Coding assignment
in TIE-52306 Computer Graphics.

General

The requirement of this assignment is to implement a basic ray tracer based on the provided skeleton program. The software must be at least interactive (> 1 frame per second) with a modern laptop.

The work is individual, and you are not allowed to share any code with other students. However, ideas can be shared. Do not copy paste code from online sources. Instead try to understand how the sources work.

The deadline is 27.11. 23:59 and the assignment is returned via email to markku.makitalo@tut.fi. If you implement just features where you only edit the shader file, then it is enough to return only that file in the email. Remember write your name and mark what functionalities you did in the beginning of the file! If you implement your own program remember to include all the files required to run and compile your program and a readme file in the email. If the size of the submission is more than 20MB, upload the 3D models and images to some webpage and include a link in the readme file.

The grading of the assignment is 0-10 based on which functionalities are implemented.

The assignment is implemented on top of the provided single-file shader, assignment.frag. It can be viewed and edited using the provided shader editor, based on WebGL. To open the editor, open index.html. Remember to pick Export -> Download file to save the file after editing. The editor should be able to continue where you left off, but don't rely on it since plugins and clearing browser data may make it unable to do so. In other words, **test how the saving works in the beginning of the first coding session or otherwise you might lose your work.**

The skeleton is provided as a basis, but you are permitted to modify any part of it. Important editing points in the code are marked with comments that end with "!". Some of them are for the required functionalities, some for extra functionalities.

Required functionalities (required for passing the assignment)

- Perspective projection. The assignment skeleton is using orthographic projection. It must be changed into perspective projection.
- Phong shading. The scene is shaded with Phong shading (or if you aim for extra points you can use more complicated shading).
- The camera must move around in the scene. Both movement and rotation are required to pass. The transformations must be controlled either by time or mouse coordinates.

Extra functionalities (required for higher grades than pass)

If required features works at least almost correctly the work is graded 1/10 and it is passed. You get one to three points from each extra feature from this list:

- Attend both visiting lectures (1p)
- More complicated shading
    - Tone mapping (0.5p)
    - PBR shading (1p)

    - Sharp shadows (1p)
    - Soft shadows (2p)
    - Sharp reflections (1p)
    - Glossy reflections (2p)
    - Refraction (1p)
    - Caustics (2p)
    - Textures (1p)
        - Either procedurally generated textures or textures loaded from a file (requires own program)
        - Must be more than just a plain checkerboard, stripes or gradient
- Your own program which shows the shader (preferably JS, C or C++) (1p)
    - Interactivity (1p)
        - Must be more than just mouse coordinates. Simple games are very welcome.
    - Progressive path tracing (2p + parrot mark)
    - Basic post-processing like bloom (1p)
    - Advanced post-processing like denoising (2p)
- Custom distance field objects
    - Simple (1p)
        - Basic shapes or their combinations.
    - Advanced (2p)
        - Fractals or otherwise non-trivial shapes
    - Animated (1p)
        - More than spinning, moving or scaling. Bending is fine.
- Any other advanced rendering technique is implemented. (1p)

Note that the functionalities on the list are examples. You can implement many advanced functionalities outside the list, and not implement some on this list, and still get 10/10. The course staff reserves the right to deduct (part of) the score for poorly implemented functionalities and add extra points for especially impressive functionalities.

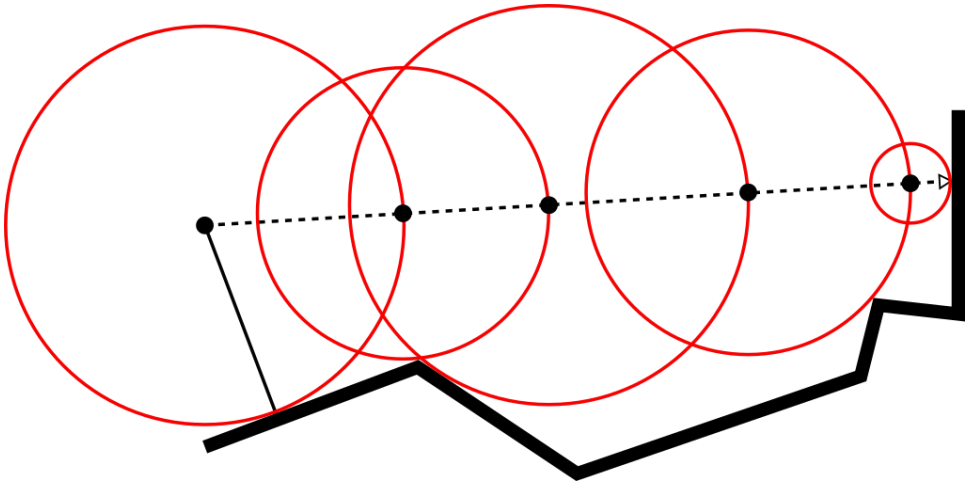
Also note that some of the functionalities are hard to show at the same time. For example, it would be odd if one scene had both hard and soft shadows at the same time. Therefore, you can make some #defines to control which functionality is on. Remember to mention this in the notes of the functionality in the beginning of the shader file.

Remember to mention all the features you implemented in the beginning of the shader file or in the readme file because some of them might be hard to spot.

Background information

The assignment is implemented using a form of ray tracing called ray marching. In it, we step along the ray direction until an intersection with the geometry is found. Technically, the type of ray marching used here is signed distance field ray marching, also known as sphere tracing.

Distance fields define the distance of an arbitrary point to the nearest surface. If the point is inside an object, signed distance fields return negative values. It is possible to find the intersection point of a given ray with the scene by stepping along the ray direction by the current distance field value, re-evaluating the distance field at the new point and repeating this process until the point is close enough to the surface.



The same distance fields allow us to also determine surface normals, and are enough to completely define the scene geometry on their own. This method is a relatively simple way to create high performance ray tracers. Because of this, it is often used in demoscene productions, especially size-limited ones. A complete implementation can fit in under 1 kilobyte.

SDF ray marching is also seeing increasingly many applications in the industry. For example, Unreal Engine 4 can use approximated distance fields for several effects such as soft shadows and ambient occlusion. Both of those effects take advantage of the way distance fields work in order to compute efficient approximations, but the details are left as an exercise to the reader. However, the biggest problem with distance fields is that artists are used to do their work with triangles and automatic tools that generate distance fields from triangle meshes do not generate efficient distance fields.