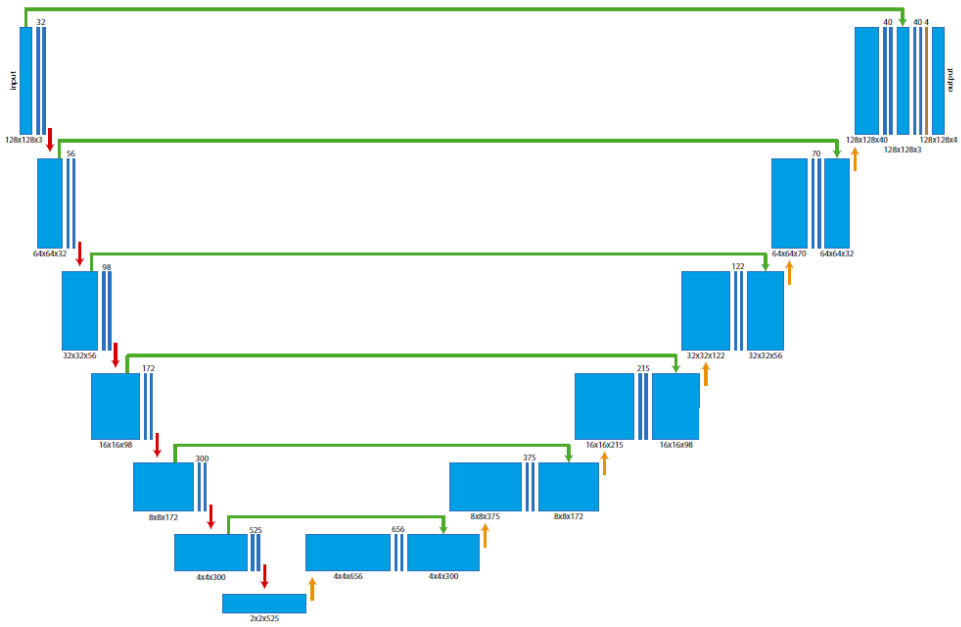
Mikko Ronkainen

Project portfolio

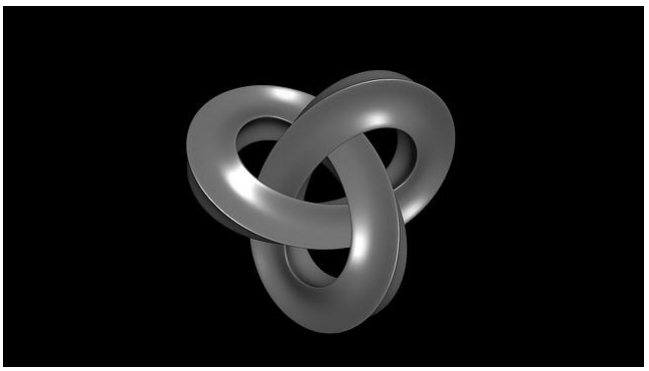
Master's thesis project
Dense tracking of human facial geometry

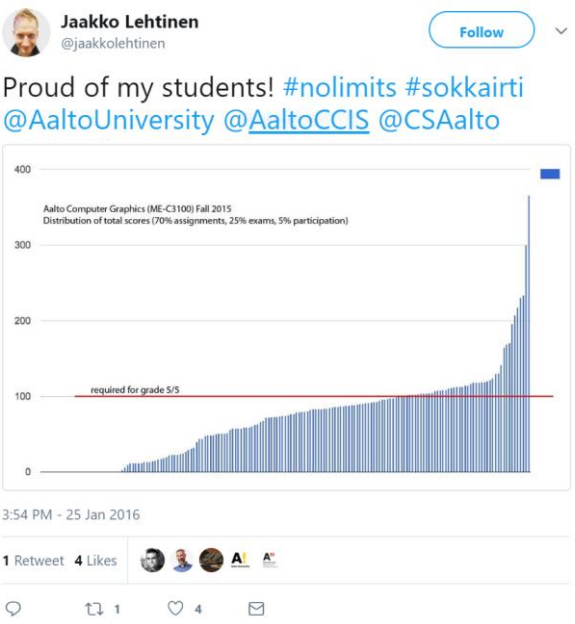


A deep neural network was trained with completely synthetic data to generate UV mappings for real-world human faces. A large number of training images were rendered with Blender. When training, these images were augmented with novel augmentation methods. The network successfully learned to segment real-world human faces from the background, detect if the faces were occluded, and finally generate a useful UV mapping.

