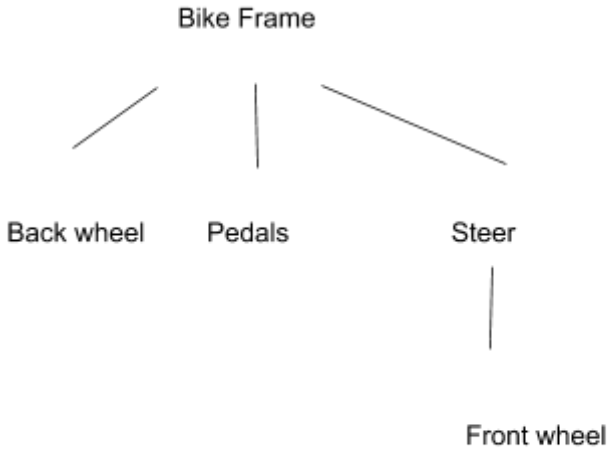
Computer Graphics – Assignment 2

Margot Sirdey & Balázs Orbán

1. Reference points

a)

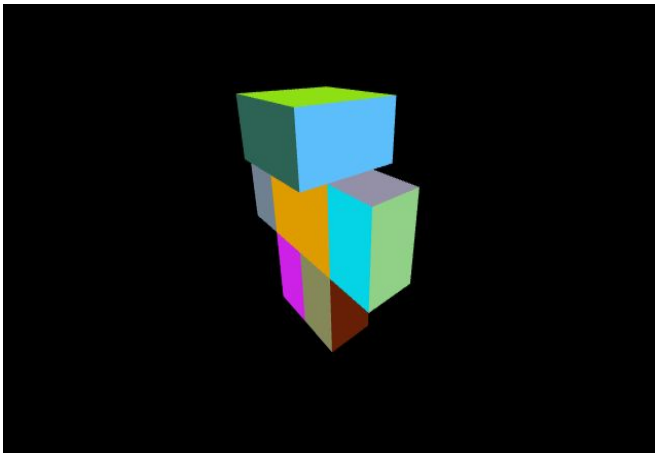
i) The Bike Frame is the root of everything because it is the actual part that connects the back wheel, the pedals and the steer. Those parts does not connect to each in any other way. (We do not consider the chain for example). The front wheel is directly connected to the steer, and through the steer to the rest of the bicycle.



ii) Three transformations are necessary :

- Translate by (-(-7) , -4), that is to say by (7,-4) along x, y and (ignoring the z-axis) from the yellow reference point.
  → now we are at the green reference point: (73,26)
- Rotate by 30° around the green reference point, z axis.
- Translate back to the yellow point : translation by (-7,4), then going to (66,30) along x and y axis

2. Getting started

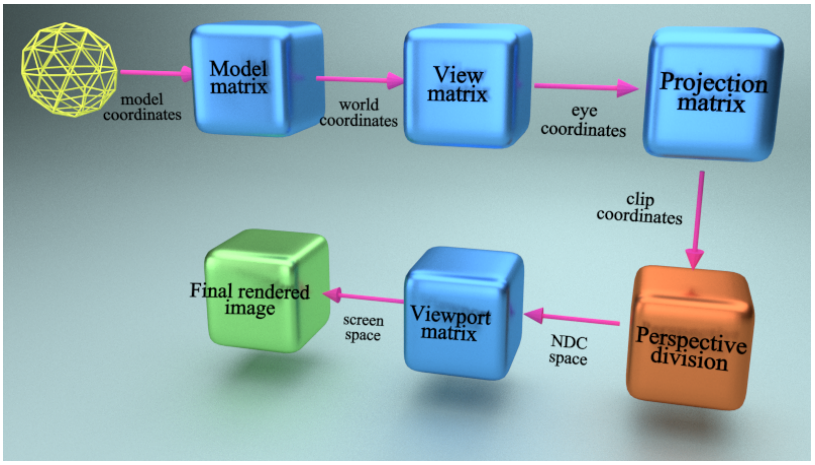
a) 

b)

i) In order to do translation transformation, we have to add a 4th coordinate: w. This component is usually used down the pipeline. It is impossible to distinguish points (w=1) and vectors (w=0) without a w component.
Note that w is usually unchanged by using homogeneous coordinates for a translation transformation.
When multiplying a vector with a 4x4 affine transformation-matrix that has a translation in its last row/column, the vector will also be translated. However, only points should be translated.
When w=0, we don't have this problem anymore.

ii) w component for a perspective transformation changes for each vertex. The w value is changed in accordance with the z value which represents the distance away from the camera. Then, w value changes the scale so that the projection matrix changes the w value based on the z value.
Thanks to this w component we can change the scale regarding the z value. We then just have to make a division (for all vertices) in order to put the w coordinate equals to 1 again (to be using homogeneous coordinates).

iii) The viewport transformation is the last matrix transformation which is applied to all vertices before they are drawn on the physical screen.
We need it to transform the normalized device coordinates' vertices to actual screen pixel coordinates. ie: all the vertices contained within the cube will be rendered, and those outside this range won't be.



3. Setting up the scene graph

See our code.

