The Future of Curved Surfaces – Summary

Moderator: Jim Van Verth, Red Storm Entertainment/Ubisoft, jimvv@redstorm.com

This roundtable examined the question of when and whether curved surfaces will be used for modeling in computer games. Issues discussed at the sessions included artist tools for curved surfaces, developing artist interest in curved surfaces, hardware support for surfaces, rendering surfaces efficiently and managing collision detection. While session attendance was small over the three days, the roundtable as a whole covered a variety of topics.

The Future of Curved Surfaces – Report

Moderator: Jim Van Verth, Red Storm Entertainment/Ubisoft, jimvv@redstorm.com

Introduction:
This roundtable examined the question of when and whether curved surfaces will be used for modeling in computer games. While curved surfaces have appeared in some games, they are currently not accepted as universal technology, and so the world of gaming is a very triangular place. However, this may change if hardware vendors and console manufacturers finally deliver on the promise of hardware-accelerated curved surfaces. While that day may not be imminent, it will come eventually. As with many other technologies, it can be worthwhile to look ahead. This roundtable covered a variety of issues with respect to curved surfaces in games.

Format:
The roundtable was broken into three sessions. The discussion topics were mostly open, with some additional topics provided by the moderator to keep things moving along. Most attendees were programmers, with experience varying from inexperienced people who wanted to learn, to a people highly experienced in the topic. Most attendees seemed lie in the middle. There were also some software and hardware vendors who were there to gauge interest so they can plan better.

Attendance:
The first and third sessions had approximately 20 attendants, primarily all programmers. The second session had approximately 15 attendants, with mostly programmers and one artist. In each of the first two sessions about 5-7 people were active participants, whereas in the last session nearly everyone participated.

Session 1:

- *Adaptive Subdivision*: After introductions, this was the first topic. Adaptive subdivision converts a surface into triangles (or tessellates), with higher detail in areas of greater curvature. It was discussed whether this was possible to achieve in a vertex shader on standard PC hardware. The issue was raised that the vertex shader only considers one vertex at a time, and neighbor information is needed for tessellation. One pre-processing solution suggested was to use a texture lookup for neighbor information. The biggest problem is avoiding cracking between the areas of varying detail – the decisions along the boundary need to match. A flatness check could be used to pre-determine the detail, but this is not dynamic and is not useful for a level-of-detail system, where you would want low tessellation if the object is far away. Full infinite regression may not be necessary – you need to spend your cycles wisely.

- *Collision Detection*: The question "What are people doing about collision detection?" was asked. The expert response was that it was difficult. One solution with NURBS used binary search and was fudged a lot – trying to eliminate the binary search is key. One possibility raised was using an approximation to collide with – but clearly if there was a fast solution for the surface data that would be better. Parallelizing the problem was also raised, but no one had a definitive solution. It's possible to break the surface into a hierarchy, but then you're talking about a space-time trade-off. Presumably you're using a surface because it is a very compressed format, so adding additional data for collision removes this advantage.
- *Why No Curved Surfaces Yet*: Lots of applications use them (movies, e.g.), why not games? In general there are two issues: artists like to know what they're going to get on-screen, and engineers want to know what's in it for them to implement it. A risk assessment needs to be done to determine whether it's worth taking the time to make the change. Hardware support would help, particularly if there were some bandwidth savings, but what surface type should be supported? Probably this will be driven by Microsoft first and then the hardware manufacturers. Until then, one baby step could be to use them as part of the production pipeline and then convert them to triangles for use in the game. Alternatively it could be used for user-created content: easier to create (maybe) and less bandwidth for network traffic.
- *Creating Curved Surfaces*: This led into a discussion of creating curved surfaces. The main question was: What are the limitations? Are hard edges okay? Do you want to model *anything*? Surfaces allow you to do the same thing with fewer points, but it's not always easier to create it. They are, however, good for facial animation. Another possibility would be to use swept curves to represent a bottle, for example. One attendee made an off-topic note that they tried patches on the Playstation VU, but ended up with cracks; a big problem, as noted before.
- *Direct Rasterization*: The question was raised of whether it was possible to render the surface directly, rather than tessellating into triangles. In this case you are solving a polynomial directly and there can be floating point issues. The general conclusion was that it was not a fruitful approach. Indeed, surfaces themselves may not be fruitful as an in-game approach when you can do so much with polygons and a displacement map.