- Code: https://github.com/mikoro/master-thesis
- Text: https://github.com/mikoro/master-thesis/raw/master/master-thesis.pdf
- Result video: https://youtu.be/iEvVTav9PN0

Computer graphics course at Aalto

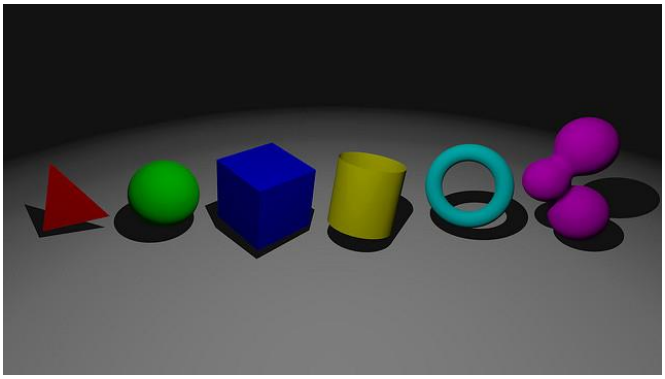


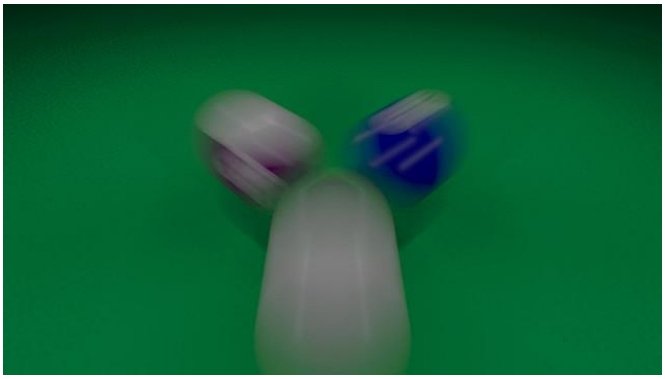


Completed the Computer Graphics course at Aalto with highest total score thus far.

Raycer

C++11/OpenMP/OpenCL raytracer with an interactive preview mode






A fully featured raytracer written in modern C++11 using multithreading and vectorization.

- Code and binaries: https://github.com/mikoro/raycer
- More images: https://www.flickr.com/photos/136293057@N06/albums/72157660998109840
- Feature video: https://youtu.be/0t6a2wXFJkE

- Runs on 64-bit Windows, Mac OS X and Linux
- Interactive preview mode on every scene
- Supports distributed rendering on large-scale computing clusters
- Plus a lot more! https://github.com/mikoro/raycer/blob/master/readme.md#features

