Master Project - Real Time Rendering of skeletal structures - Motivations Goals and Plan

OLIVIER ROUILLER - S090842
Department of Informatics and Mathematical Modelling
Technical University of Denemark

February 17, 2011

1 Introduction

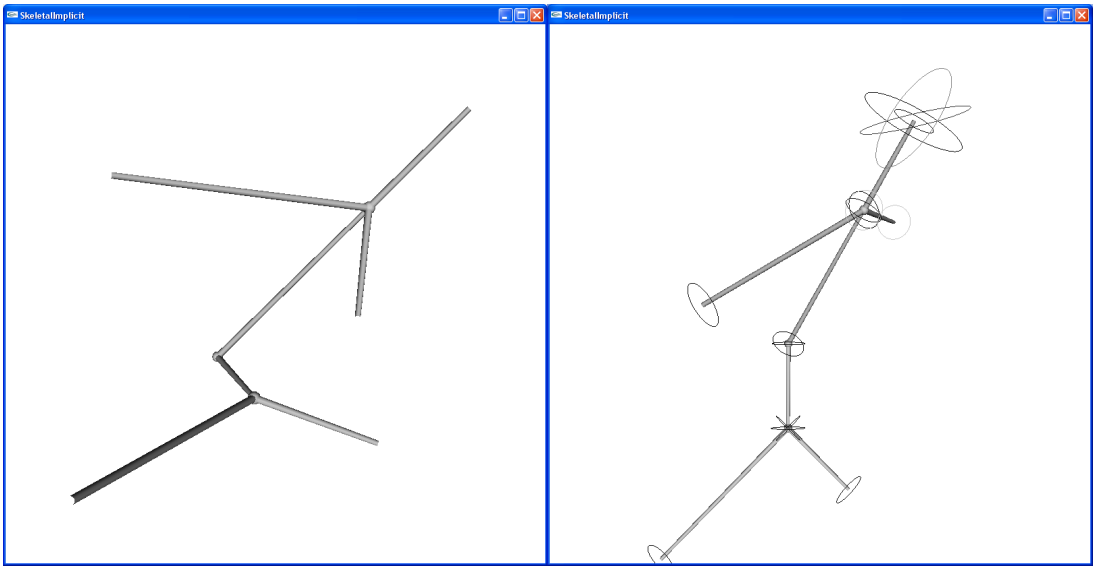
Graphics modelling is usually done at the polygonal level, models are defined by a polygonal meshes, normals, uv maps and textures. On top of this representation, we often create a skeleton for mesh animation. This representation is well suited to the real time rendering pipeline since polygons can be rendered efficiently by the graphics card.

In the other hand, this modelling process is time consuming and alternative ways of modelling are possible. We are indeed interested in modelling with implicit surfaces ans skeletons. Implicit surfaces are iso-level sets of 3d real functions. Implicit surfaces are interesting because they have a very compact representation, because they can be rendered using different techniques and because they can be used to model smooth surfaces easily by performing operations on them.

The most common examples are metaballs , which blend together

2 Skeletal representation of the model - Work done

We use a simple skeletal data structure similar to an skeleton for animation. Bones are defined by an orientation (quaternion), a length and a position. The orientations and positions are hierarchical but we also store and maintain world space informations.



(a) Skeleton (b) Primitives representation

Figure 2.1: Skeleton and primitives representation

Each bone has a list of primitives (metatubes and metaballs for now). These primitives have info on the base geometry (segments or points and radius) in local and world space.

I have implemented support for selection of bones and modification, it shouldn't be too hard to implement dynamic creation of bones and edition of the primitives.

3 Raytracing of the structure - Work done

3.1 Ray tracing

Ray tracing is a rendering technique where for each pixel of an image we try to simulate the path of the light from the scene to the pixel. To do so we throw a ray into the scene, find the intersection of the ray with the geometry of the scene, compute the color of the object at this point and write it to the pixel. To compute the color we can use a simple shading or throw other rays to get a better idea of the light incoming to this point.

