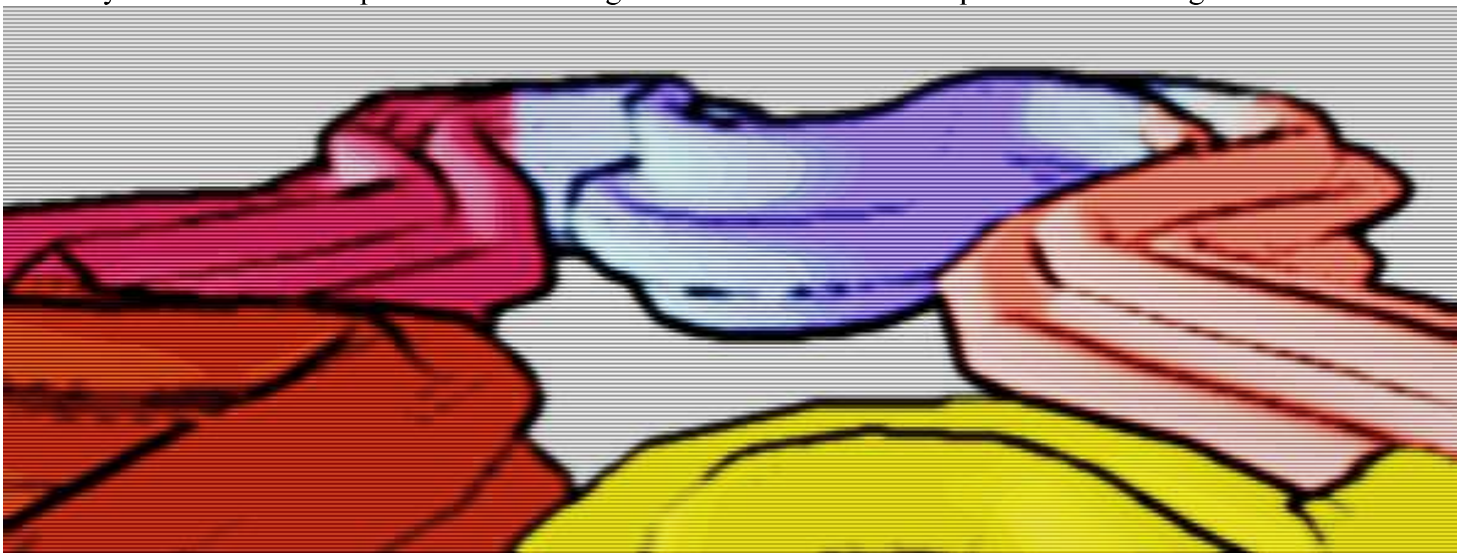


MOTE INSIDE OUT

Written by Mikko Mononen
January 1998
June 2000

Features:

- cartoon rendering
- 3D clipping
- gouraud shading w/ dithering
- multiple omni lights w/ ranges
- keyframing (clax [1])
- .3ds loader (clax)

This is a brief document which tries to describe some of the techniques used in my latest 3D engine, M0TE. M0TE is a realtime 3D engine which speciality is cartoon look-a-like rendering. This engine as well as IMHO most of toonz renderers is based on Philippe Decaudin's [2] work. If you are interested of this topic make sure to visit his homepage. It'll help understanding the following if you read Phillippe's paper forehand.

BASIC IDEA

The problem with cartoon rendering is how to get outlines of objects. Silhuettes are actually very easy to find but bare silhuettes are not enough. As you know

cartoon objects also have lines inside the object on sharp corners and intersections.

Solution to this problem is to find continuity of the surface of the objects. Phillippe Decaudin designed system which uses depth and normal of each pixel to determine continuity of surfaces. Depth buffer is used for finding object boundary detectionthe and normal buffer is used in the process of finding sharp corners of the objects. Depth and normal is stored to separate buffers, edge detectio filter is applied into both buffers and the resulting buffers are combined. Result of this process is nice looking outlines.

This method gives a good result but the drawback is that it's dog slow for realtime purposes. To apply edge detect filter into four separate buffers and rendering five buffers (also the colour buffer count here) is just too much processing for a realtime engine.

SPEEDING IT UP

It is possible to squeeze the information required for each pixels into a single byte. As well the edge detect filter needs optimazing. Of course the result is not as good as with original method but speed needs sacrifices.

