|  |  |  |
| --- | --- | --- |
|  | **Rochester Institute of Technology**  **Golisano College of Computing and Information Sciences**  **School of Interactive Games and Media**  **2145 Golisano Hall – (585) 475-7680** |  |

**Data Structures & Algorithms for Games & Simulation II**

**IGME 309**

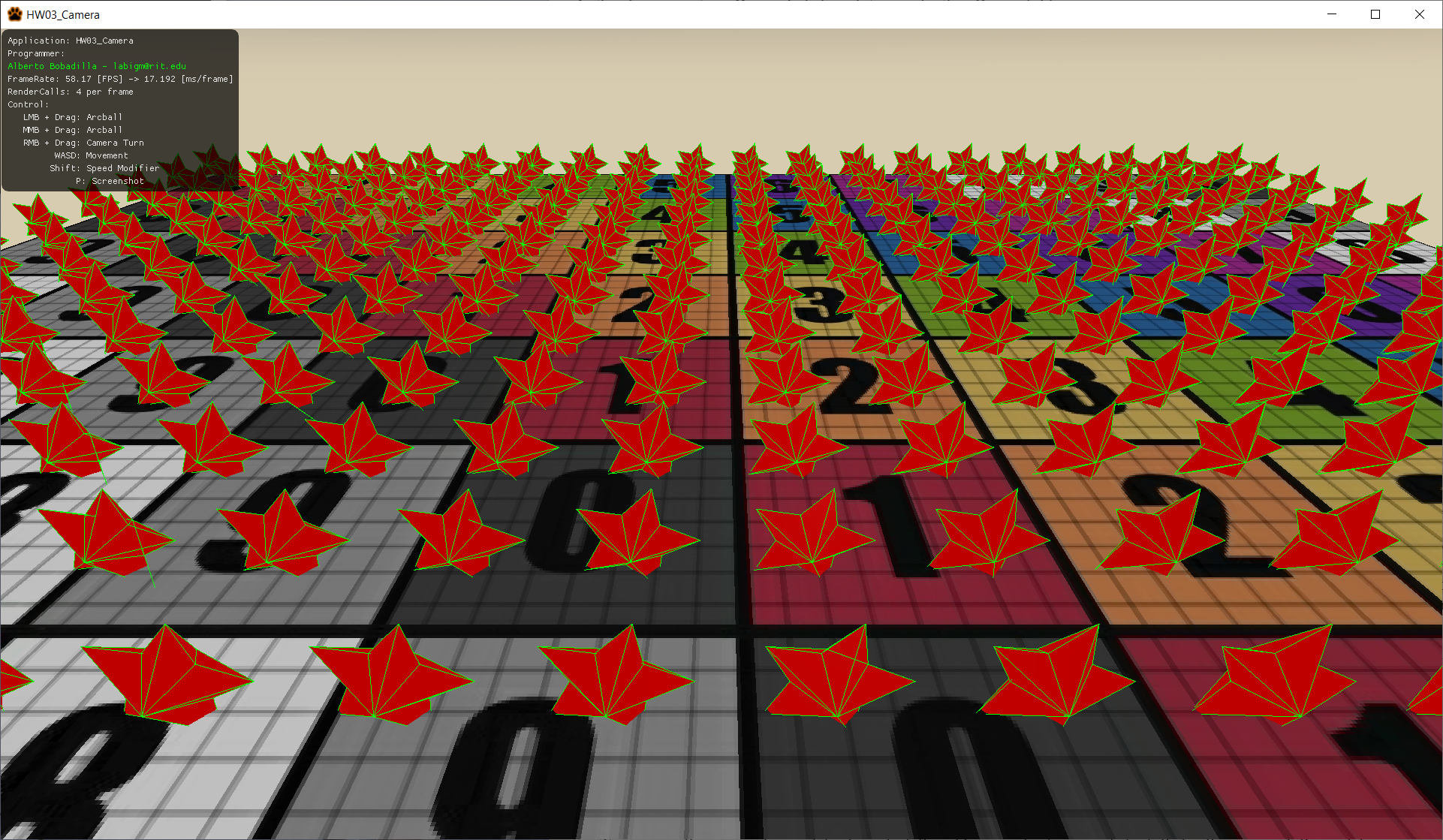
**A03 - Camera**

For this assignment, there is starter code provided in the repository, you are not required to use said starting code or the framework for it, you can use your own solution, but it will be your responsibility to translate the provided startup code on your own.

Video description: <https://www.youtube.com/watch?v=HFa2S2mAqXo>

In the class repository, I’ve included a solution under the \_Binary folder.

The starter code will give you this out of the box:



In this scene we are rendering hundreds of stars, but you cannot see them all as you are right at the center of the scene.

All controllers are connected and sending the right information to the camera class but only the move forward method is partially implemented so only W and S would move the camera. All other key binds work but the methods will not do anything with the arguments yet.

What you need to do for this assignment is to calculate the movement of the camera:

30% of the grade will come from movements like moving forward, backward, sideways and up-down (like an elevator) without turning your view (10% each pair of directions). (WASD and QE key movements please).

20% will come from turning your camera by holding the right mouse button and dragging it, only two rotations are necessary, horizontal and vertical no rolling sideways. Sending the values is already in place, the functionality needs to be implemented using quaternions.

50% of your grade will actually move forward, backwards, sideways and vertical according to your line of view (view vector), not in world coordinates; but in the space of the camera. What I mean with this is, imagine you press W for a while, that should move you closer to the floor of the world and eventually go though it; if you turn your camera 90 degrees to the right then pressing W and S would not move you in the same direction in world space but move you forward and backwards from what you are seeing. This is the usual behavior of any game camera, imagine Minecraft creative mode, except in creative mode your height is always set.

Your grade will be 100% if your solution behaves like the one provided and based on the previous description. Deductions will happen as follows:

-20% If you hardcode things in the camera.

-20% If you did not comment your code.

-30% your camera has Gimbal Lock or does not use quaternions for the rotations.

-30% any calculations are done outside of the camera class (i.e. you calculate the view in main or something of the sort)

Hints:

* Try moving the camera forward, backwards, left, right, upwards, downwards first; that is the easy part, then rotate your view and then move in that view.
* There is a complication if you look perfectly up, that will make your up vector align with your view, I will not deduct points for this but take in consideration that there are multiple solutions specially if you are avoiding the Gimbal Lock. You may limit the angle you are willing to go up, let’s say 90 degrees mean you will look perfectly up, then limit your camera to 89 degrees (although you should be using radians and quaternions this was the easiest way to explain my point) this may even avoid having to update your up vector. Remember glm::lookat internally computes a forward vector and crosses it with the upward you feed to create a rightward, it forward and upward aligns then the cross does not mean what you think.
* There is no need to move when you are in the orthographic projection, the view could be completely static as in my provided solution.

***Submit to the dropbox labeled: A03 – Camera***

As usual the required submission asks only for the project folder, not the whole solution, it should be no larger than 200kb if you are using the starter code (and you remove this document from that folder). If you are using your own framework/engine please submit the whole solution. Push your solution to your repository with the comment “**A03 Deliverable**” then zip the project (or solution) and upload it to the dropbox, in the comments section you need to specify the address of your repository.

Example:

