CST 325 Constant Buffer Lab

**Learning Objectives:**

Understanding and using constant buffers

**Reading**

*Read about the constant buffer in https://docs.microsoft.com/en-us/windows/desktop/direct3d11/overviews-direct3d-11-resources-buffers-constant-how-to*

* + *What kind of data does the constant buffer contain?*

Constant buffers contain shader constant data

* + What are the steps you have to take in order to initialize a constant buffer according to the doc?

1. Define a structure that describes the vertex shader constant data.  
2. Allocate memory for the structure that you defined in step one. Fill this buffer with vertex shader constant data. You can use malloc or new to allocate the memory, or you can allocate memory for the structure from the stack.  
3. Create a buffer description by filling in a D3D11\_BUFFER\_DESC structure. Pass the D3D11\_BIND\_CONSTANT\_BUFFER flag to the BindFlags member and pass the size of the constant buffer description structure in bytes to the ByteWidth member.  
4. Create a subresource data description by filling in a D3D11\_SUBRESOURCE\_DATA structure. The pSysMem member of the D3D11\_SUBRESOURCE\_DATA structure must point directly to the vertex shader constant data that you created in step two.  
5. Call ID3D11Device::CreateBuffer while passing the D3D11\_BUFFER\_DESC structure, the D3D11\_SUBRESOURCE\_DATA structure, and the address of a pointer to the ID3D11Buffer interface to initialize.

**Questions**

*Download the code from iLearn “starting\_point\_lab\_hourglass” and look at the code*

* + *Run the code. You should see an hourglass shape. When you click the left button, you should see a color change.*

Y

* + *What are the names of the variables in the constant buffer in the shader code?*

float cb\_a;  
 float cb\_b;  
 float cb\_c;  
 float cb\_d;

* + *What are the names of the variables in the constant buffer in the cpp code?*

float some\_variable\_a;  
 float some\_variable\_b;  
 float some\_variable\_c;  
 float some\_variable\_d;

* + *Where are the constant buffer variables being assigned in the cpp file?*

In InitDevice() starting on line 222

**Code Breakdown**

In the downloaded code, you will see the constant buffer as follows:

struct VS\_CONSTANT\_BUFFER

{

float some\_variable\_a;

float some\_variable\_b;

float some\_variable\_c;

float some\_variable\_d;

};

You will notice that in the shader file, the constant buffer looks like this:

cbuffer VS\_CONSTANT\_BUFFER : register(b0)

{

float cb\_a;

float cb\_b;

float cb\_c;

float cb\_d;

};

In the shader file, this constant buffer can be thought of as a structure. We know that values will be loaded into the constant buffer and these are the variables we will use to access that data in the shader. Notice the name of the constant buffer in both files is the same, but the variable names don’t have to be. We can load a value into some\_variable\_a on the cpp side and use that data as cb\_a on the shader side.

Notice the section in the cpp file that looks like this:

VsConstData.some\_variable\_a = 0;

VsConstData.some\_variable\_b = 0;

VsConstData.some\_variable\_c = 1;

VsConstData.some\_variable\_d = 1;

In this code, we are setting constant variables. Anytime we want to pass variables to the shader, we are going to need to pass those values through the constant buffer. For example, if we want to know mouse coordinates in order to move a shape in the vertex buffer, we have no way of knowing what those values are unless we pass them through the constant buffer. We can set some\_variable\_a to mousex and some\_variable\_b to mouse y and then in the vertex shader set up login to change the vertex position based on this new info!

**Implementation**

* + *Change the background color of the window in the render function.*

Line 283: float ClearColor[4] = { 0.3f, 0.125f, 0.3f, 1.0f }; // red,green,blue,alpha

* + *Instead of an hourglass, change the shape to a bow shape (or sideways hourglass)*

vertices[0].Pos = XMFLOAT3(-0.5f, 0.5f, 0.5f);  
 vertices[1].Pos = XMFLOAT3(0.0f, 0.0f, 0.5f);  
 vertices[2].Pos = XMFLOAT3(-0.5f,-0.5f, 0.5f);  
  
 vertices[3].Pos = XMFLOAT3(0.0f, 0.0f, 0.5f);  
 vertices[4].Pos = XMFLOAT3(0.5f, 0.5f, 0.5f);  
 vertices[5].Pos = XMFLOAT3(0.5f, -0.5f, 0.5f);

* + *Change the color that the shape changes to (right now it changes to white on left click)*

CPP: VsConstData.some\_variable\_d = pressed ? 1 : 0;

Code is in pixel shader changed value to

return float4(.3, .6, 1, 1);

* + *Set up four colors in the pixel shader and when the left button gets clicked, change the flash color to the next color (rotate between the four on left button click)*

Created a counter that increments with the mouse click and passes through the shader as VsConstData.some\_variable\_a = colorCount;

if (cb\_d)  
 {  
 if (cb\_a % 4 == 0) {  
 return float4(.3, .6, 1, 1);  
 }  
 else if (cb\_a % 4 == 1) {  
 return float4(.6, .3, .6, 1);  
 }  
 else if (cb\_a % 4 == 2) {  
 return float4(.1, .6, .3, 1);  
 }  
 else if (cb\_a % 4 == 3) {  
 return float4(1, 1, .6, 1);  
 //return float4(.3, .6, 1, 1);  
 }  
 }

* + *If the key “b” is pressed, change the color of the object to blue*

Implemented siimilar to the mouse click

* + In the render function, you will notice a variable called angle that changes. Create another variable that increases very slightly (+=.001) and set one of the constant buffer variables to it. Grab that value in the vertex buffer and update the position of the shape to move across the screen in the x direction (slide the shape across the screen)
  + Pass the x and y position of the mouse into the constant buffer. Use these values to move the shape based on where your mouse is. (have the shape follow your mouse). Remember your mouse coordinates are from 0 to width, height but your vertex coord are -1 to 1. Keep this in mind when thinking about what logic to use for this part. (Comment out the previous section that animates the shape)
  + Control the scaling of the triangle (or hourglass) with the x-axis of your mouse
  + Control the amount of blue color of the shape with the x-axis of your mouse.