Improving Interaction Performance for Ray Tracing

Daniel Kurz
Christopher Lux Jan P. Springer Bernd Fröhlich
Bauhaus-Universität Weimar, Germany

Motivation

- Ray tracing
  - Excellent image quality
  - Optical effects
    - E.g. reflections, refractions, shadows
  - Limited performance
- Virtual reality systems
  - Require interactive object manipulation
  - Challenge for ray tracing

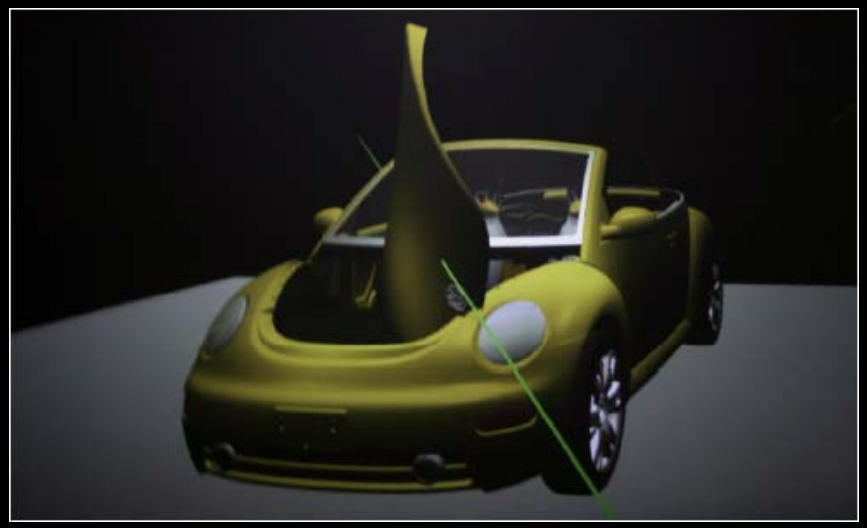
Motivation (cont'd)

