Ray Tracing Animations Using 4D Kd-Trees

Jens Olsson

Examensarbete 2007-02-19
Outline

- Introduction
  - Ray-Tracing
  - Motion Blur
  - Acceleration Structures

- Project
  - Motivation
  - Theory
  - Implementation

- Results
  - Data
  - Conclusions
  - Future Work
Outline

Introduction
  - Ray-Tracing
  - Motion Blur
  - Acceleration Structures

Project
  - Motivation
  - Theory
  - Implementation

Results
  - Data
  - Conclusions
  - Future Work
Outline

- Introduction
  - Ray-Tracing
  - Motion Blur
  - Acceleration Structures

- Project
  - Motivation
  - Theory
  - Implementation

- Results
  - Data
  - Conclusions
  - Future Work
Outline

- Introduction
  - Ray-Tracing
  - Motion Blur
  - Acceleration Structures

- Project
  - Motivation
  - Theory
  - Implementation

- Results
  - Data
  - Conclusions
  - Future Work
Ray-Tracing

- Technique used to generate convincing imagery
- Mimics a camera, but reversed light flow
- Millions of rays are traced through the scene
- Very general, extends naturally from simple to complex models
Ray-Tracing

- Technique used to generate convincing imagery
- Mimics a camera, but reversed light flow
- Millions of rays are traced through the scene
- Very general, extends naturally from simple to complex models
Ray-Tracing

- Technique used to generate convincing imagery
- Mimics a camera, but reversed light flow
- Millions of rays are traced through the scene
- Very general, extends naturally from simple to complex models
Ray-Tracing

- Technique used to generate convincing imagery
- Mimics a camera, but reversed light flow
- Millions of rays are traced through the scene
- Very general, extends naturally from simple to complex models
Ray-Tracing

- Technique used to generate convincing imagery
- Mimics a camera, but reversed light flow
- Millions of rays are traced through the scene
- Very general, extends naturally from simple to complex models
Motion Blur

- Camera film is exposed for a short moment
- Moving objects exposure are spread out over the film
- The objects appear as smeared in the accumulated image
Motion Blur

- Camera film is exposed for a short moment
- Moving objects exposure are spread out over the film
- The objects appear as smeared in the accumulated image
Motion Blur

- Camera film is exposed for a short moment
- Moving objects exposure are spread out over the film
- The objects appear as smeared in the accumulated image
Motion Blur

- Camera film is exposed for a short moment
- Moving objects exposure are spread out over the film
- The objects appear as smeared in the accumulated image
Motion Blur contd.

- Simulated by stochastic sampling, accumulation buffers or blur post-process
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification, etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
  - Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification
- etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Acceleration Structures

- Ray tracing mainly performs visibility queries
- A strategy is needed to make visibility queries fast
- All strategies pre-sort scene primitives in some way
- Grids, Bounding Volume Hierarchies, BSP Trees, Ray Classification etc etc etc
Kd-Trees

- Simple data structure in the BSP tree family
- Scene volume is recursively subdivided by arbitrarily positioned axis-aligned splitting planes
- Much research has already been done, regarded as one of the most efficient acceleration structures
- Existing algorithms for construction as well as traversal
Kd-Trees

- Simple data structure in the BSP tree family
- Scene volume is recursively subdivided by arbitrarily positioned axis-aligned splitting planes
- Much research has already been done, regarded as one of the most efficient acceleration structures
- Existing algorithms for construction as well as traversal
Kd-Trees

- Simple data structure in the BSP tree family
- Scene volume is recursively subdivided by arbitrarily positioned axis-aligned splitting planes
- Much research has already been done, regarded as one of the most efficient acceleration structures
- Existing algorithms for construction as well as traversal
Kd-Trees

- Simple data structure in the BSP tree family
- Scene volume is recursively subdivided by arbitrarily positioned axis-aligned splitting planes
- Much research has already been done, regarded as one of the most efficient acceleration structures
- Existing algorithms for construction as well as traversal
Kd-Trees

- Simple data structure in the BSP tree family
- Scene volume is recursively subdivided by arbitrarily positioned axis-aligned splitting planes
- Much research has already been done, regarded as one of the most efficient acceleration structures
- Existing algorithms for construction as well as traversal
Kd-Trees contd.
Kd-Trees contd.
Motivation

- Easy to formulate time-dependent ray tracing
- Attractive properties for rendering sequences of images
- Practical problems when actually implementing it
- Efficient techniques lays the foundation for more advanced future work (i.e. global illumination)
Motivation

- Easy to formulate time-dependent ray tracing
- Attractive properties for rendering sequences of images
- Practical problems when actually implementing it
- Efficient techniques lays the foundation for more advanced future work (ie. global illumination)
Motivation

- Easy to formulate time-dependent ray tracing
- Attractive properties for rendering sequences of images
- Practical problems when actually implementing it
- Efficient techniques lays the foundation for more advanced future work (ie. global illumination)
Motivation

- Easy to formulate time-dependent ray tracing
- Attractive properties for rendering sequences of images
- Practical problems when actually implementing it
- Efficient techniques lays the foundation for more advanced future work (ie. global illumination)
Motivation

- Easy to formulate time-dependent ray tracing
- Attractive properties for rendering sequences of images
- Practical problems when actually implementing it
- Efficient techniques lays the foundation for more advanced future work (i.e., global illumination)
We wish to adapt kd-trees to store time-dependent data

- We need to subdivide primitives temporally
- We need a new heuristic for constructing the tree
Theory

We wish to adapt kd-trees to store time-dependent data

- We need to subdivide primitives temporally
- We need a new heuristic for constructing the tree
Theory

We wish to adapt kd-trees to store time-dependent data

- We need to subdivide primitives temporally
- We need a new heuristic for constructing the tree
Theory contd.
Theory contd.
Temporal segmentation rule
find the largest $t_b > t_a$ such that

$$S\left(B\left(\bigcup_{i} B(P_{t_i})\right)\right) \leq \kappa \ S(B(P_{t_a})) \ , \ t_i \in [t_a, t_b]$$
Theory contd.

- Nodes are split using a heuristic
- Most popular heuristic for kd-trees are Surface Area Heuristics (SAH)
- Makes a local estimate of the future costs of traversal and intersection tests of the resulting subnodes
Theory contd.

- Nodes are split using a heuristic
- Most popular heuristic for kd-trees are Surface Area Heuristics (SAH)
- Makes a local estimate of the future costs of traversal and intersection tests of the resulting subnodes
Theory contd.

- Nodes are split using a heuristic
- Most popular heuristic for kd-trees are Surface Area Heuristics (SAH)
- Makes a local estimate of the future costs of traversal and intersection tests of the resulting subnodes
Theory contd.

The SAH cost function

\[ C = c_t + (1 - b_e)(p_A N_A c_i + p_B N_B c_i) \]

We reinterpreted these for the temporal case.
The SAH cost function

\[ C = c_t + (1 - b_e)(p_A N_A c_i + p_B N_B c_i) \]

We reinterpreted these for the temporal case
Theory contd.

- **Spatial case**

\[ p_A = p(B_A|B) = \frac{S_{BA}}{S_B} \quad p_B = p(B_B|B) = \frac{S_{BB}}{S_B} \]

- **Temporal case**

\[ p_A = p(B_A|B) = \frac{|t_{BA}|}{|t_B|} \quad p_B = p(B_B|B) = \frac{|t_{BB}|}{|t_B|} \]
Theory contd.

- **Spatial case**

\[ p_A = p(B_A | B) = \frac{S_{BA}}{S_B} \quad p_B = p(B_B | B) = \frac{S_{BB}}{S_B} \]

- **Temporal case**

\[ p_A = p(B_A | B) = \frac{|t_{BA}|}{|t_B|} \quad p_B = p(B_B | B) = \frac{|t_{BB}|}{|t_B|} \]
Implementation

- Developed a simple ray tracer
- Written in C++
- Plug-in inside off-the-shelf 3D software
Implementation

- Developed a simple ray tracer
- Written in C++
- Plug-in inside off-the-shelf 3D software
Implementation

- Developed a simple ray tracer
- Written in C++
- Plug-in inside off-the-shelf 3D software
Data

The 3 test scenes
Data contd.

Head Test
Data contd.

Head Test Results

<table>
<thead>
<tr>
<th>method</th>
<th>Head test common data accelerator tree memory size</th>
<th>primitives memory size</th>
</tr>
</thead>
<tbody>
<tr>
<td>accumulation</td>
<td>3D kd-tree</td>
<td>0.2 MB</td>
</tr>
<tr>
<td>stochastic</td>
<td>4D kd-tree</td>
<td>18.4 MB</td>
</tr>
</tbody>
</table>
Data contd.

<table>
<thead>
<tr>
<th>accumulation</th>
<th>8 subframes</th>
<th>16 subframes</th>
<th>32 subframes</th>
<th>64 subframes</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (min)</td>
<td>1.29</td>
<td>2.63</td>
<td>4.99</td>
<td>8.87</td>
</tr>
<tr>
<td>total rays</td>
<td>1961956</td>
<td>3909952</td>
<td>7806048</td>
<td>15599764</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stochastic</th>
<th>4 samples</th>
<th>8 samples</th>
<th>16 samples</th>
<th>32 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (min)</td>
<td>1.25</td>
<td>1.95</td>
<td>3.37</td>
<td>6.22</td>
</tr>
<tr>
<td>total rays</td>
<td>933204</td>
<td>1886628</td>
<td>3822216</td>
<td>7687188</td>
</tr>
</tbody>
</table>
### Head test rendering time breakdown (seconds)

<table>
<thead>
<tr>
<th>accumulation</th>
<th>8 subframes</th>
<th>16 subframes</th>
<th>32 subframes</th>
<th>64 subframes</th>
</tr>
</thead>
<tbody>
<tr>
<td>update data</td>
<td>4.3</td>
<td>8.3</td>
<td>16.3</td>
<td>26.1</td>
</tr>
<tr>
<td>build tree</td>
<td>2.6</td>
<td>5.5</td>
<td>10.3</td>
<td>17.6</td>
</tr>
<tr>
<td>raytrace</td>
<td>70.0</td>
<td>143.5</td>
<td>271.6</td>
<td>487.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stochastic</th>
<th>4 samples</th>
<th>8 samples</th>
<th>16 samples</th>
<th>32 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>update data</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>build tree</td>
<td>26.6</td>
<td>26.4</td>
<td>26.0</td>
<td>26.0</td>
</tr>
<tr>
<td>raytrace</td>
<td>45.8</td>
<td>87.9</td>
<td>173.7</td>
<td>344.8</td>
</tr>
</tbody>
</table>
### Stochastic rendering time comparison (minutes)

<table>
<thead>
<tr>
<th>kd-tree</th>
<th>segmented</th>
<th>4 samples</th>
<th>8 samples</th>
<th>16 samples</th>
<th>32 samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td>no</td>
<td>16.7</td>
<td>36.3</td>
<td>65.4</td>
<td>130.7</td>
</tr>
<tr>
<td>3D</td>
<td>yes</td>
<td>2.3</td>
<td>3.5</td>
<td>6.1</td>
<td>11.1</td>
</tr>
<tr>
<td>4D</td>
<td>yes</td>
<td>1.2</td>
<td>1.9</td>
<td>3.4</td>
<td>6.2</td>
</tr>
</tbody>
</table>

### Stochastic rendering time breakdown (seconds)

<table>
<thead>
<tr>
<th>kd-tree</th>
<th>segmented</th>
<th>update data</th>
<th>build tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td>no</td>
<td>1.2</td>
<td>0.5</td>
</tr>
<tr>
<td>3D</td>
<td>yes</td>
<td>1.5</td>
<td>51.9</td>
</tr>
<tr>
<td>4D</td>
<td>yes</td>
<td>1.5</td>
<td>26.0</td>
</tr>
</tbody>
</table>
Conclusions

- Possible to store animation in a 4D kd-tree
- Better performance compared to storing the animation in a 3D kd-tree
- Memory requirements were higher than expected
- Interesting applications for 4D kd-trees
Conclusions

- Possible to store animation in a 4D kd-tree
- Better performance compared to storing the animation in a 3D kd-tree
- Memory requirements were higher than expected
- Interesting applications for 4D kd-trees
Conclusions

- Possible to store animation in a 4D kd-tree
- Better performance compared to storing the animation in a 3D kd-tree
- Memory requirements were higher than expected
- Interesting applications for 4D kd-trees
Conclusions

- Possible to store animation in a 4D kd-tree
- Better performance compared to storing the animation in a 3D kd-tree
- Memory requirements were higher than expected
- Interesting applications for 4D kd-trees
Conclusions

- Possible to store animation in a 4D kd-tree
- Better performance compared to storing the animation in a 3D kd-tree
- Memory requirements were higher than expected
- Interesting applications for 4D kd-trees
Applications - Temporally instanced primitives
Applications - Temporally instanced primitives
Applications - Temporally instanced primitives
Future Work

- Triangle interpolation needs to be way faster; is it possible to cpu- and cache-optimize the triangle interpolation in an efficient way?
- Costs have to be calculated at least twice for every node; Any way of avoiding this without explicitly relating temporal and spatial axes?
- Global Illumination for animations; How can we take advantage of a time-dependent structure?
- Investigate applications further; Temporal instances could be used for crowd simulations
Future Work

- Triangle interpolation needs to be way faster; is it possible to cpu- and cache-optimize the triangle interpolation in an efficient way?
- Costs have to be calculated at least twice for every node; Any way of avoiding this without explicitly relating temporal and spatial axes?
- Global Illumination for animations; How can we take advantage of a time-dependent structure?
- Investigate applications further; Temporal instances could be used for crowd simulations
Future Work

- Triangle interpolation needs to be way faster; is it possible to cpu- and cache-optimize the triangle interpolation in an efficient way?
- Costs have to be calculated at least twice for every node; Any way of avoiding this without explicitly relating temporal and spatial axes?
- Global Illumination for animations; How can we take advantage of a time-dependent structure?
- Investigate applications further; Temporal instances could be used for crowd simulations
Future Work

- Triangle interpolation needs to be way faster; is it possible to cpu- and cache-optimize the triangle interpolation in an efficient way?
- Costs have to be calculated at least twice for every node; Any way of avoiding this without explicitly relating temporal and spatial axes?
- Global Illumination for animations; How can we take advantage of a time-dependent structure?
- Investigate applications further; Temporal instances could be used for crowd simulations
Future Work

- Triangle interpolation needs to be way faster; is it possible to cpu- and cache-optimize the triangle interpolation in an efficient way?
- Costs have to be calculated at least twice for every node; Any way of avoiding this without explicitly relating temporal and spatial axes?
- Global Illumination for animations; How can we take advantage of a time-dependent structure?
- Investigate applications further; Temporal instances could be used for crowd simulations