CMS

Celestial Motion Simulator

# Introduction

# Models

In the entire project there is mainly one model that is being used, namely the sphere. All planets and also the skybox, that actually is a sphere, originate from the same model. This model is later turned into different objects with different properties such as texture and structure.

There is a model loader implemented that loads a file of sort .obj. There is also a sphere generator that generates a sphere of given vertex resolution with normals and texture coordinates.

# Objects

Objects are used to save resources by using the same model for several similarly looking objects. Each object has a list of properties illustrated in: UML.

# Texture

To texture a sphere is a known problem. In the stitch there will be one vertex that needs two different texture coordinates depending on from what side the texture is being interpolated. Otherwise one will get one big interpolation over the entire texture in the stich.

In this project it is solved in the sphere generator. It actually inserts two vertices at the same location, one with the texture coordinate zero and one with the texture coordinate one. These are then combined correctly to their neighbours but never with each other.

# Bump map

To add a feeling of structure to for instance the earth a bump/normal map is used. If one where to add enough vertices to add real height differences on the sphere the models would become too large. The normal map is an RGB image where the three different colour channels represents x, y and z. To add these to a plane is not a big deal. Then you simply change the given normals to the ones in your normal image; not very different from reading a texure. The real difficulties begin when you want to do this to a sphere. The normal map contains normals expressed in local model coordinates, which for a plane are simply x, y and z, but for a sphere this becomes the local normal, a tangent and a bitangent. These need to be aligned for all positions on the sphere for the mapping to look good. The normal is given in the model. To find the first tangent the cross product between the current position and the top of the sphere is used. To avoid problems at the top of the sphere the point to cross multiply with is located just slightly above the top. This way the cross product is always valid. To find the bitangent the cross product between the the normal and the first tangent is calculated.

ILLUSTRATION

# Specularity map

To add the effect of water being more specular than land on the earth a specularity map is used. This is used to adjust the specularity coefficient in the phong shader.

# Shadows

By implementing a plücker based ray caster in the fragment shader shadows where easily achieved. The computational complexity grows fast though. For every pixel you need to cast a ray from every light source to the pixel and then check the distance for every planet to this line. If the distance is smaller than the planet radius it generates a shadow in the current pixel. This does not give any soft shadows.

There exists different methods to get rid of the sharp shadow edges. The one mentioned in this course is to randomly alter the position of the light source and sum the contribution from every random position for every pixel. This method quickly becomes very computational heavy.

Another method that we invented on our own is to use an activation function of the light source position and radius and the blocking objects position and radius. The activation function could be for instance a sigmoid function.

ILLUSTRATION

# Shaders

To avoid unnecessary branches in the shader several different shaders are used designed specifically for their purpose. The earth is the object with the most texture data available and therefore it got a dedicated shader to handle texture blending, bump mapping, specularity mapping and shadows. The more common planets do not have a night texture, bump map or specularity map and hence need none of these features in their shader.

# Physics