CS562 PROJECT 1: DEFFERED SHADING

SYNOPSIS

Simple vertex transformation in Vertex shader and lighting calculation for all pixel in fragment shader is how we generally implement lighting. This methodology is called as **"Forward Rendering".** The downside of this approach is that, it's not scalable. Imagine a 3D complex scene where a single pixel is shared by multiple objects. Fragment shader just needs to run on the topmost object in that pixel otherwise we are wasting GPU cycles. Light calculation is done for all the pixel no matter how far the object really is. With multiple light sources the calculation gets more and more expensive and Forward Rendering fails.

Deferred shading in simple terms is postponing rendering to a later stage (multiple passes). In deferred shading we have 2 passes. First pass we process all the geometry information and save it in a buffer and in second pass called lighting pass we render on a FSQ (full screen quad) with all the calculation with respect to geometrical information stored in the buffer.

IMPLEMENTATION

From what I understood we are separating the geometry calculation and lighting calculation in 2 passes. We are saving the output of the first pass in G-Buffers and using that information to calculate light in the second pass.

I have total of 2 passes.

GbufferPass() to calculate all the geometric details and to fill various buffers (listed below).

GlobalLightingPass() this pass refers to the buffers filled up in the GBufferPass and performs lighting calculation to acquire result.

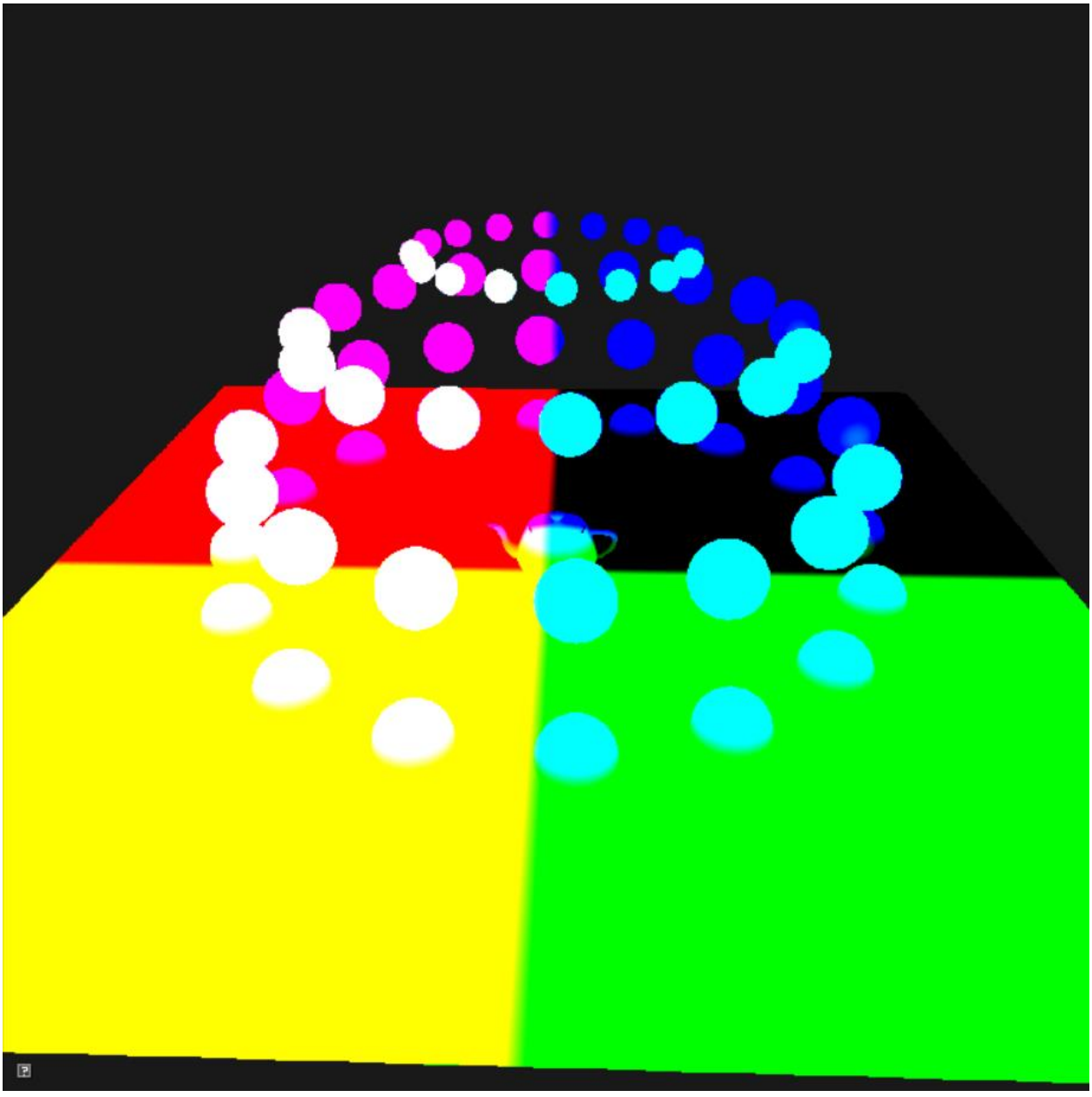
I have successfully rendered 100 point lights.