Valo

C++11/OpenMP/CUDA physically based renderer with an interactive preview mode



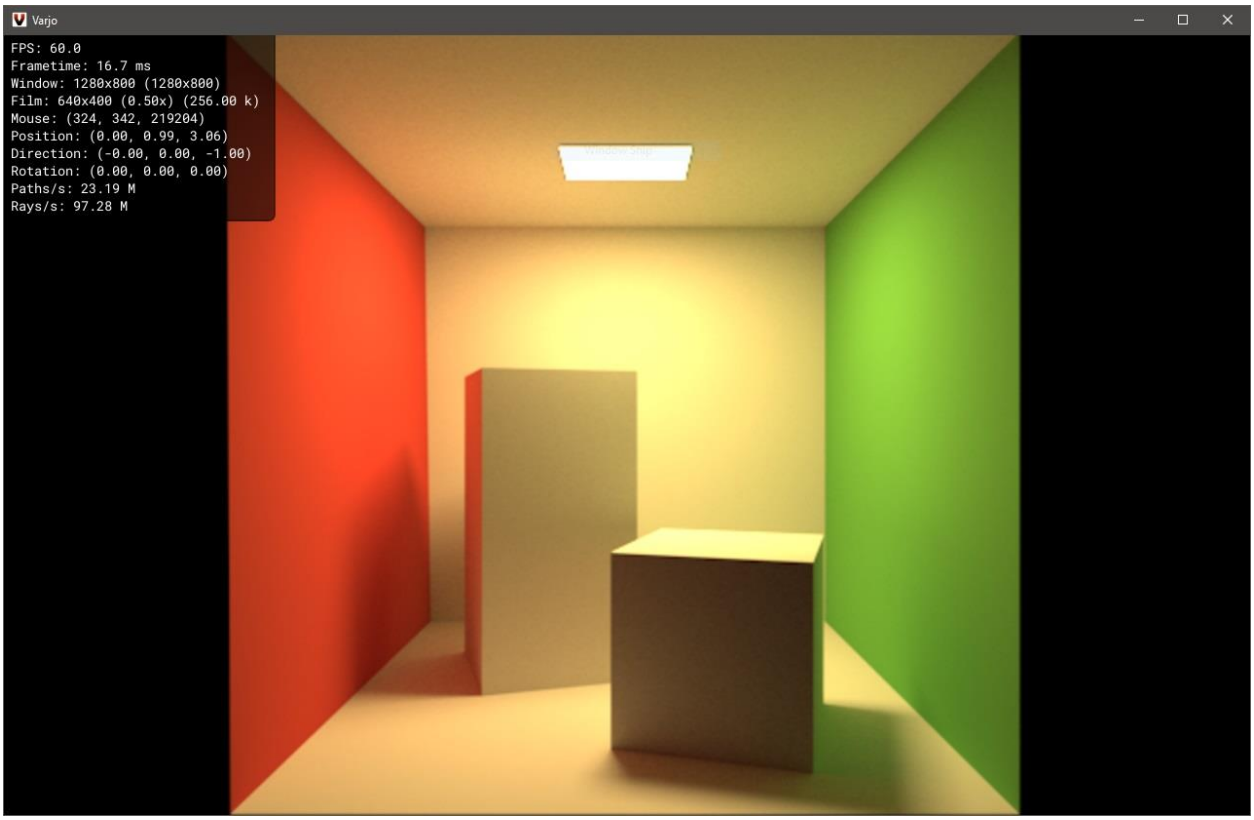

A physically based renderer written in modern C++11. Uses CUDA for rendering on GPUs (based on megakernel approach, not so fast).

- Code and binaries: https://github.com/mikoro/valo
- More images: https://www.flickr.com/photos/136293057@N06/albums/72157665827123423
- Feature video: https://youtu.be/rqemjBD6Lj4

- Runs on 64-bit Windows, Mac OS X and Linux

Varjo

Physically based GPU (CUDA) wavefront path tracer

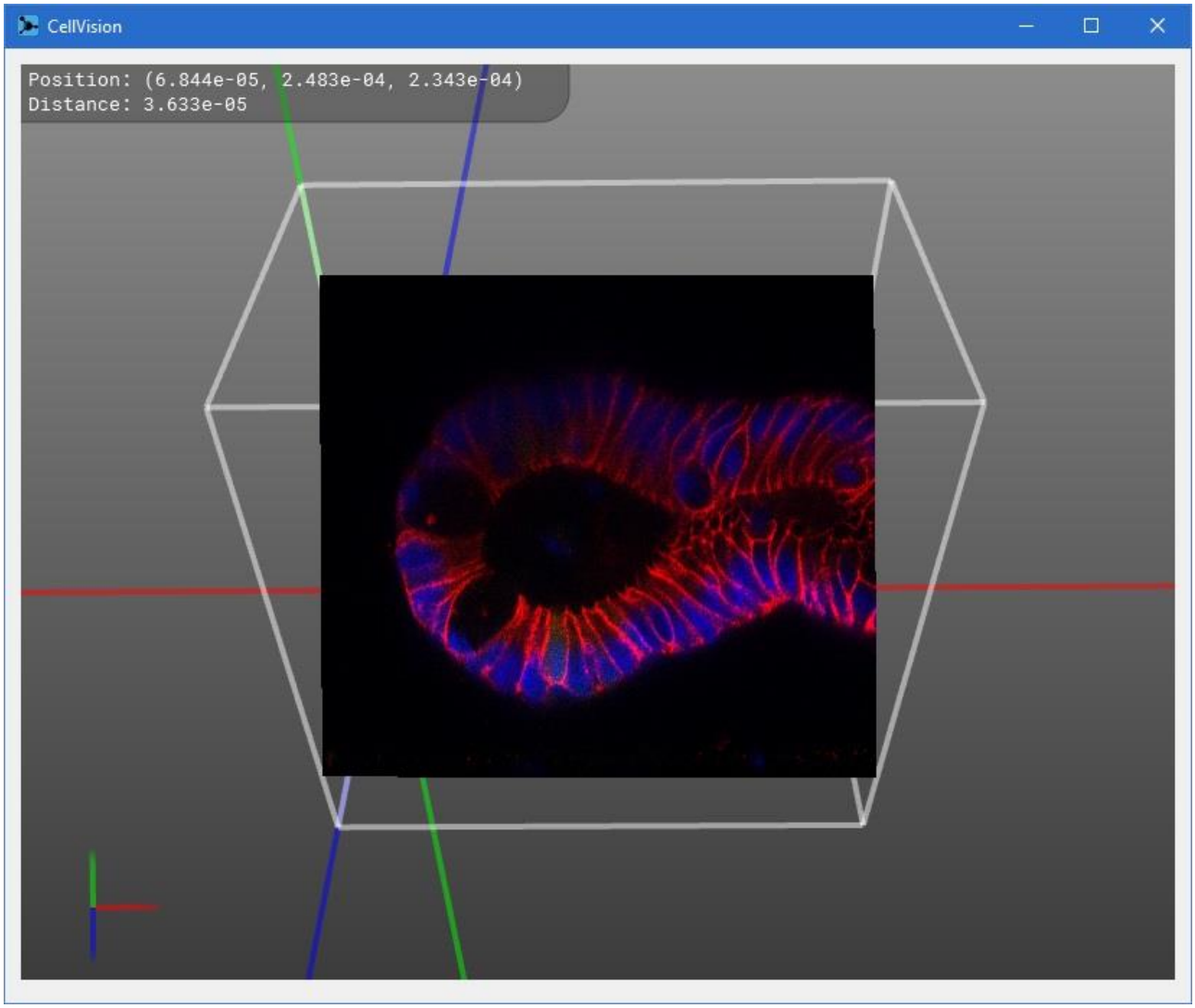


Improvement of the path tracing implementation in Valo. The previous version used the megakernel approach, which was not very performant. The wavefront method improved performance, but not that significantly, as GPUs have become better at handling megakernel type of workloads.

