Game development / graphics programming portfolio

Roni Koitermaa *

April 17, 2021

Links to my projects can be found on my website `https://roninkoi.net/projects`.

1 Game jam games

These games were created for game jams (game development competitions) in 2015-2019. Most of them are submissions to Ludum Dare, and were developed within 72 hours. Eternal Eski was developed for the AGBIC 2015 game jam in the summer months of 2015.

Figure 1: Eternal Eski (Dart / WebGL).

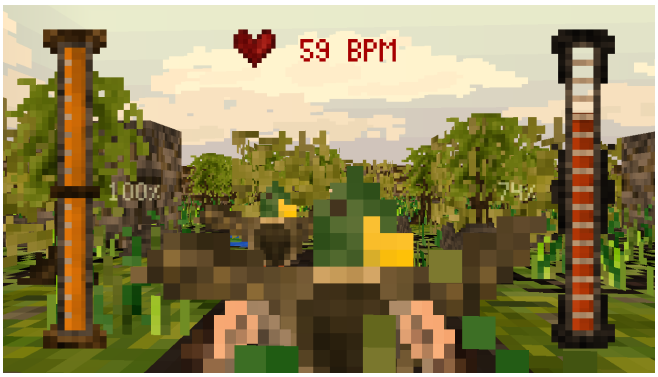

Figure 2: Revenge of the Ducks (Dart / WebGL).

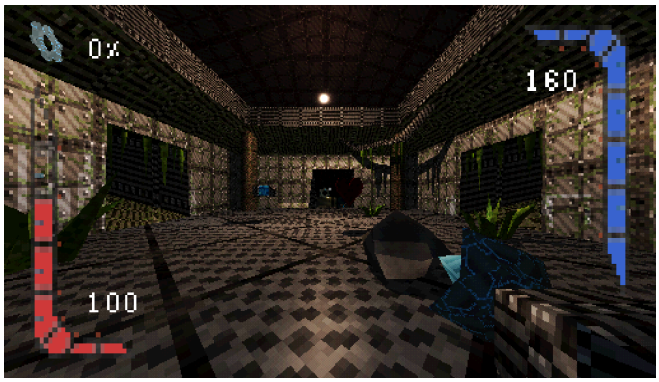

Figure 3: Tech Raid (Corium, C++ / OpenGL).

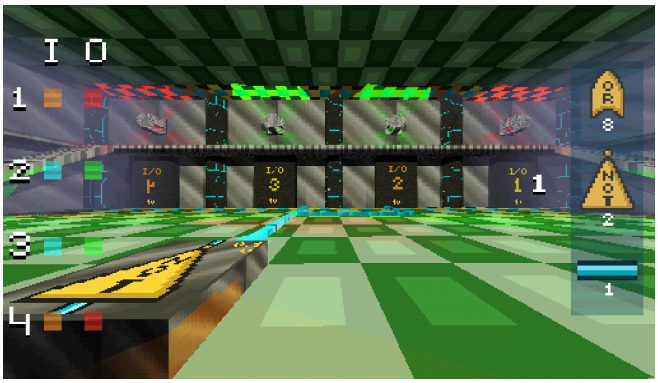

Figure 4: Moore, No! (Corium, C++ / OpenGL).

*roni.koitermaa@iki.fi

Figure 5: Tomb Robber (Go / WebGL).

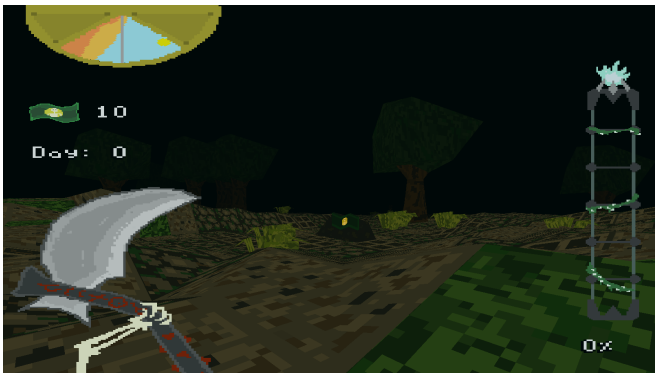

Figure 6: Worked to Death (Go / Vulkan).

2 Game engine development

Much of my development has been focused on making game engines. All of my games use a custom engine. Technologies I've used include SDL, OpenGL, Vulkan and WebGL.

