# 600086 Lab Book

# Week 2 – CUDA Lab 4. CUDA OpenGL Interoperability & Image processing

Date: 24th Feb 2022

## Exercise 1. Create an OpenGL-CUDA program based on a CUDA SDK sample

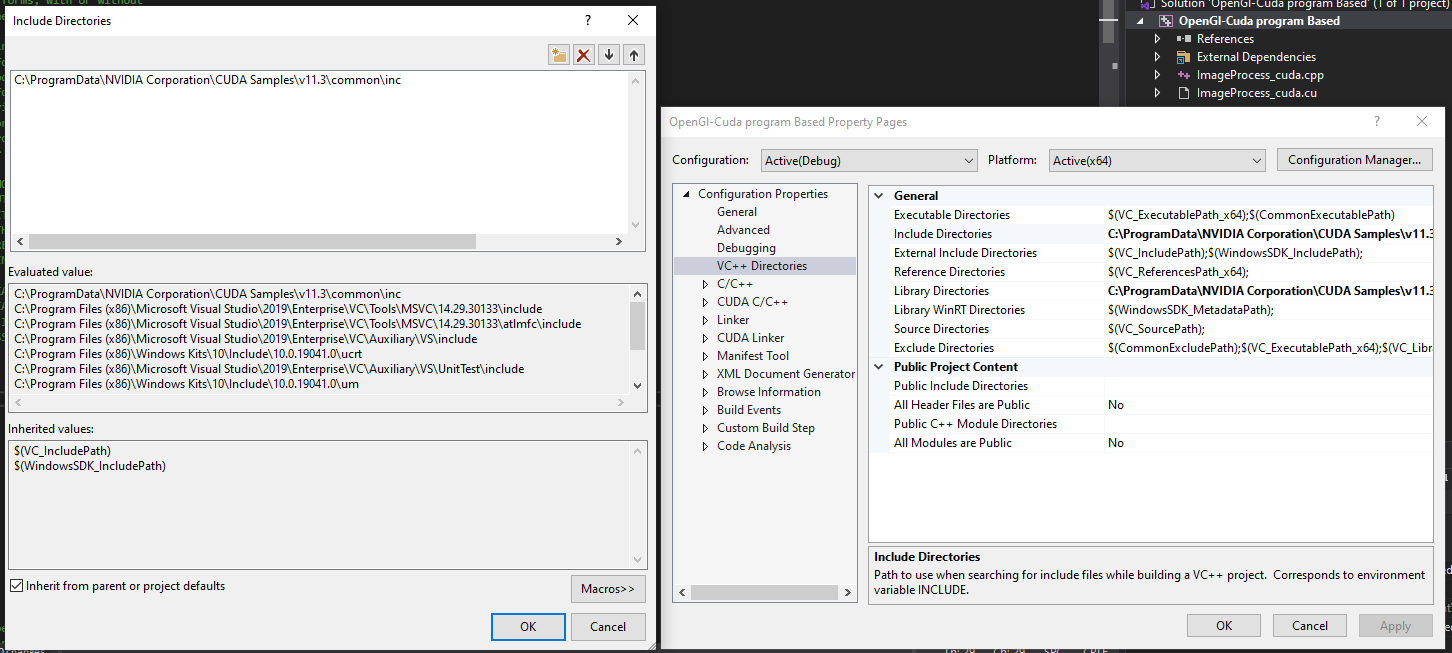
### Question:

Create an OpenGL-CUDA program based on a CUDA SDK sample

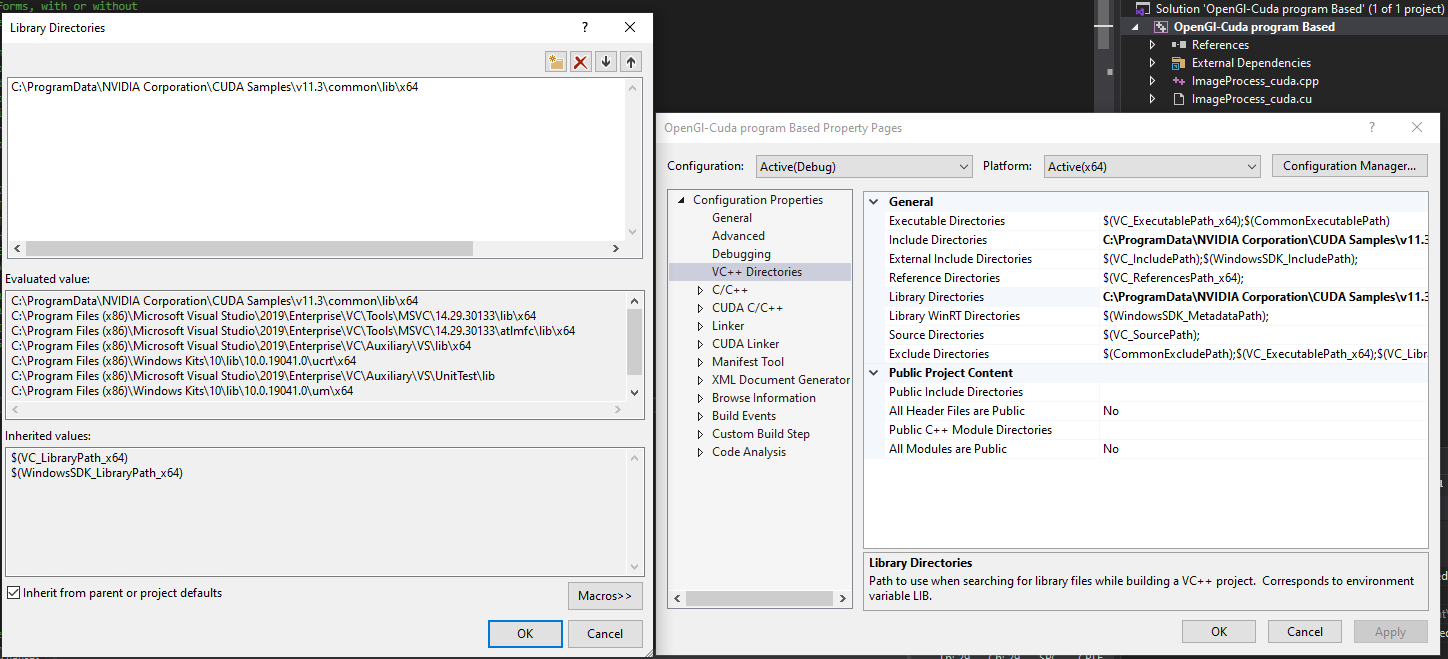
### Solution:

No sample code to show

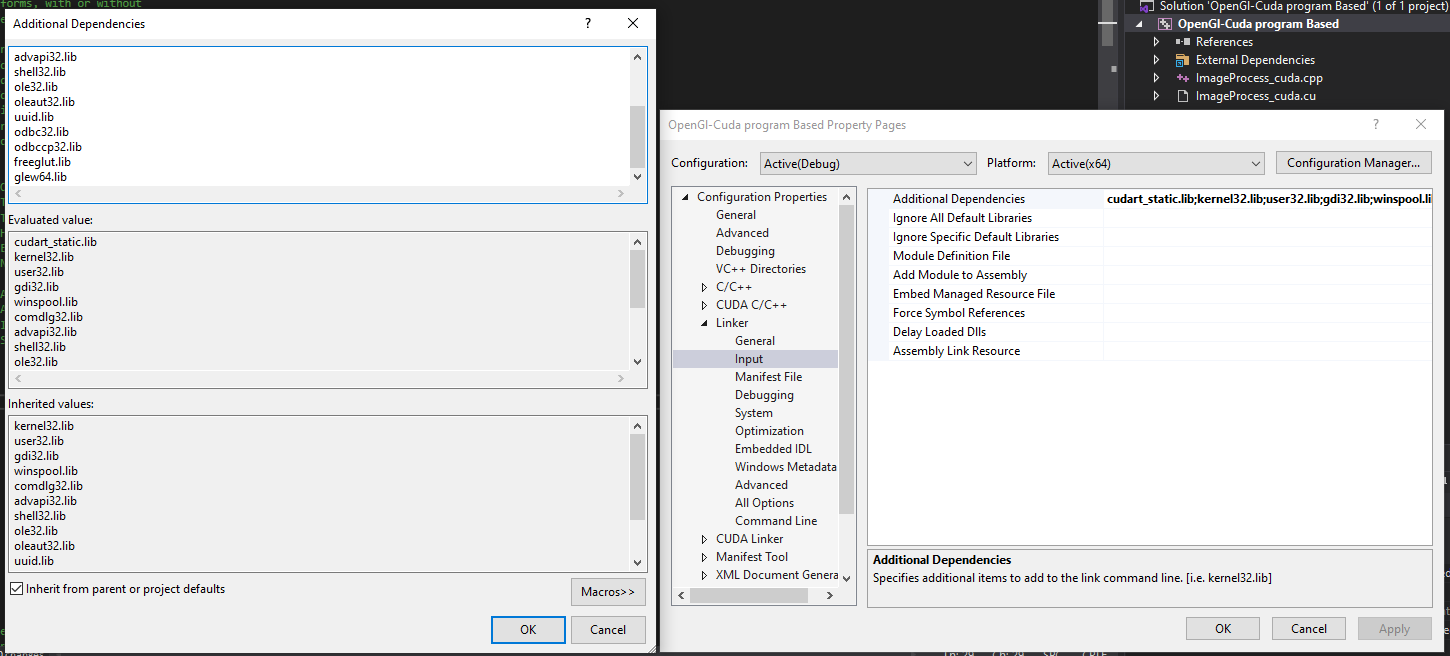
Step 1 : adding include directories to project