Session 2:

- *Impact on Art Pipeline*: After a late start and introductions, this was the first topic. The first point was that starting in the middle of production was the wrong approach. You need to decide whether you are trying to use them for real-time in-game data, just for pre-production, or elsewhere in the art pipeline. The motivation for real-time is that they add curvature with high compression – which is also the motivation for ATI's TrueForm N-patches implementation. But this can't just be used for rendering, as the application needs to place all objects correctly so they don't intersect – not just on polygonal surfaces.

- *Convincing Artists to Use Them*: It was posited that what artists want are subdivision surfaces with displacement maps, for characters in particular. However, doing this in real-time is bad; again, cracking is an issue and the computations are expensive. The motivations need to be compelling: it looks better, or triangles are not as efficient when building, or it's better for texture stretching.
- *What would hardware implementation be*? As Cell had been formally introduced the day before, this was a hot topic. The problem is that surfaces are inherently recursive. The Cell architecture might be possibly better than other architectures for this, because you could subdivide on the Cell, and then send the result to the GPU for rendering. Displacement maps in hardware are inherently easier, but for artists, displacement may not be enough.
- *Which types: Bezier, NURBS?* At this point the discussion was winding down. The general conclusion was that bi-cubic patches are best for terrain and not so much for characters. Subdivision surfaces would probably be better for characters.
- *What is the future of curved surfaces*? As one of the primary foci of the discussion was creating curved surfaces for characters, aiming for movie-quality characters is clearly a primary goal. Beyond that there didn't seem to be a driving need; polygons and displacement maps might be enough, particular since the problems with curved surfaces are so great right now.

Friday

- *Current State of Support:* After the introductions, one attendee wanted to know what the current state of curved surfaces was. Rhino is one modeling package that supports NURBS, but it may not have real-time support and the high-end consumers are driving its features. ATI's N-patches were mentioned, as well as the PSP. Curves can also be faked with displacement mapping, which can be created with Z-Brush or 3DMax's Claybrush. In general, curved surfaces are good for organics.
- *Curve fitting:* After this, there was some discussion of whether tools could be created to fit a curve to a high-res model (much as displacement/normal maps modify a low-res model). This could be good for managing silhouettes better, and for LOD. However, cracking still needs to be managed. The technology to convert exists – curve fitting as a science is relatively understood. However, it's not so good for flat areas and hard edges. They have to be adaptive to the underlying continuity. But one argument for curve fitters is that they could be used as a more intuitive method for modeling, as NURBS can be unintuitive.
- *Support:* Ideally GPU support would be needed. Support may exist at the high-end (Boeing), but it needs to trickle down to PCs. It might be possible to handle it with highly programmable shaders. It might also be desirable: If only a fixed hardware implementation were provided it might not be the one you want. One note: GPUs are so programmable these days using the term hardware may be a misnomer. But ultimately the chipset controls what you can do, e.g. CISC vs. RISC. Ray tracing

theoretically makes it easier to render curves, but it really requires a “processor” per-pixel, with a deep pipeline.

- *Collision:* One possible motivation for curved surfaces is using them for collision; one could use tools to create displacement maps for rendering and curves for collision. The one flaw in this is that it’s difficult to manage collision with curves. Possibly you could reuse tessellation from the rendering process, or vice versa. However, extruding NURBS for time-of-impact calculations could get messy. Might be better to think of a swept sphere vs. NURBS terrain – however it’s unclear how difficult this is.
- *Motivation:* Other motivations exist for curve surfaces. One possibility is to use curved surfaces for navigation areas in natural environments instead of navigation meshes. The main values of curves are that they have a compact description and have variable scale (you can zoom in). The compressed format helps one manage bandwidth. Some already discussed disadvantages are collision detection and highly discontinuous objects (hard edges).
- *Convincing Artists:* It comes down to tools. Z-Brush provides a lot of functionality; you can do a lot with normal mapping or displacement. Artists want to use the right tools, whether pre-built or custom construction. One previously mentioned package is Rhino, which is still available. One thing that could help is a new metaphor for building curves, for example, a haptic interface for virtual sculpting.
- *The future:* Curves do provide flexibility in options. Right now triangles are the main force, but other options are coming. However, one conclusion is that they don’t have much of a future. Instead, tools might be directed towards smoothing low-res meshes, and rendering technology directed towards atmospheric effects. NURBS are used in movies because they’re highly scalable, and even then not everyone does that because they want better photo-realism. Hair implementations began as NURBS and are now tending towards polygons (e.g. Maya). The best one might hope for is a mixed world of organic curves and non-organic polygons: trees in cities.