Rendering diffuse objects using particle systems inside voxelized surface geometry

Thorsten Juckel
Steffi Beckhaus

University of Hamburg
interactive media / virtual environments

Outline

1. Introduction
2. Brief overview about common techniques for rendering diffuse objects
3. Introduction to particle systems and voxelization
4. Description of our design approach
5. Interaction, manipulation, and dynamics
6. Results

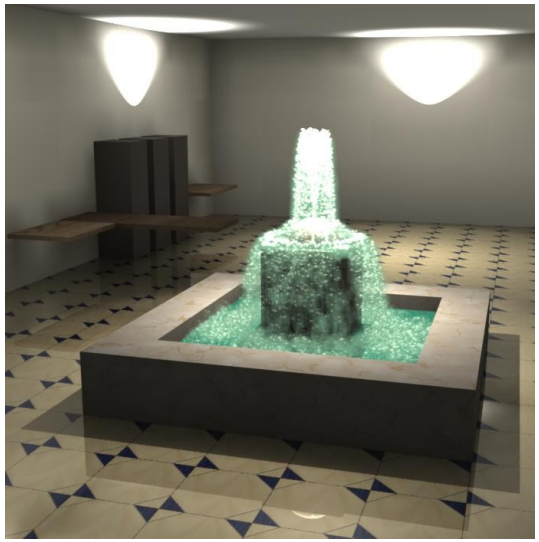
Introduction

In computer graphics

- Clouds, dust, fire, water, smoke, explosions are needed

Solid objects can be „modelled“ and added to the scene

„Ghosts“ or other more complex diffuse objects need complex rendering techniques or shaders for rendering

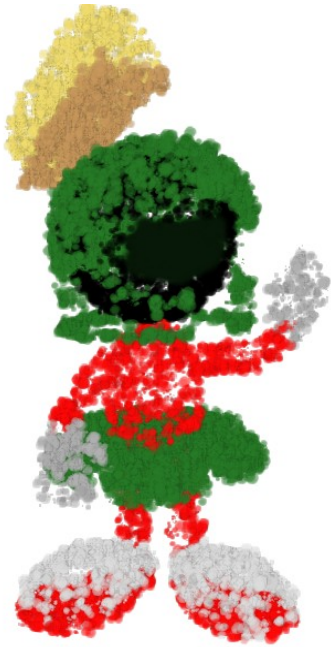


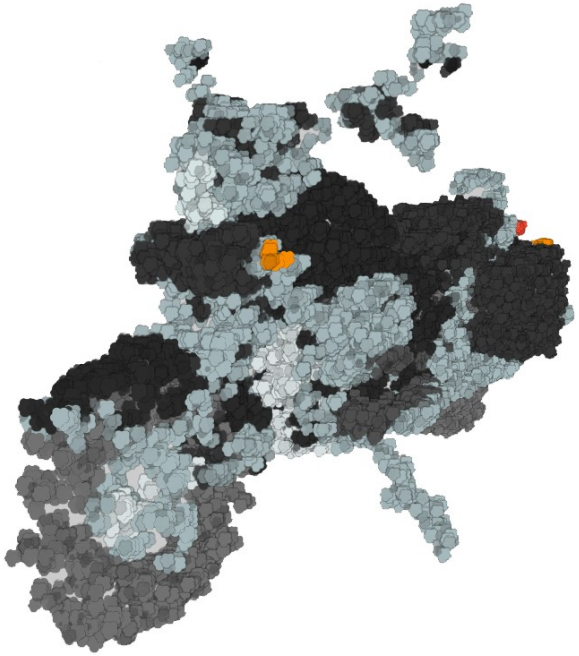
© McAllister, ParticleSystems.org

Introduction

How can someone add a character to the application?

- Based on surface geometry
- Constantly changes shape
- Is diffuse in appearance
- Fuzzy and dynamic shape



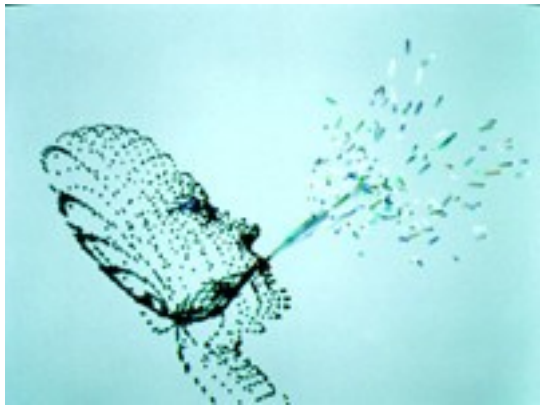


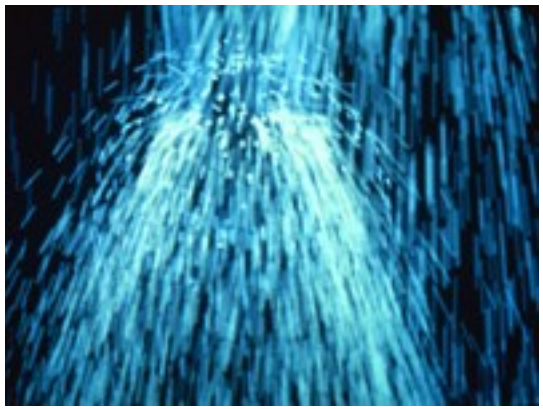
Particle systems

Introduced in 1983 by Reeves for the movie „Star Trek: The Wrath of Kahn“

Basic principle of a particle system

- Particles are autonomous, free moving points on screen
- Current state is stored inside the particles
- Rendering as points, lines, small objects
- Emitter controls generation, update, and extinction
- Now used in all fields of CG





© K.Sims, „Particle Dreams“, 1988

Particle systems

Particle systems are a good choice for dust, smoke, and fire

Problems with particles

- Movement of particles is „pre-determined“ by the programmer
- Forces are mostly assumed to be constant inside the system
- Particle interaction is expensive => $O(n^2)$

Designers want to model a figure

- Hide programming from designer
- Focus on modelling ascpect

What do we need?

We need something to display that data:

- ➔ Particle systems are a good choice!

We need something to convert surface geometry

- Designer should focus on modelling aspect

We need a data stucture for storing attributes:

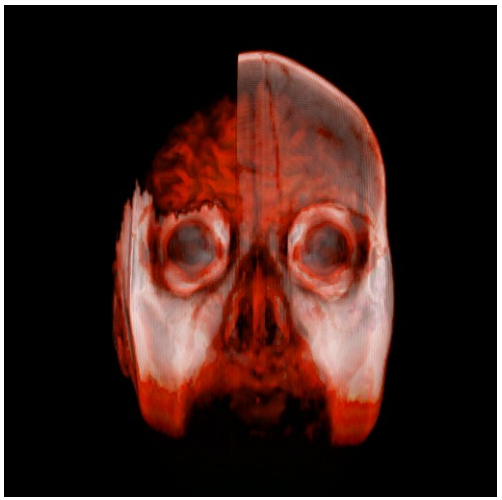
- Color, size, transparency, texture, forces, and fading behavior
- Allow fast and easy access
- ➔ Voxels can store lots of data and be used for spatial arrangement

Voxelization

Voxels are small cubes in space

- Commonly used for storing scientific data (MRI)
- Coloring indicates density
- Converted using the Marching Cubes algorithm

But: We need the other way around!



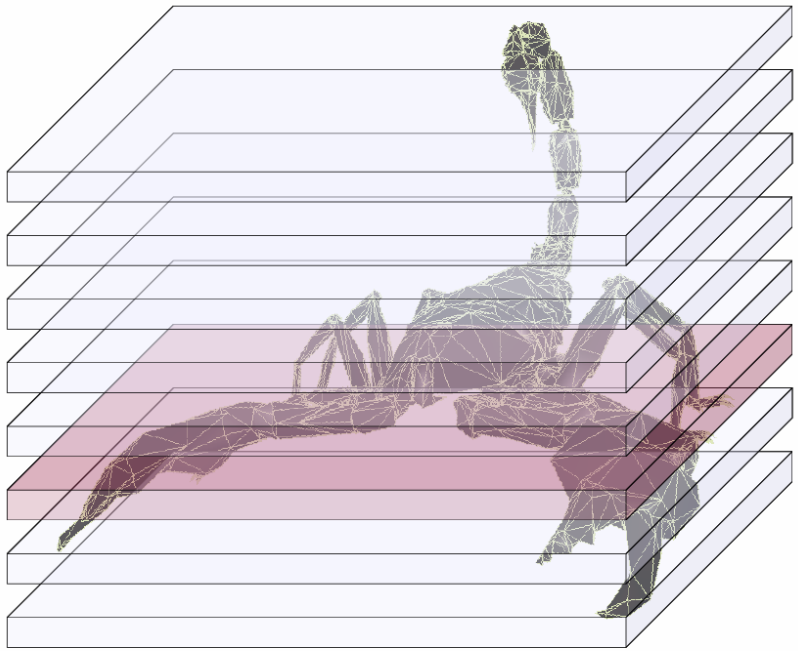
From:
http://anusf.anu.edu.au/anusf_visualization/viz_showcase/volume_rendering/

Voxelization

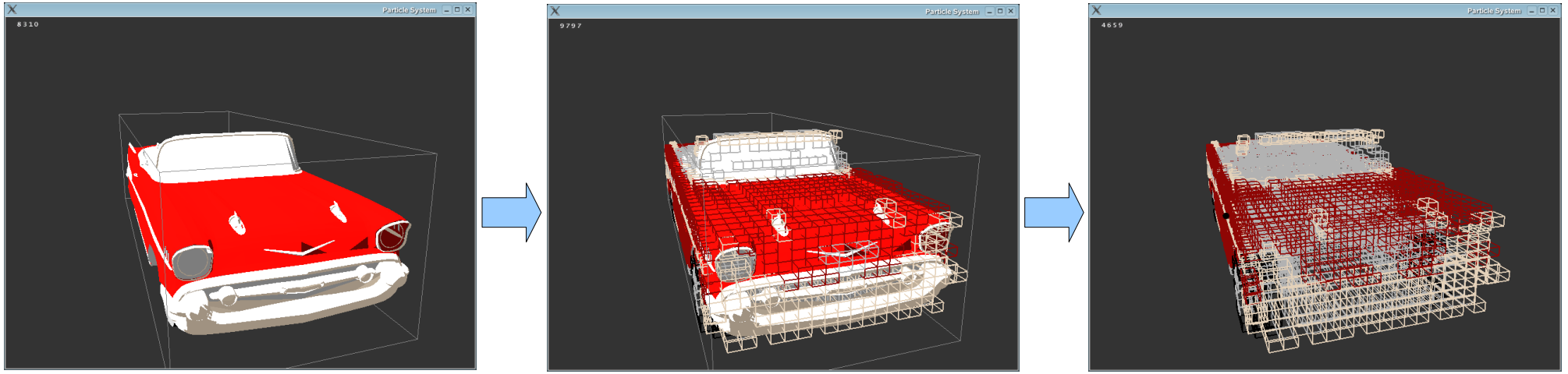
Beckhaus et al. used a hardware accelerated approach

- Scene is rendered into the frame-buffer
- Pixels are extruded
- Voxels used for spacial analysis and collision detection

Objects can be converted from surface geometry to voxels



Extending voxelization algorithm



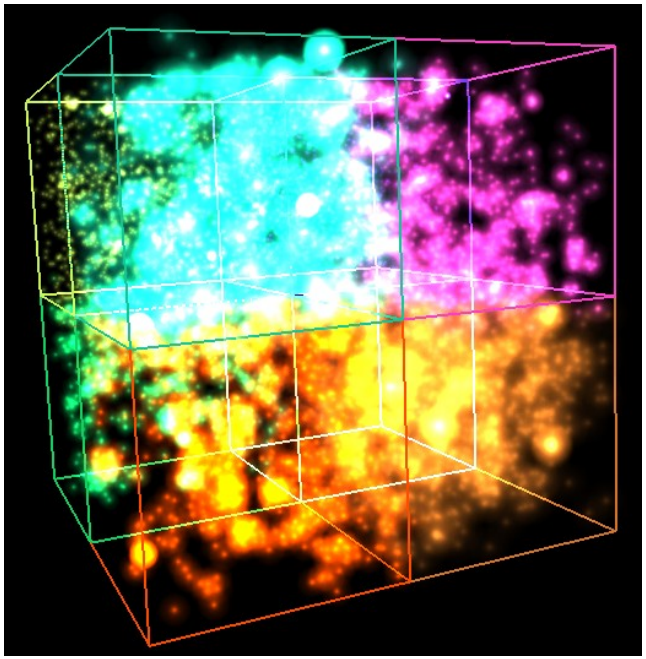
Initial values

- Color and surface information of the model can be stored inside the voxels
- Additional information, such as size, forces, density, or fading of particels are assigned during initialization
- Voxels outside the geometric model are deactivated
- All voxels are stored inside an array to allow fast random access
- Structure is easily extendible

Particle Creation

Using the underlying data structure, particles can be created

- Emitter is handling creation and update cycle
- Creation is done randomly inside activated voxels
- Initial values for particles are assigned by the current voxel
- Movement is determined by forces inside the voxel
- Easy access of voxel attributes through array structure



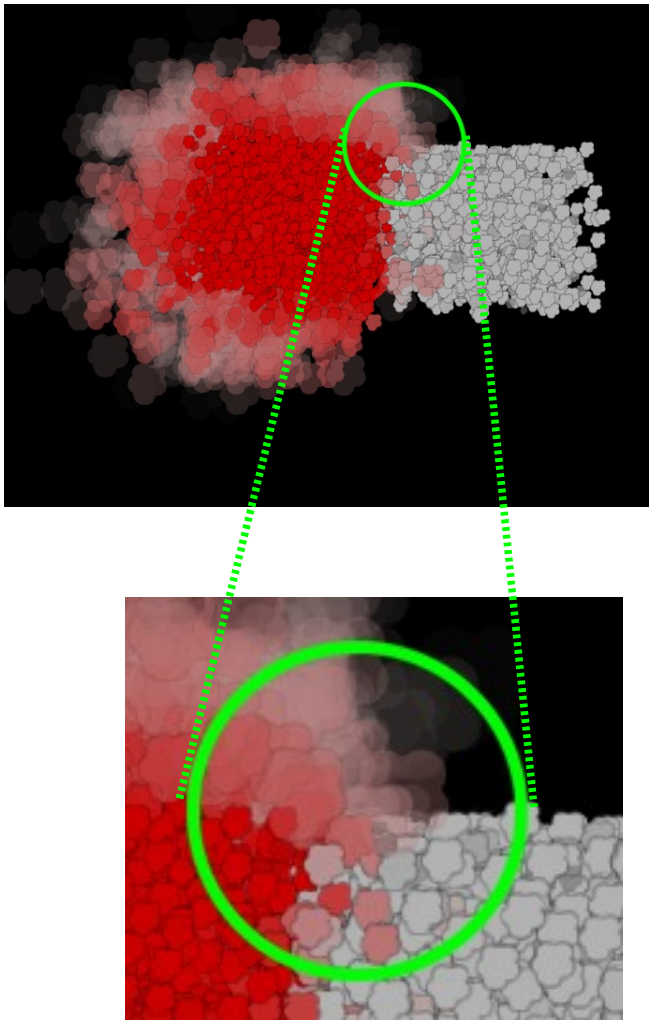
Creating a particle shape

System characteristics

- Particle never die inside an active voxel
- Particle attributes are interpolated between voxels
- Particles die if they leave an active voxel

Additionally

- Different types of forces can act in different voxels



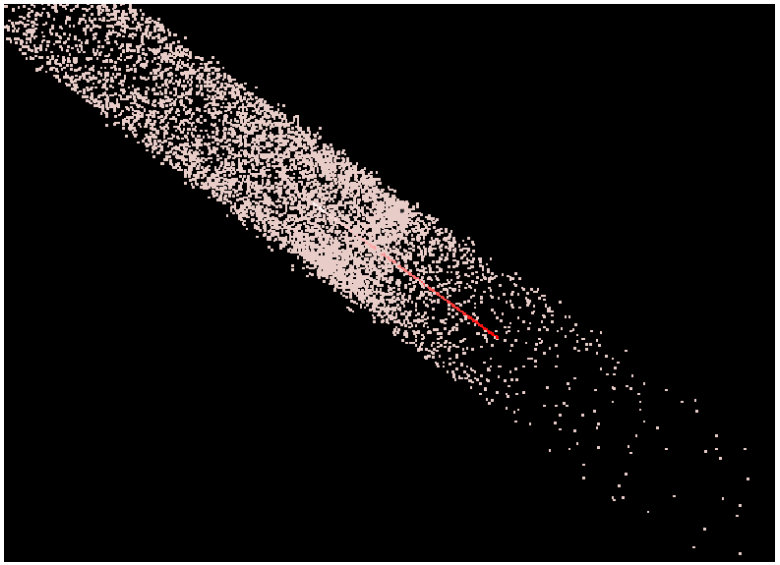
Forces inside the voxel

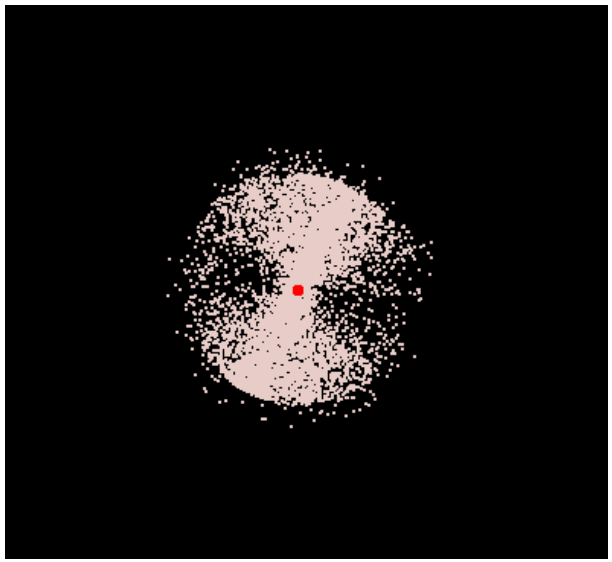
One force type assigned per voxel

- Acting at the center

Implemented forces:

- linear, rotational, spiral movement, gravity points, and helix type of motion





Outside forces and interaction

Outside forces still are constant to the whole system

- Gravity
- Wind
- ...

Translation and rotation is treated as a force

- Old particles move stay where they are
- New particle inherit an initial inertia

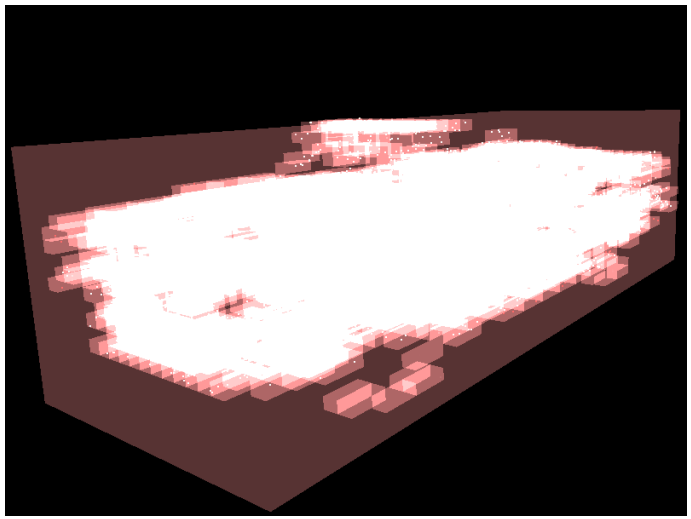
Special care regarding particles – object collisions

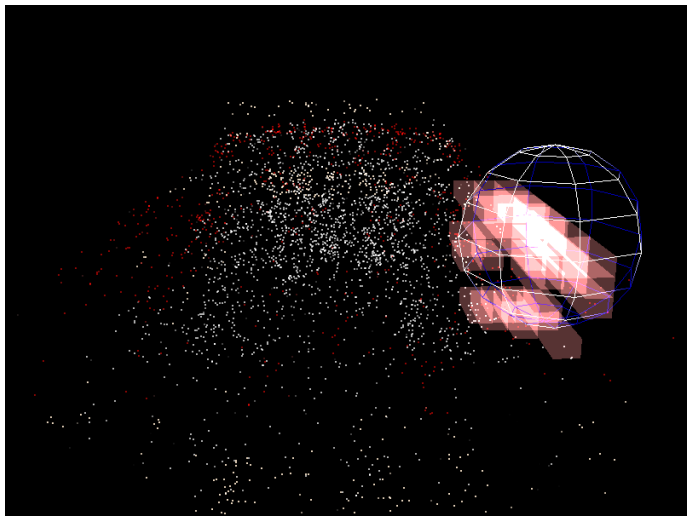
Particle collisions

Collision calculations should be limited to a minimum

Hierarchial bounding box organization

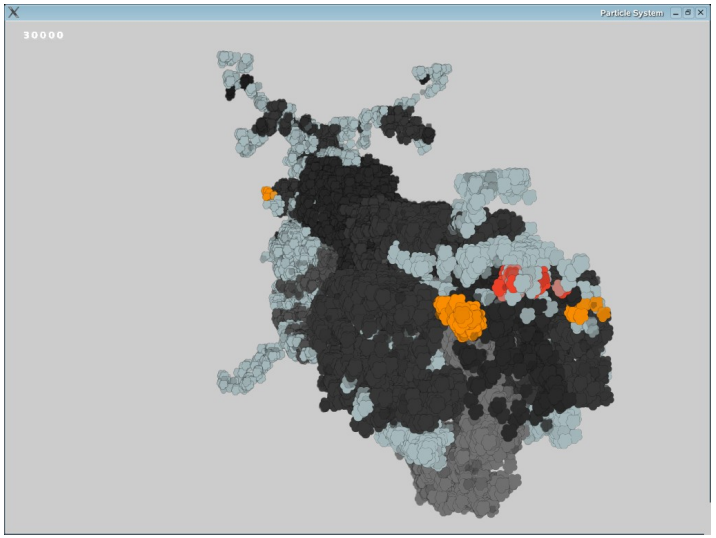
- Collisions only have to be checked if the outer BB penetrates another object
- Checking against voxel-BB's can be done quickly
- Particles need to be checked only inside a few voxels
- Colliding voxels should be deactivated

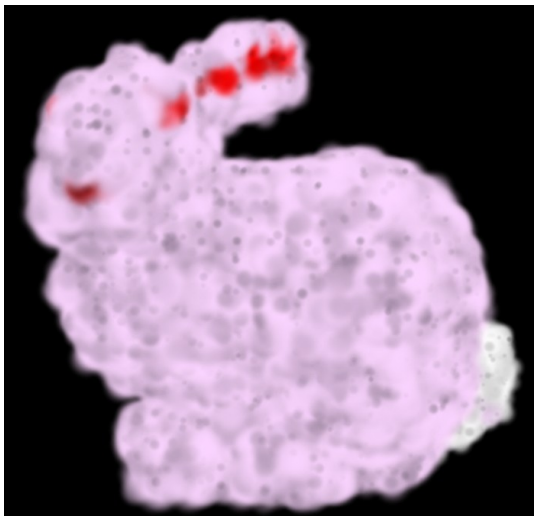


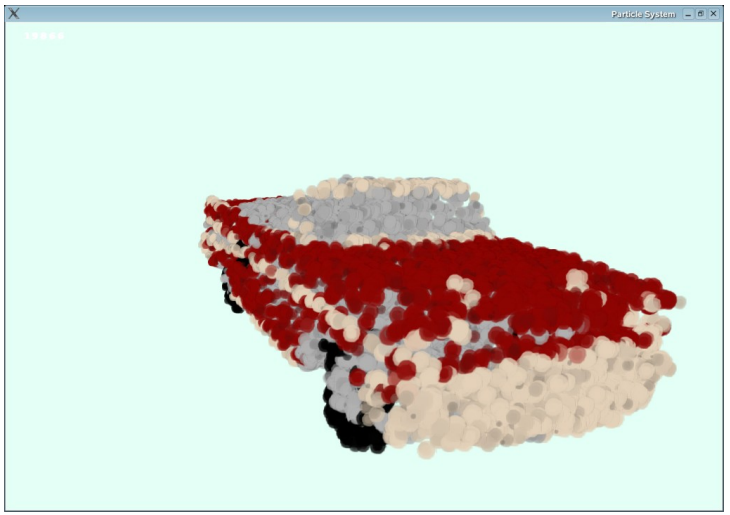


Results


Particle System
40000


Particle System
30000




Particle System

Problems / Additions

Issues

- Voxelization not perfect
- Rendering artefacts leave holes in the shell
- Automatic filling of the shape cannot be done

Additions

- Particle interaction is not included yet
- Fluid simulations should be included to extend movement realism

Conclusion

We presented a simple way to display fuzzy characters and dynamic objects using particle systems inside a voxel-structure

- Designer can focus on modelling
- Particle system for real-time rendering
- Forces act on the particles inside the voxels to move them inside the structure
- Collision detection is done using a structured bounding box hierarchy
- Architecture can easily be extended

Questions?

Contact:

juckel@blurredvision.de
www.blurredvision.de

steffi.beckhaus@uni-hamburg.de
imve.informatik.uni-hamburg.de