3.2 Ray tracing of implicit surfaces

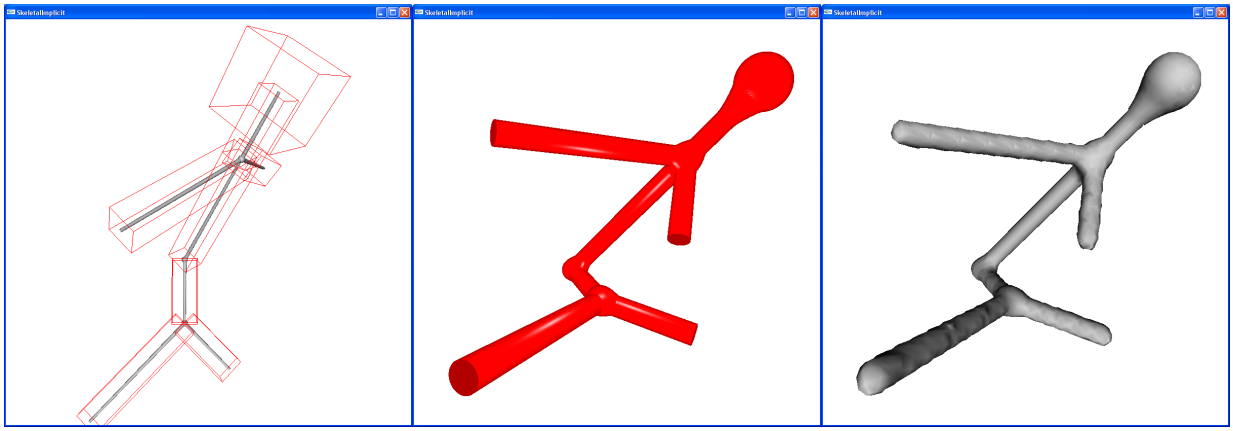
Implicit surfaces can be ray traced because it is easy to find the intersection of the surface with the ray, it boils down to finding the zeros of a function along the ray.

The main issue is to find these zeros efficiently. The way we did it now is the most naive possible, we go through the ray with a constant step length and stop when the sign of the function value changes.

3.3 Basics of GPU raytracing of implicit surfaces

Although ray tracing have been mostly used for offline renderers because of the cost of this technique, with programmable pipelines it is now possible to do ray tracing in real time. The ray tracing is done in the fragment shader. To gain efficiency, we need to throw rays only where we know that they are likely to intersect the surface.

To limit the number of rays shot, bounding meshes for the primitives are drawn Figure3.2a. Info about position and radius of the primitives are sent to the fragment shader in world space. Rays are shot from the bounding mesh in world space to find the isosurface. To find the isosurface, all primitives are evaluated which should be avoided. Primitives are sent to the shader as a list of points and radius. Since there is no dynamic allocation of arrays on shaders?? a large enough array is allocated. The surface is the same as the polygonised surface. But it is a lot faster faster to render it from polygons (2000fps against < 100).