DATA PACKING

Squeezing the four values (depth, and noamrl's each component) into one byte is very easy. We just have to bear in mind that we only need the discontinuity information.

In my first tests I found out that you can have quite good result just with Depth value and normal's z component. The two values were scaled to range 8 bit range and interpolated separately. Then I figured out that it's no use interpolating two values. They can be combined into a just one byte. I just scaled them in range 0 and 127 and summed them up. It worked just fine. And compared to the two separate value method only speed was difference.

I used this method in our (M0ppi prod.) demo called Kolme Pientä Pukkia (Three Little Goats). After we had released the demo I found out that there's even better way. Since there are four values (depth value and normal's three components) I scaled them all between 0 and 63 and summed them up. And it the result looked more better! I got rid of many artifacts which appeared in my earlier method. I think the result can be adjust a bit by finding better ranges for each of the four values we sum together.

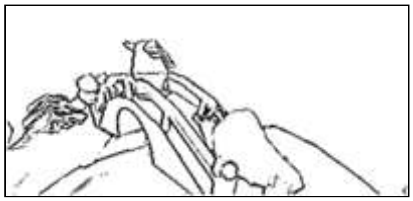
THE FILTER

Optimazing the filter is another case. When I tried to optimize my filter I meet one big problem: compares. In a edge detect filter it is required to clamp the value between certain range because of the nature of the filter. In an edge detect filter you substract values instead of adding them (blur filter). I ended up with just scaling the edge detection ouput and clipping some bits off just to filter the result a bit. A basic edge detection kernel kernel was used:

```
 0  -1   0
-1   4  -1
 0  -1   0
```

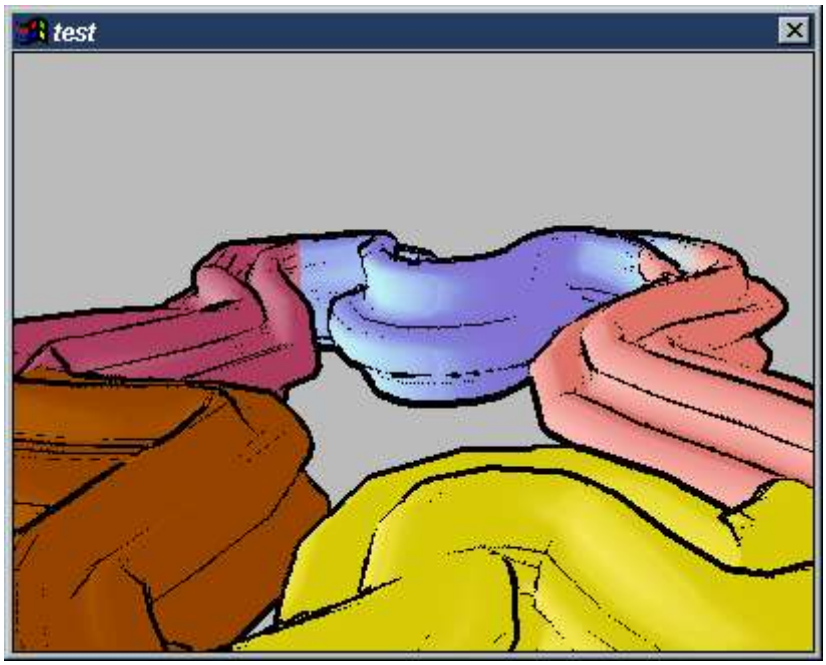
IMAGES

Rendered nz-buffer. Buffer is slightly blurred to get fatter outlines.



Nz-buffer after edge detect filter applied in it.

Final image. Filtered nz-buffer is put on gouraud shaded scene.



A Screen shot of the Windows test application.

DEMOS

Download Kolme Pientä Pukkia demo by M0ppi Productions. The demo was 4th at TheParty'97. Only works in DOS (most like not in DOS-prompt).

- m0p-3pp.zip (1816 kB)

  This is the source of the engine. I have modified it a bit and port it to Windows. The code is badly written and there are no comments, but if you like to have it, here it is.

- M0te Win32 test application (304 kB)

  This is new version of the engine. It uses 16-bit depth/normal buffer for determining the outline.

- M0te 1.1 Win32 test app (254kB)

---

[1] Clax - portable keyframing engine (link broken)
[2] Philippe Decaudin