Step 2 : adding Lib directories



Step 3 : adding to the Linker files



Step 4 : I then compiled the project resulting in the command line ouput shown in sample output data

### Test data:

n/a

### Sample output:

### 

### Reflection:

Nothing to report was fairly perfunctory

### Metadata:

N/A

### Further information:

N/A

## Exercise 2. Understand pixel colour

## Question:

a) An image is simply a 2D array of pixels. Each pixel has a colour value which can be digitally

represented as a list of numbers, depending on the data format adopted. In the framework, the

Colour of each pixel is represented in RGBA format using 4 integers, each of which ranging from 0

to 255. Open ImageProcess\_cuda.cu and go to the method d\_render( ), modify the 4 numbers

shown in make\_uchar4( ..., ..., ..., ... ) in the following line:

d\_output[i] = make\_uchar4(c \* 0xff, c \* 0xff, c \* 0xff, 0);

say,

d\_output[i] = make\_uchar4(0xff, 0, 0, 0);   
 and then

d\_output[i] = make\_uchar4(0, 0xff, 0, 0);   
d\_output[i] = make\_uchar4(0, 0, 0xff, 0);

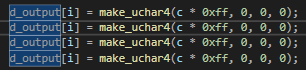
b) The original image is a grey value image, the pixel intensity at a pixel position at (u,v) is read using float c = tex2DFastBicubic(texObj, u, v); where c is in [0, 1].

c) Now modify the value d\_output[i] using image pixel value c read from image location at

(u, v) with the following colour and observe how the image colour is changed.

d\_output[i] = make\_uchar4(0, 0, c\*0xff, 0);

### Solution:



Running each of these one at a time and commenting out the other to display the different resulting outcomes

### 

### Test data:

n/a

### Sample output:

|  |  |
| --- | --- |
| code | output |
| d\_output[i] = make\_uchar4(0xff, 0, 0, 0); |  |
| d\_output[i] = make\_uchar4(0, 0xff, 0, 0); |  |
| d\_output[i] = make\_uchar4(0, 0, 0xff, 0); |  |
| d\_output[i] = make\_uchar4(0, 0, 0xff, 0); |  |

### Reflection:

Nothing to report was fairly perfunctory

## Exercise 3. Image Transformation

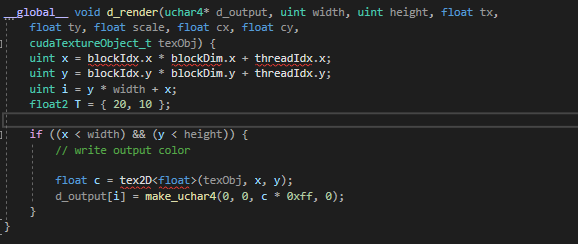
## Question:

Demonstrate Image transformation.

