DCGI

DEPARTMENT OF COMPUTER GRAPHICS AND INTERACTION

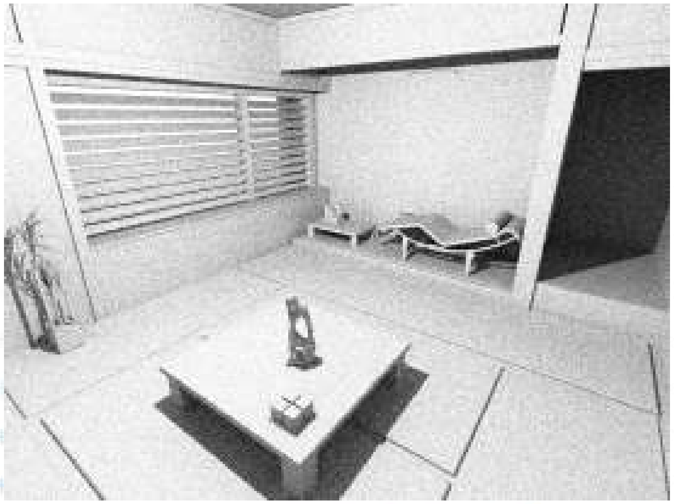
# When it makes sense to use uniform grids for ray tracing

M. Hapala, O. Karlík, V. Havran

Czech Technical University in Prague
Faculty of Electrical Engineering

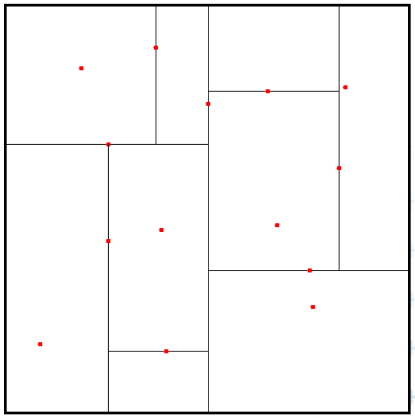
# Intro

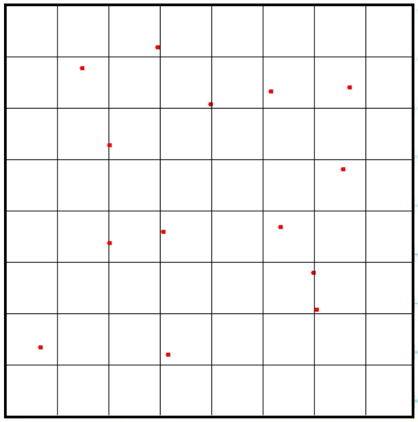
- Ray tracing/casting
  - Basic visibility operation
  - Finding closest intersections between rays and objects in a scene
- Intersection search complexity
  - Naïve in O(N)
  - Acceleration data structure as fast as O(log N)
- Applications almost always use one data structure



# Intro 2

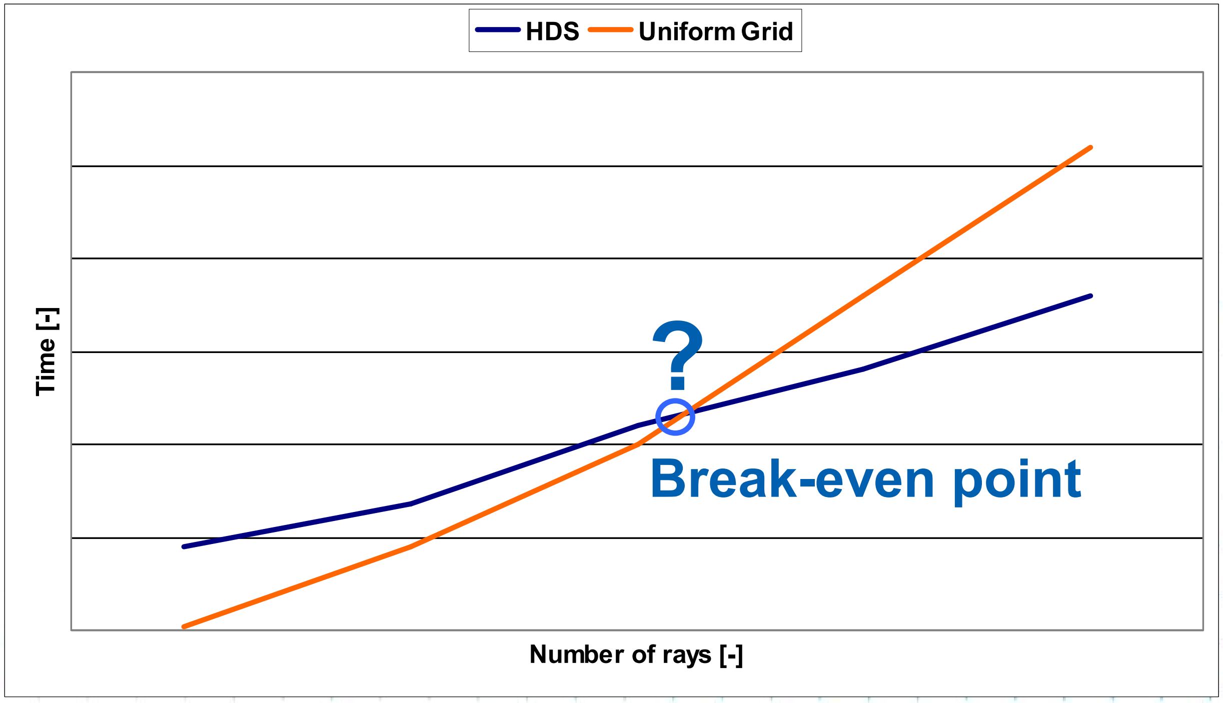
- Uniform grid
  - Build in O(N)
  - Traversal in $O(\sqrt[3]{N})$
- Hierarchical data structures (HDS)
  - Build in O(N · log N)
  - Traversal in O(log N)
- Hiddent constants for HDS traversal
  - “Quality” of the structure, how it can adapt to the scene
  - Implementation and hardware performance





