


Point-Based Rendering

Matthias Zwicker
Computer Graphics Lab
ETH Zürich

Point-Based Computer GraphicsMatthias Zwicker1



Point-Based Rendering

- Introduction and motivation
- Surface elements
- Rendering
- Antialiasing
- Hardware Acceleration
- Conclusions

Point-Based Computer GraphicsMatthias Zwicker2



Motivation 1




Quake 2
1998



Nvidia GeForce4
2002

Point-Based Computer GraphicsMatthias Zwicker3




Motivation 1

- Performance of 3D hardware has exploded (e.g., GeForce4: 136 million vertices per second)
- Projected triangles are very small (i.e., cover only a few pixels)
- Overhead for triangle setup increases (initialization of texture filtering, rasterization)

➔ A simpler, more efficient rendering primitive than triangles?


Point-Based Computer GraphicsMatthias Zwicker4



Motivation 2


- Modern 3D scanning devices (e.g., laser range scanners) acquire huge point clouds
- Generating consistent triangle meshes is time consuming and difficult

➔ A rendering primitive for direct visualization of point clouds, without the need to generate triangle meshes?




4 million pts.
[Levoy et al. 2000]

Point-Based Computer GraphicsMatthias Zwicker5



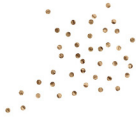
Points as Rendering Primitives

- Point clouds instead of triangle meshes [Levoy and Whitted 1985, Grossman and Dally 1998, Pfister et al. 2000]



triangle mesh (with textures)

➔



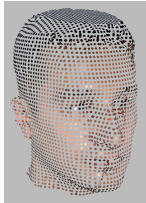
point cloud

Point-Based Computer GraphicsMatthias Zwicker6

Point-Based Surface Representation



- Points are *samples* of the surface
- The point cloud describes:
 - 3D geometry of the surface
 - Surface reflectance properties (e.g., diffuse color, etc.)
- There is no additional information, such as
 - connectivity (i.e., explicit neighborhood information between points)
 - texture maps, bump maps, etc.

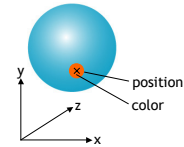


Surface Elements - Surfels



- Each point corresponds to a surface element, or *surfel*, describing the surface in a small neighborhood
- Basic surfels:

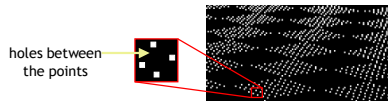
```
BasicSurfel {
  position;
  color;
}
```



Surfels



- How to represent the surface between the points?



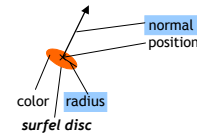
- Surfels need to *interpolate* the surface between the points
- A certain *surface area* is associated with each surfel

Surfels



- Surfels can be extended by storing additional attributes
- This allows for higher quality rendering or advanced shading effects

```
ExtendedSurfel {
  position;
  color;
  normal;
  radius;
  etc...
}
```



Surfels



- Surfels store essential information for *rendering*
- Surfels are primarily designed as a *point rendering primitive*
- They do not provide a mathematically smooth surface definition (see [Alexa 2001], point set surfaces)

Model Acquisition



- 3D scanning of physical objects
 - See Pfister, acquisition
 - Direct rendering of acquired point clouds
 - No mesh reconstruction necessary



[Matusik et al. 2002]

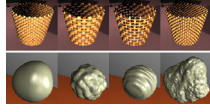
Model Acquisition



- Sampling synthetic objects
 - Efficient rendering of complex models
 - Dynamic sampling of procedural objects and animated scenes (see Stamminger, dynamic sampling)



[Zwicker et al. 2001]



[Stamminger et al. 2001]

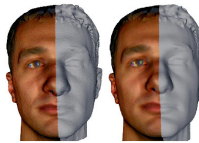
Point-Based Computer Graphics

Matthias Zwicker 13

Model Acquisition



- Processing and editing of point-sampled geometry



spectral processing
[Pauly, Gross 2002]
(see Gross, spectral processing)

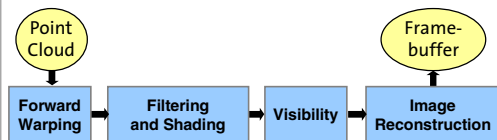


point-based surface editing
[Zwicker et al. 2002]
(see Pauly, Pointshop3D)

Point-Based Computer Graphics

Matthias Zwicker 14

Point Rendering Pipeline

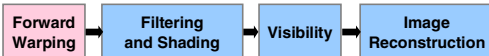


- Simple, pure forward mapping pipeline
- Surfels carry all information through the pipeline („surfel stream“)
- No texture look-ups
- Framebuffer stores RGB, alpha, and Z

Point-Based Computer Graphics

Matthias Zwicker 15

Point Rendering Pipeline



- Perspective projection of each point in the point cloud
- Analogous to projection of triangle vertices
 - homogeneous matrix-vector product
 - perspective division

Point-Based Computer Graphics

Matthias Zwicker 16

Point Rendering Pipeline

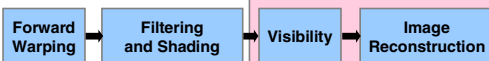


- Per-point shading
- Conventional models for shading (Phong, Torrance-Sparrow, reflections, etc.)
- High quality antialiasing is an advanced topic discussed later in the course

Point-Based Computer Graphics

Matthias Zwicker 17

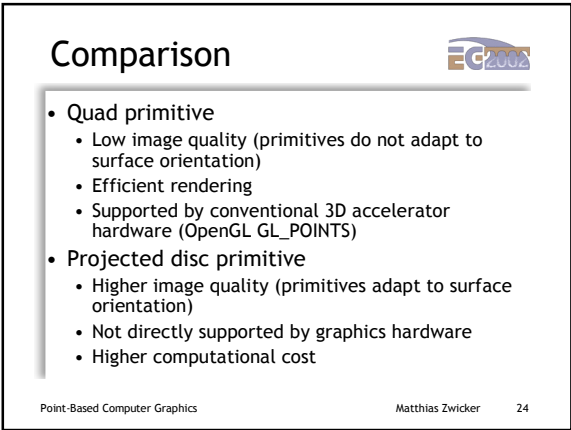
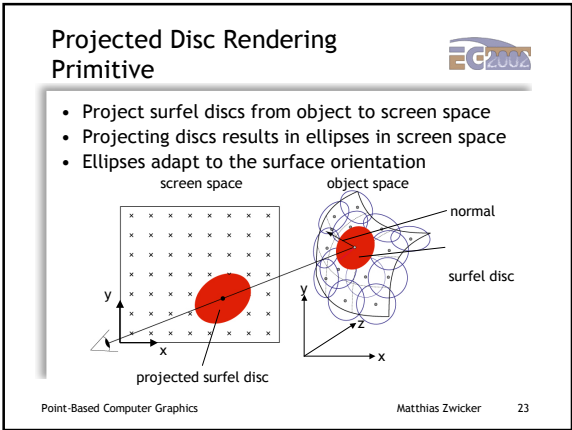
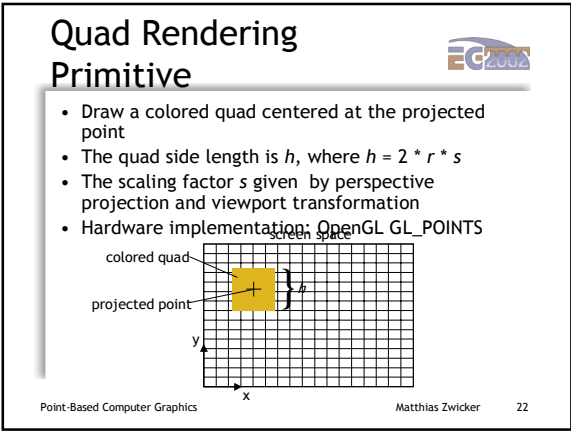
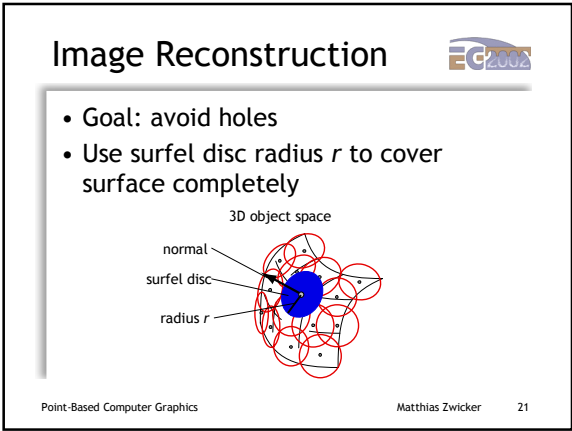
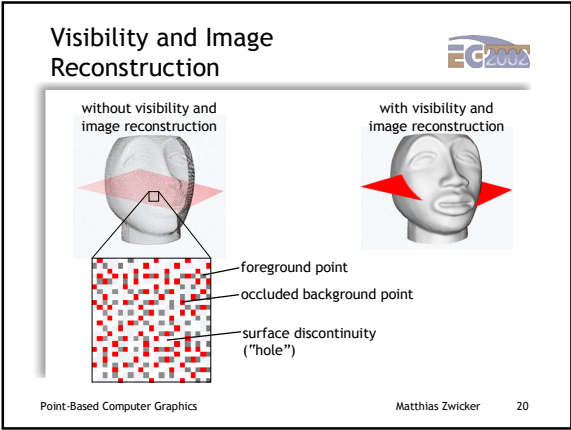
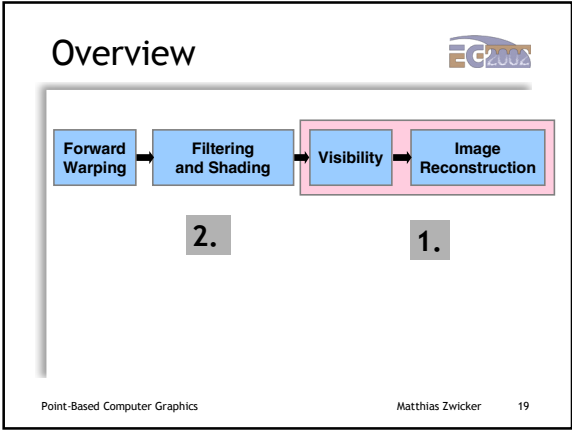
Point Rendering Pipeline



- Visibility and image reconstruction is performed simultaneously
 - Discard points that are occluded from the current viewpoint
 - Reconstruct continuous surfaces from projected points

Point-Based Computer Graphics

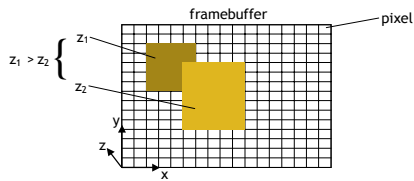
Matthias Zwicker 18



Visibility: Z-Buffering



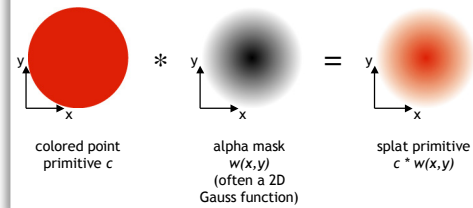
- **No blending** of rendering primitives



Splatting



- A splat primitive consists of a colored point primitive and an alpha mask



Splatting



- The final color $c(x,y)$ is computed by **additive alpha blending**, i.e., by computing the weighted sum

