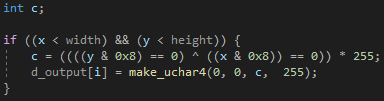
**Part 1 –** Modify the code to produce a checkerboard image

**Part 2 –** Modify the code to produce larger red-blocks in the checkerboard

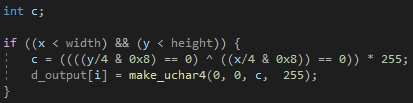
**Part 3 –** Modify the code to produce a red disc

### Solution:

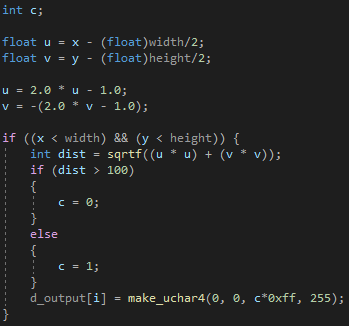
**Part 1 –**

****

**Part 2 –**

****

**Part 3 –**

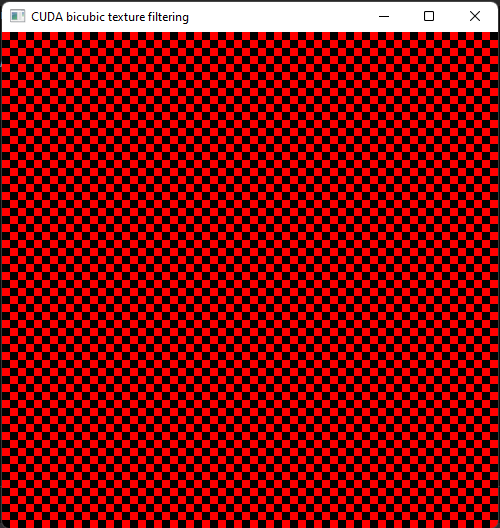
****

### Test data:

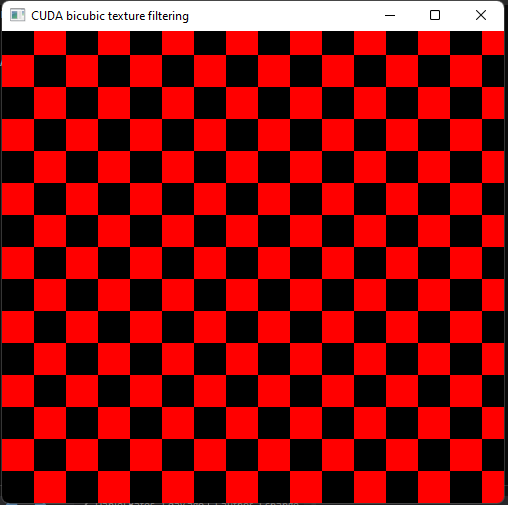
**All Parts –** N/A

### Sample output:

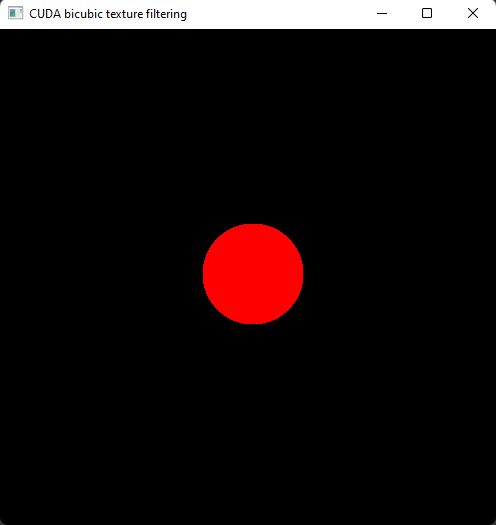
**Part 1 –**



**Part 2 –**

****

**Part 3 –**

****

### Reflection:

The circle drawing took me a while to do but I think I was just over complicating it. Other than that, everything else was very straight forward

### Metadata:

Checkers & Discs

### Further information:

N/A

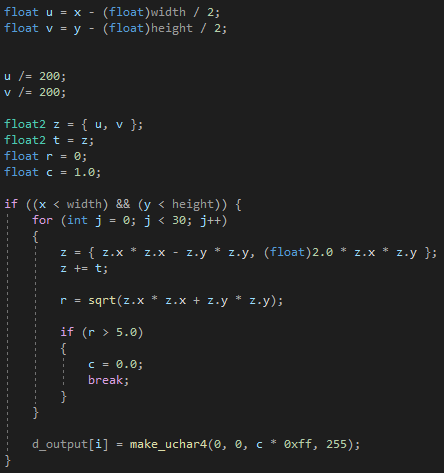
## Q3. Drawing the Mandelbrot and Julia Sets

### Question:

Implement Mandelbrot and Julia sets

### Solution:

**Mandelbrot Set –**



**Julia Set –**

Same as Mandelbrot set except the following:

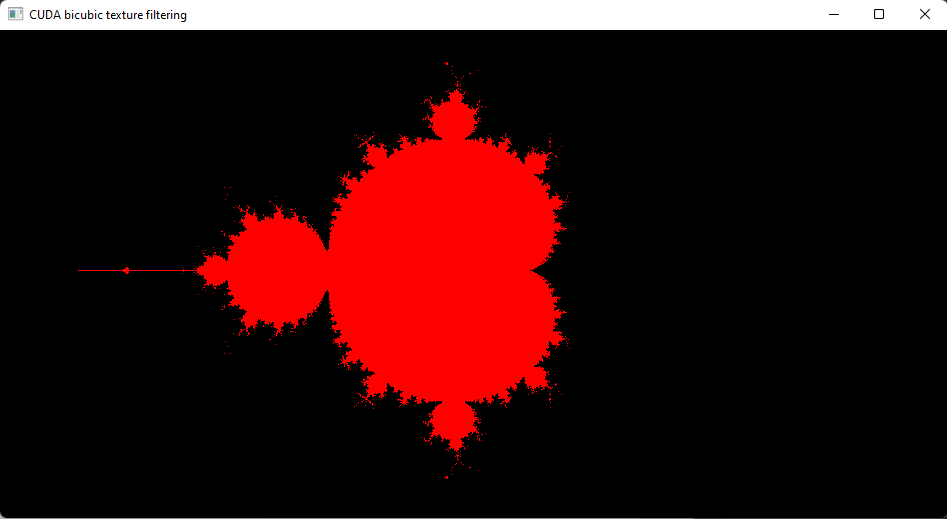


### Test data:

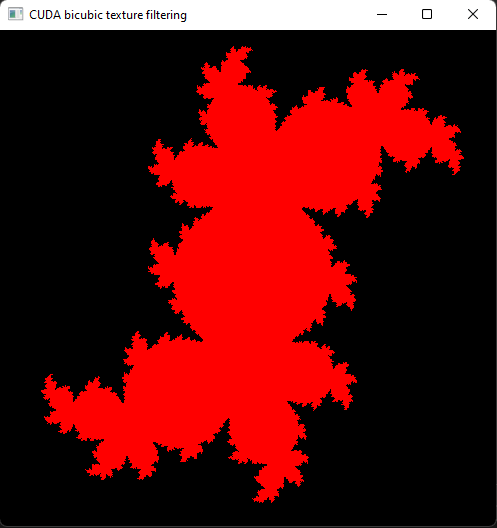
N/A

### Sample output:

**Mandelbrot Set –**



**Julia Set –**



### Reflection:

Fairly straightforward but to scale the image size I have to divide the u and v by 200 instead of multiply by 4.0 and I’m not sure why this was the case.

### Metadata:

Mendelbrot & Julia

### Further information:

N/A

# Week 7 – Lab 7

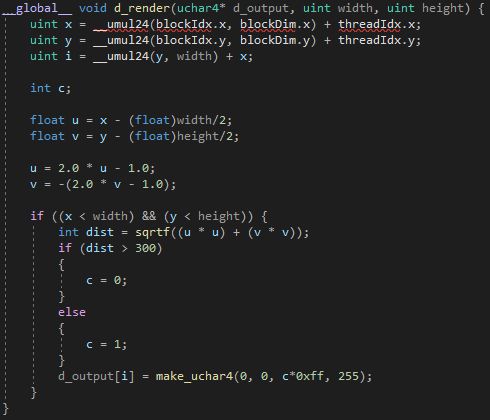
Date: 16/03/2022

## Q1. Drawing based on a canvas of size [-1, 1]x[-1, 1]

### Question:

Draw a disc based on pixel coordinates defined in float type variables in [-1, 1]x[-1, 1]

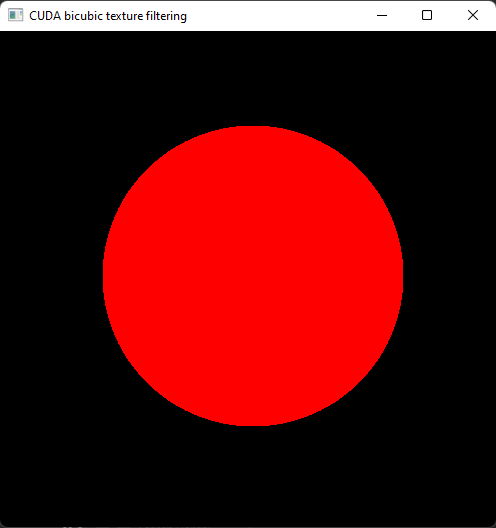
### Solution:



### Test data:

N/A

### Sample output:



### Reflection:

This was just repeating a task of lab 6 so I just copied the code I used for task 4 of lab 6