- Code and binaries: https://github.com/mikoro/varjo

CellVision

3D volume image visualizer

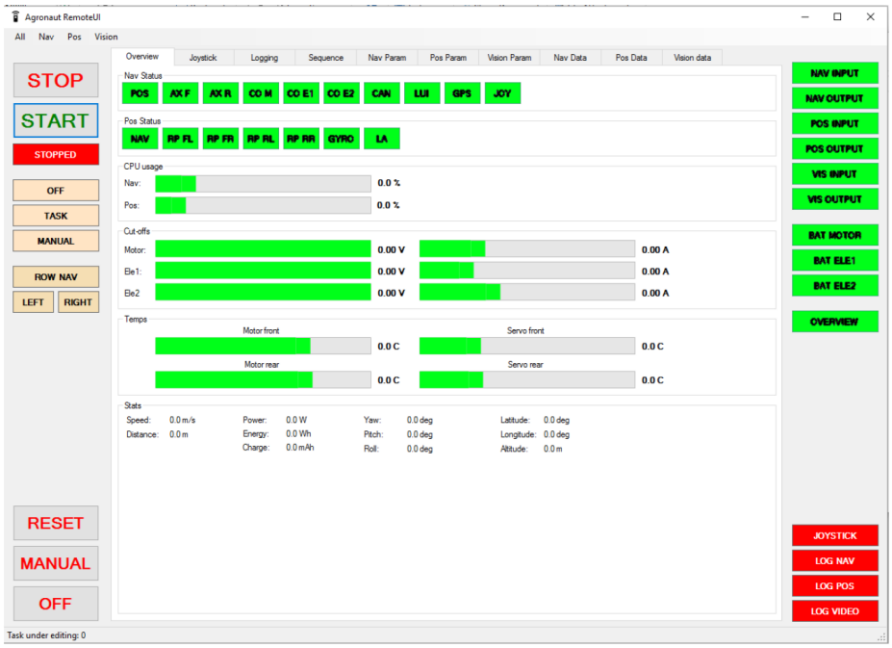


A specialized visualization software for 3D images. Used mainly for visualizing volume scans of different human cells. The cells were imaged by removing very thin slices and taking pictures at every level. The data could then be imported to the program which could colorize the images and enable visualization of the cell from any direction. The visualization works by having a plane fixed to the camera slicing through the 3D volume of the image. The surface of this plane is textured from the volume data by 3D interpolation.

- Code and binaries: https://github.com/mikoro/cellvision
- Runs on Windows and macOS
- Uses Qt for the UI and modern OpenGL for the rendering

