Parallax Occlusion Mapping

By: Shantanu Chauhan

Project Description:

This project is an extension of the current framework that I am using for this class. All the previous passes and features are still in the framework but some of them are just not currently being used in the scene. The main objective of this project was to do Parallax Occlusion Mapping. Normal mapping was already done in this project so this project will replace that portion of the code in GBuffer.frag file as well so that our light matches the texture displacements as well.

First, I started with doing Parallax Mapping, then added steep Parallax mapping and finally Parallax Occlusion mapping.

Various references were used like:

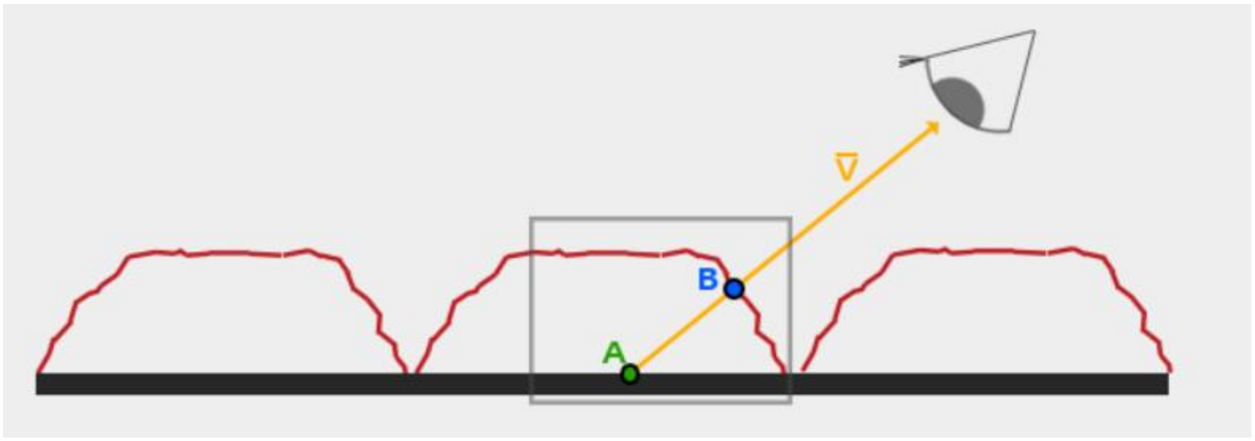
- https://faculty.digipen.edu/~gherron/references/References/SurfaceDetail/Tatarchuk-POM-SI3D06.pdf
- https://learnopengl.com/Advanced-Lighting/Parallax-Mapping
- I have met with you (Prof. Herron) to ask for help understanding some concepts in this algorithm, mainly how to transfer the view vector in the tangent space of the object.

Algorithm:

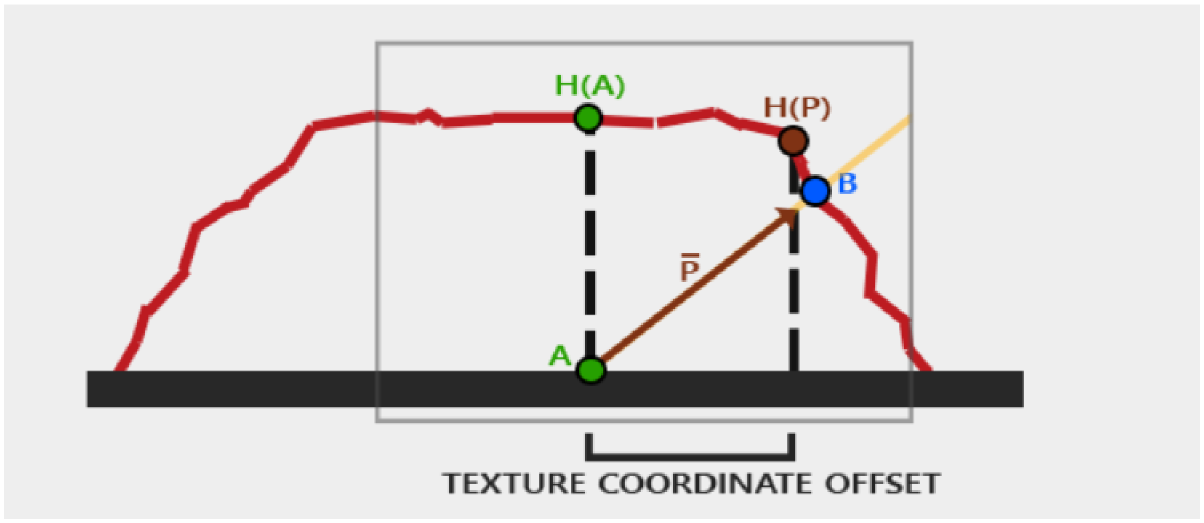
Parallax Mapping:

Parallax mapping in simple terms is taking the current texture lookup(TexCoord) to get the normal value from a normal map and actual texture details like diffuse, specular and texture from an image and displacing it to a suitable degree to mimic a sense of depth in the scene using a HeightMap.

Like in the picture below. The red rough line represents the values in the height map as a geometric representation of for e.g. a brick. The V vector represents the View vector. Now in reality if a person views an actual brick then he would see the point B but as we have a flat plane then the viewer sees the point A instead of B. So, the aim of parallax mapping is to offset texture coordinates at point A to B so that we can do the texture lookups for point B instead of A.



The main challenge in doing that is how to get texture coordinates at point B from point A. We do this by scaling the view vector by the height at point A. We scale the vector V by the height at point A so vector V becomes scaled vector P. We then take this vector P and take its vector coordinates that align with the plane as the texture coordinate offset. This works because vector P is calculated using a height value from the heightmap so the higher a fragment's height, the more it effectively gets displaced. But this is a crude approximation, if we change the height rapidly then we might not end up close to point B. The view vector is in the tangent space/texture space to take into account of an object's rotation.

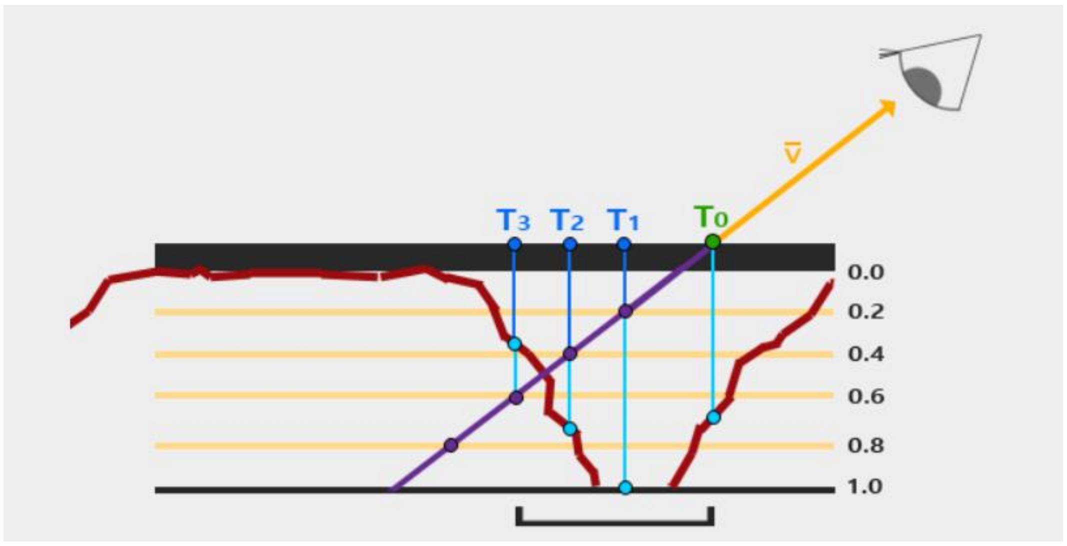


Steep Parallax Mapping:

Steep Parallax Mapping is an extension on top of Parallax Mapping in that it uses the same principles, but instead of 1 sample it takes multiple samples to better pinpoint vector P to B.

