|  |  |  |
| --- | --- | --- |
|  | **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, 2016-17 Spring**

**A8: Camera Class**

**Due: Wednesday April 5th 2017 at 23:59Hrs.**

Instructions:

This homework assignment is meant to be done in teams (teams of two).

For this homework assignment you will design and develop a camera class, there are just a couple of restrictions and instructions but at the end of the day you your implementation will need to provide two matrices: The View matrix and the Projection matrix.

Both the View and the Projections can be obtained using glm but if you want to calculate them by hand I’m not opposed to. If you decide to use a math library other than glm (this includes your own) to calculate any math related matrix you need to also provide a method that translates to glm::mat4 (matrix4)

**Restrictions:**

Your camera class will be handle through a CameraManager singleton in a future assignment, plan ahead of time.

Your orientations should be done through quaternions.

You should avoid the dreadful gimbal lock (which kind of ties in to the previous point)

**Grade:**

Your grade will be assigned based on the implementation of the following methods in your camera class:

5% - matrix4 GetView(void); //Should get the View Matrix from your class

5% - matrix4 GetProjection(bool bOrtographic); //Should get the Projection Matrix from your class, this function should let me choose between a Perspective or an Orthographic projection depending on the bool argument.

5% - void SetPosition(vector3 v3Position);//Sets the position of the camera

5% - void SetTarget(vector3 v3Target);//Sets the target of the camera

5% - void SetUp(vector3 v3Up);//Sets the up vector of the camera

10% - void MoveForward(float fIncrement);//Move the camera forward (or backwards) depending on the “view vector”

10% - void MoveSideways(float fIncrement);//Move the camera based on the “right” vectordepending on the “view vector”

10% - void MoveVertical(float fIncrement);//Move the camera up or down depending on the “up vector”

10% - void ChangePitch (float fIncrement);//Orient your camera (meaning the forward and up vectors) changing its angle in x

10% - void ChangeRoll (float fIncrement);//Orient your camera (meaning the forward and up vectors) changing its angle in z

10% - void ChangeYaw (float fIncrement);//Orient your camera (meaning the forward and up vectors) changing its angle in y

5% You should provide a project that has a scene populated with a couple of objects in such a way that we can move around and notice a change.

5% You should provide actual controllers related to your keyboard input W,A,S,D should let us move like a FPS controllers Q and E should let us move up and down respectably. You can use any library to control the camera, but you need to provide any related .lib or .dll files, you cannot assume it is installed in the lab computers or if you are using a controller that we will have one available for grading your project; that said if you can use a controller that is great! But we will be grading based on the keyboard.

5% You should provide actual controllers related to your view using the mouse, changing the camera View and Pitch

No starter code is provided, if you want to use the ReEngine you can clone a project within the solution.

You can use ReEngine in order to make the project or create your own solution with the framework of your choice, if you decide to use ReEngine you will need to set the View matrix and the Projection to the object to render in a line of code like this:

/\*

Assuming m\_pCamera is the name of your camera object.

\*/

m\_pCone->Render(

m\_pCamera->GetProjectionMatrix(), //Gets the projection matrix in a glm::mat4 format

m\_pCamera->GetViewMatrix(), //Gets the current view matrix in a glm::mat4 format

mModelToWorld); //current matrix associated with your model in a glm::mat4 format

There is no extra credit for this assignment.

You will need to add a txt file with the name of the teammates that work on this assignment in the root folder.

For your submission you need to create a repository and provide us the address is stored into (even if it’s the repository you have been using for the class for any of the team members). There is a dropbox in MyCourses and you can submit your project to it (only the project, not the whole solution, so your submission should be about 100kb) and in the comments you need to specify the address of your repo so we can clone it.

Submit to the dropbox labeled A8\_CameraClass