Field Robot Project

Medium-scale autonomous outdoor agricultural robot

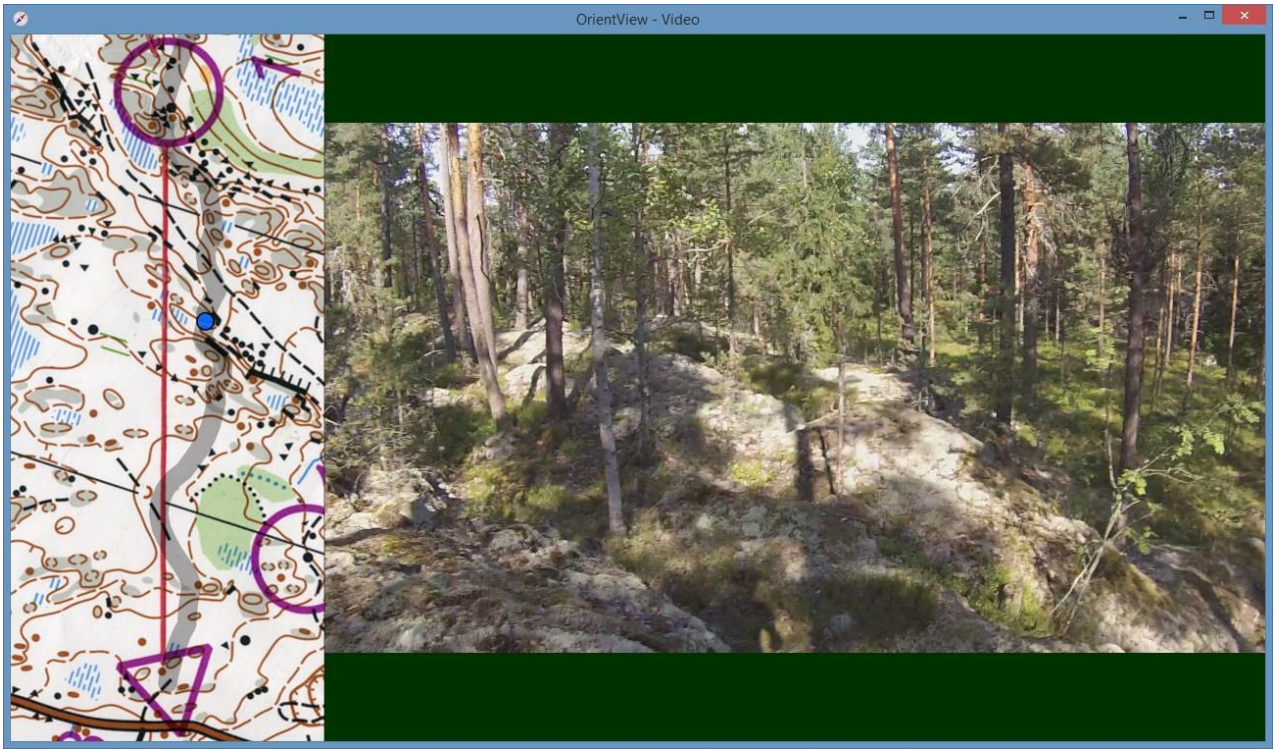


Participated in a year-long autonomous field robot designing and building project. The team consisted of nine students. The robot competed against others at the Field Robot Event in Germany. Some of my responsibilities were:

- Design and implement all the GUI software running outside the robot. One software functioned as a remote control (pictured). One other was used for real-time computer vision algorithm parameter tuning.
- Implement all the glue code between the C++ code generated from Simulink and .NET/C#. Simulink was used to design the algorithms, but the actual main program running on the robot was written in C#.
- Enable remote control of the robot using an Xbox 360 controller.
- Design and and implement the computer vision logic of one task using OpenCV.

OrientView

Real-time analysis of orienteering runs with video and map displays



OrientView is an orienteering video analyzing program which displays the video and the map side-by-side in real-time. Many image and video file formats are supported. Routes are created and calibrated using 3rd-party QuickRoute-software. Resulting video can be exported to an MP4 video file.

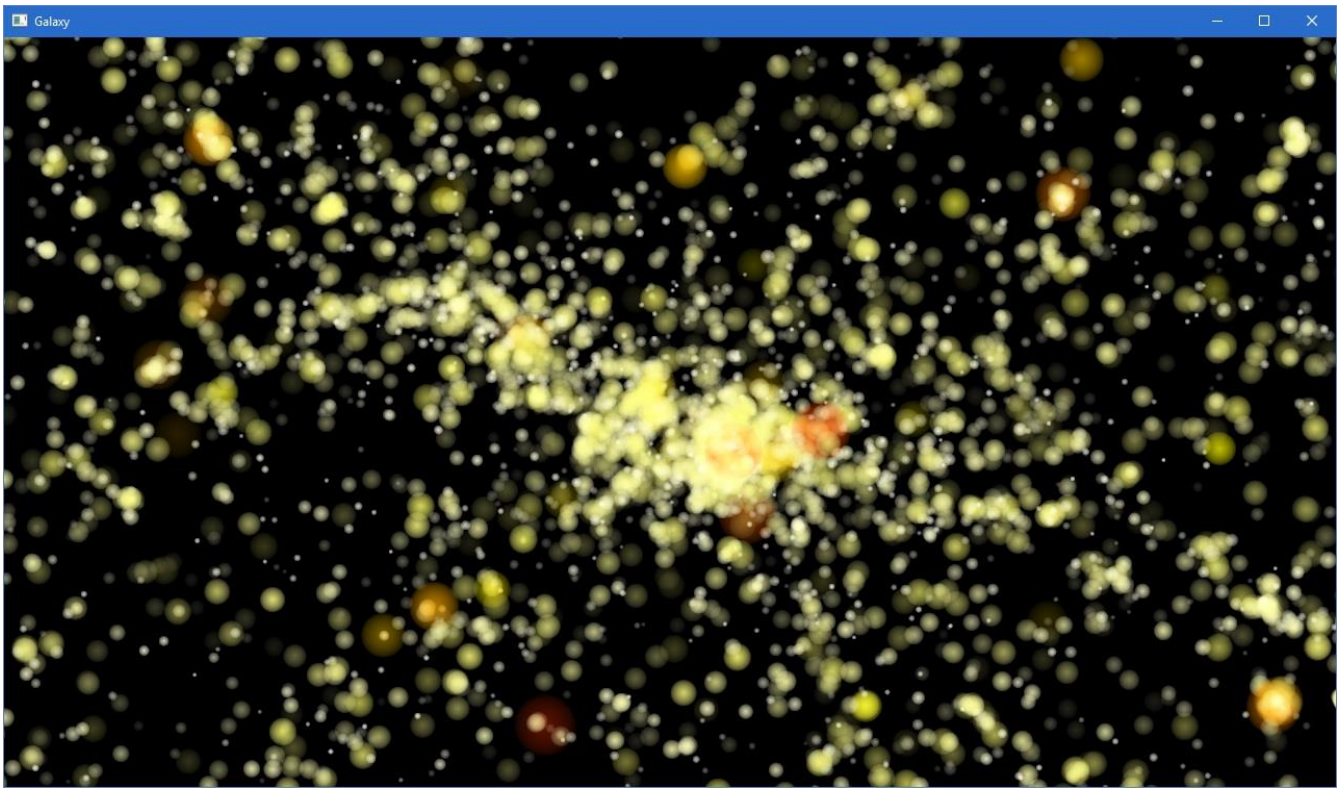
- Code and binaries: https://github.com/mikoro/orientview
- Example video: http://youtu.be/4jh9KmYjdq8
- Runs on Windows, Linux, and macOS

