Sketchy Drawings – A Hardware-Accelerated Approach for Real-Time Non-Photorealistic Rendering

Marc Nienhaus Jürgen Döllner[*]
Hasso-Plattner-Institute at the University of Potsdam

Introduction

In Non-Photorealistic Rendering (NPR), sketchy drawings are essential to visually communicate and illustrate drafts and ideas, for instance, in architectural or product design. However, current hardware-accelerated, real-time rendering techniques do not concentrate on sketchy drawings of arbitrary 3D scene geometries.

We present an image-space rendering technique that uses today's texture mapping and fragment shading hardware to generate sketchy drawings of arbitrary 3D scene geometry in real-time. We stress sketchiness in our drawings by simulating uncertainty. For simulating uncertainty we have to adjust visibility information using depth sprites, which allow us depth testing and 3D scene composition.

Sketchy Drawing

Our sketchy drawings primarily include 1) visually important edges and 2) simple surface-style rendering to convey scene objects.We consider silhouette and crease edges as visually important edges of 3D scene geometry. We obtain these edges by extracting discontinuities in the normal and depth buffer [Decaudin 1996]. The assembly of edges and their constituting intensity values forms a single texture T_{Edge} (Figure a) as described in [Nienhaus and Döllner 2003].

We opt for unlit geometry as simple surface-style representation of 3D scene geometry (Figure b). Therefore, we render designated geometry directly into the texture $T_{Surface}$ using a render-to-texture implementation.

Sketchiness is controlled by the degree of uncertainty, which is applied for rendering edges and surfaces. To simulate uncertainty, we create a screen-aligned quad that fits completely into the viewport of the canvas and texture that quad using the product of T_{Edge} and $T_{Surface}$. Furthermore, we apply an additional texture T_{Noise} whose texture values have been determined by a noise function [Perlin 1985]. T_{Noise} serves as an offset texture when accessing T_{Edge} and $T_{Surface}$, i.e., texture values of T_{Noise} slightly perturb texture coordinates that access T_{Edge} and $T_{Surface}$. To perturb texture coordinates of T_{Edge} and $T_{Surface}$ non-uniformly, we apply two different 2×2-matrices – one shifts perturbed coordinates of T_{Edge} and one shifts perturbed coordinates of $T_{Surface}$. Then, we merge texture values of T_{Edge} and $T_{Surface}$ resulting in a sketchy drawing. Figure a' and b' show intermediate results after perturbing texture coordinates.

Adjusting Visibility Information

When rendering a screen-aligned quad that is textured with the texture of 3D scene geometry, z-values as visibility information of that geometry get lost. Furthermore, visibility information of 3D scene geometry is not available in its periphery when uncertainty has been applied.

To control visibility we use depth sprites. Conceptually, depth sprites are common 2-dimensional sprites that provide an additional depth component at each pixel for depth testing.

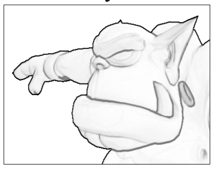
We implement depth sprites using fragment programs [Kilgard 2003]. Initially, we generate a high precision depth texture T_{Depth} derived from 3D scene geometry (Figure c). Then, we render the screen-aligned quad textured with T_{Depth}. Thereby, we replace fragment z-values produced by the rasterizer with texture values received from T_{Depth} using the fragment program.

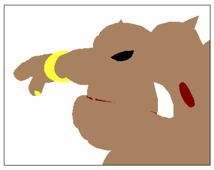
To adjust visibility information of the preceding sketchy drawing we additionally access T_{Depth} twice while applying the same perturbations to its texture coordinates. The first perturbation adopts the offset used for accessing T_{Edge} and the second perturbation adopts the offset used for accessing $T_{Surface}$. The minimum value of both texture accesses produces the final fragment z-value (Figure c').

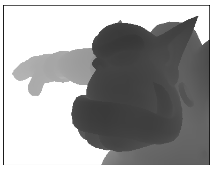
As a result our sketchy drawings include perturbations of visually important edges and simple surface-style rendering, and adjusts visibility information of 3D scene geometry (Figure d).

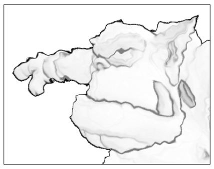
Conclusions and Future Work

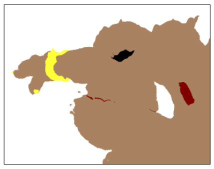
Our approach presents a first sketchy rendering technique that takes fully advantage of graphics hardware fragment programming capabilities and that actually achieves real-time performance. In our future work, we expect to mimic hand-drawn sketches more realistically by considering geometrical properties derived from 3D scene geometry to precisely control uncertainty offsets.

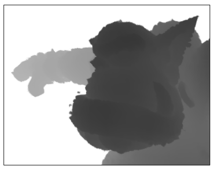

a) Visually important edges

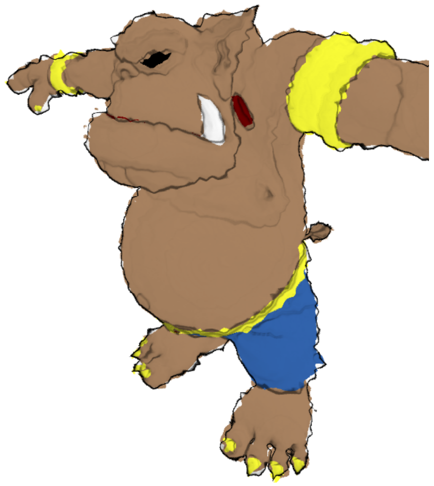

b) Simply shaded geometry


c) Depth values of geometry


a') Uncertainty applied to visually important edges


b') Uncertainty applied to simply shaded geometry


c') Uncertainty applied to depth values


d) The final sketchy drawing of Olaf includes uncertainty applied to edges and surface style.

References

DECAUDING, P. 1996. Cartoon-looking rendering of 3D-Scenes. Technical Report INRIA 2919. Université de Technologie de Compiègne, France.

KILGARD, M. 2003. NVIDIA OpenGL Extension Specifications. NVIDIA Corporation.

NIENHAUS, M., DÖLLNER, J. 2003. Edge-Enhancement – An Algorithm For Real-Time Non-Photorealistic Rendering. In Journal of WSCG'03. 11(2):346-353.

PERLIN, K. 1985. An Image Synthesizer. In Proceedings of ACM SIGGRAPH '85. 19(3):287-296.

[*] {nienhaus,doellner}@hpi.uni-potsdam.de