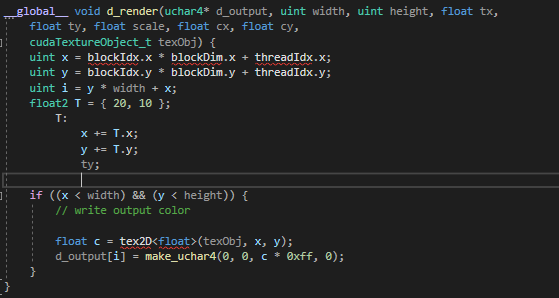
### Solution:

Translate the image.

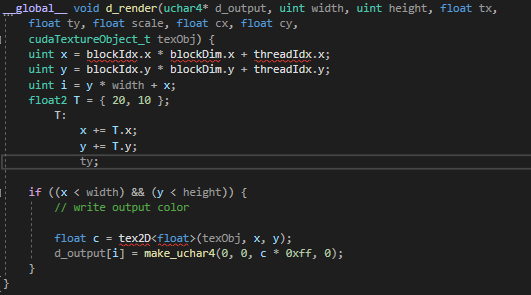
1. Define a translation as a 2D vector, say float2 T={20, 10};



1. Translate (x, y) with vector T: x +=T.x; y +=T.y;



1. Read pixel colour with translated coordinates x, y: float c = tex2D(texObj, x, y);



1. Compile the run your program and observe if the image is translated according to your wish.
2. Observe how the image is transformed by defining different translation vectors.

### Sample output:

|  |  |
| --- | --- |
| Translation | result |
| float2 T = { 20, 10 }; |  |
| float2 T = { 200, -100 }; |  |

### Reflection:

The image has been translated by moving the image in the way described in the vector T. the first value adjusts the translation in the xs axis and the second value adjusts it in the Y axis. When translating to the right the image is replaced by black plixels where the image has moved but when translating to the left the image appears stretched.

### Further information:

Why doe the image appear stretched whgen translating in negative directions?

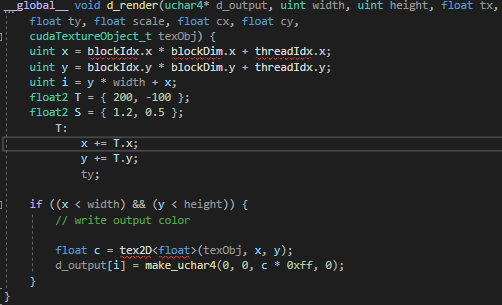
## Question:

Demonstrate Image Scaling

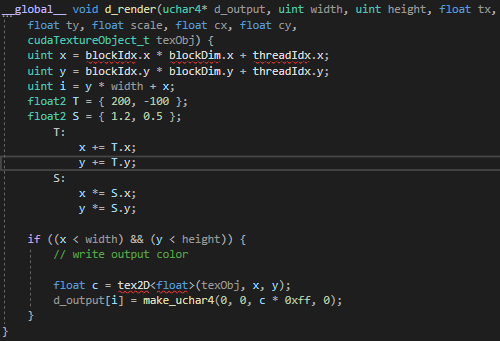
### Solution:

Scale the image

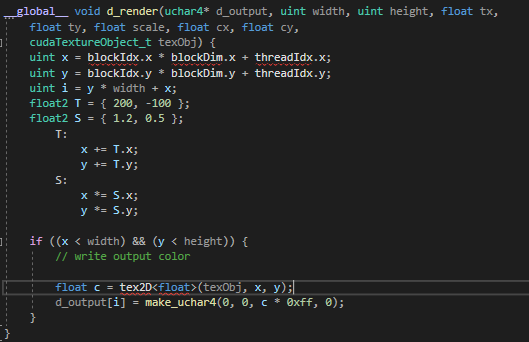
1. Define a scaling transformation as a 2D vector, say float2 S= {1.2, 0.5};



1. Scale (x, y) with vector S: x \*=S.x; y \*=S.y;



1. Read pixel colour with scaled coordinates x, y: float c = tex2D(texObj, x, y);



1. Compile the run your program and observe if the image is scaled according to your wish.
2. Observe how the image is scaled by defining different scaling vectors.

### Sample output:

|  |  |
| --- | --- |
| Translation | result |
| float2 S = { 1.2, 0.5 }; |  |
| float2 S = { 2, -0.5}; |  |

### Reflection:

The image has been translated by scalin it according to the vector S in the second image I experimented by using a negative value this resulted in a strange image

### Further information:

Why doe the image appear as shown when scaled in negative directions?