## Intermediary Results

# Models

The models will be box like to keep things simple, e.g. collision detection and to keep the focus on the programming task.

There is the need to read the vertex information from a file into a vertex array.

Every model is represented through a mesh component, which stores all the vertices.  
Meshes and texture are only initialized once and objects uses references of them.

Rotation is stored as matrix. This allows easy concatenation of the rotation.

Every object is in a list/vector, so it can be loop for rendering, updating and collision detection

To render textures on the objects, the shaders need to be adjusted.   
Different algorithms like Bilinear, Trilinear or anisotropic. Antialiasing.

A “PickupManager” manages the spawn of the pickups. It controls where they spawn and what type of pickup spawns.

The skybox is represented by a cube and textured on the inside (Anticlockwise indices). The skybox requires the highest value in the z-buffer, so it is always in the background.

Intersection check:

First check distance between objects, when they are close enough check for intersection

function intersect(a, b) {

return (a.minX <= b.maxX && a.maxX >= b.minX) &&

(a.minY <= b.maxY && a.maxY >= b.minY) &&

(a.minZ <= b.maxZ && a.maxZ >= b.minZ);

}

Explosion/Fire/Dust

Tutorial: <http://www.rastertek.com/dx11tut33.html>  
Shaders which manipulate noise Textures.  
Just set a plane at certain position with a transparent texture, which is always facing towards the player.

Sources:   
Millington Ian, Game Physics Engine Development, Second Edition, Boca Raton 2010  
Essential Mathematics for Games and Interactive Applications, Van Verth James M., third Edition, 2016 Boca Raton

https://developer.mozilla.org/en-US/docs/Games/Techniques/3D\_collision\_detection