Prof. Dr.-Ing. Ralf Steinmetz
Multimedia communications Lab

Dipl. Inf. Robert Konrad
Polona Caserman, M.Sc.

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Game Technology
Winter Semester 2017/2018

Solution 6

General Information

- The exercises may be solved by teams of up to three people.
- The solutions have to be uploaded to the Git repositories assigned to the individual teams.
- **The submission date (for practical and theoretical tasks) is noted on top of each exercise sheet.**
- If you have questions about the exercises write a mail to game-technology@kom.tu-darmstadt.de or use the forum at https://www.fachschaft.informatik.tu-darmstadt.de/forum/viewforum.php?f=557

P6. Practical Tasks: Bumps and Animations (5 Points)

Get the source code of this exercise from **https://github.com/TUDGameTechnology/Exercise6.git**. For a reference to how the exercise`s solution should look like, please look at the video on the course homepage.

Please remember to push into a branch called "exercise6".

You can find the solution source code in the exercise at https://github.com/TUDGameTechnology/Solution6.git.

P6.1 Normal Maps (2 points)

On the right side of the exercise, you can find a box. For this box, a tangent-space normal map is provided.

a) Implement the creation of the tangent space basis (see comments in Exercise.cpp). Use the vertices of the mesh and their UV coordinates for this task. Please finish Task T6.1 first, as you will be deriving the necessary formula there.

b) Implement the missing part of the pixel shader (see comments in shader.frag).

P6.2 Vertex Animation (3 points)

On the left side of the exercise, you can find a pie-chart with a peculiar shape... The object is provided as a mesh with the center of the shape at the model`s origin. Animate the model in the vertex shader so that the opening at the right side appears to open and close rhythmically.

Hints:

- You can treat the problem as 2D and ignore the z-coordinate of the model
- Think about how one vertex (e.g. at the "mouth") has to move and try to find functions which realize this movement, then apply them to all vertices.

T6. Theoretical Tasks: Graphics Mix (5 Points)

T6.1 Tangent Space Basis (1 point)

As a preparation for P6.1 a), please derive the formulae for calculating T and B, the tangent and binormal vectors, from the differences in the vertex positions and UV coordinates of the triangle:

deltaPos1 = deltaU1 * T + deltaV1 * B
deltaPos2 = deltaU2 * T + deltaV2 * B

Hint: Treat the two equations as a system of linear equations and solve it.

As hinted at in the exercise, we write out the equations and solve the resulting system of equations.
The easiest way to carry this out is by re-writing this as using a matrix and inverting it to solve the system:

$$\begin{pmatrix}\Delta x_1\\ \Delta x_2\end{pmatrix} = \begin{pmatrix}\Delta \mathrm{u}_1 & \Delta v_1\\ \Delta u_2 & \Delta v_2\end{pmatrix}\begin{pmatrix}T\\ B\end{pmatrix}$$

$$\begin{pmatrix}\Delta \mathrm{u}_1 & \Delta v_1\\ \Delta u_2 & \Delta v_2\end{pmatrix}^{-1}\begin{pmatrix}\Delta x_1\\ \Delta x_2\end{pmatrix} = \begin{pmatrix}T\\ B\end{pmatrix}$$

Calculate the inverse:

$$det\begin{pmatrix}\Delta \mathrm{u}_1 & \Delta v_1\\ \Delta u_2 & \Delta v_2\end{pmatrix} = \Delta \mathrm{u}_1 \Delta v_2 - \Delta u_2 \Delta v_1 = d$$

$$\begin{pmatrix} \Delta u_1 & \Delta v_1 \\ \Delta u_2 & \Delta v_2 \end{pmatrix}^{-1} = \frac{1}{d}\begin{pmatrix} \Delta v_2 & -\Delta v_1 \\ -\Delta u_2 & \Delta u_1 \end{pmatrix}$$