# Idea

- Take the best from both worlds
- Which is more efficient for a particular scene?
- Change from grid to HDS when advantageous
- Need rough number of rays to be computed



# Calibration

- Executed once
- Set of representative scenes
- Build a HDS and measure
  - Time to build the data structure
  - Time to compute a single ray
- What do we need these for?
  - Build time constant

Build time

Average over all scenes

$$\boxed{\frac{1}{s}\sum_{i=1}^{s}}\frac{\boxed{T_B^H(i)}}{\boxed{N(i)\cdot\log_2 N(i)}}$$

Build time complexity

  - Shooting a single ray constant

$$\frac{1}{s}\sum_{i=1}^{s}\frac{T_R^H(i)}{M(i)\cdot\log_2 N(i)}$$

# Calibration

- Executed once
- Set of representative scenes
- Build a HDS and measure
  - Time to build the data structure
  - Time to compute a single ray
- What do we need these for?
  - Build time constant

$$\frac{1}{s}\sum_{i=1}^{s}\frac{T_B^H(i)}{N(i)\cdot\log_2 N(i)}$$

  - Shooting a single ray constant

Shooting all rays time

$$\boxed{\frac{1}{s}\sum_{i=1}^{s}}\frac{\boxed{T_R^H(i)}}{\boxed{M(i)}\cdot\boxed{\log_2 N(i)}}$$

Average over all scenes

Number of rays

Shooting complexity

# Application

- Build a uniform grid
- Compute a small set of representative rays
- Estimate HDS performance
  - Build time

$$\boxed{N\cdot\log_2 N}\cdot\boxed{\frac{1}{s}\sum_{i=1}^{s}\frac{T_B^H(i)}{N(i)\cdot\log_2 N(i)}}$$

Time complexity

  - Shooting a single ray time

$$\boxed{\log_2 N}\cdot\boxed{\frac{1}{s}\sum_{i=1}^{s}\frac{T_R^H(i)}{M(i)\cdot\log_2 N(i)}}$$

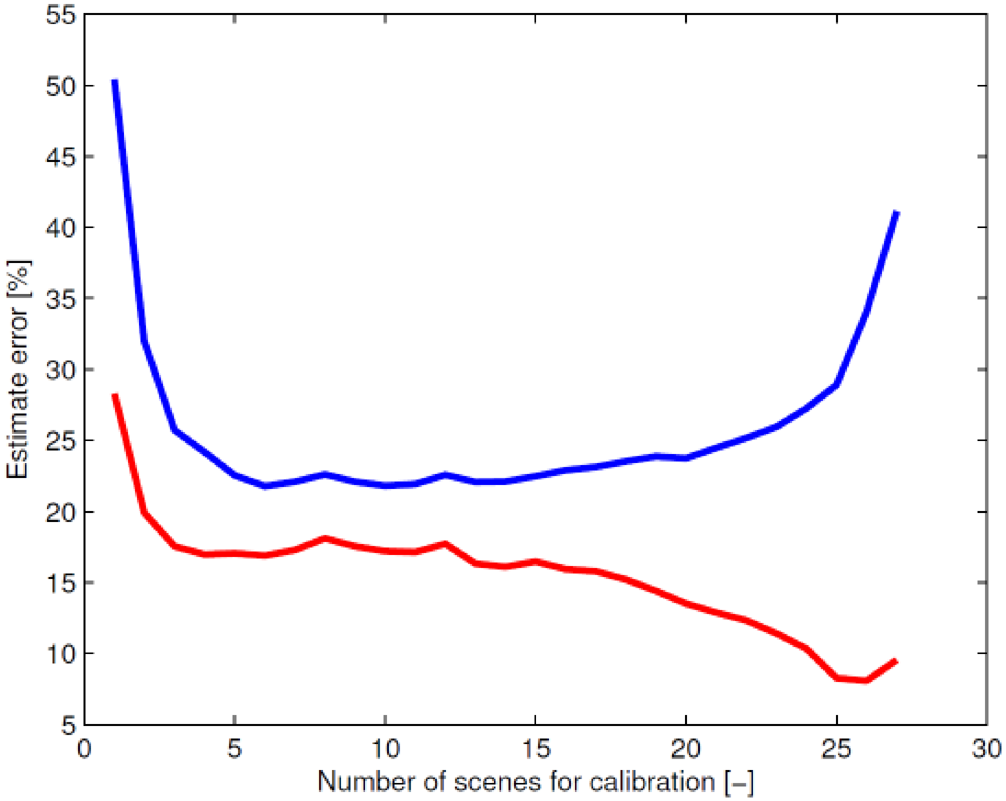
Implementation/Hardware constants

- Compute break-even point
- Decide if we need to build and use HDS
- Shoot all the rays

# Results

- Tested on 28 scenes
  - Primitive count 500 – 1.6M
  - Various levels of uniformity
  - X scenes for calibration
  - 28-X scenes estimated
- 2M rays
  - Randomly generated
  - Uniform distribution
- Estimate accuracy
- Speedup

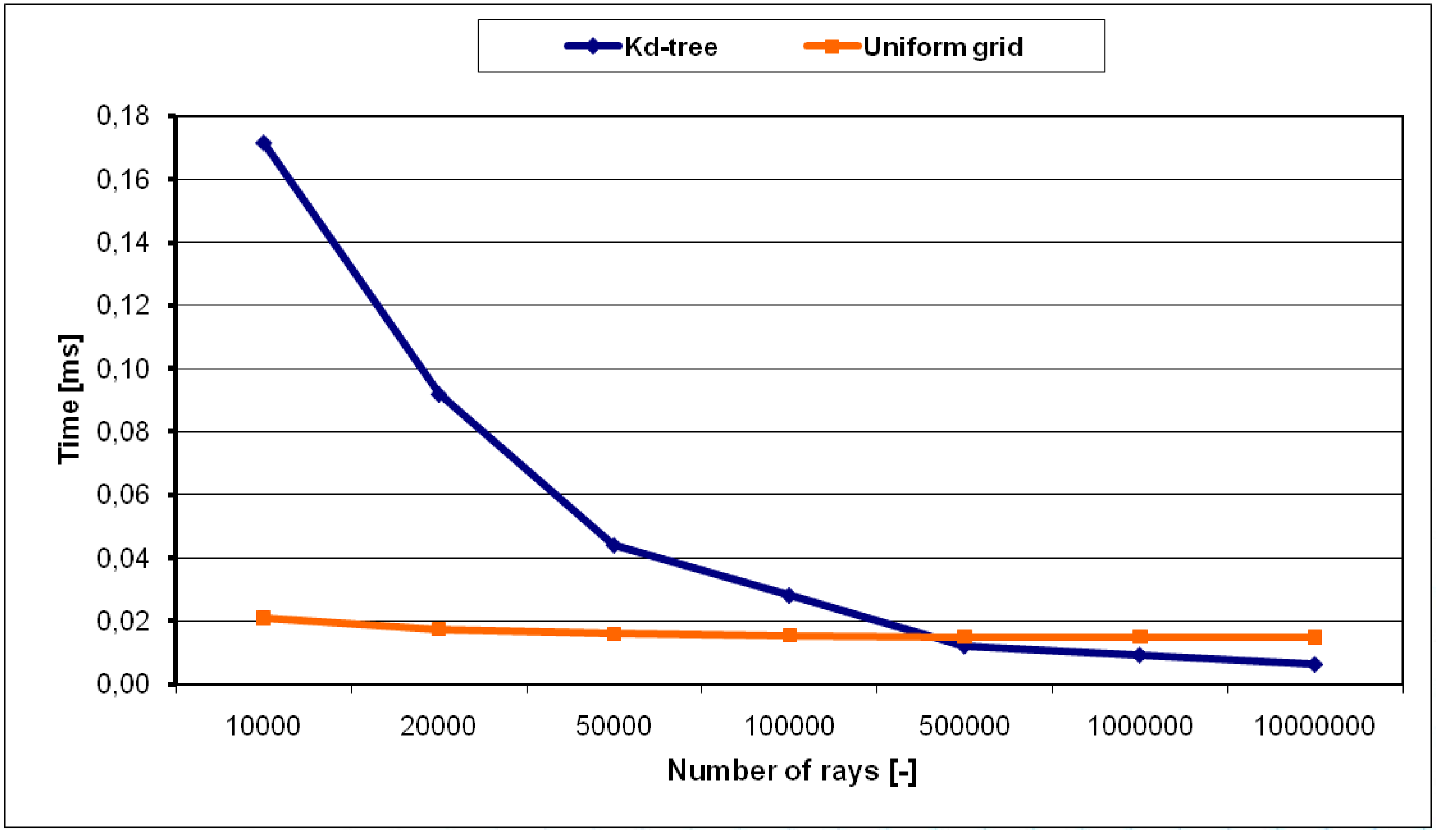
# Break-even point estimate accuracy



- Relative estimate error [%] = $100 \cdot \frac{R_{est} - R_C}{R_C}$
- Red - sum of relative errors
- Blue - sum of absolute values of relative errors

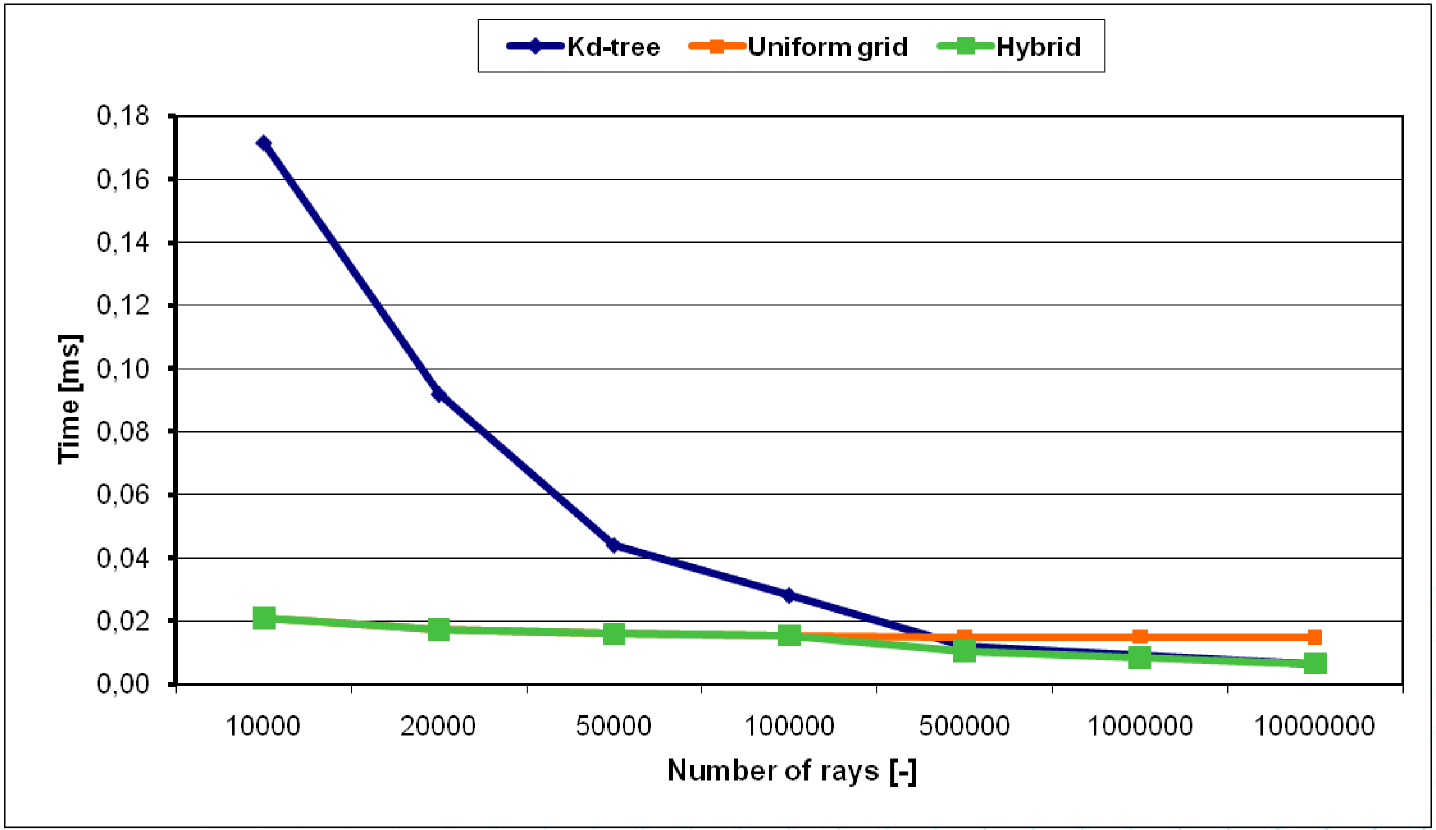
# Speedup

- Median time per ray



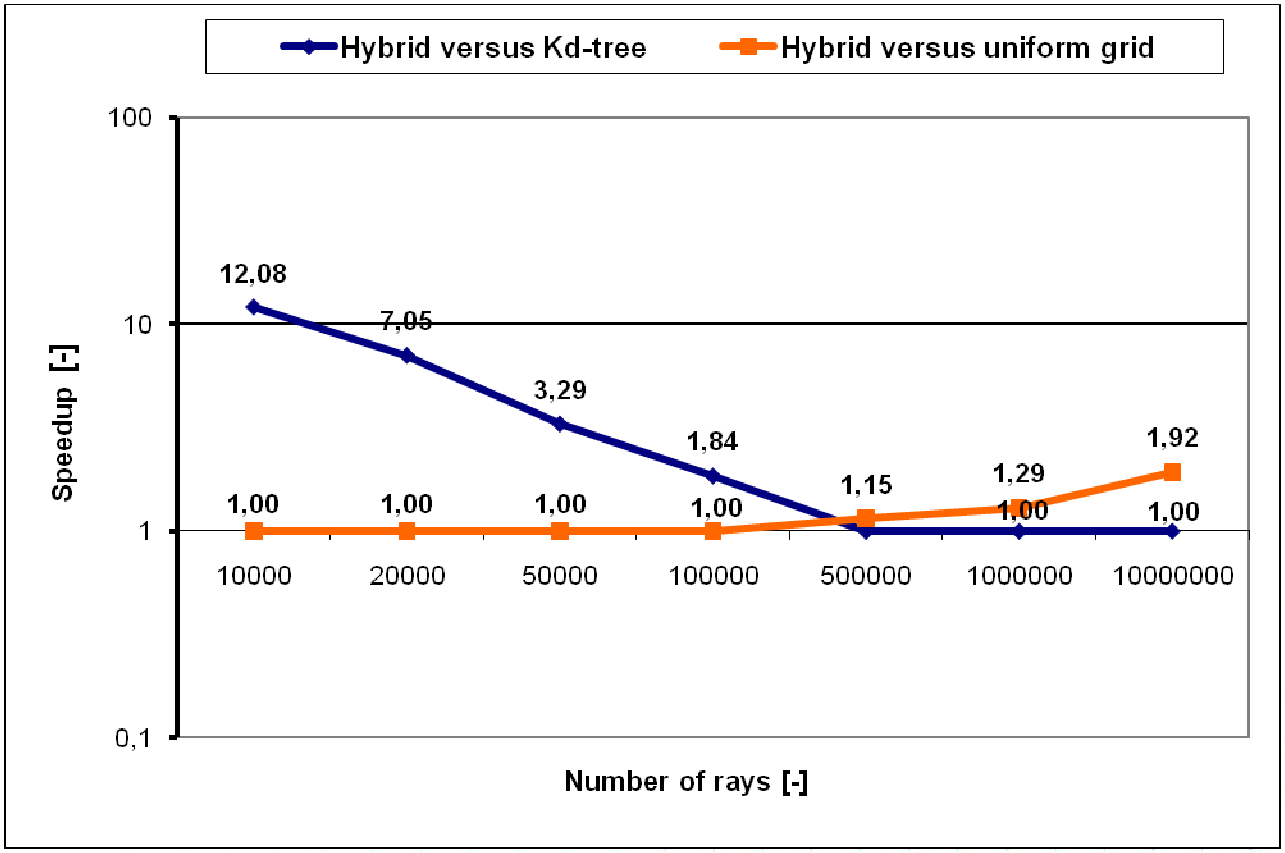
# Speedup

- Median time per ray



# Speedup

- Median hybrid algorithm speedup versus using only one data structure



# Conclusion

- Choose a data structure based on the number of rays
- Minimal overhead
- High speedup
- Uniform grid efficient even for a significant number of rays
  - In the range of millions