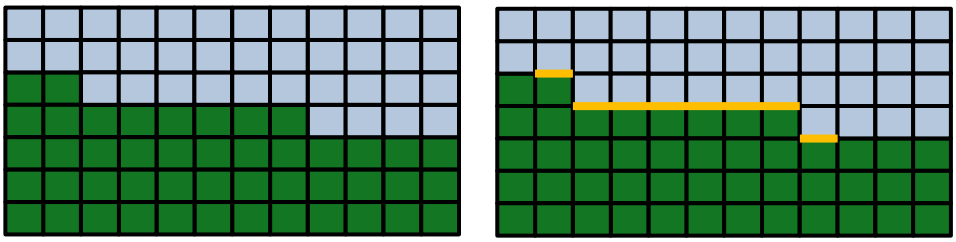
Hello dear developers,

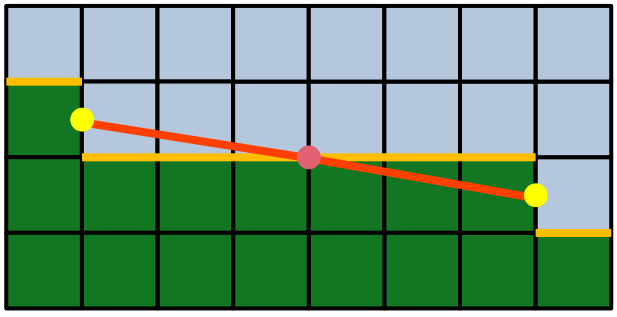
I would like to introduce you to my new anti-aliasing method which I called **LFAA-HV_(Line-Fitting Anti-Aliasing_Horizontal_And_Vertical)**. I developed it in hopes that it could be used in VR games. This method allows for high-quality smoothing of aliasing on edges and in textures using a simple calculation. Its only weakness is the dashed lines and aliasing occurring between frames. For a perfect result, it would have to be combined with another method that would solve these shortcomings. Also, the method would still need to be optimized to run in real time in VR. In any case, I would like to share this method and release the code for further development of the method.

Principle of the **LFAA – HV** method:
The code contains 2 main parts: horizontal smoothing and vertical smoothing. The first part works with individual rows of the image, and the second with columns. Their principle is almost identical, so we will only explain horizontal smoothing.
For horizontal antialiasing, I allocate one entire image line to each GPU thread to process. I'll go pixel by pixel from left to right looking for a distinct light transition between this line and the line below it. I identify all horizontal edges in this row. For each such edge, I find out how it connects to the edges in the surrounding rows on the left and right. Based on that, I estimate how the edge needs to be tilted (uphill, downhill, left straight). In addition, each such edge is divided into 2 parts - the left shoulder and the right shoulder (the center point of the edge remains unchanged). For each arm, I will independently estimate this tilt to the center of this edge. Then I recalculate the R G B values of all the pixels in the row and the row below that enclose this edge and create the optical illusion of the edge tilting. Let's show it graphically in steps:

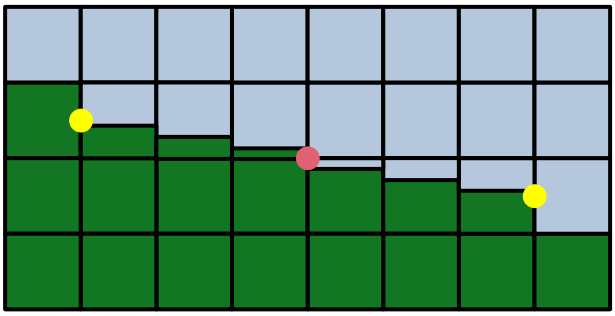
1/ We identify the edge and its connections



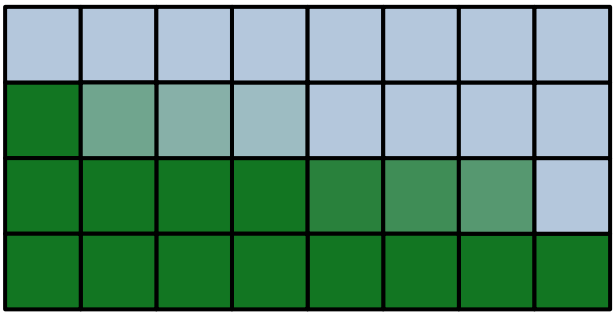
2/ We determine the center of the edge, divide the edge into left and right arms and determine the direction of inclination of the arms to the center of the edge based on their connections



3/ We calculate for each position x the percentage ratio of pixels so that it copies this tilt
If we were to represent the percentages graphically, it would look like this



4/ Now we recalculate the colors of the pixels holding the inclined arms according to the percentage representation.



Note: If we smoothed out all the edges we see in this image, it might look something like this: