This week I was interested in creative code around typography after coming across it on the p5 reference page. I decided instead of twitter I would look on Instagram to find some pieces. The first one is by a user named @mattiazucchelli\_. Their piece is the word fade with a fade effect that makes it look like the word is dust being carried away by the wind. The image is quite minimalistic with a green/brown background with white text. The second example is a countdown video by @josefluispelz. This video reminds me of a colour-blind test in the design of it. It’s quite hard to look at it as it has a lot going on. The design reminds me of a particles diagram from chemistry. The numbers are accentuated by being bigger than the background – this has an effect of pushing them to the foreground. The third piece is by user @andreiongd. This is a quote which has a wave warp effect running across the right side of the video. This distorts the text a little bit, but it still remains readable. The lined paper background adds to the melancholic aesthetic that the wave warp and black text are trying to produce.

Interested in how to do this I looked up “creative coding typography” into YouTube and they were all titled “kinetic typography”. This informed me I had been looking into the wrong thing, so I went back to Instagram and found some more examples. My first example of kinetic typography is a design uploaded by @contemporarytype. It’s a video of the word “design” wobbling around the screen but leaving a trail of blank letters when it moves. It has quite a psychedelic effect as well as natural feel to the movement. I believe a design like this could potentially be possible for my creative coding assignment. The second example is by user @cristiania who has made the word snow fill in white and then be changed to black just as quick. I believe this is being done but quickly pushing the black letters off the screen to the right when the white letters from the left come in and make contact with the black letters. This is different to how I imagined it was done at first glance. On first impressions I thought it was a change of background(). The final example by @\_hesselucas is an example of motion graphics with typography. This video gives the details they wanted to convey in a simplistic way with contrasting colours to make everything clearer to the audience. It has been performed in a 4-point pivot around a central square. It reminds me of the 3d camera tool in adobe after effects as the perspective (field of depth) of the video seems to change throughout as they move side to side to show new information.

This week I used the p5 reference page and went to the typography section, while I scrolled, I saw the camera section and after thinking about 3d camera I decided to check the camera section out as well. The textalign() tool reminded me of html but I also recall using it for one of the worksheets. textleading() sets the number space in between each line of text. textAscent() and textDescent() shows the distance of the tallest character above and below the baseline. loadFont() will be a useful tool as it allows for the import of fonts. p5 showed the syntax to be loadFont(path,[callback,[onError]). This path is the file or URL to load for the font. The other two were optional. onError seemed to be the more useful tool as it works like a backup plan and will execute another function if an error were to occur. The text() reference page had an interesting example of rotation being used. It used rotateZ() which gave the text a 3d effect as it rotated on the z axis.

The camera section was more complicated to get my head around and I am unsure if I have. The camera() tool had one example of a square moving back and forward in the screen – looked to be on a z axis rather than scaling the square up and down. The other example involved sliders. The sliders controlled 6 parameters of the cube. Depending on how you moved the sliders the camera would follow and show you the cube from different angles. The last thing I looked at was perspective(). perspective() involves a 3d sketch which I assume has a static camera in the middle you rotate around. Using this, any image closest to the perspective you are looking at appears larger while any further appear to be in the background. It adds a range of depth to the image.