For the programming task you have to use C++
For questions and help refer to the course's discord server
Or the course's e-mail:
raytracingcourse@chaos.com

Slides: CRT 05 Triangle 02

Task 1.

Generate an image with a resolution of your choice. For each pixel, generate a camera ray (similar to Homework 3) and check if this ray intersects with the triangle:

```
CRTTrinagle tri {
        CRTVector(-1.75, -1.75, -3),
        CRTVector(1.75, -1.75, -3),
        CRTVector(0, 1.75, -3)
};
```

Assume the camera is located at (0, 0, 0) in the coordinate system, directed to "look" towards the -Z direction, with the image plane located 1 unit in front of the camera, i.e., the center of the image plane is at (0, 0, -1).

- *Generate triangles normal vectors: normalize(cross(E0, E1))*
    - *At each pixel:*
        - *Generate camera ray R: 3rd Lecture*
        - *If R is not parallel to the triangle's plane: dot(N, R) != 0*
        - *If R is towards the triangle's plane: dot(V0, N) < 0*
            - *Find R-plane intersection point P: t = rpDist / rProj; p = t * rDir*
                - *Check if P is on the left of E0: dot(N, cross(E0, V0P)) > 0*
                - *Check if P is on the left of E1: dot(N, cross(E1, V1P)) > 0*
                - *Check if P is on the left of E2: dot(N, cross(E2, V2P)) > 0*
                - *If P is on the left of the 3 edges, we have an intersection*

Task 2.

Use another triangle (with different coordinates, your choice for the vertices).

Task 3.

Add a second triangle, and when checking for ray intersections, you must iterate through all the triangles.

Task 4.

Create a simple 3D shape with several triangles like a fan, pyramid, hexagon, etc. When checking for intersections, consider that you must take the triangle closest to the start of the ray!