Splat Based Raytracing

Sriram Kashyap M S

Guide: Prof. Sharat Chandran

Indian Institute of Technology, Bombay

November 2008
1 Introduction
   - Points and Splats
   - Raytracing

2 Raytracing point models
   - Splat Generation
   - Raytracing
     - Overview
     - Octree Generation
     - Ray Splat Intersection
   - Normal Vectors
     - Phong Splatting

3 Results

4 Conclusion
Points and Splats (Iphigenie sculpture)

Source: Kobbelt et al., Optimized Sub-Sampling for Surface Splatting
What information is associated with a splat?
What information is associated with a splat?

- Position
- Normal Vector
- Radius
- Color
Raytracing

- Send out primary rays from the camera position through the center of each pixel of the resulting image onto the scene
- Compute the intersection of the primary rays with the objects of the scene using ray-splat intersections
- From the intersection points, send out secondary rays (shadow, reflection and refraction rays)
- Recurse reflection and refraction rays till ray-trace depth
Raytracing

Examples of point based rendering

Source: Jensen et al., Raytracing point sampled geometry
Input: A large number of points, representing the surface of the object
Output: A set of disks (splats) which cover the entire surface
Overview of Splat Generation

Initialisation

- Let there be a splat $S$ at point $p$
- Normal vector of $S = \text{normal vector at } p$
- Initial radius of $S = 0$
Overview of Splat Generation

**Initialisation**
- Let there be a splat $S$ at point $p$
- Normal vector of $S =$ normal vector at $p$
- Initial radius of $S = 0$

**Splat Growth**
- Grow the radius of the splat, till an error threshold is reached
- Error $= \max(\forall p_i \in S, \text{shortest distance between } p_i \text{ and } S)$
Splat Generation

Threshold
Splat Density

Reducing Splat Density

- Consider a splat $S$ at a point $p$, with radius $r$
- Let $D = k \cdot r$, $k \in [0, 1]$
- Points within a distance $D$ of $p$ are no longer candidates for Splat Generation

○ Inactive Points
○ Active Points
Fundamental operations in Raytracing

- Ray-surface intersection
- Reflection/Refraction based on surface normal
Fundamental operations in Raytracing

- Ray-surface intersection
- Reflection/Refraction based on surface normal
- Finding the point of intersection of ray with objects in the scene
- Calculating surface normal at that point
Raytracing: Overview

- **Light Source**
- **Object**
- **Image Plane**
- **Viewpoint**
Ray-Splat Intersection

Splits

Image plane

Ray

Viewpoint
Ray-Splat Intersection

Octree of Splats

Ray
Image plane
Viewpoint
Octree generation

The generation of the octree is done as follows:

- Start with an empty octree which is the bounding box of the entire scene
- Iteratively insert each splat into that leaf cell that contains the center of the splat
- When one leaf cell contains more than a given number of splats, the cell gets subdivided
- After the entire tree is built, insert the splats into all leaf cells they intersect
Octree generation

Splat spanning multiple leaves of the octree

Source: Rosenthal et al., Splat-based Ray Tracing of Point Clouds
Ray-Splat Intersection

Finding the splats that intersect with the ray:

- Each ray is tested for intersection against the root node
- Find the leaf cell at the point of intersection
- Check for splat intersections in this leaf
- If there are no intersections, move to next leaf in ray direction
- Else compute the precise intersection point with the splat(s)
Ray-Splat Intersection

Finding the most 'appropriate' splat in case of multiple intersections

Source: Rosenthal et al., Splat-based Ray Tracing of Point Clouds
Surface normals

- Point models are rendered using splats
- Each splat has a single normal vector
- A splat spans several pixels
Surface normals

- Point models are rendered using splats
- Each splat has a single normal vector
- A splat spans several pixels

Finding the normal vector at different points on the splat
Surface normals

- Point models are rendered using splats
- Each splat has a single normal vector
- A splat spans several pixels

Finding the normal vector at different points on the splat

Use Phong splats
Phong Splatting

Associate a linearly varying normal field with each splat
Orthogonal principal vectors: $u_j$ and $v_j$

The normal vector at a point $(u, v)$ on the splat $S_j$ is:

$$N_j(u, v) = n_j + u\alpha_j u_j + v\beta_j v_j$$
Each normal vector is represented by 2 parameters \((x,y)\), formed by intersecting the normal with an offset tangent plane.

Source: Botsch et al., Phong Splatting
Normal field calculation

- \( \mathbf{n}_j = (x_c, y_c) \)
- Let \((u_i, v_i)\) denote the coordinates of a point on the splat
- The normal at this point:
  \[ N_j(u_i, v_i) = \mathbf{n}_j + u_i \alpha_j \mathbf{u}_j + v_i \beta_j \mathbf{v}_j = (x_i, y_i) \]

\[ x_c + u_i \alpha = x_i \]
\[ y_c + v_i \beta = y_i \]
Normal field calculation

For each point $p_i \in S_j$, we have its corresponding projection on the splat, $(u_i, v_i)$ and normal vector $(x_i, y_i)$

\[
\begin{align*}
    x_c + u_1\alpha &= x_1 \\
    x_c + u_2\alpha &= x_2 \\
    x_c + u_3\alpha &= x_3 \\
    x_c + u_4\alpha &= x_4 \\
    \vdots \\
    x_c + u_n\alpha &= x_n
\end{align*}
\]

Solve for $x_c$ and $\alpha$ by fitting a straight line to these points with least square error
Gaussian Blending vs Phong Splatting: 350k Splats

Source: Botsch et al., Phong Splatting
Phong Splatting: Resolution Scaling

Gaussian Blending vs Phong Splatting: 110k Splats

Source: Botsch et al., Phong Splatting
Gaussian Blending vs Phong Splatting: 35k Splats

Source: Botsch et al., Phong Splatting
Normal field discontinuity

When two splats $S_i$ and $S_j$ overlap, ray splat intersections need to move smoothly across the point $y$

Source: Rosenthal et al., Splat-based Ray Tracing of Point Clouds
Normal field discontinuity

Weighted averaging of normals

Let $S_1, \ldots S_p$ be all the splats that are hit by a ray within a small environment $\xi$ around the intersection point. Let $(u_1, v_1), \ldots (u_p, v_p)$ be the coordinates of the ray intersection points. Let $n_1, \ldots n_p$ be the normals at the intersection points. Then, the normal $n$ at the intersection point is given by:

$$n = \frac{\sum_{i=1}^{p} (1-\|\langle u_i, v_i \rangle \|_2) n_i}{\sum_{i=1}^{p} (1-\|\langle u_i, v_i \rangle \|_2)}$$
Improving the Normal Field

- Small splats → minimal overlap
- Large splats → smooth normals
- Proposal: Use different splat radius for splat generation, and normal field generation
Skeleton Hand data set

Source: Rosenthal et al., Splat-based Ray Tracing of Point Clouds
Buddha data set

Source: Rosenthal et al., Splat-based Ray Tracing of Point Clouds
Conclusion

- Point based rendering techniques have advanced to an extent where they can perform most tasks that can be accomplished through traditional rendering methods.
- Splat rendering can be an efficient alternative to traditional rendering methods for complex scenes.
Pitfalls

- Lack of connectivity requires that implementations rely heavily on thresholds to distinguish between different surfaces.
- Efficient only for dense and complex models (not suited for rendering large plain surfaces).
- Triangle based rendering is very well established and current algorithms for triangle rendering are far more efficient.
- Existing graphics hardware have been optimised for triangle mesh rendering.
- Currently, there is no advantage of point based raytracing as compared to traditional methods.
References