LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK
MEDIENINFORMATIK
PROF. DR. ANDREAS BUTZ, CHANGKUN OU, DAVID ENGLMEIER
COMPUTERGRAFIK 1, SOMMERSEMESTER 2020

Final: Exam Review

This is not an assignment but a document for you to prepare the exam.

1 Feedback

First of all, we would like to thank you for your participation in this online semester, and we would love to hear further feedback from you regarding the course content after the course is complete.

To submit your feedback, write a short paragraph (< 300 words) and tell us your thoughts regarding the following topics (mostly just one sentence for each subject):

- Your original expectation *before* joining the course. For example, "interested in CG", "want to know how to build a video game", "interested in graphics modeling", "just want to get ECTS", or any other possible reason from your side.
- The best, worst, most loved, and hated course content, including lectures and tutorials.
- Your suggestions to make this course better. For instance, "add more interesting topics about XYZ", "stop using JavaScript and switch to C++", "no programming at all"etc.
- Your further plan (if you have one) to proceed with graphics knowledge after the course. For example, "take another advanced graphics course", "start building your new graphics project", "start creating your dreamed characters", "doing a thesis in graphics", "never touch graphics again" etc.
- Anything that you would like to let us know.

Unlike the mid-term survey, this feedback is not anonymous, as we *might* want to do a follow up on your feedback and hear more from your side. Thus, submit your feedback in a text file (`.txt`) to Uni2Work if you would like to share your further feedbacks.

2 Course Overview

We prepared a mind map for the course overview, as shown in Figure 1. You can find a bigger version here[1]. Note that the mind map is for your inspiration, and we recommend creating your version of the course mind map because it helps you to see the connection between different topics.

In the mind map, the course is separated into four major parts: *rasterization*, *rendering*, *animation*, and *interaction*, where the most frequently discussed topics are located in rasterization and rendering. This does not mean that the animation and interaction are not essential for this course. Instead, they are more advanced topics that go slightly beyond a basic computer graphics course. For instance, you need more physics background (e.g., mechanics) to create a physically-based 3D world in animation and more human side knowledge from psychology and even biology in interaction. You need to acquire these pieces of wisdom from other courses that are not in the prerequisites.

[1] http://www.medien.ifi.lmu.de/lehre/ss20/cg1/tutorials/cg1-tutorial-appendix-mindmap.pdf

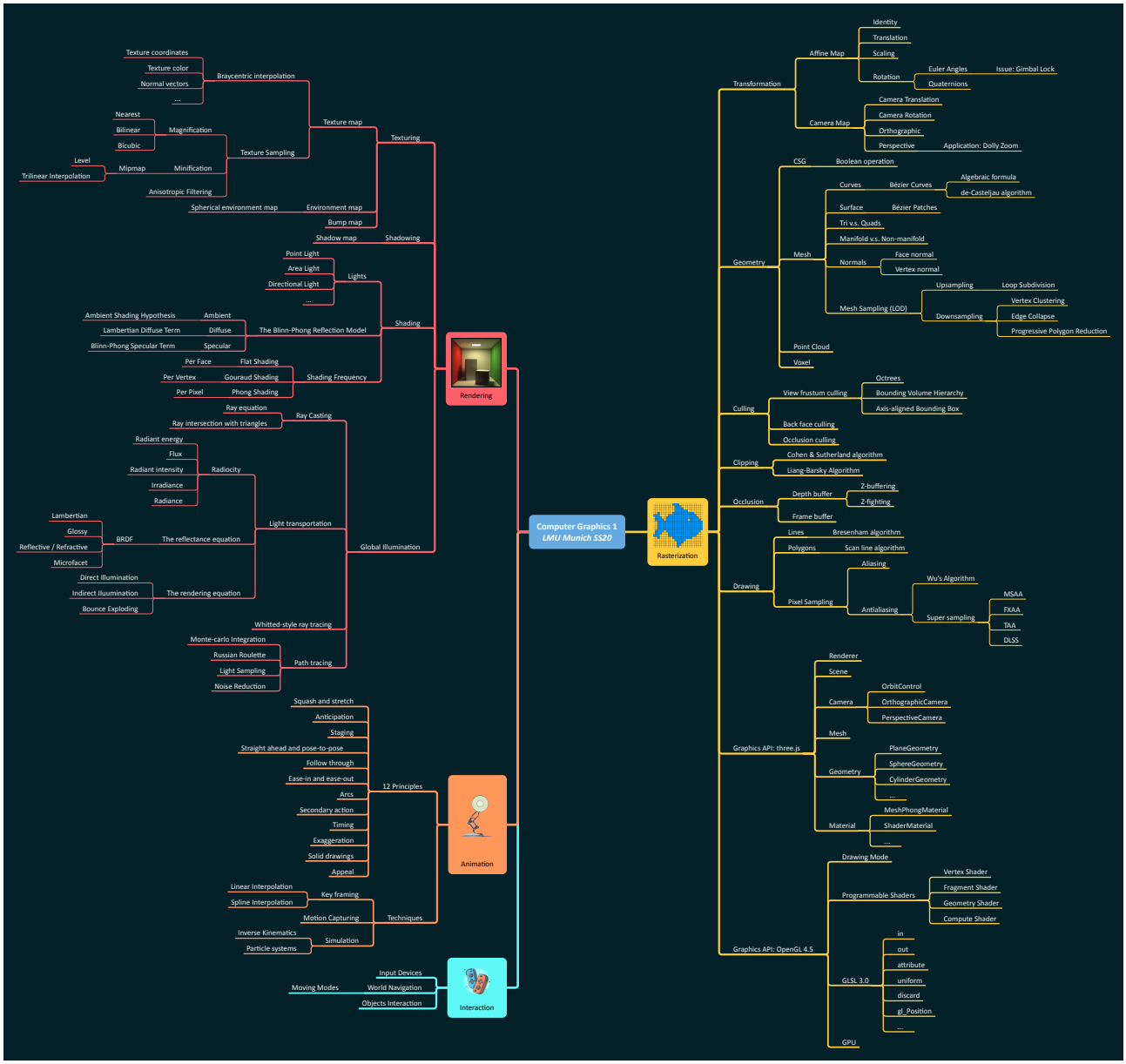

Figure 1: A CG1 mind map: This mind map is only for demonstration purpose.

In rasterization, we spend almost the entire tutorial (except the last one) on constructing the whole graphics pipeline and also demonstrated an example regarding how a graphics API (OpenGL) builds its pipeline to support user customization. The meaning of the word "shader", in this context, has been extended to manipulating produced interminate results along the graphics pipeline.

Along the pipeline, a rasterizer transforms the world in camera space, then projects the view frustum into a canonical cube. With the canonical cube, we can rescale the scene into the viewport using x-y coordinates, as well as testing occlusion with depth buffers using the z-coordinates in the subsequent processing. Moreover, several techniques are applied to speed up the rasterization processing, e.g., culling and clipping, etc.

We bring everything in a frame buffer using drawing algorithms for the discretization to draw and display everything on to our monitor. Finally, the frame buffer sends its content to our display. Remember, the devil is in the details. We have oversimplified the rasterization process in this overview. You should be aware of many related methods, e.g., sampling, antialiasing, etc.

For better appearance modeling, we also discussed texturing, and local illumination models in rendering, where different types of textures play different roles for faking materials. Local illumination models tell you the nature of light and shadows and how to mock them in real-time. Furthermore, the discussion of global illumination with ray tracing opens a new rendering framework compared to local shading. The ray tracing pipeline is different from the rasterization pipeline, and thus, severe issues are intrinsically not easily solved with rasterization.

To sum up, in the "Online-Hausarbeiten", you won't get a task whose solution can be found directly in the slides of the lecture/tutorial. Instead, you should be able to apply what you have learned and understood in the lectures and tutorials to problems that were not discussed in detail, i.e., *transfer knowledge questions*. For instance, you learned to keyframe a virtual camera along a spline curve with rasterization and animation. It is reasonable for you to transfer this knowledge to move an arbitrary object.

3 Question List

Maybe you were overwhelmed by this vast graphics field. There are so many topics, aren't they? We also created a list of questions for you to test yourself, use it for good.

3.1 Transformation

- What are the differences between point and vector?
- How to compute the vector norm?
- How to perform a vector dot/cross product?
- What is the geometric meaning of the cross product? Give an example
- How to perform matrix multiplication?
- How to express cross product using matrix notation?
- What is the geometric meaning of matrix multiplication? Give an example
- What are the span and basis?
- What is the geometric sense of span and basis? Give an example
- What is an orthonormal basis, and why do we need it?
- How to compute an orthonormal basis?
- What is a determinant?
- How to calculate 2x2, 3x3, and 4x4 determinants?

- What is the geometric meaning of a determinant? Give an example
- Name differences of left-handed coordinate frame and right-handed coordinate reference frame
- How to express coordinates in a spherical coordinate reference frame?
- What are homogeneous coordinates, and why do we need it?
- Why (x, y, z, 1) and (wx, wy, wz, w) are equivalent?
- What is a linear transformation, and how to verify a transformation is linear?
- How to express scale, rotation, shear, reflection, translation, as affine transformation?
- What are the properties of linear/affine/isometric transformation?
- What are the Euler angles?
- What are the types of Euler angles?
- How to express 3D rotations with Euler angles?
- What is a quaternion?
- How to represent 3D rotations with quaternions?
- What is a gimbal lock? Explain how it occurs
- How does using quaternions solve the gimbal lock issue?
- What is a scene graph, and how to compute a model (object) transformation?

3.2 Geometry

- What is CSG? Why and why not use it?
- What are the boolean operations? How to construct a geometry with CSG?
- What are the geometric primitives?
- Give an object example that can be constructed using extrusion
- Which geometric primitive is used in your case?
- What is a voxel, and when are they useful?
- What so special with point-based representation?
- What is interpolation?
- What is the Perlin noise, and how it works?
- How to draw a Bézier curve?
- What is the de Casteljau algorithm, and how it works?
- How to express a Bézier curve using Bernstein basis?
- What are the properties of Bézier curves?

- What is the piecewise Bézier curve, and why do we need it?
- What is a Bézier surface?
- Which data structure can be used to store a mesh?
- How to determine the front side of a polygon?
- Why triangles and why quads?
- Distinguish non-manifold from manifold surfaces
- What is the face normal and the vertex normal?
- What is mesh simplification?
- What is vertex clustering, and how it works?
- What are the drawbacks of vertex clustering?
- What is Melax's progressive polygon reduction, and how it works?
- What is mesh subdivision?
- What is the loop subdivision, and how it works?
- Name a subdivision approach for quad-meshes
- Why do we need mesh simplification and subdivision?
- What are mesh downsampling and upsampling?
- What is mesh aliasing?
- What is LOD, and why do we need it?

3.3 Camera

- What is the motivation for camera view transformation?
- What is the orthographic projection?
- What are the advantages and disadvantages of orthographic projection?
- What are the parameters we need to define the view frustum in orthographic projection?
- What is perspective projection?
- What are the advantages and disadvantages of perspective projection?
- What is a vanishing point, and what is the maximum vanishing points in perspective projection?
- What are the parameters we need to define the view frustum in perspective projection?
- What are the differences between orthographic and perspective projection?
- How to derive the perspective projection matrix?
- What are MVP matrices?

- What is viewport transformation and how to compute it?
- What is Dolly Zoom, and how can it be implemented?
- Why is Dolly Zoom not always perfect?
- Describe the transformation pipeline from a 3D geometry to a 2D viewport

3.4 Rasterization

- What is the Painter's algorithm, and what are the drawbacks?
- What is the depth buffer, and how it works?
- What problem can occur with depth buffer in perspective projection and how to solve it?
- What is frame buffer, and why do we need it?
- What are the types of culling, and what are their differences?
- What is BVH, and why is it better than Octrees in culling?
- What is AABB, and why is it more useful than the other types of bounding boxes?
- What is clipping, and what are the differences compare with culling?
- What are the purposes of clipping, and how is it involved in the transformation pipeline?
- What is Cohen and Sutherland algorithm, and how it works?
- What is the Liang-Barsky algorithm, and how it works?
- What is Bresenham's algorithm, and how it works?
- What is the Scan-Line algorithm, and how it works for triangles?
- What is point aliasing, and how to deal with it?
- What is supersampling?
- Explain how MSAA works, and what is the cost of it?

3.5 Material

- What is a texture map?
- What is the barycentric interpolation, and why do we need it?
- How to compute barycentric coordinates given a triangle?
- How to interpolate colors using barycentric coordinates?
- How to interpolate texture coordinates?
- What is magnification/minification? Give three examples of different types of methods for it.
- What is mipmap, and why do we need it?
- What is the storage overhead of mipmap?
- How to select an appropriate level in a mipmap?
- What is the environment map, and what is the application of it?
- What is the bump map? Give an example and how it is applied then explain the limitation of it.
- What is the shadow map, and how to create it?
- What are the types of shadow maps, and what are the limitations of them?
- What is BRDF, and how it relates to materials? What are the inputs and outputs of it?
- Draw a picture that demonstrates light reflection behavior regarding specular, glossy, diffuse surface
- How to calculate and measure a BRDF?

3.6 Illumination

- What are the differences between local illumination and global illumination?
- What is shading? What is the difference comparing to shadowing?
- What are ambient, diffuse, and specular?
- What is ambient shading, and what is the assumption of ambient shading?
- What is the influence of the ambient coefficient?
- What is Lambertian shading, and what is the assumption of Lambertian shading?
- What is the influence of the diffusion coefficient?
- What is the impact of the specular coefficient?
- What are Phong, Blinn-Phong shading, and what is the assumption in the Phong and Blinn-Phong shading term?

- What are the Phong and Blinn-Phong reflection model? Write the complete formula.
- What is the impact of the shininess factor?
- What are flat, Gouraud, and Phong shading, and what are their differences?
- What is the implied conclusion from shading frequency?
- What is point light, area light, directional light?
- What is the rendering equation? Draw a picture then mark the components and explain all symbols
- What makes computing the rendering equation so hard?
- What is ray tracing, and how is it different from rasterization?
- What Whitted-style ray tracing, and what is wrong with it?
- What is radiant energy, and what is intensity?
- What is irradiance, and what is radiance?
- What is the difference between radiance and irradiance?
- What is direct illumination, and what is indirect illumination?
- What is path tracing, and what is different from (Whitted-style) ray tracing?
- What is the distributed ray tracing?
- What is the expected value, variance, bias, PDF, and CDF?
- What is Monte Carlo Integration, and how to calculate it?
- How is the Monte Carlo method used to solve the rendering equation?
- What is Russian Roulette, and why do we need it?
- What are biased and unbiased estimators?
- What is noise reduction, and why do we need it?

3.7 Animation & Interaction

- What are the 12 principles in animation?
- Give an animation example then explain each principle
- What is the keyframing and how to interpolate between those keyframes?
- What are splines, and why is it useful in creating animations?
- What are the types of splines commonly used in computer graphics?
- Why are these splines popular? What properties make them most useful?
- Which physical law do we need for object acceleration simulation? Write the complete formula.

- What is motion capturing?
- What is the difference between forward and inverse kinematics?
- What is the particle system? Name three examples.
- What must be concerned to create an input device for 3D interaction?
- How to perform world navigation in a 3D world? Name an example.
- Name an input device and explain how it applies to object interaction in a 3D world?

3.8 Graphics APIs: `three.js` and `OpenGL`

- When do we need to use const (const) other than variable (let)?
- What are the differences between a standard function and an arrow function?
- How to check the equality of two numbers in a flow control statement?
- What is the class, and what is the required method?
- What are the data types in JavaScript? Name two operations/methods for each of the data types
- How to handle errors in JS?
- What is Node.js, and why do we need it?
- What is npm, and why do we need it?
- What are the required steps to create a scene in three.js?
- Explain these concepts: Scene, Camera, Renderer, Geometry, Material, Mesh, Face, OrbitControl.
- How to express rotation in three.js?
- How to animate objects/camera in three.js?
- How to load a model in three.js?
- How to perform texture mapping in three.js?
- How to connect customized GLSL shaders in three.js?
- Why are GPUs considerably more suitable than CPUs in graphics processing?
- What are the drawbacks of using a graphics API?
- What is the relationship and differences between OpenGL and GLSL?
- What are the stages in the OpenGL graphics pipeline?
- What are the geometric primitives in OpenGL?
- What are the drawing modes in OpenGL?

- What is a shader, and where is it executed?
- What are the programmable shaders in OpenGL?
- What are the inputs and outputs of different shaders in OpenGL?
- How to write an OpenGL shader?
- What are in/out/attribute/uniform/discard?
- What are the required variables in every vertex/fragment shader?
- What is missing in WebGL 1.0 and 2.0 regarding the OpenGL graphics pipeline?

4 Final Words

You have learned a lot in this course: In theory, you learned many topics start from absolute beginner to several state-of-the-art techniques; In programming, you learned how to program and reproduce pictures and scenes that you have seen using `three.js` and `OpenGL` shaders; in writing, you practiced the skill to write Markdown in a plain-text file; ... now you should feel more comfortable with this digital virtual world ;-)

Do not worry about the exam since you have go through a lot difficult topics. The exam will be much easier than what you encountered along the semester, and you will pass the exam if 1) you can answer 90% of questions that is listed in Section 3; 2) you can solve all non-programming tasks in the assignments; 3) you can solve 50% of the programming tasks in the assignments. You will get an excellent grade if you can also complete all programming tasks individually without double checking the solution.

Good luck!