Key	Output
1	Final Image
2	Position vector Buffer
3	Normal vector Buffer
4	Vector buffer formed from camera position
5	Vector formed by source light

DATA IN GBUFFER:

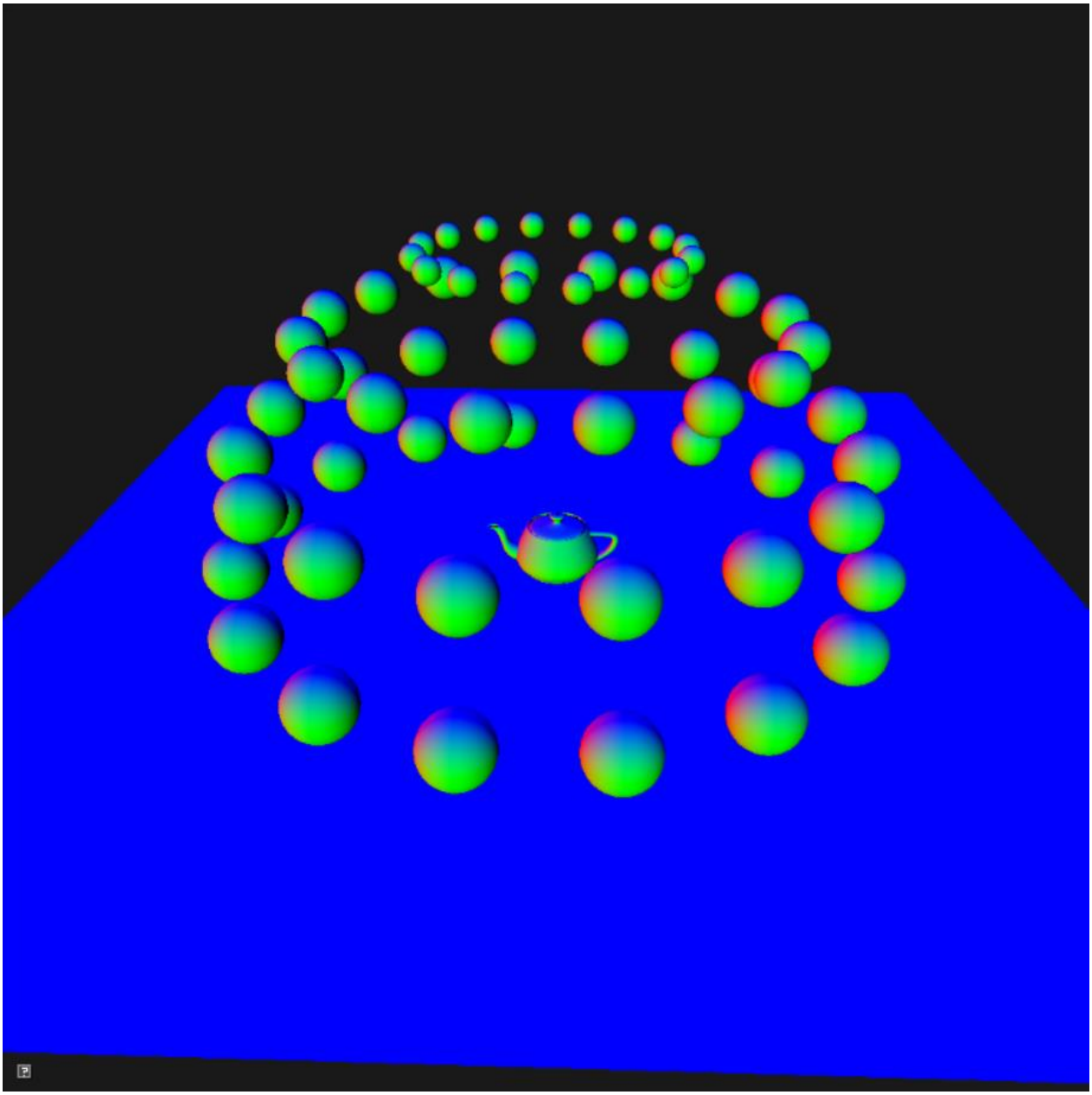
BUFFER 1

World Position: Position vector of all vertex which is used for lighting calculation.