- First, go on an orienteering run and record it on video and GPS.
- Scan the map and use QuickRoute to calibrate the GPS track onto the map.
- Import the map and the video to OrientView and sync them up.
- Press play and analyze your mistakes.

- GUI parts are made using Qt.
- The whole display is rendered through OpenGL.
- Video data is read with FFmpeg.
- Supports real-time video stabilization using OpenCV primitive features. Stabilization can be improved with a two-pass mode.
- Can export the whole playback as H.264 MP4 –video.

Galaxy

Simple OpenGL 2D star N-body gravity simulator

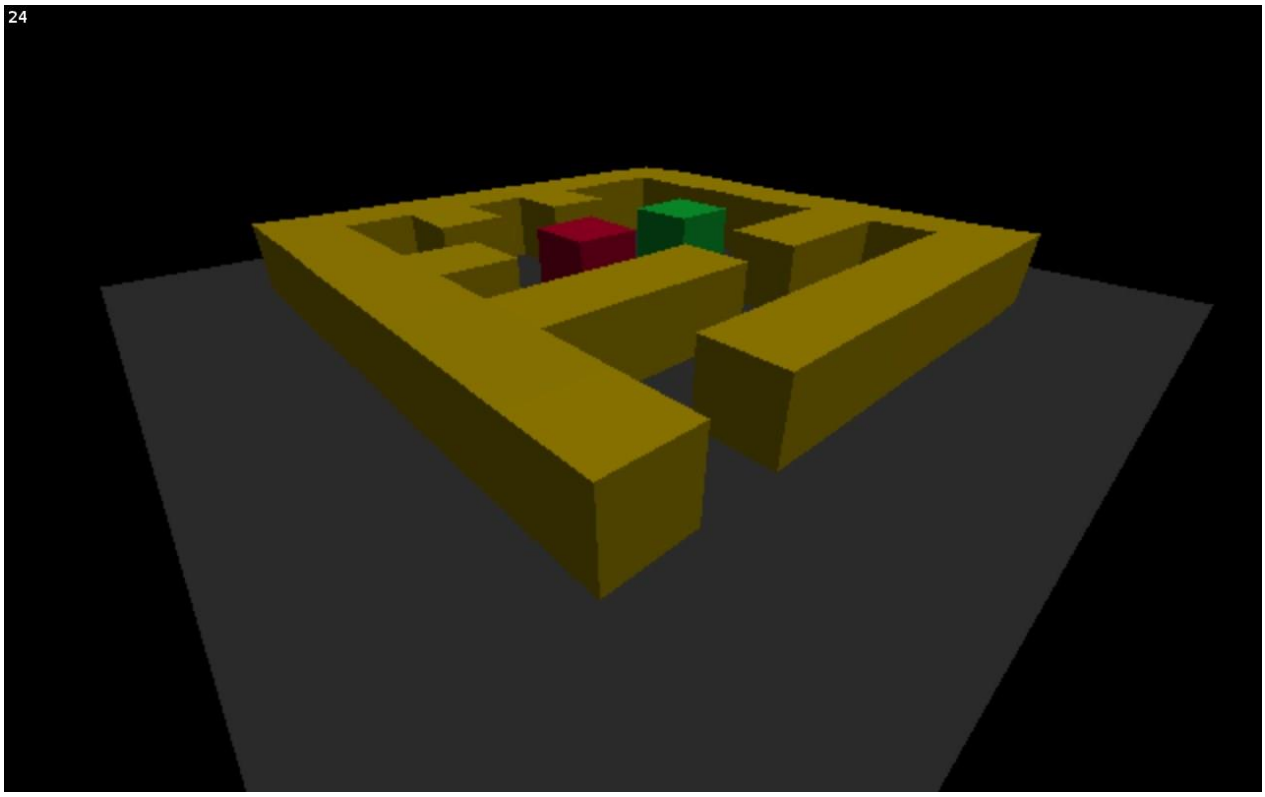


- Code and binaries: https://github.com/mikoro/galaxy
- YouTube video: https://youtu.be/vU7oqISAM1k