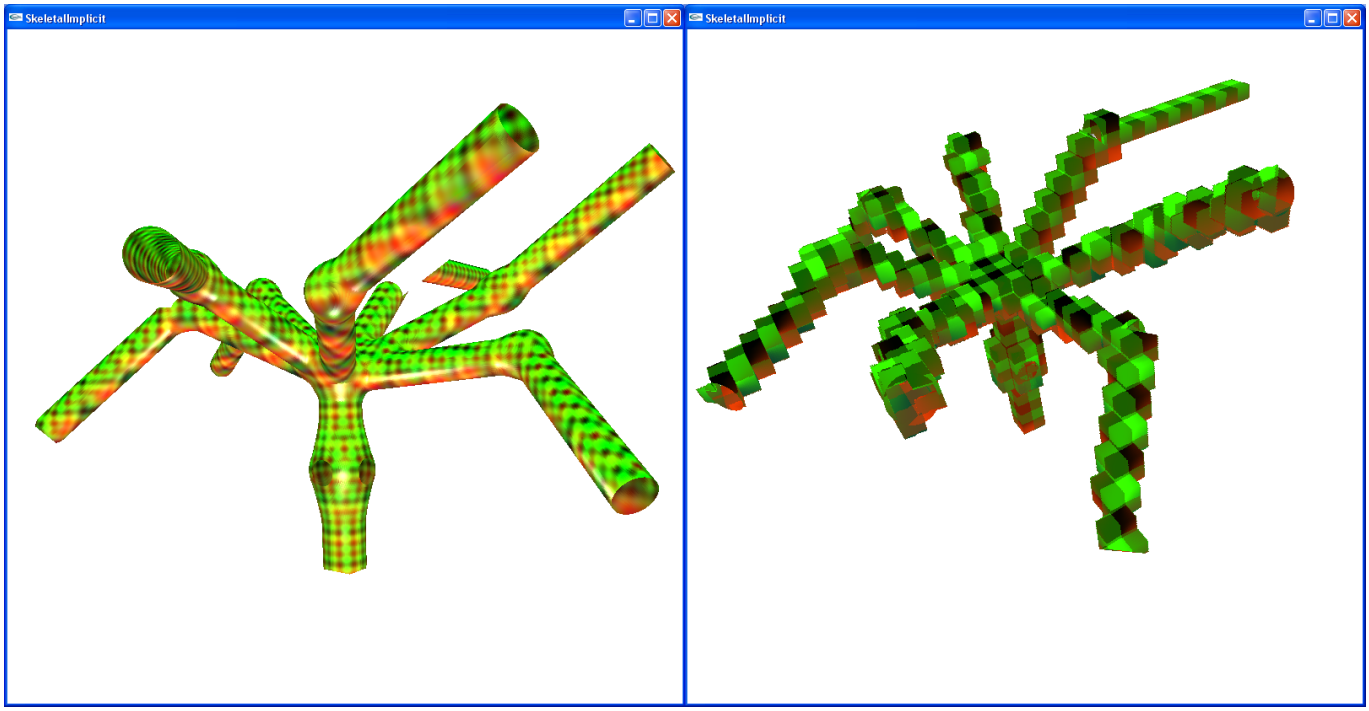
(a) Primitives Bounding meshes (b) Raytraced isosurface (c) Polygonized surface.

Figure 3.1: Skeleton to surface

3.4 By-products

3.5 Remarks

- Blending is good but too much blending is bad. forearm blends with upper arm but not with left foot! It gives me the idea that when ray marching, only the primitives of adjascent bones should be evaluated.



(a) Textured with a 3D noise function (b) Rendered in a voxel style

Figure 3.2: Other skeleton with effects

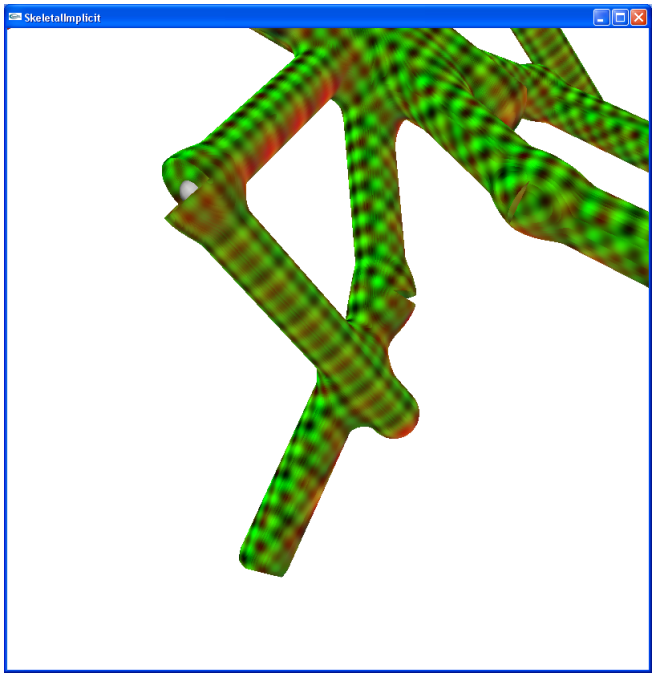


Figure 3.3: wrong blending

4 Optimizing the ray tracing - Reading Notes

4.1 Goal

Design an algorithm to raytrace efficiently a implicit surface defined by a skeleton.

- The algorithm must raytrace the surface efficiently, if possible as fast as rasterizing a mesh.
- The algorithm must get the entire surface.
- The algorithm must allow dynamic skeleton.
- The algorithm must allow as many effect as possible.

How to speed up the ray-tracing?

- Shoot ray only where there is a surface intersection. Involves spatial subdivision techniques such as BVH, Octree, ... Or as we did send bounding primitives to the pipeline.
- Speed up intersection. Choice of root finding algo, using geometry.
- Evaluate only the primitives that contributes at the point of evaluation. spatial sorting.

5 Techniques for root finding

Hart gives a good intro to different techniques in Ray-Tracing implicit surfaces.

5.1 Polynomial Root Solving

Works for algebraic surfaces (in our scope).

First work seems to be from Hanrahan [Han83]. His method used "a symbolic algebra system to automatically derive the equation of intersection between the ray and the surface and then solves this equation using an exact polynomial root finding algorithm". This implies converting the space function of the surface to a univariate function along the ray and then solve this polynomial equation. Having the coefficients of the ray polynomial, numerical methods for root finding are used.

The same approach is used in [LB06] but the univariate ray polynomial is computed using Bezier tetrahedra.

5.2 Interval analysis

A common method to find root of a function along ray is to use interval analysis, isolating the root in an interval then refining this interval to approximate the root. [Mit90].

5.3 Lipschitz methods

Interval analysis can be improved by using the local Lipschitz constant of the function at each iteration.

5.4 Sphere Tracing

Another method to raytrace an implicit surface is to use sphere tracing [Har96]. This method does not involve root finding but finds the first intersection of the ray with the surface. This relies on knowing the distance of the current point on the ray to the surface. If this distance is less than ϵ, we found the intersection, otherwise we know that we can move forward with the distance to the surface. It seems that in the case of convolution surfaces, we know this distance.

6 Acceleration data structure

To address the problems of shooting rays only where the surface is and to evaluate only the primitives that contributes when marching the ray, we require a spatial data structure.

With metaballs or tubes (convolution surfaces) and with a distribution of compact support, we know that the primitives are zero outside a bounding volume (a sphere for a point primitive and a capsule for a tube).

One idea is to maintain on the GPU a bounding volume hierarchy as done [GPP+10]. This structure is used to limit the number of shooted rays and to decide which primitives have an effect.

February				March			
W5	W6	W7	W8	W9	W10	W11	W12
				DADIU PRODUCTION!			
Skeletal structure Surface representation Visualisation by tesselation Draft of skeletal data structure Raytracing of skeletal structure with balls and tubes Raytracing of a single metaball Raytracing of a single metatube	Redisign of the skeletal structure and editor More primitives and convolution	Continue on the structure Brute force ray tracing of the structure	Design of Ray tracing algo	Design of Ray tracing algo	RayTracing optimization	RayTracing optimization	RayTracing optimization

April				May			
W13	W14	W15	W16	W17	W18	W19	W20
RayTracing optimization	RayTracing optimization	Improve shader Texturing Design texturing method	Design texturing method	Texturing	Texturing	Animation of the tree structure Might broke raytracing and texturing	Fix

June				July			
W21	W22	W23	W24	W25	W26	W27	W28
Fix	Integrate in classical pipeline	Shader effects	Shader effects	Shader effects	Shader effects	Writting Polishing	Writting Polishing

August	
W29	W30
Writting Polishing	Writting Polishing

References

[GPP+10] Olivier Gourmel, Anthony Pajot, Mathias Paulin, Loic Barthe, and Pierre Poulin. Fitted bvh for fast raytracing of metaballs. *Computer Graphics Forum*, 29(2):281–288, may 2010.

[Han83] Pat Hanrahan. Ray tracing algebraic surfaces. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '83, pages 83–90, New York, NY, USA, 1983. ACM.

[Har96] John C. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12:527–545, 1996. 10.1007/s003710050084.

[LB06] Charles Loop and Jim Blinn. Real-time gpu rendering of piecewise algebraic surfaces. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 664–670, New York, NY, USA, 2006. ACM.

[Mit90] D. P. Mitchell. Robust ray intersection with interval arithmetic. In *Proceedings on Graphics interface '90*, pages 68–74, Toronto, Ont., Canada, Canada, 1990. Canadian Information Processing Society.