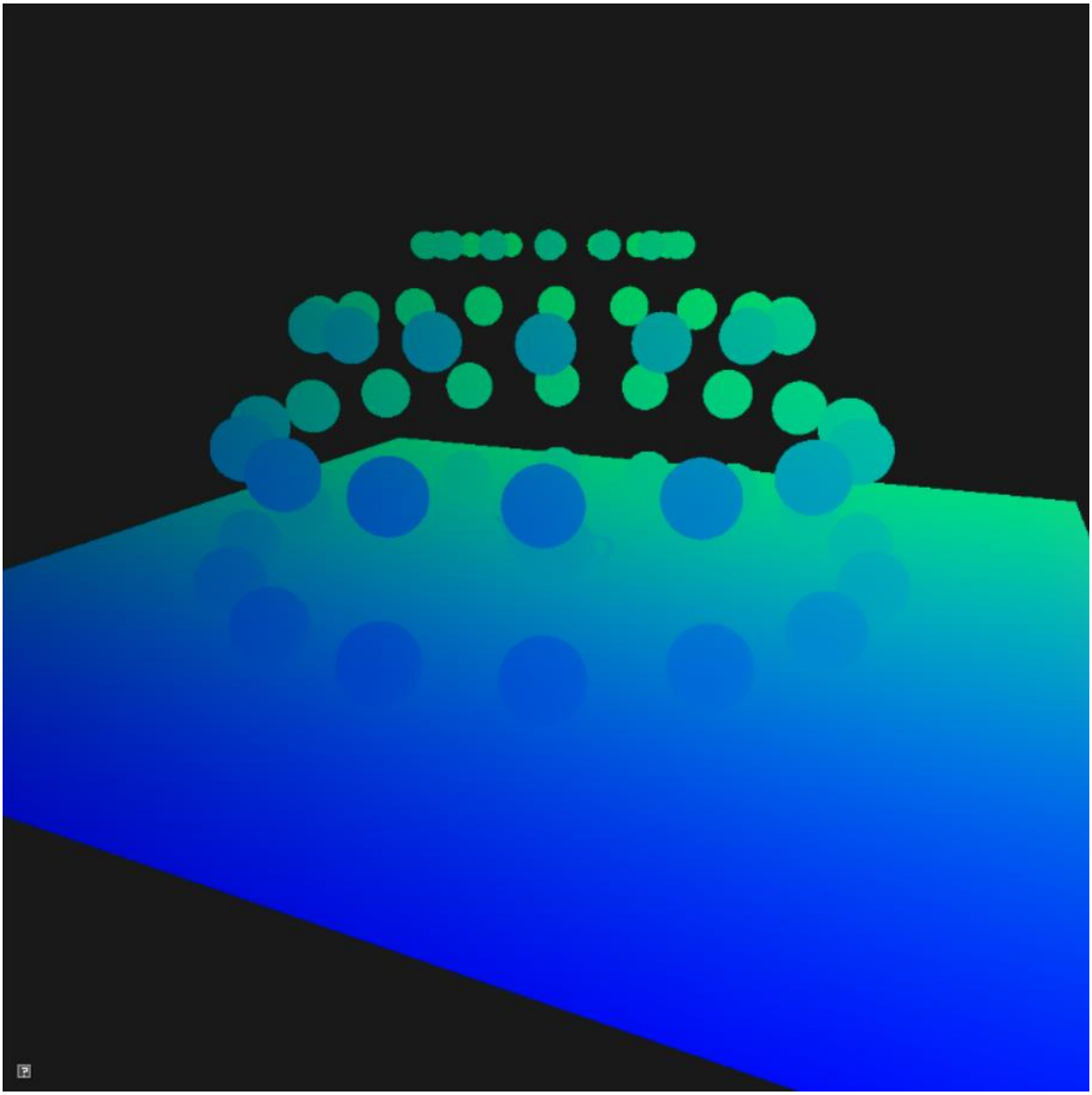
BUFFER 2

Normal Vector: Vectors used to determine the surface slope



