Computer Graphics - Exercise 13

Maximilian Richter

Winter Semester 2022/23

1 Mipmaps

Let an image being used as a texture with side length resolution $2N$, where $n \in (1, 2, ..., N)$ is the mipmap level.

a) **Derive the function for the memory consumption $m(n)$ in byte of a corresponding mipmap with respect to the mipmap level n, as well as the function for the total memory consumption $\sum_{n=1}^{N} m(n)$. What is the result for $N = 12$?**

Let m_p be the memory consumption for on pixel in byte.

$$
\begin{aligned}
m(n) &= m_p 2^{2n} \\
\sum_{n=1}^{N} m(n) &= m_p \sum_{n=1}^{N} 2^{2n} \\
&= m_p 2^{2N} \sum_{n=1}^{N} \frac{2^{2n}}{2^{2N}} \\
&= m_p 2^{2N} \sum_{n=0}^{N-1} \frac{1}{2^{2n}} \\
&= m_p 2^{2N} (1 + \frac{1}{4} + ...) < m_p 2^{2N} \frac{4}{3}
\end{aligned}
$$

$$
\begin{aligned}
m(12) &= m_p 2^{2 \cdot 12} = 2^{24} m_p = 1,677,216 \\
\sum_{n=1}^{12} m(n) &= 22,369,625 \\
\frac{\sum_{n=1}^{12} m(n)}{m(12)} &= 1.33333325
\end{aligned}
$$

b) **Name at least two scenarios, in which you would use mipmaps**

- Precomputation for Antialiasing (Trilinear interpolation)
- Precomputation for small pictures with high resolution

c) **Provide some exemplary lines of code to realize mipmapping in OpenGL and explain the different commands.**

```
glBindTexture(GL_TEXTURE_2D, textureID);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glGenerateMipmap(GL_TEXTURE_2D);
```

The above code assumes that a texture has been loaded and bound to a texture ID `textureID`. The following is an explanation of the different commands:
`glBindTexture(GL_TEXTURE_2D, textureID)`: This command binds the texture with the specified ID to the 2D texture target, allowing for subsequent commands to affect this texture.
`glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR)`: This command sets the texture's minification filter to use mipmapping with linear interpolation between mip levels. This results in smoother transitions between mip levels as the texture is scaled down.
`glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)`: This command sets the texture's magnification filter to use linear interpolation. This is used when the texture is scaled up.
`glGenerateMipmap(GL_TEXTURE_2D)`: This command generates mipmaps for the texture bound to the specified target. Mipmaps are pre-filtered versions of the texture at different resolutions, allowing for efficient texture sampling at different distances and scales.

By using mipmapping, textures can be rendered more efficiently and with better visual quality at different scales and distances. The glGenerateMipmap command generates the mipmaps automatically, while the glTexParameteri commands set the filters to use when sampling the texture.

2 Geometry and Tessellation Shaders

The geometry shader in combination with the tessellation shader are useful tools, e.g., to achieve some grid refinement like in the case of terrain rendering. In general, one can implement two different tessellation shaders, namely the tessellation control and tessellation evaluation shaders.

a) **Describe the functionality of the geometry, tessellation control, and tessellation evaluation shaders. Consider the input and output parameters and**

describe the most important differences between them.

Geometry: Transformation of primitives.
Input: 1 primitive, Output: $0 - n$ primitives.
Tessellation Control: Sub-divide triangles by adding new vertices.
Input: Array of vertices, Output: Array of vertices.
Tesselation evaluation: Change the position of the vertices.
Input: Array of vertices, Output: Position of vertex (All vertices are computed in parallel)

b) **Think of at least two other interesting use cases, in which the geometry and tessellation shaders may be of use. Roughly distribute core functionalities to the individual shaders in order to realize each use case and justify your decision in short.**

From Slide 6.176: Use tessellation shader to produce sphere from cube.