POST PROCESSING ON SOLAR SYSTEM

NANDA KISHORE KALIDINDI,
nkk263, N16138926

This assignment aims at providing multiple post processing shaders for a particular scene using multiple framebuffers

Assignment Requirements
- Add post-processing shaders to a previous assignment (Solar system) ✔
- It must combine at least two post processing effects ✔

Additional implementations
- Universe **Skybox** using cube maps
- Arc ball from previous scene (still buggy!)

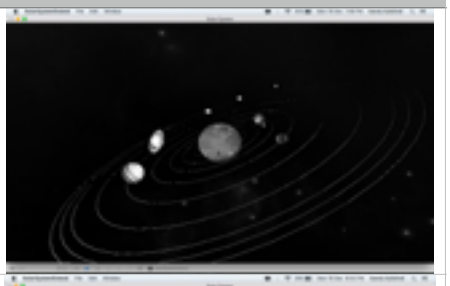
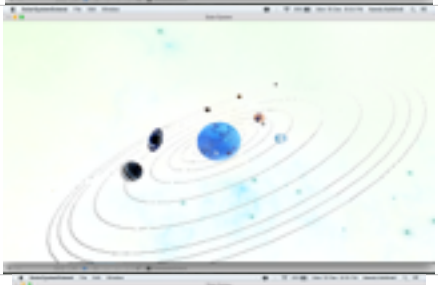
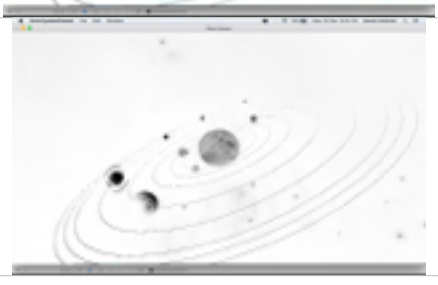
SKYBOX:

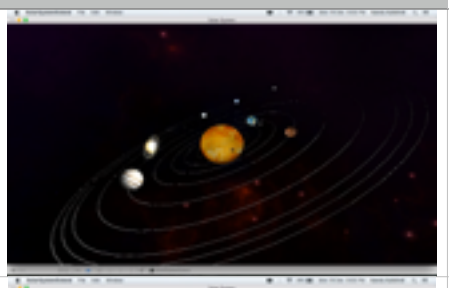
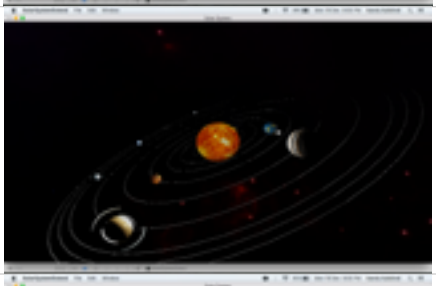
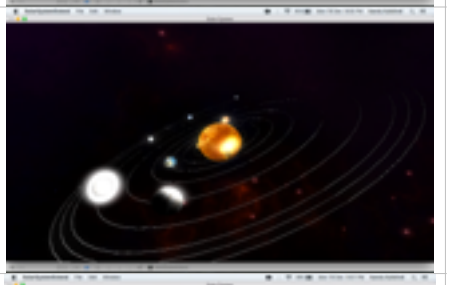
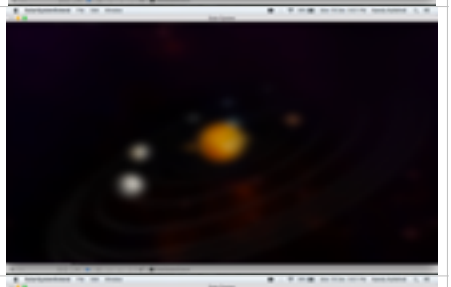
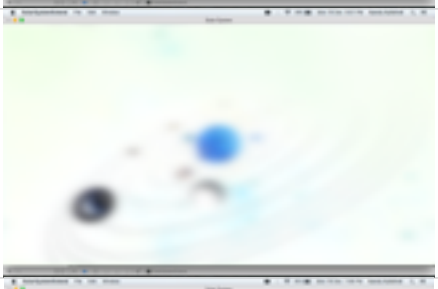
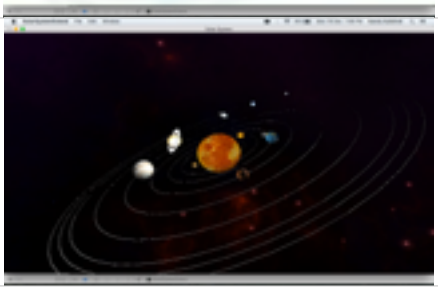
All the implementation details for the scene are present in the previous write up, please refer to **SolarSystem Writeup** in the same folder location.

Skybox images are generated using a software called **Spacescape** which allows an infinite customisation of the surrounding via XML. Upon export it generates a set of 6 images which are the front, back, left, right, up and down images. During loading and setting the cube map using C++ code, a few modifications were made to render the skybox without any abrupt changes like changing the order in which the image files are pushed into the cube map vector.

POST PROCESSING SHADERS:

A total of **7** post processing shaders are implemented of which some are a combination of two or more effects. All the post processing shaders and their key bindings to invoke and see real time post processing rendering are provided below.

Key Binding	Post processing effect	Number of frame buffers used	Image
1	Black and white	1	
2	Color inversion	1	
3	Black and white + Color inversion	2	

Key Binding	Post processing effect	Number of frame buffers used	Image
4	FXAA	1	
5	HDR tone mapping	1	
6	Glare	3	
7	Blur	2	
8	Blur with Invert colors	3	
0	None (renders a plain scene)	0	

NOTE:

The code was not properly segregated into respective structures due to time constraints and there is a lot of duplication of global variable which could have been avoided. Each and every post processing function(eg: blur(), hdr(), fxaa(), glare() etc.) is self explanatory in terms of how the rendering is being done.