We get:

$$\begin{pmatrix} T \\ B \end{pmatrix} = \frac{1}{d}\begin{pmatrix} \Delta v_2 & -\Delta v_1 \\ -\Delta u_2 & \Delta u_1 \end{pmatrix}\begin{pmatrix} \Delta x_1 \\ \Delta x_2 \end{pmatrix}$$

$$\begin{pmatrix} T \\ B \end{pmatrix} = \frac{1}{d}\begin{pmatrix} \Delta v_2 \Delta x_1 - \Delta v_1 \Delta x_2 \\ -\Delta u_2 \Delta x_1 + \Delta u_1 \Delta x_2 \end{pmatrix}$$

$$T = \frac{1}{d}(\Delta v_2 \Delta x_1 - \Delta v_1 \Delta x_2)$$

$$B = \frac{1}{d}(-\Delta u_2 \Delta x_1 + \Delta u_1 \Delta x_2)$$

We see that we can re-use the determinant in the calculations.

T6.2 Particles (1 point)

Particle systems consist of lots and lots of semi-transparent billboards. Depth buffer-based 3D rendering does not handle transparency well. What problem must be avoided and can that be done efficiently? When all is set and done, what is likely to be the biggest performance burden when rendering particles?

The problem we need to avoid is drawing the alpha-blended semitransparent particles in the wrong order. To counteract this, particles have to be sorted by distance.

If particles are aligned to always show towards the camera, this can be done somewhat efficiently, since they cannot intersect in this case.

The biggest performance problem is typically overdraw because the graphics chip renders lots of semi-transparent pixels on top of each other. Overdraw is the problem that the same pixel in the output image is drawn several times. When a z-buffer is used, this problem is usually lessened, since pixels that do not pass the z-buffer-test are rejected and not drawn at all. However, for semi-transparent geometry, this is not possible.

T6.3 Normal Mapping vs. Displacement mapping (2 point)

Which view angles are most advantageous and which ones are most disadvantageous when we use “normal mapping”? What about when we use “displacement mapping”?

Very flat angles are bad, since we can see that there is no outline generated, i.e. the effect stops at the border of the surface. Conversely, steep angles are good, since we can't see the borders and the effect is not lost.

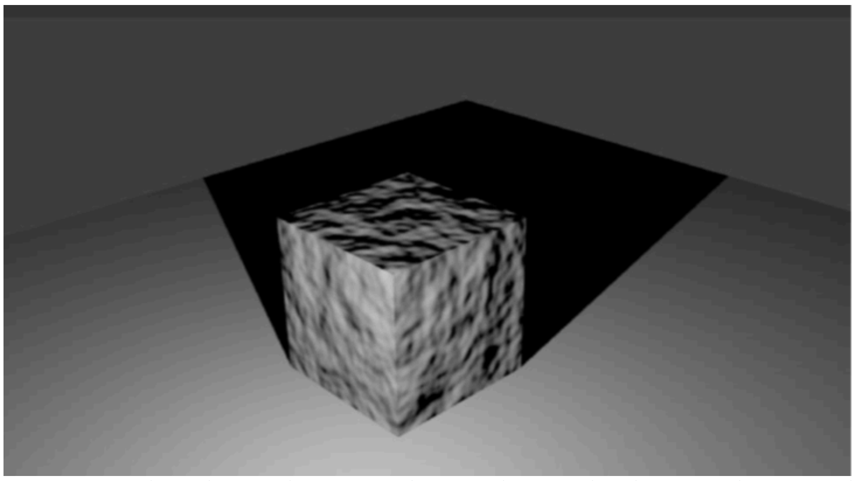

Figure 1: A cube with normal mapping. The normal map makes the material appear to be rough.