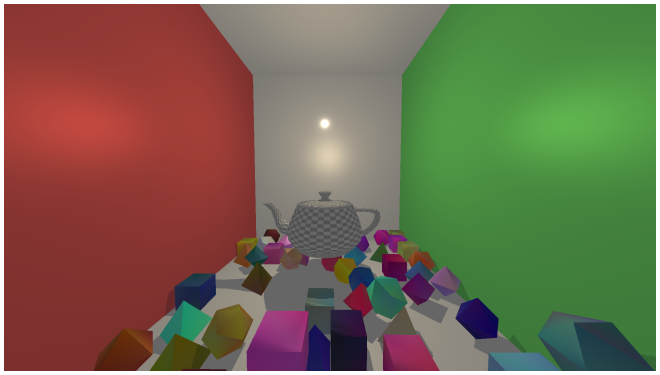

Figure 7: Cornell box with Utah teapot (no effects).

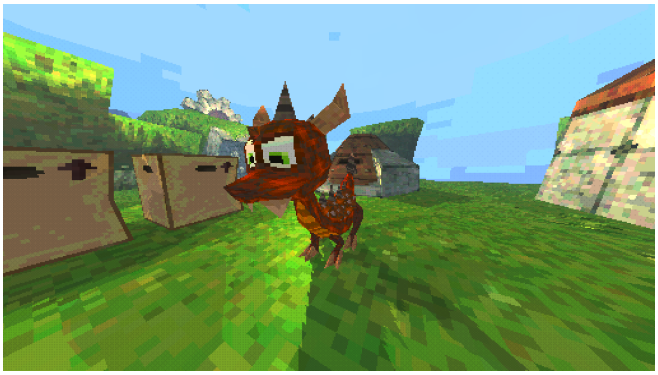

Figure 8: Demo with animated dragon character.

Corium is my personal C++ / OpenGL game engine for making small games. It focuses on a retro aesthetic with the goals of high performance and simplicity. It has simple rigid body physics simulations and animation, as well as its own scripting system. The renderer supports dithering, point lights and shadows (using geometry shaders). Colors are downsampled to create a color banding effect and scenes are rendered to a low resolution texture which is renderered to the screen.

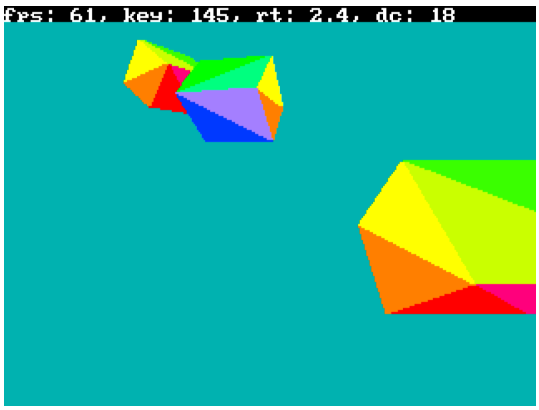

Figure 9: CRENDER drawing 3D cubes in DOSBox.

CRENDER is a software renderer for DOS written in C using mode 13h. The purpose of the project is to study

low-level rendering in x86 assembly. It has support for line draws, triangle fills and 3D projected triangle mesh draws (with sorting and face culling). There is also a keyboard interrupt handler for multi-key input.

3 Shaders

I enjoy shader programming as a hobby because it combines art and code. These GLSL shaders were published on the site ShaderToy. The shaders are run in the browser using WebGL. They can also be viewed on my own site `https://roninkoi.net/projects` using a simple Go / WebGL renderer. Many of the shaders use raymarching to render signed distance fields (SDFs).

Figure 10: Voxel flower.

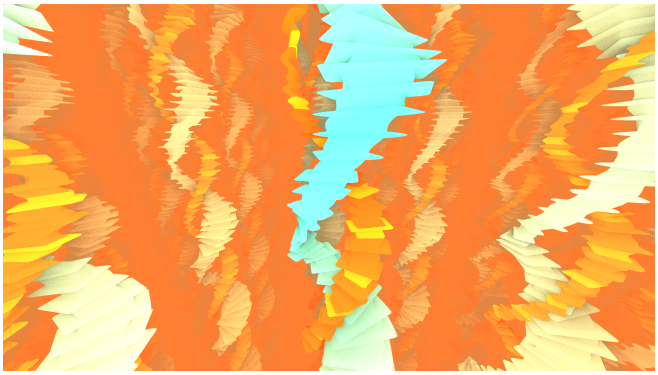

Figure 11: Compressing helices.

Figure 12: Dragon tail.

Figure 13: Mug torus homeomorphism.