### Metadata:

Drawing Disc

### Further information:

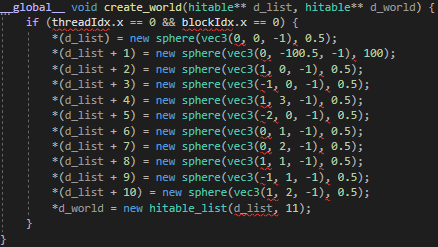
N/A

## Q2. Write a simple ray caster

### Question:

Write a simple ray caster by implementing the code given in the lab sheet. Then add ten new spheres to the world

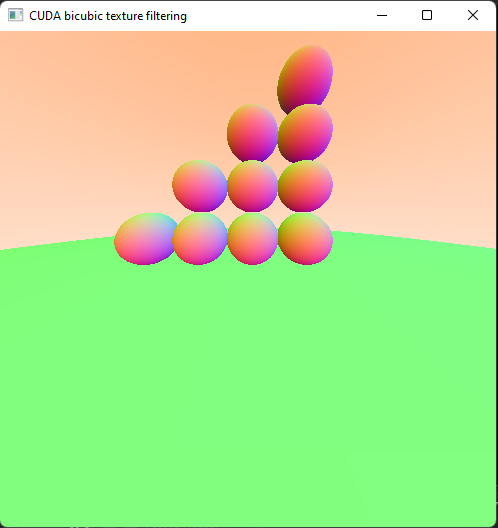
### Solution:



### Test data:

N/A

### Sample output:



### Reflection:

I’m not sure what the following line is doing:



I think its serving as a skybox but I’m not sure.

### Metadata:

Spheres

### Further information:

N/A

# Week 8 – Lab 8

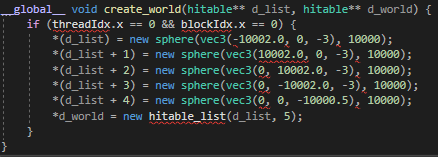
Date: 23/03/2022

## Q1. Draw a box without front wall

### Question:

Using the spheres create a box without a front wall in the create world method

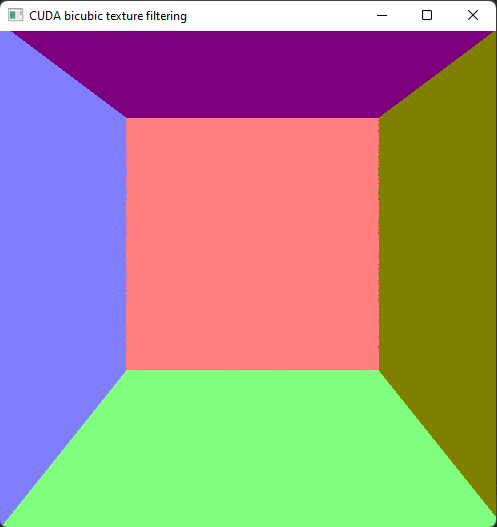
### Solution:



### Test data:

N/A

### Sample output:



### Reflection:

This was very simple to implement. I’m not sure how to implement the other way the lab sheet mentions though.

### Metadata:

Box

### Further information:

N/A

## Q2. Free motion animation

### Question:

Draw a ball rotating by the centre of the box

### Solution:

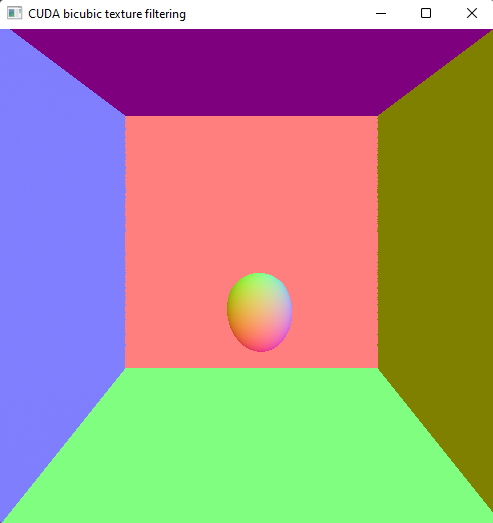




### Test data:

N/A

### Sample output:



### Reflection:

The rotating sphere becomes distorted into and egg shape but I’m not sure why this is

### Metadata:

Rotation

### Further information:

N/A

## Q3. Ball-box walls collision animation

### Question:

Animate a ball in motion with an initial velocity that is reflected when the ball collides with a wall. When a collision takes place the ball should also change colour

### Solution:

### Test data:

### Sample output:

### Reflection:

I can’t get it to work properly at all and the way it was explained to me to do in the lab sessions causes multiple errors which means the code won’t compile. I’m not sure if this is because I’m on version 11.6 instead of 11.5

### Metadata:

### Further information: