Getting to the Point…?

Markus Gross
Computer Graphics Laboratory
ETH Zürich, Switzerland
To appear in “Graphically Speaking”, IEEE Computer Graphics and Applications

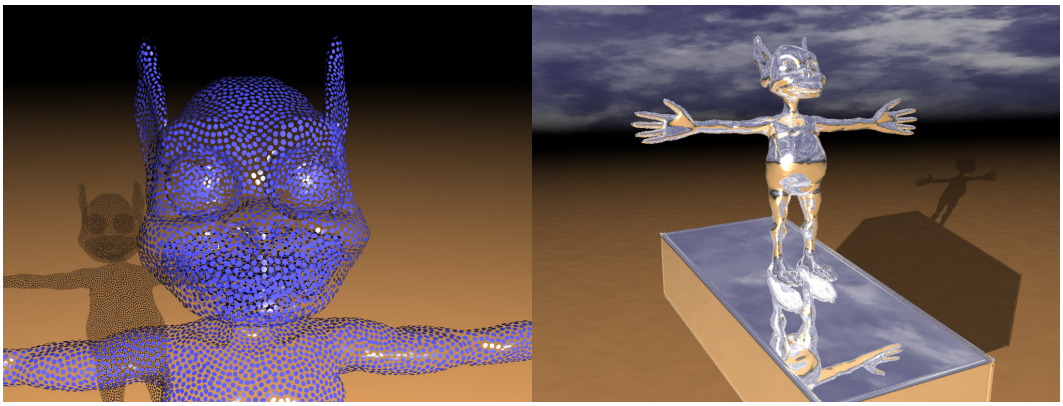

Figure 1: Ray-Traced images of point-sampled objects

In recent years, point primitives have received a growing attention in graphics visualization, and modeling. There are various reasons for this new interest in points: On one hand, we have witnessed a dramatic increase in the polygonal complexity of computer graphics models. The overhead of managing, processing, and manipulating very large polygonal meshes has led researchers to question the future utility of polygons as the fundamental modeling primitive. On the other hand, modern 3D digital photography and 3D scanning systems facilitate the ready acquisition of complex, real-world objects. These techniques generate huge volumes of point samples and create the need for advanced digital processing of points.

In this short article I will share with you my thoughts and experience on the usefulness and versatility of point primitives for graphics, visualization, modeling, and animation. My observations and findings are based on almost 10 years of research experience in point based graphics, from 1997-2006, and many discussions I have had with key researchers in this rapidly developing field. The article is intended to provide both a critical analysis and an inspiration for researchers and developers.

History and Significance

Historically, points have not been utilized very much in computer graphics. In spite of a few exceptions, such as Reeves’ particle systems, Levoy’s and

Whitted's 1985 report probably constitutes the first serious attempt to employ point primitives for graphics display and thus marks the beginning of point-based graphics. Their actually revolutionary concept, however, was abandoned by the authors soon after and it was only in the late 90ies that points regained interest in graphics. Of course, related disciplines, such as computational geometry, have investigated point clouds constantly over the years, mainly for the purpose of (triangular) surface reconstruction.

In order to better understand the significance and potential of points as a fundamental primitive it is instructive to compare them to two other, very popular representations: *Triangle meshes* and *voxel grids*. While the former one stands for "conventional" computer graphics and for geometry processing, the latter one serves as an icon for scientific visualization. For my comparisons I will focus on three main subfields of graphics: rendering, modeling, and physically-based animation.

The Triangle – Queen of Graphics Primitives

Let me start with *triangles*. Triangle meshes consist of reasonably simple, piecewise linear primitives whose efficient rendering is supported in modern graphics hardware. A closer look onto triangle *rendering* reveals one of the conceptually most central features of this representation and, more generally, of conventional graphics pipelines: the separation of *geometry* and *appearance*. Most contemporary graphics architectures distinguish properly between vertex and fragment processing and assume that model appearance is encoded and conveyed separately by means of texture maps. The recombination of both paths occurs dynamically in the rasterization and fragment processing stages. This requires both high internal texture memory bandwidth and processing power to alleviate aliasing.

Regarding *modeling*, a lot of progress has been made over the past years in digital geometry processing of triangle meshes. When stored in a proper datastructure, triangles encode connectivity explicitly and statically, that is, through edge/vertex/face-adjacency relations. Constant-time access to neighbored vertices in combination with adequately discretized differential operators makes meshes well suited for a variety of modeling and surface processing operation. Large deformations, however, can produce long, thin triangle slivers and create the need for dynamic mesh restructuring – an often costly operation. In less sophisticated representations, for instance in "soups", triangle primitives are mostly unrelated so that individual vertices are transferred multiple times to the graphics subsystem. The hardcoded connectivity of a mesh also makes level-of-detail management complicated and non-trivial. Likewise, progressive compression and transmission of meshes becomes a rather difficult endeavor. The vast body of literature in this area serves as evidence.