Static part

Slow client

Manipulated part

Fast client

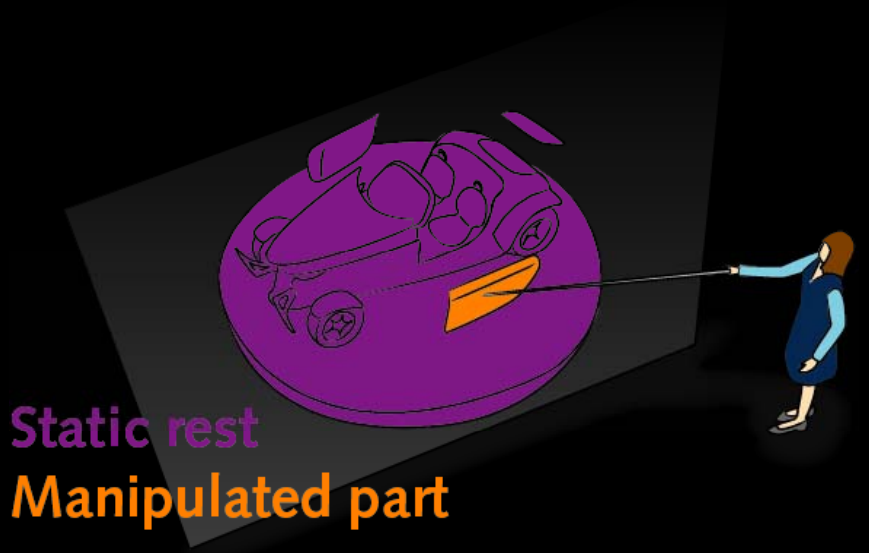


Composite

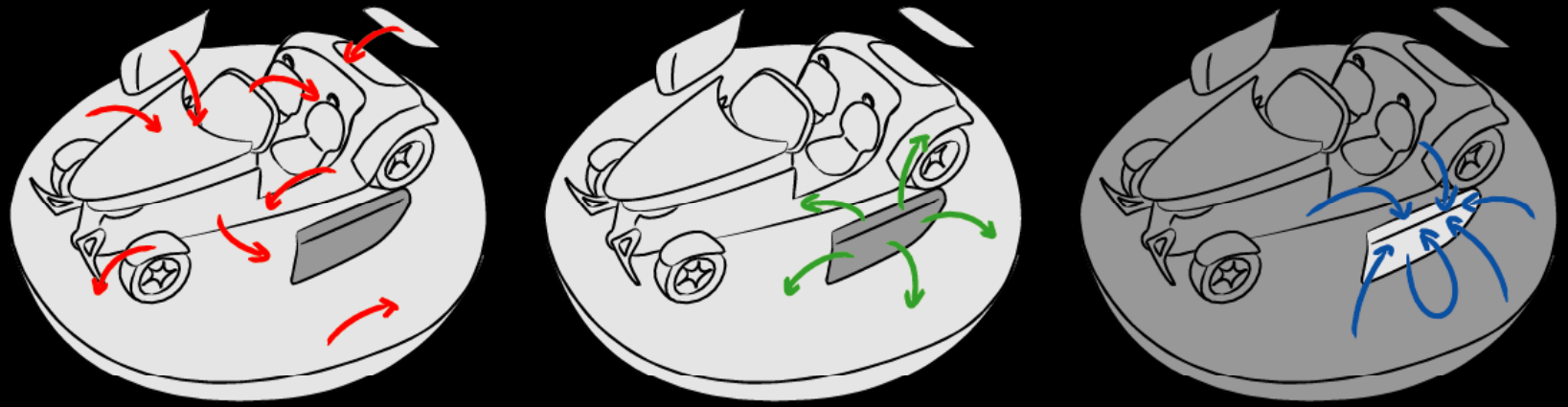
[Springer et al. IEEE VR 2007]

- Projection-based VR systems
  - High frame rates
    - Object manipulation
    - System control
  - Low(er) frame rates
    - Head tracking
    - Navigation
- Dividing the scene
  - Asynchronously rendered on different GPUs
  - Display digital composite
- Object manipulation at the fast client's frame rate
  - Multi-frame rate rendering

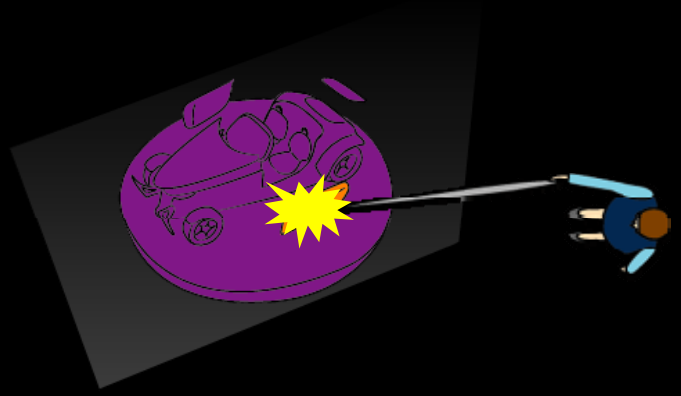
Rendering Approach



- Extending this approach to ray tracing
- Scene is divided
  - Manipulated part
  - Static rest
- First order reflections

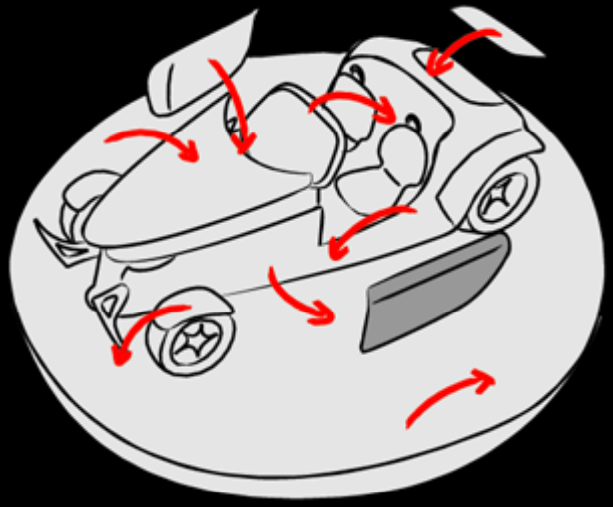


Rendering Approach (cont'd)