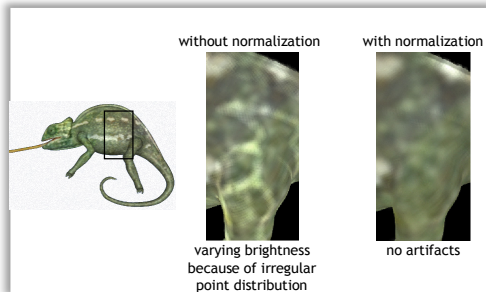
$$c(x,y) = \frac{\sum_i c_i w_i(x,y)}{\sum_i w_i(x,y)}$$

color of splat i alpha of splat i at position (x,y)

- Normalization is necessary, because the weights do not sum up to one with irregular point distributions

$$\sum_i w_i(x,y) \neq 1$$

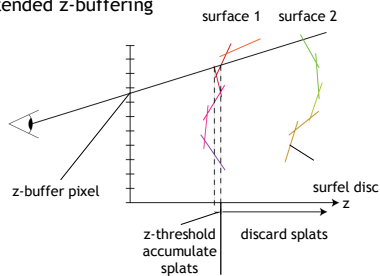
Splatting



Splatting



- Extended z-buffering



Extended Z-Buffering



```

DepthTest(x,y) {
    if (abs(splat z - z(x,y)) < threshold) {
        c(x,y) = c(x,y) + splat color
        w(x,y) = w(x,y) + splat w(x,y)
    } else if (splat z < z(x,y)) {
        z(x,y) = splat z
        c(x,y) = splat color
        w(x,y) = splat w(x,y)
    }
}
    
```

Splatting Comparison

elliptical splats circular splats with min. radius surface splatting

minif. ↑
↓ magnif.

128 x 192 128 x 192 128 x 192

Point-Based Computer Graphics Matthias Zwicker 31

High Quality Splatting

- High quality splatting requires careful analysis of **aliasing** issues
 - Review of signal processing theory
 - Application to point rendering
 - Surface splatting [Zwicker et al. 2001]

Point-Based Computer Graphics Matthias Zwicker 32

Aliasing in Computer Graphics

- Aliasing = Sampling of continuous functions below the **Nyquist frequency**
 - To avoid aliasing, sampling rate must be twice as high as the maximum frequency in the signal
- Aliasing effects:
 - Loss of detail
 - Moiré patterns, jagged edges
 - Disintegration of objects or patterns
- Aliasing in Computer Graphics
 - Texture Mapping
 - Scan conversion of geometry

Point-Based Computer Graphics Matthias Zwicker 33

Aliasing in Computer Graphics

- Aliasing: high frequencies in the input signal appear as low frequencies in the reconstructed signal

Point-Based Computer Graphics Matthias Zwicker 34

Occurrence of Aliasing

Spatial Domain Frequency Domain Spatial Domain Frequency Domain

continuous signal $A_{\text{cont}}(\omega)$ reconstruction filter $A_{\text{recon}}(\omega)$

sampling grid $A_{\text{sam}}(\omega)$ reconstructed signal (aliased) $A_{\text{recon}}(\omega)$

sampling signal $A_{\text{sam}}(\omega)$ aliasing

Point-Based Computer Graphics Matthias Zwicker 35

Aliasing-Free Reconstruction

Spatial Domain Frequency Domain Spatial Domain Frequency Domain

continuous signal $A_{\text{cont}}(\omega)$ reconstruction filter $A_{\text{recon}}(\omega)$

sampling grid $A_{\text{sam}}(\omega)$ reconstructed signal (correct) $A_{\text{recon}}(\omega)$

sampling signal $A_{\text{sam}}(\omega)$ no aliasing

Point-Based Computer Graphics Matthias Zwicker 36

Antialiasing

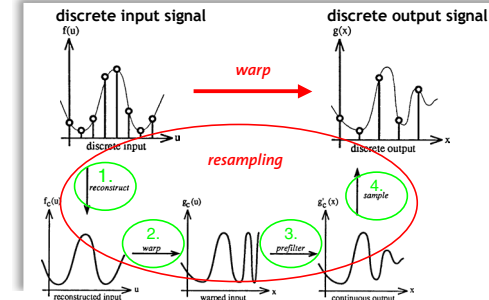


- Prefiltering
 - Band-limit the continuous signal before sampling
 - Eliminates all aliasing (with an ideal low-pass filter)
 - Closed form solution not available in general
- Supersampling
 - Raise sampling rate
 - Reduces, but does not eliminate all aliasing artifacts (in practice, many signals have infinite frequencies)
 - Simple implementation (hardware)

Point-Based Computer Graphics

Matthias Zwicker 37

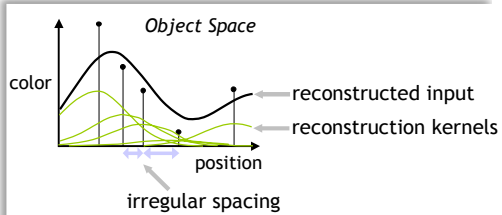
Resampling



Point-Based Computer Graphics

Matthias Zwicker 38

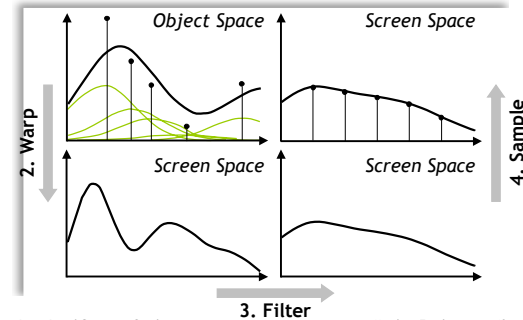
Resampling Filters



Point-Based Computer Graphics

Matthias Zwicker 39

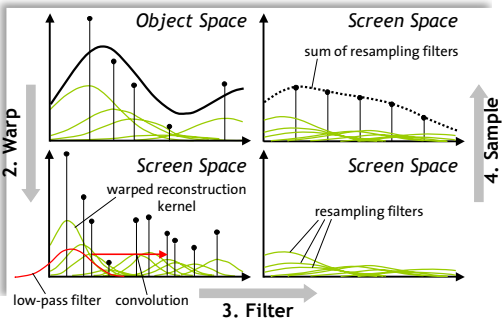
Resampling Filters



Point-Based Computer Graphics

Matthias Zwicker 40

Resampling Filters



Point-Based Computer Graphics

Matthias Zwicker 41

Resampling



- Resampling in the context of surface rendering
 - Discrete input function = surface texture (discrete 2D function)
 - Warping = projecting surfaces to the image plane (2D to 2D projective mapping)

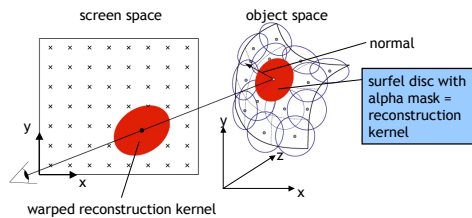
Point-Based Computer Graphics

Matthias Zwicker 42

2D Reconstruction Kernels



- Warping a 2D reconstruction kernel is equivalent to projecting a surfel disc with alpha mask



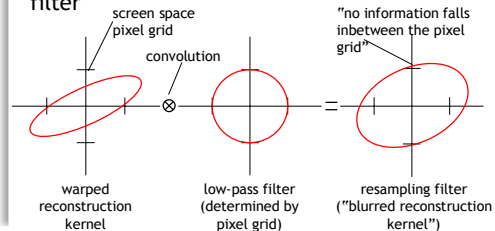
Point-Based Computer Graphics

Matthias Zwicker 43

Resampling Filters



- A resampling filter is a convolution of a warped reconstruction filter and a low-pass filter



Point-Based Computer Graphics

Matthias Zwicker 44

Mathematical Formulation



$$c(x, y) = \sum_k c_k r_k(m^{-1}(x, y)) \otimes h(x, y)$$

pixel color

reconstruction kernel color

warping function

reconstruction kernel

low pass filter

Point-Based Computer Graphics

Matthias Zwicker 45

Gaussian Resampling Filters



- Gaussians are closed under linear warping and convolution
- With Gaussian reconstruction kernels and low-pass filters, the resampling filter is a Gaussian, too
- Efficient rendering algorithms (*surface splatting* [Zwicker et al. 2001])

Point-Based Computer Graphics

Matthias Zwicker 46

Mathematical Formulation



$$c(x, y) = \sum_k c_k r_k(m^{-1}(x, y)) \otimes h(x, y)$$

Gaussian reconstruction kernel

Gaussian low-pass filter

screen space

screen space

Point-Based Computer Graphics

Matthias Zwicker 47

Mathematical Formulation



$$c(x, y) = \sum_k c_k r_k(m^{-1}(x, y)) \otimes h(x, y)$$

$$= \sum_k c_k G_k(x, y)$$

Gaussian resampling filter

Point-Based Computer Graphics

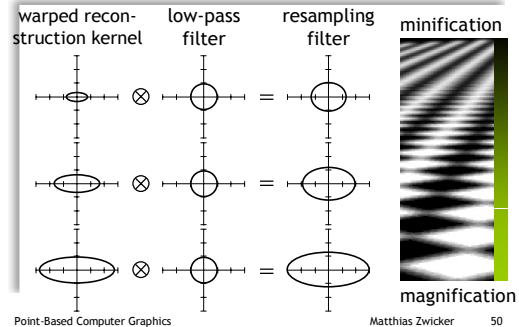
Matthias Zwicker 48

Algorithm



```
for each point P {
  project P to screen space;
  shade P;
  determine resampling kernel G;
  splat G;
}
for each pixel {
  normalize;
}
```

Properties of 2D Resampling Filters



Hardware Implementation



- Based on the object space formulation of EWA filtering
- Implemented using textured triangles
- All calculations are performed in the programmable hardware (extensive use of vertex shaders)
- Presented at EG 2002 ([Ren et al. 2002])

Surface Splatting Performance



- Software implementation
 - 500 000 splats/sec on 866 MHz PIII
 - 1 000 000 splats/sec on 2 GHz P4
- Hardware implementation [Ren et al. 2002]
 - Uses texture mapping and vertex shaders
 - 3 000 000 splats/sec on GeForce4 Ti 4400

Conclusions



- Points are an efficient rendering primitive for highly complex surfaces
- Points allow the direct visualization of real world data acquired with 3D scanning devices
- High performance, low quality point rendering is supported by 3D hardware (tens of millions points per second)
- High quality point rendering with anisotropic texture filtering is available
 - 3 million points per second with hardware support
 - 1 million points per second in software
- Antialiasing technique has been extended to volume rendering

Applications



- Direct visualization of point clouds
- Real-time 3D reconstruction and rendering for virtual reality applications
- Hybrid point and polygon rendering systems
- Rendering animated scenes
- Interactive display of huge meshes
- On the fly sampling and rendering of procedural objects

Future Work



- Dedicated rendering hardware
- Efficient approximations of exact EWA splatting
- Rendering architecture for on the fly sampling and rendering

References



- [Levoy and Whitted 1985] The use of points as a display primitive, technical report, University of North Carolina at Chapel Hill, 1985
- [Heckbert 1986] Fundamentals of texture mapping and image warping, Master's Thesis, 1986
- [Grossman and Dally 1998] Point sample rendering, Eurographics workshop on rendering, 1998
- [Levoy et al. 2000] The digital Michelangelo project, SIGGRAPH 2000
- [Rusinkiewicz et al. 2000] Qsplat, SIGGRAPH 2000
- [Pfister et al. 2000] Surfels: Surface elements as rendering primitives, SIGGRAPH 2000
- [Zwicker et al. 2001] Surface splatting, SIGGRAPH 2001
- [Zwicker et al. 2002] EWA Splatting, to appear, IEEE TVCG 2002
- [Ren et al. 2002] Object space EWA splatting: A hardware accelerated approach to high quality point rendering, Eurographics 2002