Prof. G. Zachmann
J. Roßkamp (j.rosskamp@cs.uni-bremen.de)

University of Bremen
School of Computer Science
CGVR Group
November 22, 2018

Winter Semester 2018/19

Assignment on Virtual Reality and Physically-Based Simulation - Sheet 2

Due Date December 2. 2018

For this exercises, take a look at the Unreal Engine 4 Documentation and in case of problems also the UE4 AnswerHub.

Exercise 1 (View Frustum, *5 Credits*)

In the lecture the calculation of the view frustum for two eyes was discussed. Let us now discuss the case of a one-eyed moving spectator. The eye position is changing, so we have to compute the frustum everytime the eye position is changed. For example see `https://www.youtube.com/watch?v=hvrT7FqpPQE`. The view frustum in OpenGL is set with glfrustum(left,right,bottom,top,near,far). Your task is to calculate these values for the setup shown in figure 1. The following exercises should guide you through the calculation.

a) Calculate the orthonormal basis $(\vec{v}_r, \vec{v}_u, \vec{v}_r)$ of the projection plane by using the screen corners p_1, p_2, p_3.
b) Compute the vectors $(\vec{v}_1, \vec{v}_2, \vec{v}_3.\vec{v}_4)$ from the eye position p_e to the corners of the projection plane
c) Calculate the distance d between eye position and projection plane.
d) Compute left, right, bottom, top (l,r,b,t)
e) Convert l,r,b,t from the projection plane to the near plane, because glfrustum is defined for the near plane

In order to check, if your calculations are correct, use the following numbers: $n = 1$(near plane), $d = 3, \vec{v}_r = (1, 0, 0), \vec{v}_1 = (2, 1, -10)$. This results in $l = 2/3$. Please note, that these numbers are completely made up and have no meaning.

Exercise 2 (Unreal editor (C++), *4 Credits*)

In this exercise, the goal is to implement the results from exercise 1 in UE4. There is an example project you can download from `http://cgvr.informatik.uni-bremen.de/teaching/vr_1819/uebungen/UEExerciseProjection.zip`. In the following there will be short introduction on how to use this example project. Create the Visual Studio solution file (.*sln) and navigate to *Plugins/UE4-Plugin-OffAxis/Source/OffAxisProjection/Private/OffAxisLocalPlayer.cpp* and compile. Open the project in UE4. You should now see an empty scene with an OffAxisActor in placed at (0,0,0). It is important that the position of the OffAxisActor is not changed. After you hit play you can switch to OffAxis mode by pressing P. You can navigate by using the arrow keys.

You should add your code to *UOffAxisLocalPlayer::GenerateOffAxisMatrix_Internal_Slow*. After you changed the code you should recompile using VS and restart UE4. Otherwise Unreal won't detect your changes.

a) Add a message on the screen (use *GEngine->AddOnScreenDebugMessage*)

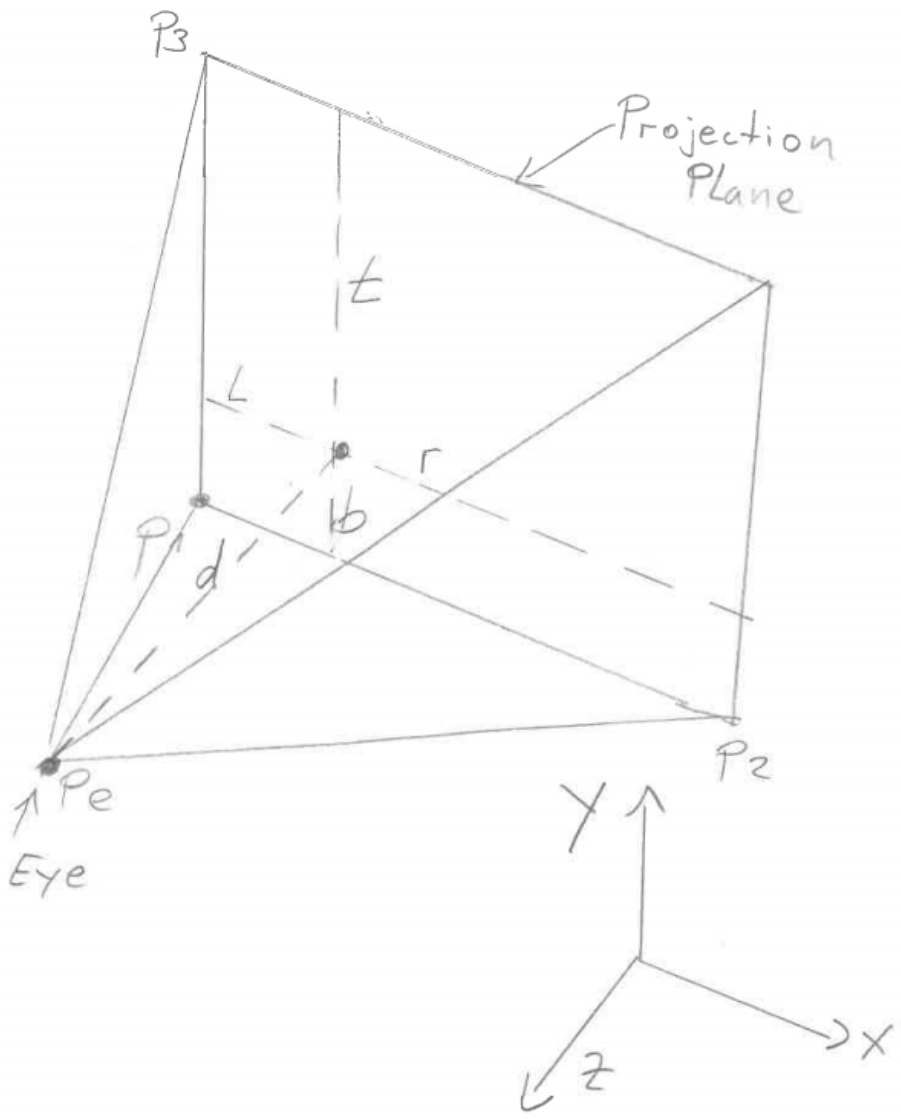

Figure 1: Setup one-eyed moving spectator

b) Implement the calculations for l,r,b,t from exercise 1
c) Print the values for l,r,b,t to the screen
d) Create a simple scene to test your implementation