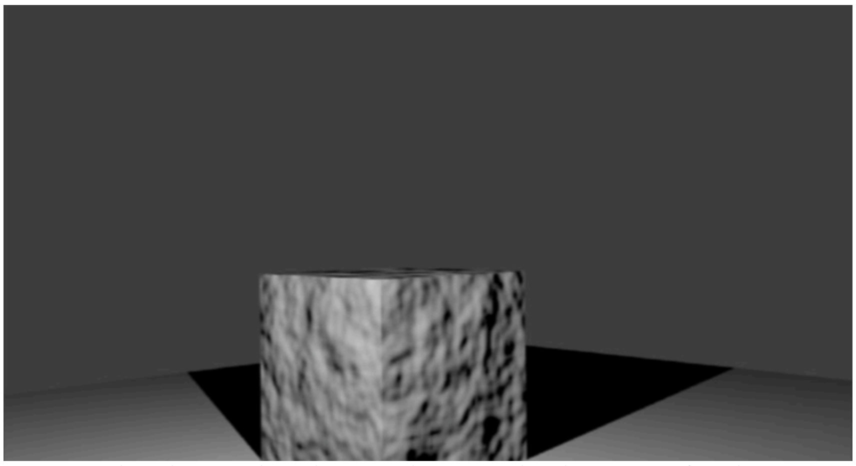

Figure 2: The cube viewed at a lower angle. We see that the upper surface does not appear to be rough any more.

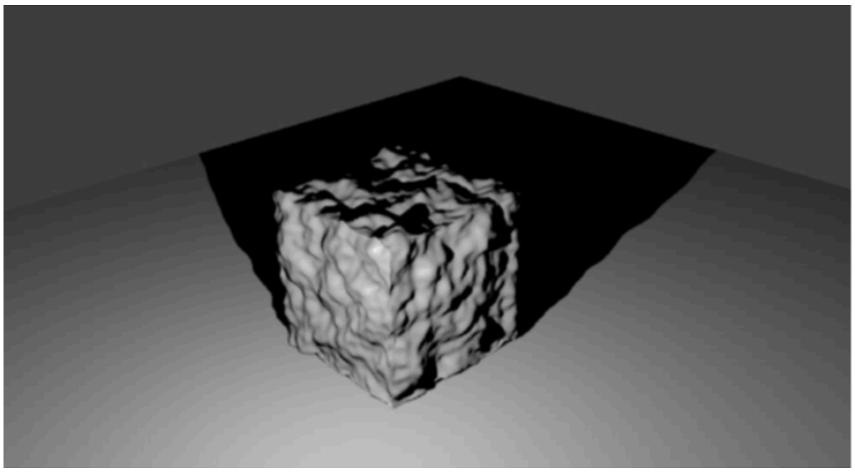

Figure 3: The cube, with displacement mapping. We see that the outline and the shadow is not straight. However, this involves rendering more triangles or tessellating the mesh in realtime.

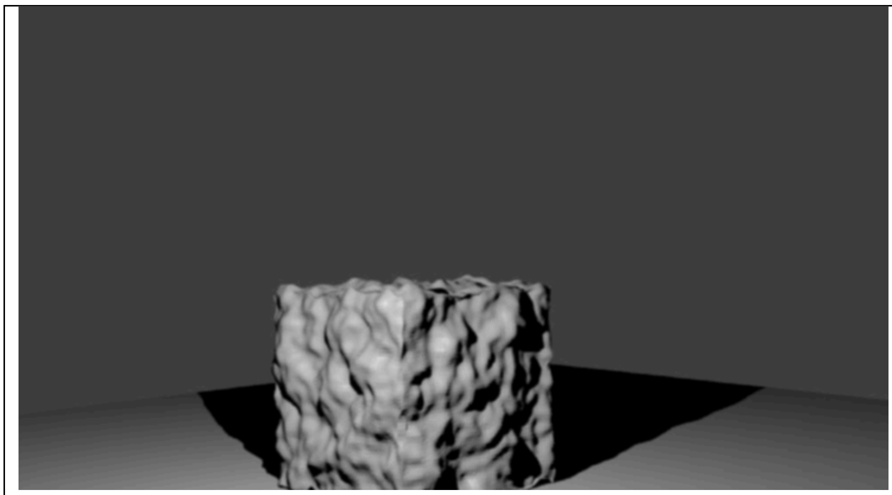
Figure 4: The displacement-mapped cube seen from a flat angle.

T6.4 Skeletal Animations (1 point)

Will the problem we referred to as "Achselhölle"/"Candy Wrapper Problem" in the lecture also occur if the most number of bones that influence any vertex in the mesh is 1, i.e. if every vertex is only bound to one bone? Explain your answer.

Yes, it will also occur in this configuration.

The vertices will still be connected with edges. If a loop of vertices v1 is bound to bone b1 only and a second loop v2 is connected to it and bound to bone b2 only, if one or both bones are rotated, the effect will occur.

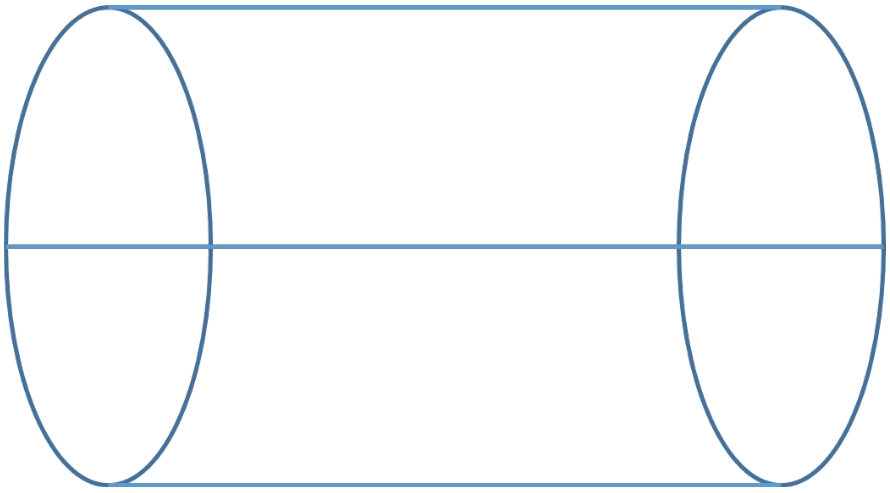
Figure 5: Two loops of vertices connected by edges.

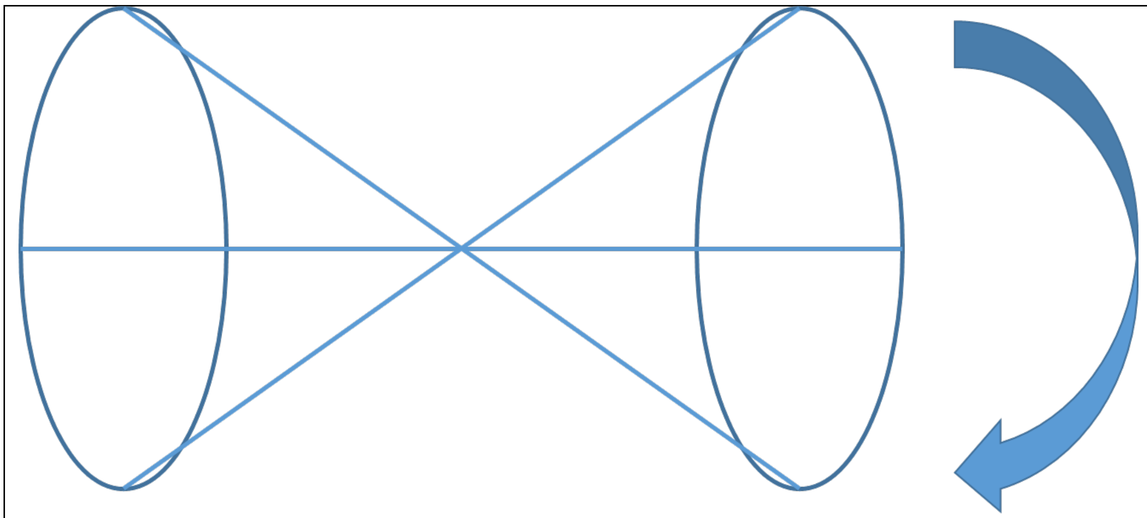

Figure 6: The same configuration, with the right bone rotated by 180 degrees. The candy wrapper effect occurs.