- 32768 differently sized and colored stars, all affecting each other.
- All data stored in a single vertex buffer.
- Compute shaders are used in two passes. First pass calculates the forces for each star and the second pass does the velocity/position integration.
- Geometry shader is used for generating quads from the star positions.
- Fragment shader is used for modeling the star shape and color.

Pymazing

3D software rasterizer written in Python



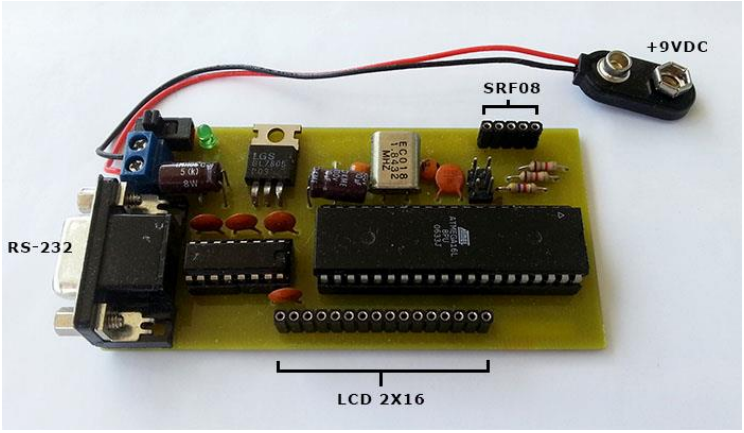
Pymazing is a basic 3D software rasterizer that does all the drawing/rasterization and 3D math in pure Python and Numpy. Levels ("mazes") can be loaded from image files in which each pixel represents a colored cube in the 3D world.

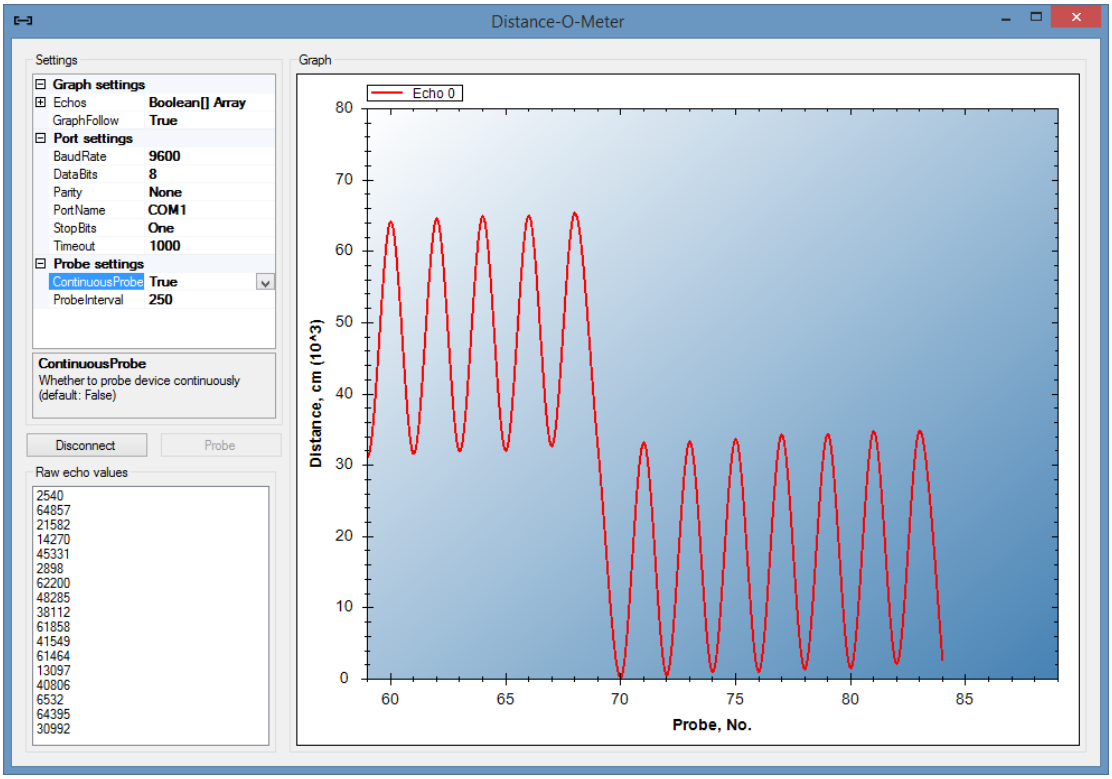
- Code and binaries: https://github.com/mikoro/pymazing
- Example video: http://youtu.be/01qt1wwhz1k

- Runs interactively at about 20 fps. User can fly around freely.
- Implements all the basic algorithms needed for vertex transformations, line/triangle culling, line/triangle rasterization, and lighting.
- The rasterization happens at a smaller scale and the rasterized image is displayed and scaled to the window using OpenGL.

