Computer Graphics

View PDF

Instructor(s):

László Szécsi
Attila Kárpáti

Aim of the Course:
This course teaches the principles of computer graphics and enables students to implement modeling, rendering, and animation systems as well as 3D video games in a C++/OpenGL/Cg environment.

Prerequisites:
The course requires fundamental skills in Trigonometry, Mathematical Analysis, Linear Algebra, Programming, and Physics.

Basic trigonometry is used extensively throughout the course.
From Mathematical Analysis, the concepts of single- and multivariate functions, derivatives and integrals are often referred to.
From Linear Algebra, understanding vector-matrix multiplication and the concept of linearity are required. Transformation between different coordinate spaces is a major tool in Computer Graphics, and is explained in detail. Even more prevalent are simple 3D vector operations like dot and cross products.
Some kind of programming background is required: C or Java is sufficient. C++ experience and knowledge of basic object-oriented programming helps.
From Physics, we assume familiarity with standard units of measurement, ratio or density measures, and Newton's laws of motion.

Method of instruction:
The course consists of lectures (2 hours per week), practices (2 hours per week) and 4 home assignments. Parallel to the lectures, students have to develop programs in C++ aiming at the discussed graphics problems. Students are provided with a simple C++/GLUT/OpenGL framework program which should be extended. During practices, parts of these assignments are solved together under the teachers' supervision. The remaining parts should be finalized at home individually or forming small teams.

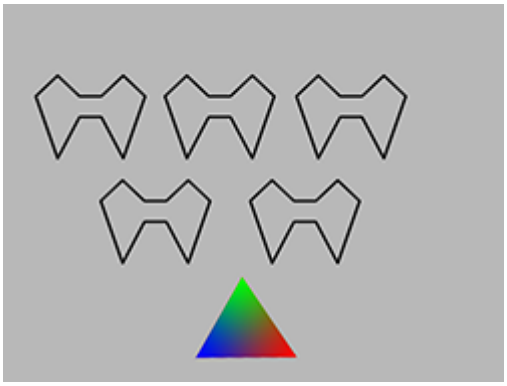
Assessment:
The final assessment is based on the quality of the home assignments (60%), on the solutions of smaller theoretical problems given at lectures (30%) and on the final presentation about the topic of one of the assignments.

Detailed Program and Class Schedule:

1. **Introduction:**
   Modeling, virtual world, and rendering.

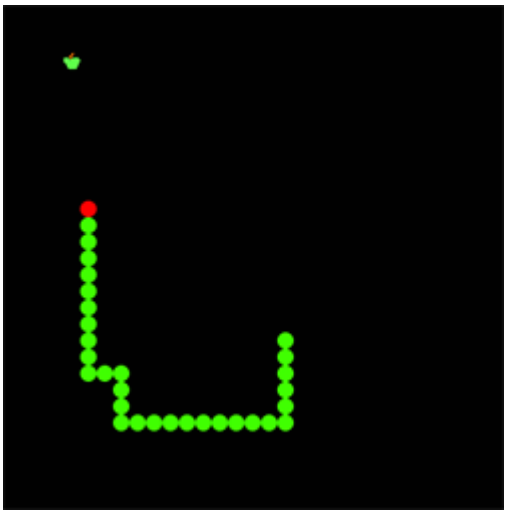
Practical: Development environment setup.
Compilation and linking of a simple OpenGL application.

2. **Fundamentals of graphics software**
   Event driven programming, device independence, graphics libraries. GLUT and OpenGL.



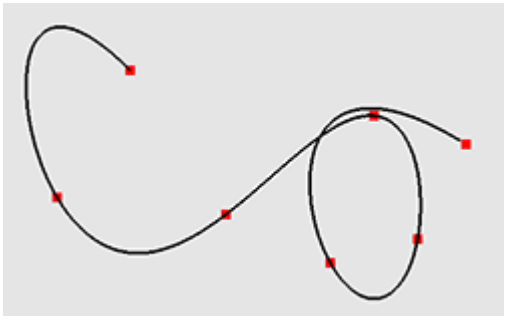
Practical: Drawing triangles and polylines. Moving them with keys and the mouse.

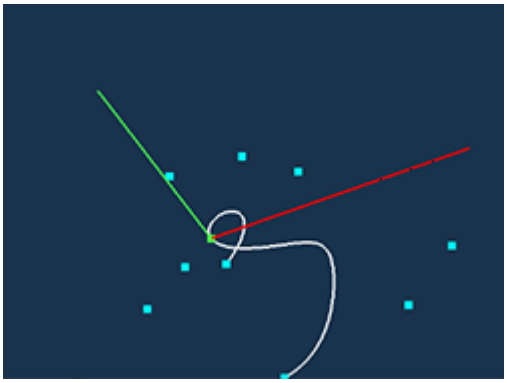
3. **Assignment 1: Simple 2d game in opengl**
   This assignment serves for students to familiarize themselves with the basic drawing operations and the event handling framework of OpenGL and GLUT.



4. **Geometric modeling**
   Coordinate systems. Explicit, implicit and parametric forms. Curves with interpolation and

approximation. Lagrange interpolation, Bezier curves, B-spline, Catmull-Rom spline. Parametric and implicit surfaces. Subdivision surfaces. Solids.



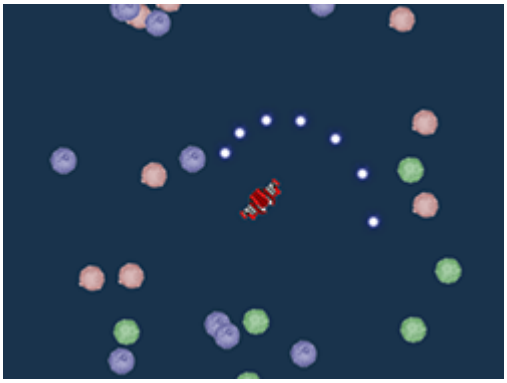


Practical: Drawing various curves, adding/moving control points interactively.

5. **Geometric transformations**
   Homogeneous coordinates. Affine transformations and their matrix representation. Projective geometry.

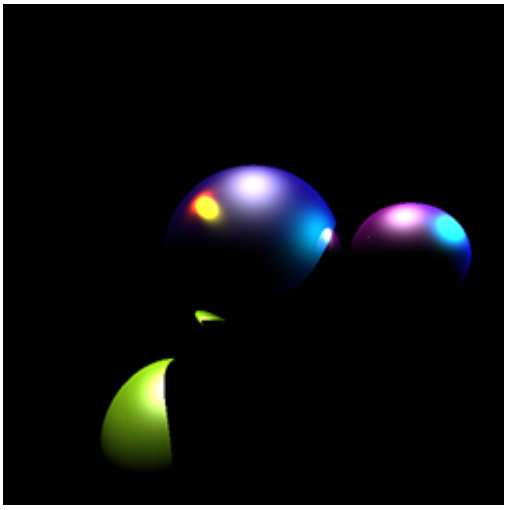
Practical: Rotating and moving objects.

6. **2d image synthesis**
   2D rendering pipeline: vectorization, transformations, clipping, line and polygon rasterization. Software architecture of 2D drawing programs.



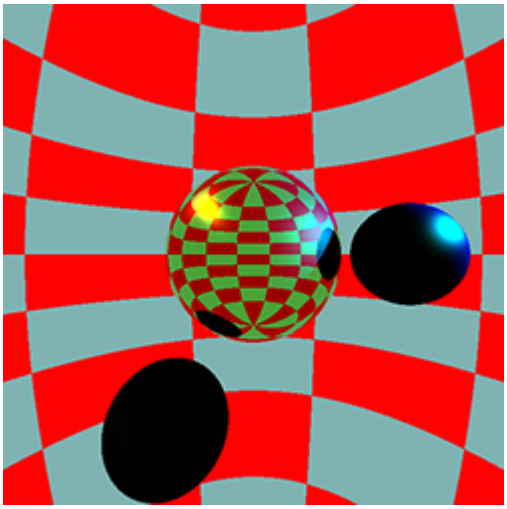
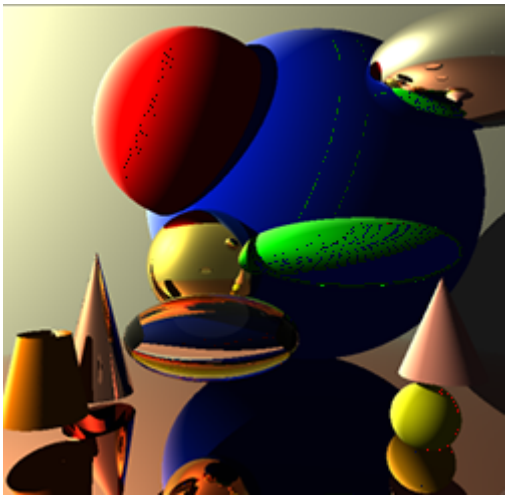
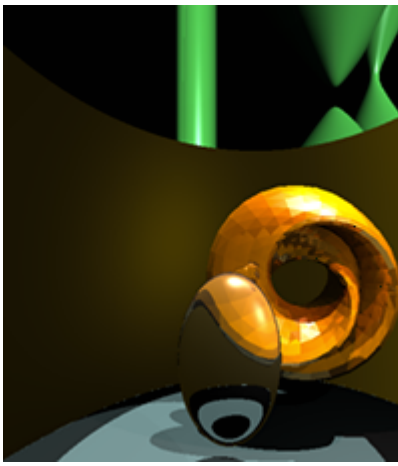
7. **Assignment 2: 2d vector editor**
   Students receive this assignment after practical sessions where they have already implemented drawing Bezier and Lagrange curves, and adding new control points using mouse clicks. The assignment requires them to apply their knowledge to different types of curves (which is not automatic), and to add new editing features, demonstrating their understanding of the event based application model.
   See example for assignment text
8. **3d rendering and optical laws**
   Power and radiance. Laws of light-matter interaction. BRDFs.

9. **Ray tracing**
   Local illumination model and ray casting. Solution of the visibility problem. Recursive ray tracing. Software architecture of ray tracers. Ray tracing acceleration.



Practical: Ray casting with spheres.

10. **Assignment 3: ray tracing program**
    Students receive this assignment after practical sessions where they have already implemented ray tracing spheres with shadows and reflections. They have to implement ray intersection for somewhat more complex geometry, and make creative use of tweaking the ray tracing process for various effects

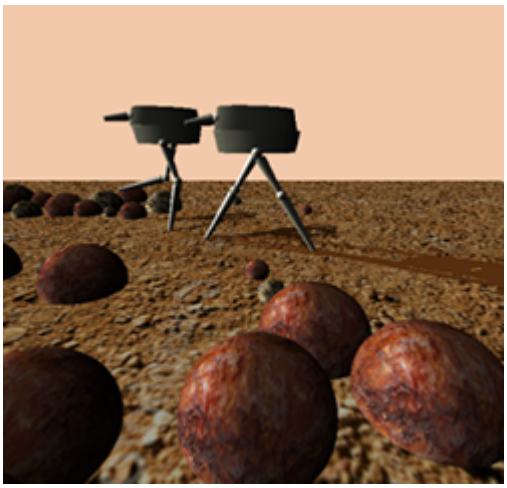




See example for assignment text

11. **Incremental image synthesis**
    3D rendering pipeline: tessellation, transformations, 3D clipping in homogeneous coordinates, visibility determination, shading, and texturing.

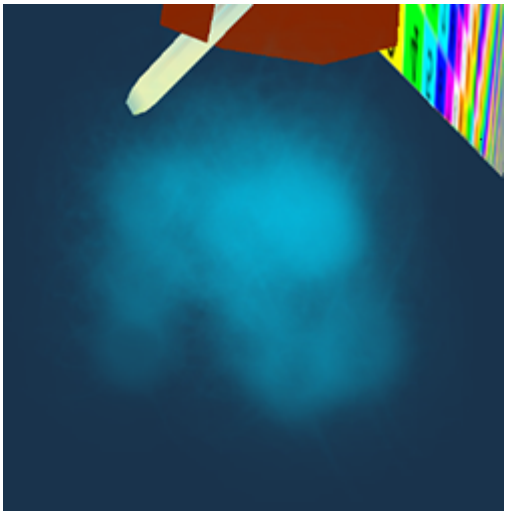
Pratical: 3D graphics with OpenGL

12. **Assignment 3: interactive 3d game**
    Students receive this assignment after practical sessions where they have already implemented rendering and animation of triangle mesh models, and also complete a 2D Asteroids game earlier in the semester. They have to combine the game architecture elements with 3D graphics, with more complex control and animation challenges.



See example for assignment text

13. **Animation, computer games**
    Requirements of realistic animation. Newton’s law. Key-frame animation, path animation, character animation, **dynamics**. Motion capture. Game architecture: virtual world, game loop, user interfaces.

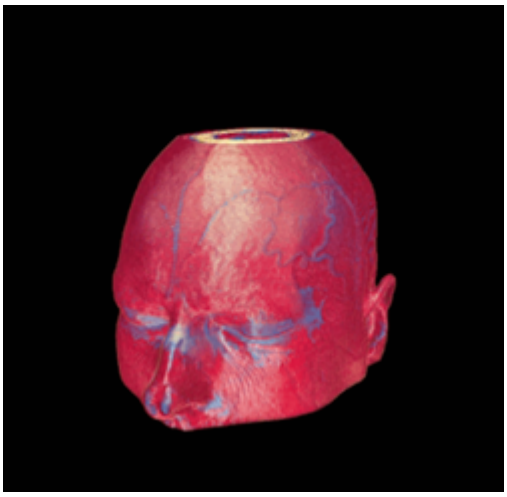


Practical: Particle systems and billboards

14. **GPU** programming
    GPU architecture. **Vertex shader and fragment shader programs.** Examples: Phong shading, shadow mapping, and environment mapping. GPGPU concepts. CUDA.

Practical: Particle systems and billboards

15. **Extra**
    Special topics depending on the student interests, e.g. mobile graphics, medical visualization, scientific visualization, computational aesthetics, game engines, global illumination rendering, etc.



Textbook:
Tomas Akenine-Moller, Eric Haines, Eric Haines: *Real-Time Rendering*, A.K. Peters.
László Szirmay-Kalos: *Computer Graphics* (in *Algorithms of Informatics II* edited by Antal Iványi, Mondat Kiadó, available electronically)

Instructors' bio:

László Szécsi (born 1978) is an associate professor at the Department of Control Engineering and Information Technology at the Budapest University of Technology and Economics. He also lectured Computer Graphics at the János Selye University in Slovakia. He received his Ph.D. in 2010. His research interests include real-time computer graphics, non-photorealistic rendering, computer game programming, and programming of graphics cards both for rendering and for general-purpose computations. He is author of 43 publications, including 1 book, 7 book chapters, 10 journal articles, and 25 conference papers. He won the Bolyai Scholarship of the Hungarian Academy of Sciences in 2012.

Attila Kárpáti (born 1995) is a senior software engineer at Graphisoft SE and PhD researcher at the Department of Control Engineering and Information Technology at the Budapest University of Technology and Economics. He has been teaching computer graphics, GPU programming, and game development since 2017 and published several papers in prestigious international conferences, with subjects including: real-time simulation of interactions between porous bodies and fluids, AI-assisted surface reconstruction of metaball surfaces, controlling characters made of liquid substances, and applications of natural language models in teaching.