A further important subfield of graphics has been physically-based *animation*. In particular, the computational power of the most recent generation of entertainment consoles will bring realtime physics effects into the realm of computer games. Traditionally, the simulation of material or fluid effects is based on finite element modeling utilizing regularly or irregularly sampled grids and meshes. While such methods do have advantages regarding numerical robustness, realtime performance is mostly still some way ahead. Furthermore, some effects, such as fracture, change topology and depend on mesh restructuring which quickly becomes a serious bottleneck for hardware-acceleration. For this reason, researchers have been longing for meshless simulation where interactions between samples are maintained through dynamic connectivity graphs.

The Voxel – A Success Story in Visualization

Voxel representations, on the other hand, are likewise simple samples of a volumetric function and live on a regular grid. The grid structure is often considered an advantage, because it is adapted to the (medical) data acquisition process and, more importantly, implies implicit connectivity. In addition, voxel grids allow for a straightforward application of advanced signal processing algorithms, such as FFT, filtering, enhancement, diffusion, compression and so forth. A wide spectrum of methods has been developed over the years for the efficient processing and display of volume primitives.

Volume *rendering* is probably one of the most important achievements of scientific visualization. A vast body of literature, many efficient algorithms and flagship applications, like the medical field, confirm this observation. High-quality methods have been accelerated in hardware and dedicated volume rendering architectures have been proposed. Again, the regular structure of the representation supports efficient computations. In this context, it is also important to distinguish between volumetric rendering and isosurface reconstruction/display.

The capability of voxels to *model* complex geometry, however, is somewhat sobering. While much progress has been made in the field of implicit surface representations, the obvious limitation of volume primitives is their difficulty of representing high resolution surfaces including surface microgeometry. Not only makes the implicit nature of the representation interactive modeling difficult, but increasing geometric detail requires excessive upsampling of the underlying volumetric representation. In addition, volume deformations require adaptations of the grid structure and resolution. In general, the overhead of volume samples for large datasets and high resolutions sets bounadries to their practical use. Hierarchical extensions can only help to a limited extend.

Regular, volumetric grids have been the bread and butter of physically-based *animation* and are often referred to as the Eulerian approach to PDEs. The

domain is discretized by regular sampling, so are the differential operators. Yet, for many problems, the regularity of the representation is too rigid and imposes unnecessary constraints. Instead, more flexible structures, such as tetrahedral grids have been proposed and utilized in the numerical modeling community.

The Point – A Lingua Franca?

Points are clearly the simplest graphics primitive. In some sense, they generalize pixels and voxels towards irregular samples of geometry and appearance. The conceptually most significant difference to triangles is that points – much as voxels or pixels – carry all attributes needed for processing and rendering. As such, there is no distinction between vertex and fragment anymore. As a sample of geometry and appearance, points, in their purest form, do not store any connectivity or topology. Since many 3D acquisition algorithms generate point clouds as an output, points naturally serve as a canonical representation for 3D acquisition systems, whereas, for instance, triangle meshes are the result of a 3D reconstruction algorithm and require prior assumptions on topology and sampling. The lack of topology and connectivity, however, is strength and weakness at the same time. The atomic nature of a point sample gives the representation a built-in level-of-detail making it possible, for instance, to stream and render pointclouds progressively.

Point *rendering* is the most traditional part of point graphics. As a sampled representation including geometry and (prefiltered) appearance point representations allow one to carry over some of the computationally expensive fragment processing, such as filtering, to the preprocessing stage. Their very “sameness” of geometry and appearance creates the potential of designing leaner graphics pipelines. Of course, this simplified processing comes at a price. Straightforward framebuffer projection leaves holes in the representation which have to be filled for close-up views. Point models also require a denser sampling compared to triangle meshes. The higher resolution of the representation potentially leads to increased bandwidth requirements between CPU and GPU. In some sense, bandwidth has to be traded with processing speed. We have investigated these issues in greater detail and have designed a prototype hardware architecture for point based rendering at ETH Zürich. ASIC and FPGA implementations serve to analyze its performance (see Figure 2).

Besides object space oriented rendering methods, some authors have suggested high-quality ray tracing of point models. Figure 1 gives an example.

Points have proven their ability to *model* complex geometry. Their lack of connectivity enables one to conveniently resample without the need to restructure the representation on the fly. Resampling, one of the key ingredients of many point graphics algorithms can be accomplished in many different ways and continuous surface reconstructions are provided by a plurality of versions of moving least squares (MLS). The lack of connectivity makes changes of model

topology more accessible, but comes at a cost. K-nearest neighborhood, such as being needed form many surface processing algorithms, has to be computed on the fly. This, in turn, requires more elaborate data structures including KD-trees or spatial hashing. Also, improperly sampled point models do not give guarantees on topological correctness which may or may not be a problem. The flexibility of dynamic adjacency computation is specifically efficient if the model size is large and the operations are local. Some researchers have resorted to cache strategies to retain some static adjacency in the representation.

Traditionally, point samples have been utilized successfully in particle systems for the physically based *animation* of fluids or smoke. So-called smoothed particle hydrodynamics (SPH) and, more generally, Lagrangian methods for solving Navier-Stokes equations have become an indispensable toolset for simulation. Although they share limitations regarding the reproduction of some physical effects including incompressibility, their implementation is comparatively simple and efficient. This makes them well suited for interactive applications, such as computer games. More recently, meshless methods have successfully been extended to compute elastic and plastic deformations as well as fracturing of solid objects. It has been shown that the absence of a rigid mesh structure facilitates the modeling of phase transitions as for instance melting. The proposed methods are robust and render visually plausible results. In addition to the usage of points for the discretization of computational domains some research has been done to reconstruct and animate the corresponding surfaces using point samples. Again, the previously discussed properties of point representations help to conveniently change topology (fracture, melting) or resample dynamically (deformation). Very often, a high-resolution point skin is combined with a lower resolution volumetric particle system.

Where we Stand

In summary, point primitives constitute a simple and versatile low-level graphics and visualization primitive. As a representation points have different strengths and weaknesses compared to other graphics primitives. As such, they are not going to replace the existing ones, but have rather proven their ability to complement them. Many technical issues related to point based graphics boil down to reconstruction and resampling. As a sample-based approach to graphics, points can be thought of being building blocks for "*visual signals*" and stimulate us take a signal processing view onto graphics and visualization.

After 10 years of research in point based graphics, I feel that I can safely say that the primitive has clearly demonstrated its usefulness and versatility in major subfields of computer graphics. Research on point primitives has found its place in the scientific community, is considered a new subfield of graphics, and has created its own subcommunity. Numerous paper submissions to all major graphics conferences and journals further evidence the dynamics and vibe of this

new area. Additionally, annual Symposia on Point Based Graphics are being held and [1] provides an excellent resource for the subject.

Future Potential

I predict a bright future for this field, both in computer graphics, and beyond. As of today, we face numerous significant open research problems in point based graphics. For instance, there is no sampling theory for general (manifold) surfaces and it is not entirely clear to us what adequate sampling really means. Also, surface reconstruction continues to be an issue and we should look beyond MLS and related methods. Points seem to constitute a proper representation for procedural geometry and might be utilized for the modeling of terrain or other complex datasets. Point based graphics architectures might become part of future generation GPUs complementing triangle rendering. To reduce bandwidth, we could synthesize samples dynamically on chip. In physically-based modeling, a broad spectrum of phenomena, such as shells or reaction-diffusion systems, is waiting for further exploration. In addition, we should investigate the question of what is the additional information that we have to store with each point in order to accelerate spatial search.

Beyond graphics, points have not been investigated very much in scientific visualization. For instance, they are ideally suited to model multidimensional, irregular data. I foresee them migrating into a broad range of applications including medical imaging, CAD, scientific computing, and many others.

A good overview of our own work on points can be found on [2]. As I write this article, we are in the process of compiling a textbook on the subject to appear in 2007. May this book serve as a reference for researchers and developers starting work in this area.

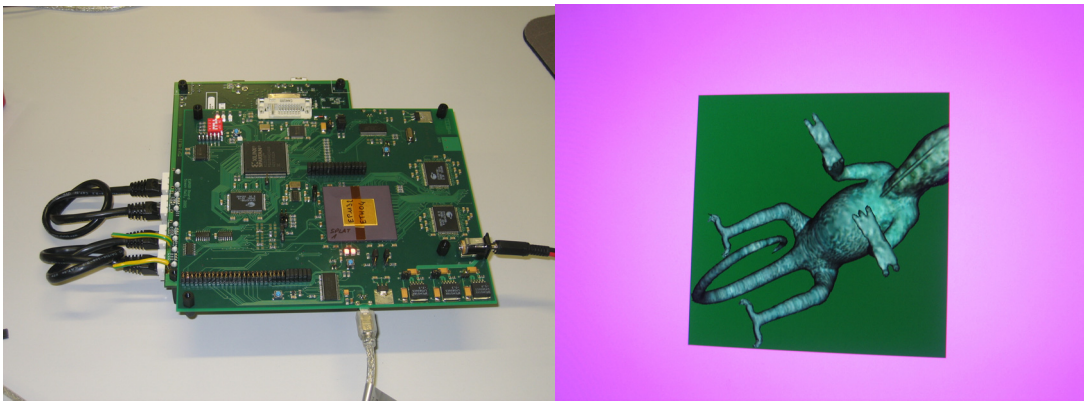

Figure 2: The world's first point graphics chip and board: An experimental ASIC implementing EWA splatting in hardware. The prototype was built at ETH as a student project in 0.25 μ technology and comprises 500 K gates. Its peak performance is 30 Mio. splats/sec @512x512 framebuffer resolution. The chip

carries the rasterization engine as well as a reordering cache and performs n extended z-testing. The image on the right displays the framebuffer output.

[1] http://myweb.hinet.net/home7/hks/Resource/PointBasedPaper.html

[2] graphics.ethz.ch->research->point based graphics