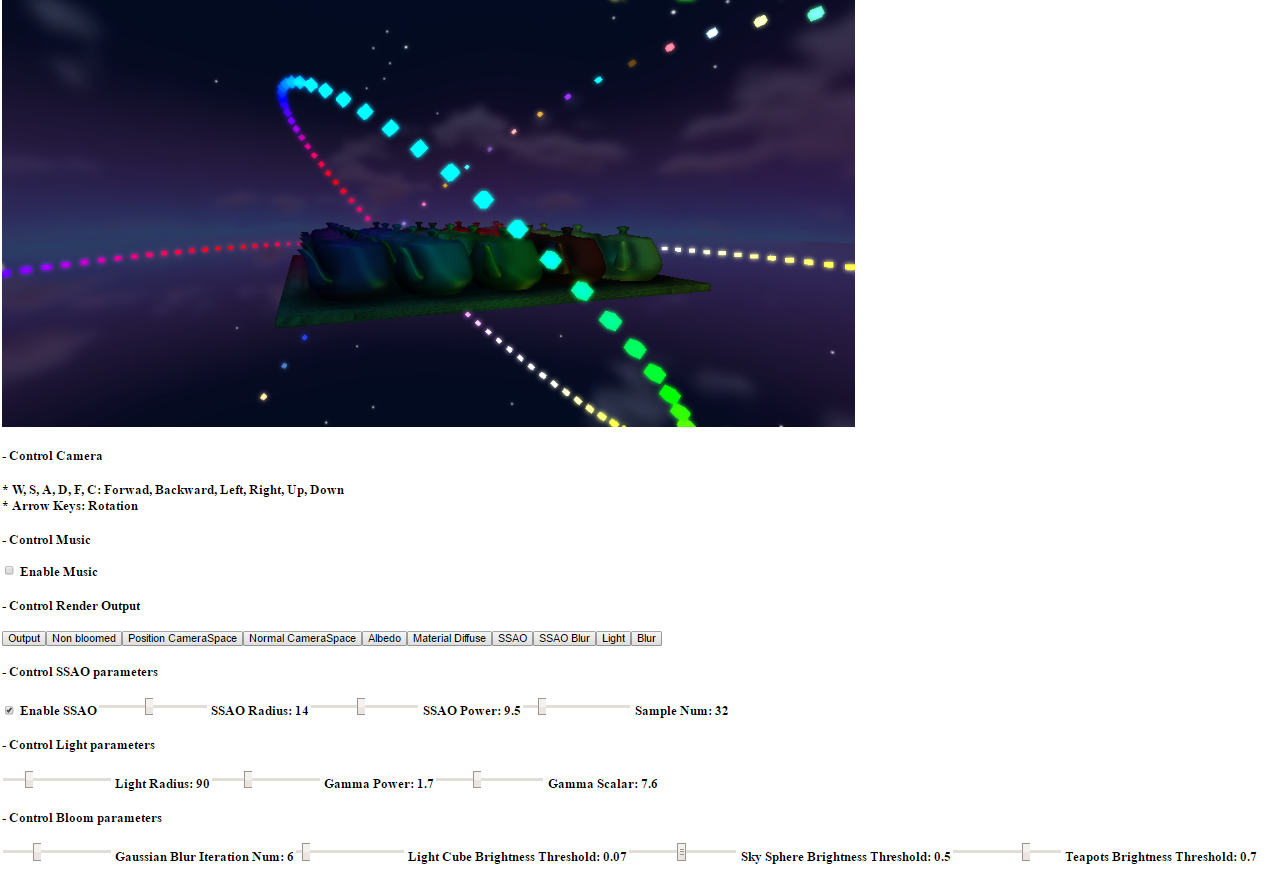
Readme of RT Rendering Lab5:

1. How to use the program:



Control Camera:  
W, S, A, D, F, C: Forwad, Backward, Left, Right, Up, Down  
Arrow Keys: Rotation

Control Music:  
When the music is turned on, the animation will start

Control Render Output:  
This lab is implemented by using Deferred Shading, Screen Space Ambient Occlusion (SSAO), Bloom, and Normal Mapping, each rendering pass will generate one or more screen space textures. To see what these textures looked like, press these buttons.

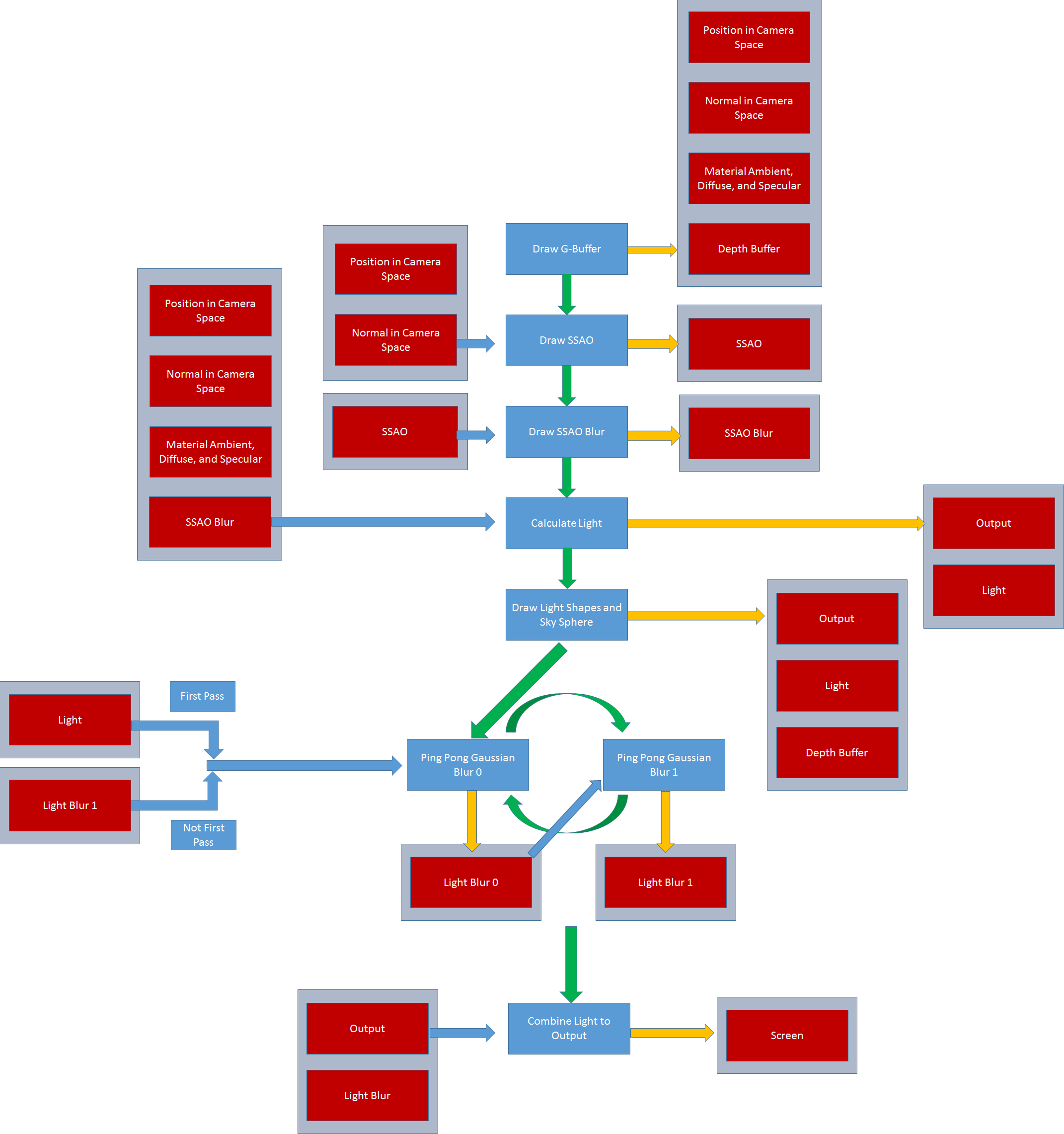
Control SSAO:  
Enable SSAO: Turn SSAO on/ off;  
SSAO Radius: Control the radius of sample hemisphere when rendering SSAO texture  
SSAO Power: Control the power of the occlusion function output, i.e.  
occlusion = pow( original occlusion, SSAO power )  
Sample Num: Control how many sample points taken when rendering SSAO texture

Control Light:  
Light Radius: Control how far the light will affect the mesh objects. By adopting a distance checking when calculating lights, I can conserve computing power and avoid unnecessary interference from the lights in outer light cube cycle.  
Gamma Power: Control the Gamma Correction power parameter when rendering the inner light cube cycle, because the color of cube will not be correct to human eyes by simply scaling a color with a scalar from 0 to 1.  
Gamma Scalar: Control the Gamma Correction scalar parameter when rendering the inner light cube cycle

Control Bloom:  
Gaussian Blur Iteration Num: Control how many time to blur the light texture in order to render blur texture  
Brightness Threshold: Control how the threshold for passing to light texture

2. Lists which browser/OS you developed your code on (just in case)  
 Chrome/ Win10: Although this lab supports Firefox, it is pretty laggy on it.

3. Implementation details:  
 The figure below shows the rendering sequence for each frame:



Blue arrows: Input texture  
 Yellow arrows: Output texture  
 Green arrows: Control flow

4. Reference Materials:  
 In order to avoid academic misconduct, here I list all the reference I used when implementing this lab. I tried my best to not to write exactly the same shader code. However, some of the code structure is still similar.

<http://learnopengl.com/#!Advanced-Lighting/SSAO> – I referenced the shader code of SSAO

<http://john-chapman-graphics.blogspot.com/2013/01/ssao-tutorial.html> – I referenced the shader code of SSAO

<http://learnopengl.com/#!Advanced-Lighting/Deferred-Shading> – I only referenced the main idea, code is not referenced

<http://learnopengl.com/#!Advanced-Lighting/Bloom> – I referenced the shader code of light texture generation and Gaussian Blur

<https://www.patrick-wied.at/blog/how-to-create-audio-visualizations-with-javascript-html> – I referenced the part of how to retrieve the frequency data. I did all the rest things by myself, including frequency data visualization.

<http://stackoverflow.com/questions/25582882/javascript-math-random-normal-distribution-gaussian-bell-curve> – I referenced the code of generating normal distribution for streetlight intermittent. It’s just a recreational effect and did not make too much difference compared with uniform distribution. Still I can change back to uniform distribution if you think it’s inappropriate.

<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/> – I referenced the part of how to generate TBN matrix, the other parts are trivial.