## Task 2 Cube rendering

First, we setup the cube with an array of 3d vectors and its Model Matrix. The Model matrix starts as an Identity Matrix, which means that it is right at the center of the world (so no transformations are applied to the cube). Each vector in the array resembles the position of a vertex.

Second, I setup the view matrix, which is the position of the camera for our scene. I place it at position (0,0, -2) and let it look at the center (0,0,0) and define the up vector to be the y axis (0,1,0). I place it at z -2 so I have some space between the camera and the cube.

Third, we define the projection matrix, by setting the FOV to 90 degrees and min and max depth as well as the screen aspect ratio.

Fourth, we apply some transformations to the model matrix of the cube –a 45 degree rotation around the z-axis.

Then we log all the necessary data, and, in the end, we calculate all the screen positions. Therefore, we take the precalculated MVP matrix and multiply it to each vertex of the cube – which gives us the normalized pixel coordinate. This value then needs to be transformed to pixel space, which I do with the function transformToPixelSpace().

For transforming, I deliberately ignored the clipping and a screen offset. That's the reason why we also can receive negative pixel coordinates after rotating the cube.

## Positions – copied from the terminal

```
Model Matrix: [
mat4x4((0.707107, 0.707107, 0.000000, 0.000000), (-0.707107, 0.707107, 0.000000,
0.000000), (0.000000, 0.000000, 1.000000, 0.000000), (0.000000, 0.000000, 0.000000,
1.000000))
]
View Matrix: [
mat4x4((-1.000000, 0.000000, -0.000000, 0.000000), (0.000000, 1.000000, -0.000000,
0.000000), (0.000000, -0.000000, -1.000000, 0.000000), (-0.000000, -0.000000, -2.000000,
1.000000))
]
Projection Matrix: [
mat4x4((0.750000, 0.000000, 0.000000, 0.000000), (0.000000, 1.000000, 0.000000,
0.000000), (0.000000, 0.000000, -1.001001, -1.000000), (0.000000, 0.000000, -0.100100,
0.000000))
]
Mvp Matrix: [
mat4x4((-0.530330, 0.707107, 0.000000, 0.000000), (0.530330, 0.707107, 0.000000,
0.000000), (0.000000, 0.000000, 1.001001, 1.000000), (0.000000, 0.000000, 1.901902,
2.000000))
]
Cube vertices: [
Vertex 0: vec3(1.000000, 1.000000, 1.000000)
Vertex 1: vec3(1.000000, -1.000000, 1.000000)
Vertex 2: vec3(-1.000000, -1.000000, 1.000000)
Vertex 3: vec3(-1.000000, 1.000000, 1.000000)
Vertex 4: vec3(1.000000, 1.000000, -1.000000)
Vertex 5: vec3(1.000000, -1.000000, -1.000000)
Vertex 6: vec3(-1.000000, -1.000000, -1.000000)
Vertex 7: vec3(-1.000000, 1.000000, -1.000000)
]
NDC positions: [
0:vec3(0.000000, 1.414214, 2.902903)
1:vec3(-1.060660, 0.000000, 2.902903)
2:vec3(0.000000, -1.414214, 2.902903)
3:vec3(1.060660, 0.000000, 2.902903)
4:vec3(0.000000, 1.414214, 0.900901)
5:vec3(-1.060660, 0.000000, 0.900901)
6:vec3(0.000000, -1.414214, 0.900901)
7:vec3(1.060660, 0.000000, 0.900901)
]
Screen positions: [
0:vec2(400.000000, 124.264061)
1:vec2(-24.264050, -300.000000)
2:vec2(400.000000, -724.264099)
3:vec2(824.264038, -300.000000)
4:vec2(400.000000, 124.264061)
5:vec2(-24.264050, -300.000000)
6:vec2(400.000000, -724.264099)
7:vec2(824.264038, -300.000000)
]
```

## Discussion

Notice: The screen has a resolution of 800x600 pixels.

As we can see, that some vertices are outside the screen. I think this comes from the fact that we rotate the cube around the z-axis moving it from a position, where it is small enough for the screen to a position where some vertices are outside. Rotating the cube in the z-axis for 45 degrees moves the vertices of its front surface to the top, bottom, left and right, making them reach outside the screen. So it is only naturally that some of the vertices are not visible anymore.

I also noticed, that some of the vertices have the same values (vertex 7 and 3 e.g.). I think that happens because we only removed the 3rd axis of the ndc vector when transforming to pixel space. For that reason, these vertices have the same screen position, but one of them is hidden by the other.