Distance-O-Meter

Ultrasonic range finder + visualization software





- Code: https://github.com/mikoro/distance-o-meter

- Self-designed and manufactured PCB with an AVR microcontroller and an ultrasonic range finder.
- Firmware written in C. Firmware reads the data from the ultrasonic sensor and sends it to the serial port.
- Visualization software written in .NET/C#. Reads the data from the serial port and plots it.

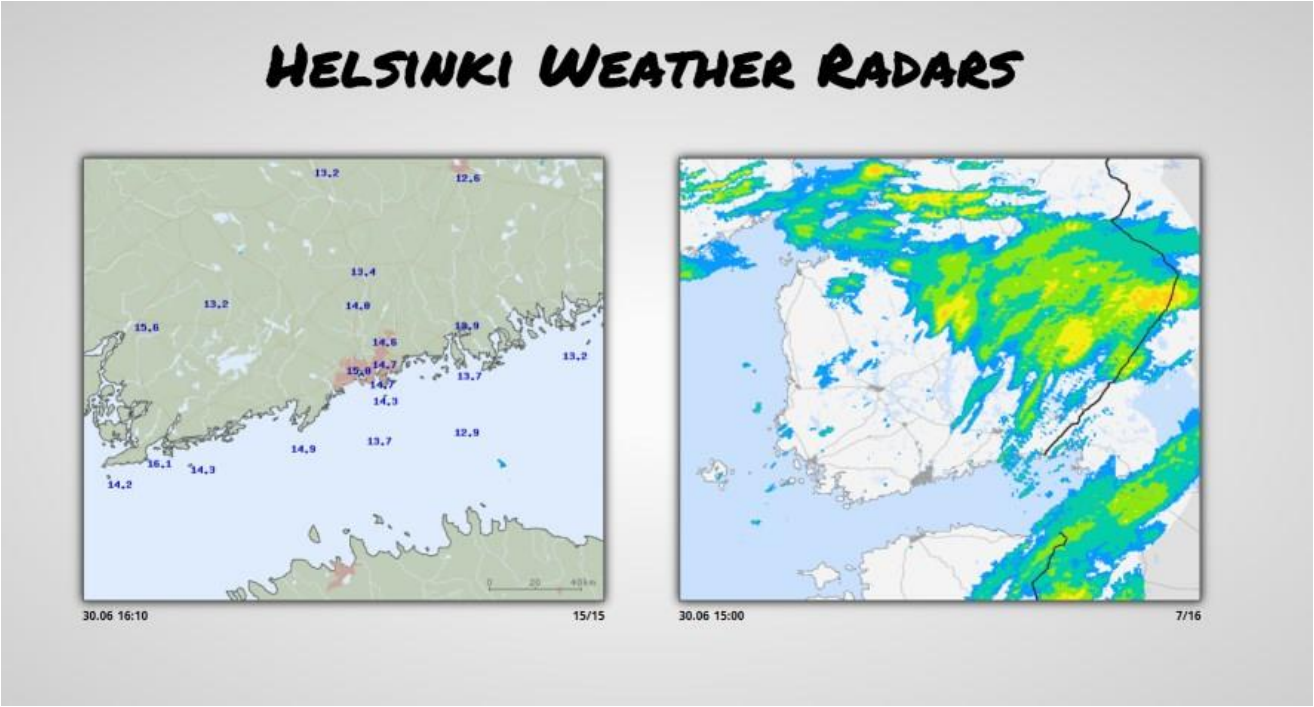
Ramparted

Simple 3D clone of the classic game Rampart

- Code: https://github.com/mikoro/ramparted
- Gameplay video: http://youtu.be/6dfNiyhZ7r8

- A three person project
- Made with C++ and the OGRE3D graphics engine

Helsinki Weather Radars



A simple web page displaying two weather radars of the Helsinki area. Images are loaded from the Finnish Meteorological Institute.

- Code: https://github.com/mikoro/helsinki-weather-radars
- Live site: http://radars.mikkoronkainen.com/
- HTML5 and JavaScript
- Runs on Google App Engine. Data fetching and storing is done with Python in the backend.