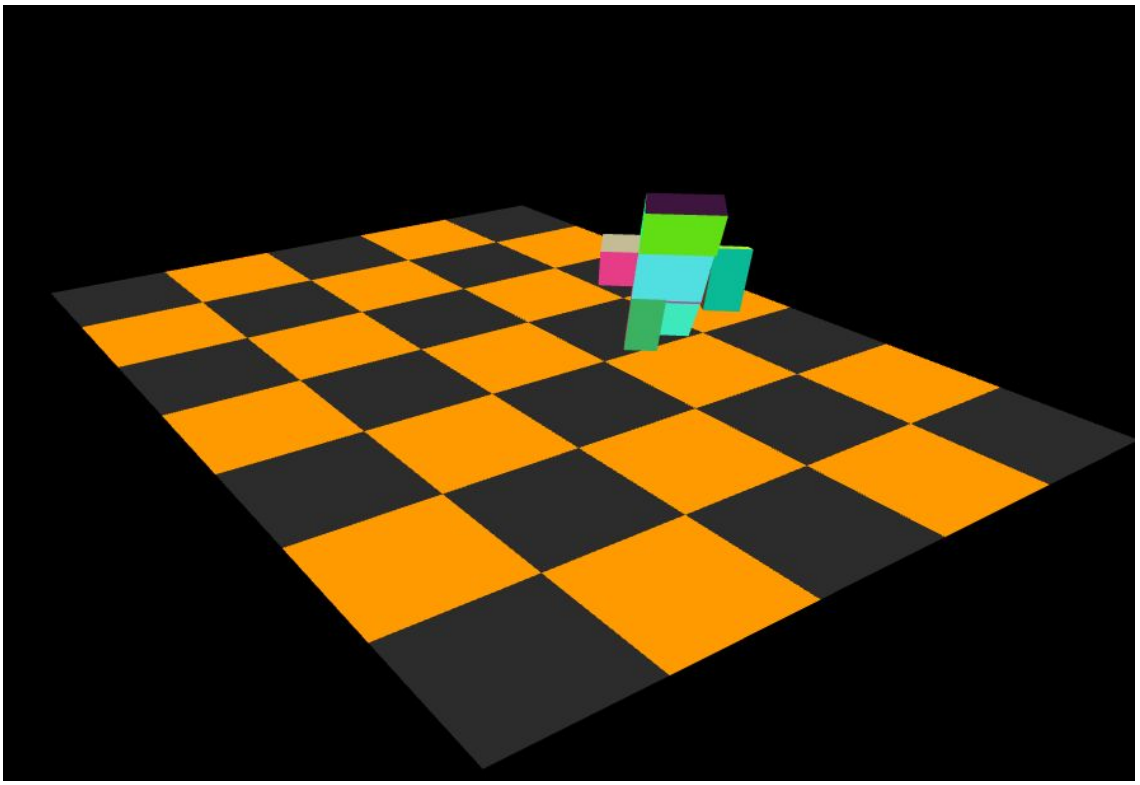
4. Traversing and updating the scene graph

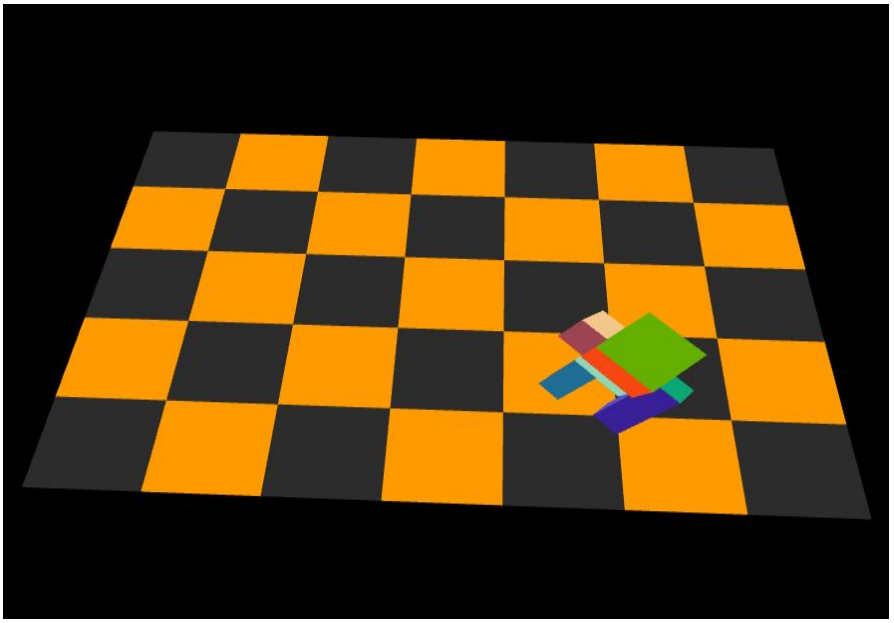
See our code.

5. Rendering the scene graph

See our code.

Screenshots :





6. Sources

- https://www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/ (b) i) and ii))
- http://www.thecodecrate.com/opengl-es/opengl-viewport-matrix/ (b) iii))