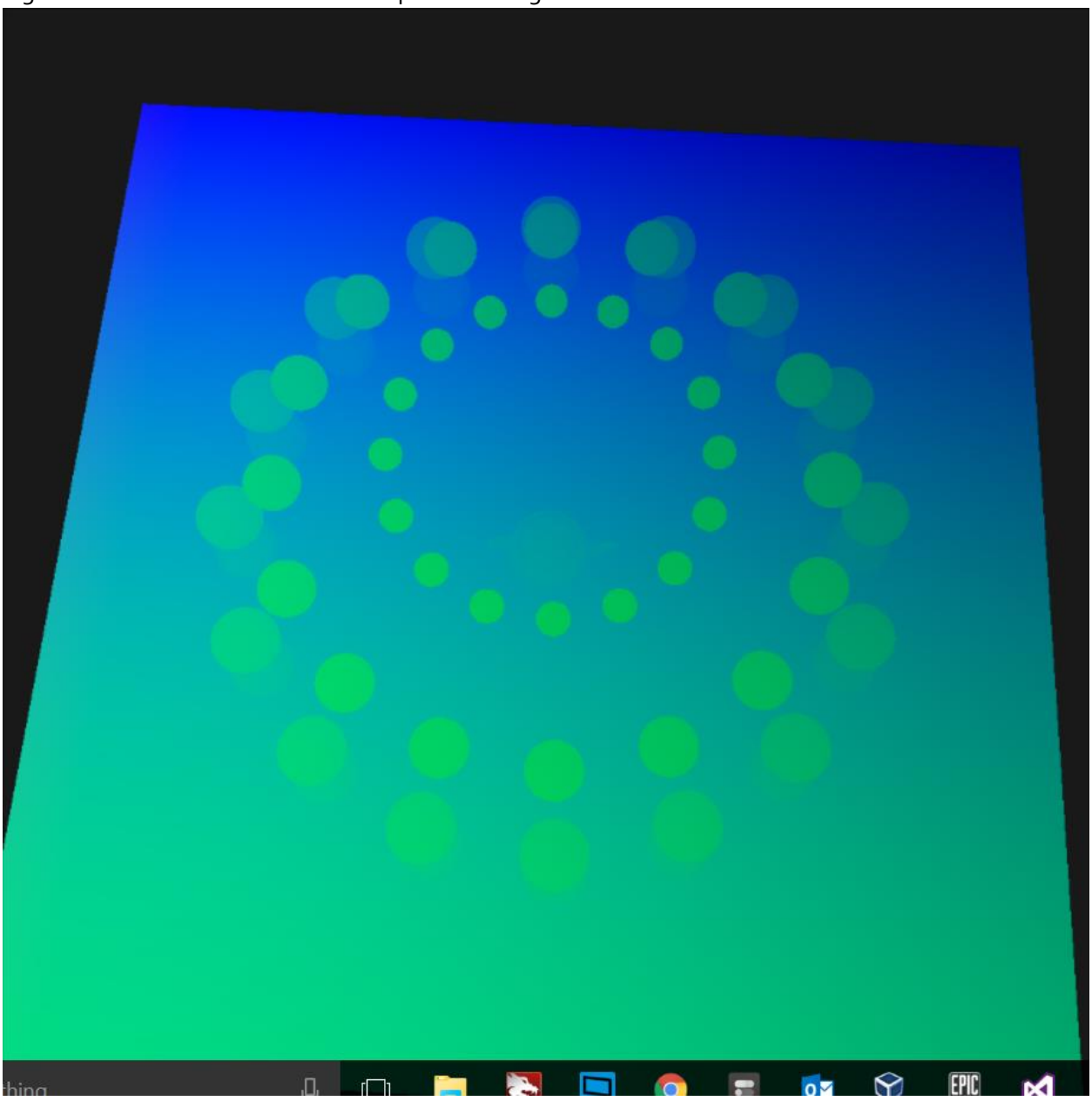
BUFFER 3

Eye vector: Vector of each vertex with respect to the camera position.



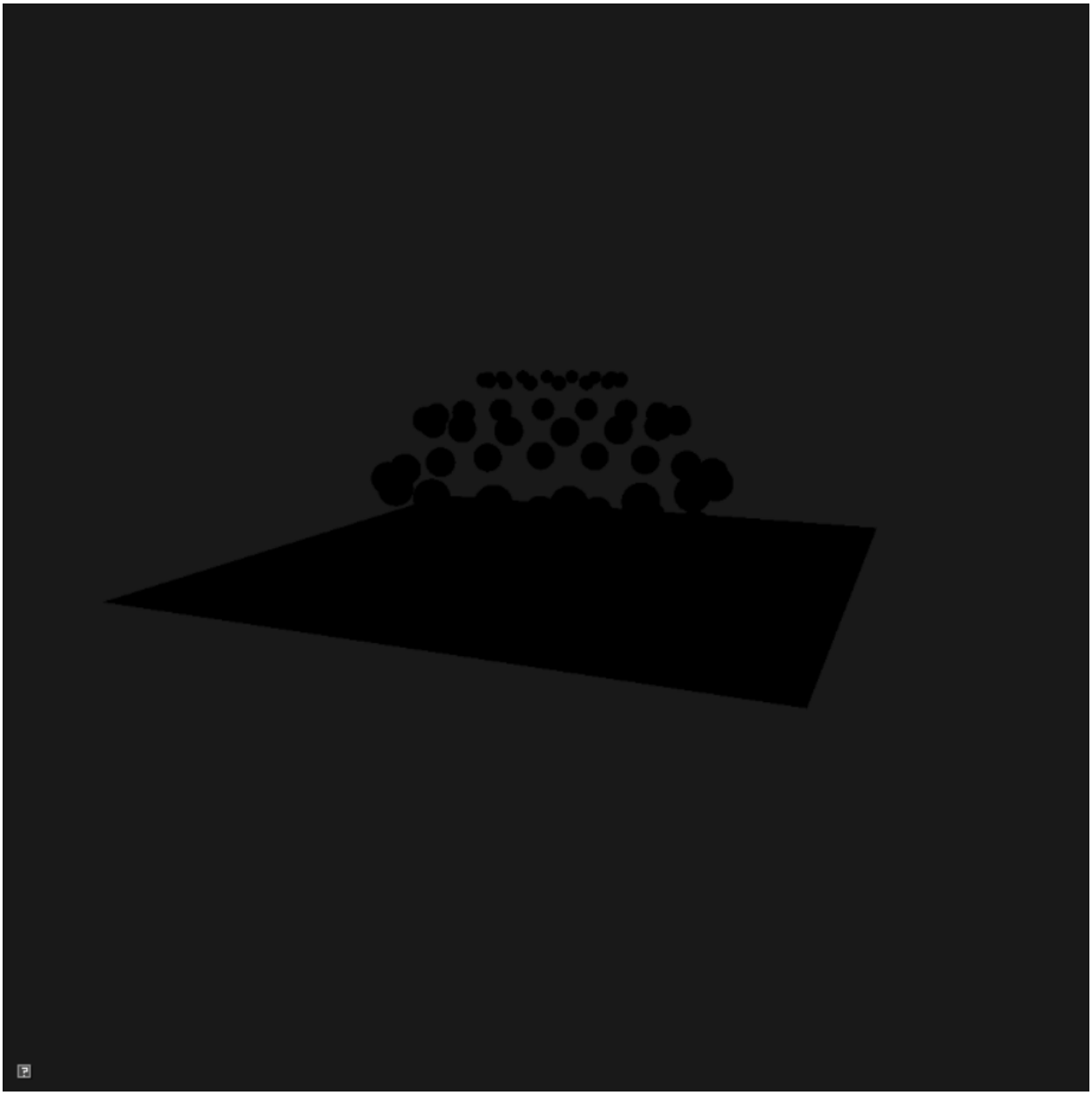
BUFFER 4

Light Vector: Vectors formed with respect to the light source

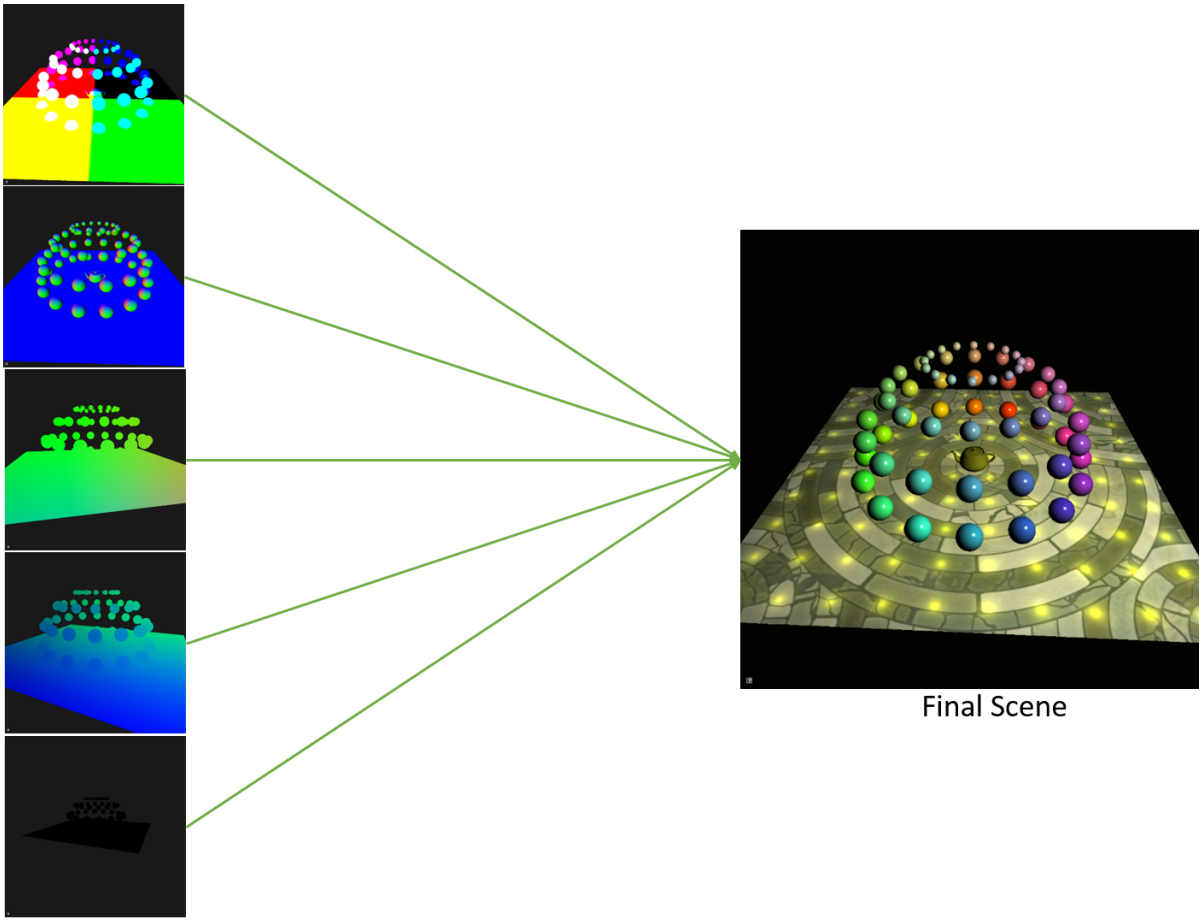


BUFFER 5

Shadow vector:



RESULT



GBuffer

Final Scene

The major advantage of this approach is that all the buffer value contains the value of the pixel which is useful for lighting calculation as it filters out all the overlapping pixels. This saves a lot of GPU cycles.