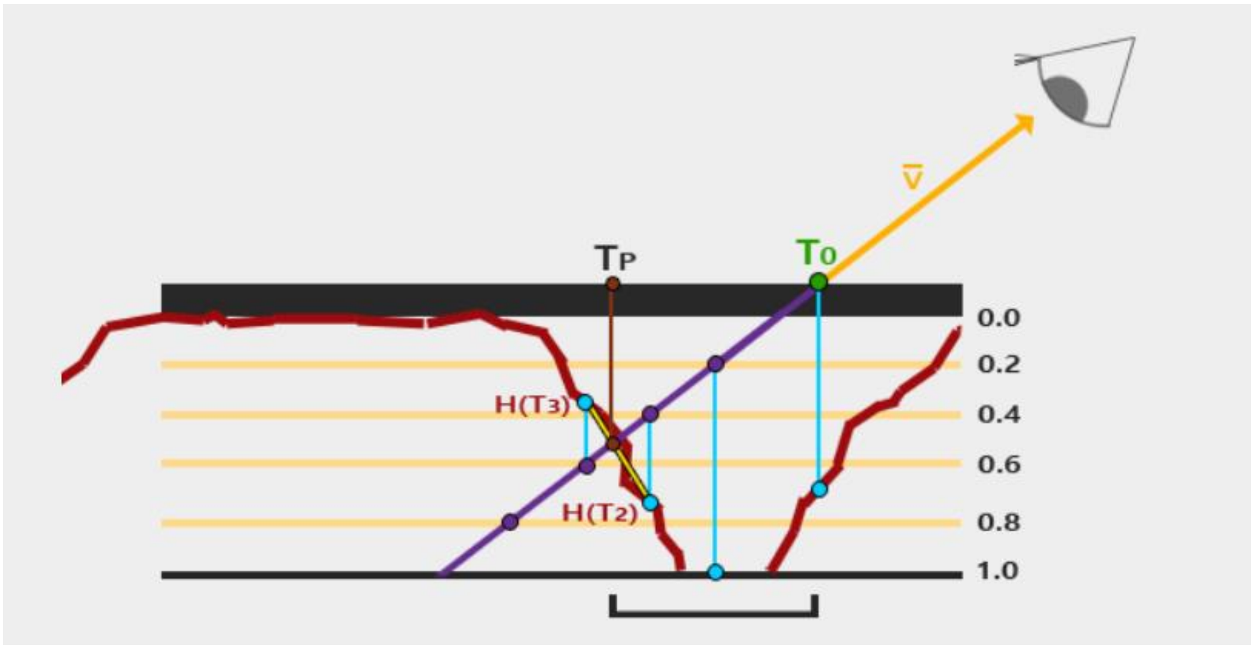
The general idea of Steep Parallax Mapping is that it divides the total depth range into multiple layers of the same height/depth. For each of these layers we sample the depth map shifting the texture coordinates along the direction of P until we find a sampled depth value that is below the depth value of the current layer.

We traverse the depth layers from the top down and for each layer we compare its depth value to the depth value stored in the depth map. If the layer's depth value is less than the depth map's value, it means this layer's part of vector P is not below the surface. We continue this process until the layer's depth is higher than the value stored in the depth map: this point is then below the (displaced) geometric surface.



Parallax Occlusion Mapping:

Parallax Occlusion Mapping is based on the same principles as Steep Parallax Mapping, but instead of taking the texture coordinates of the first depth layer after a collision, we're going to linearly interpolate between the depth layer after and before the collision. We base the weight of the linear interpolation on how far the surface's height is from the depth layer's value of both layers.



Parllax Occlusion mapping is similar to Steep Parallax Mapping with as an extra step the linear interpolation between the two depth layers' texture coordinates surrounding the intersected point. This is again an approximation, but significantly more accurate than Steep Parallax Mapping.

So, after getting the new texture coordinate, we use this to get the value of the normal from the normal map so that our light in the scene acts more naturally to the displacement of the surface.

All of this is done in the GBuffer to manipulate the Normals and the Texture(colors) rest of the shader are unaffected.

Framework Controls:

The controls are the same for what is used in CS-541:

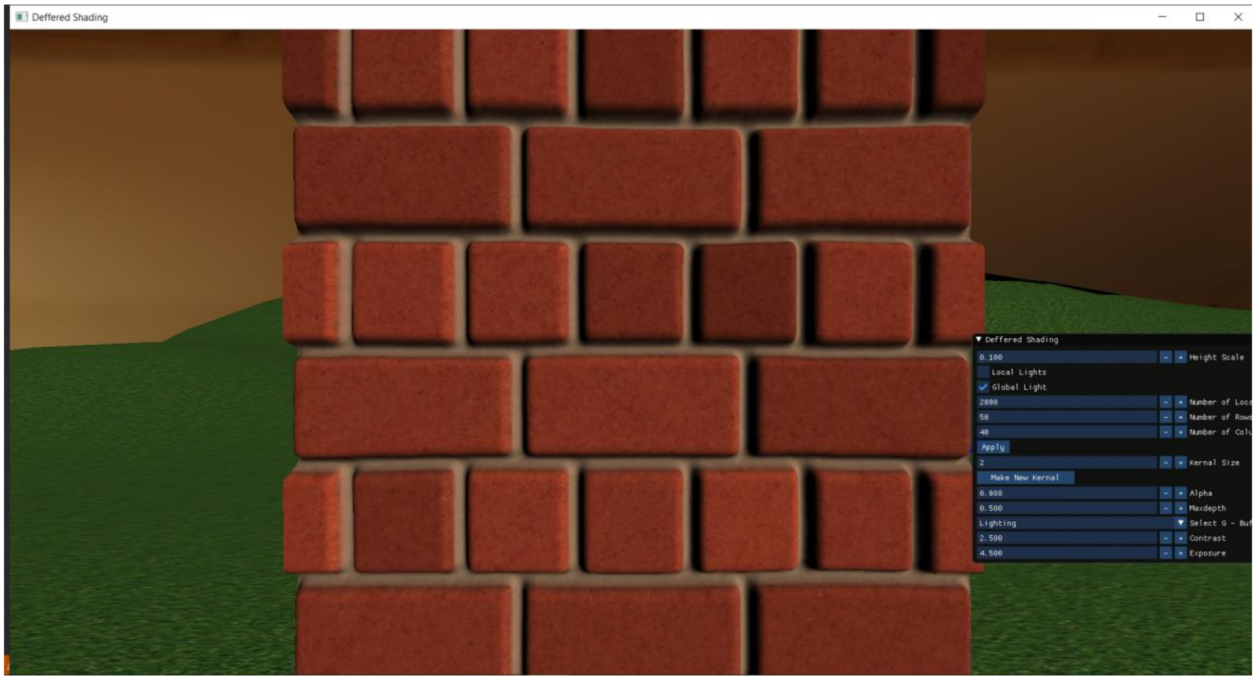
- ‘E’ - To toggle the camera from world view to a First-Person view.
- ‘W A S D’ – To move around in the First-Person view.
- ‘Left Click + Mouse movement’- Moves the camera around in both views.
- ‘Right Click + Mouse Movement’ – Moves the center of the camera around in world view.
- ‘Escape’- To close the application.
- ‘Scroll Wheel’- To zoom in and out of the scene in world view.
- ‘Shift + Left Click + Mouse Movement’- To move the global light around.
- ‘Shift + Scroll’- To get the light closer and farther from the scene.

In Debug Menu:

- You can change the height scale used in the parallax mapping to get more depth in the object.

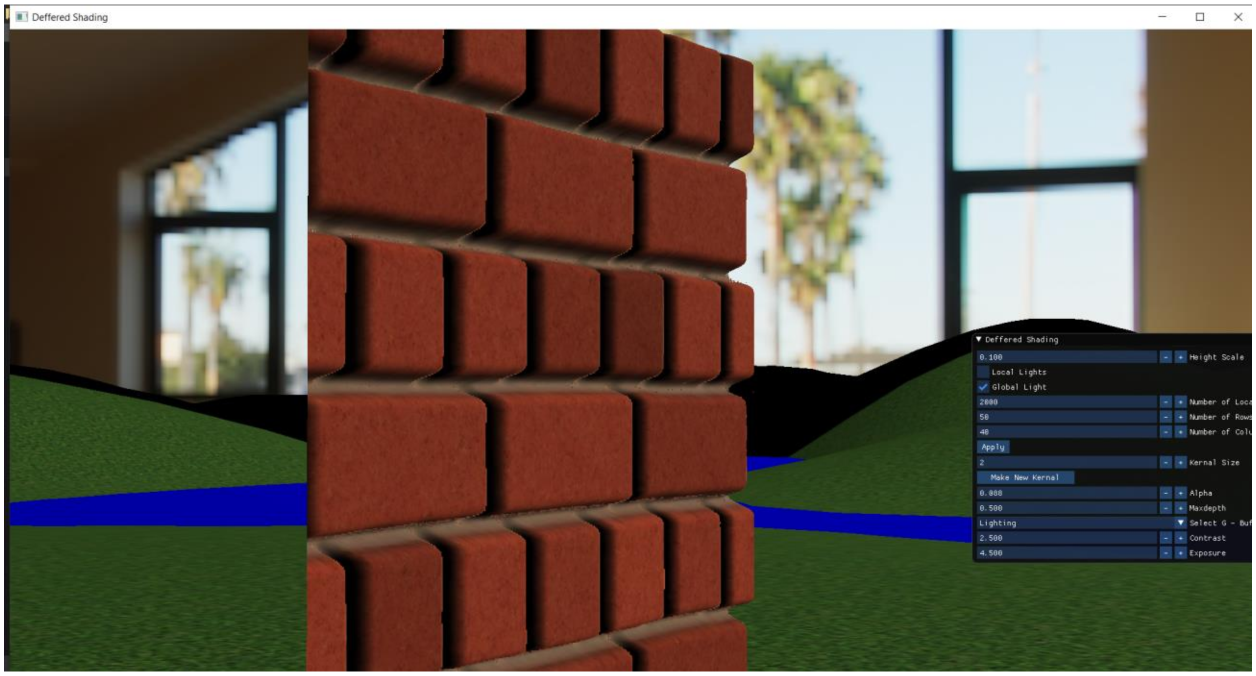
Screenshots:

- Front View of the brick quad with height scale of 0.1



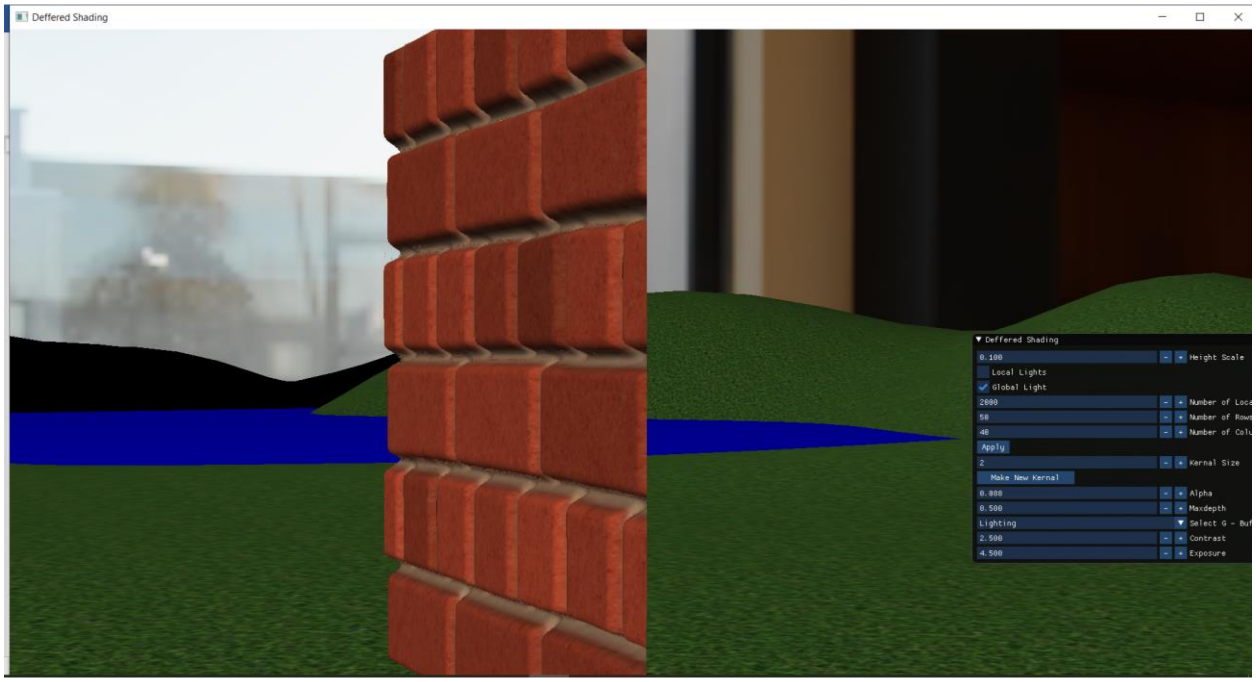
```
▼ Deffered Shading
0.100        - + Height Scale
[ ] Local Lights
[✓] Global Light
2000         - + Number of Loca
50           - + Number of Rows
40           - + Number of Colu
Apply
2            - + Kernal Size
Make New Kernal
0.080        - + Alpha
0.500        - + Maxdepth
Lighting       ▼ Select G - Buf
2.500        - + Contrast
4.500        - + Exposure
```

- Side view(left) of the brick wall



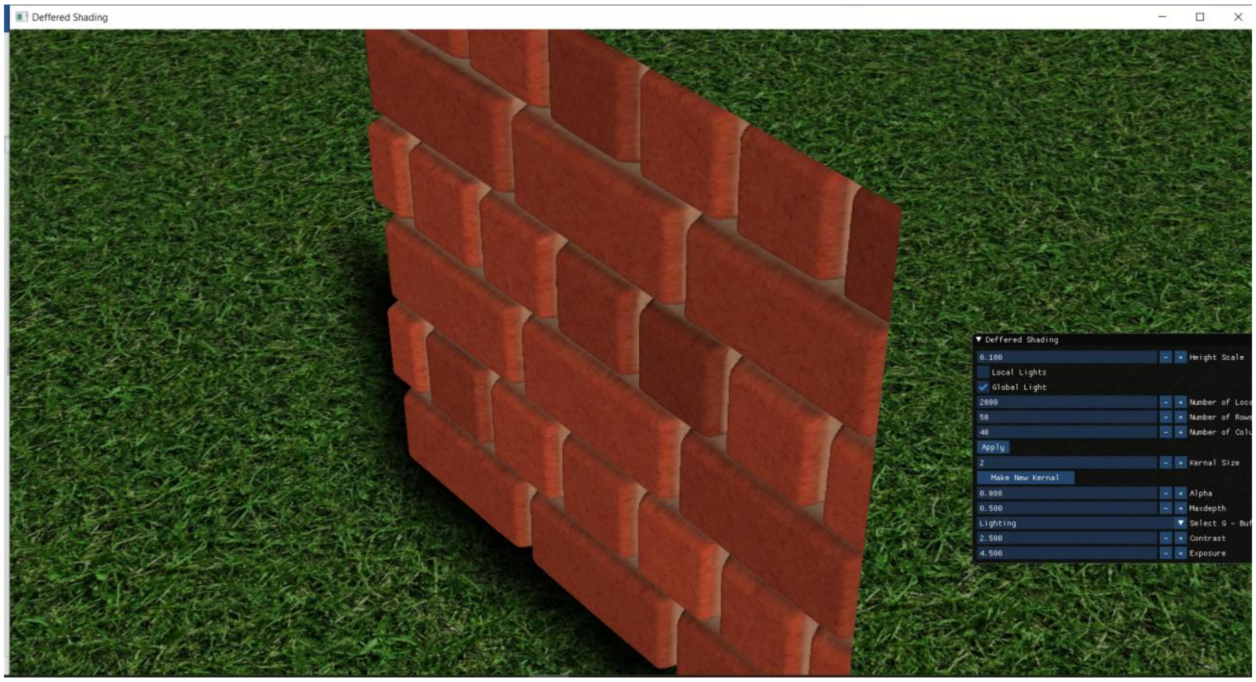
```
▼ Deffered Shading
0.100        - + Height Scale
[ ] Local Lights
[✓] Global Light
2000         - + Number of Loca
50           - + Number of Rows
40           - + Number of Colu
Apply
2            - + Kernal Size
Make New Kernal
0.080        - + Alpha
0.500        - + Maxdepth
Lighting       ▼ Select G - Buf
2.500        - + Contrast
4.500        - + Exposure
```

- Side view(right) of the brick wall



```
▼ Deffered Shading
0.100        - + Height Scale
☐ Local Lights
☑ Global Light
2000         - + Number of Loca
50           - + Number of Rows
40           - + Number of Colu
Apply
2            - + Kernal Size
Make New Kernal
0.888        - + Alpha
0.500        - + Maxdepth
Lighting       ▼ Select G - Buf
2.500        - + Contrast
4.500        - + Exposure
```

- Top view of the brick wall



```
▼ Deffered Shading
0.100        - + Height Scale
☐ Local Lights
☑ Global Light
2000         - + Number of Loca
50           - + Number of Rows
40           - + Number of Colu
Apply
2            - + Kernal Size
Make New Kernal
0.888        - + Alpha
0.500        - + Maxdepth
Lighting       ▼ Select G - Buf
2.500        - + Contrast
4.500        - + Exposure
```