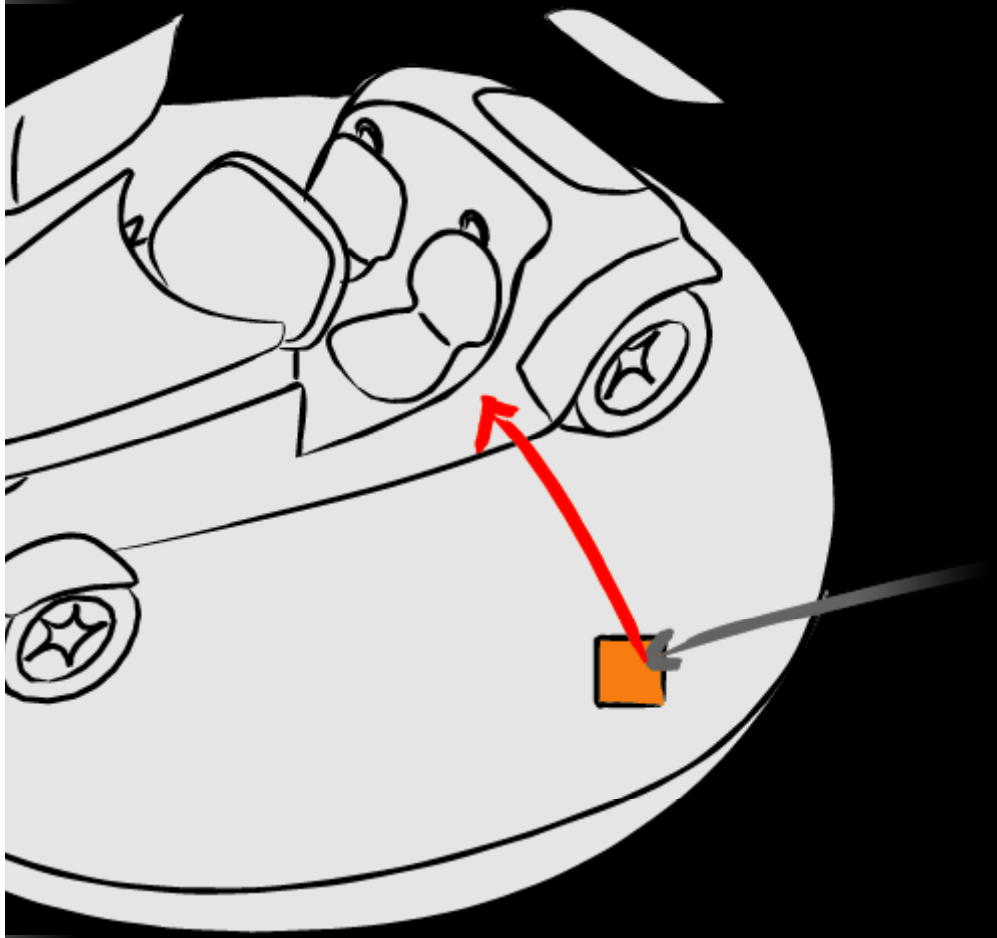
- Navigation and object manipulation are performed alternating
  - Navigation
    - Regular ray tracing
  - Object manipulation
    - Static view point
    - Reflections within the static rest remain the same
    - Can be pre-computed
    - Composed with updated reflections of the object
- Object manipulation at much higher frame rates

Rendering Approach (cont'd)



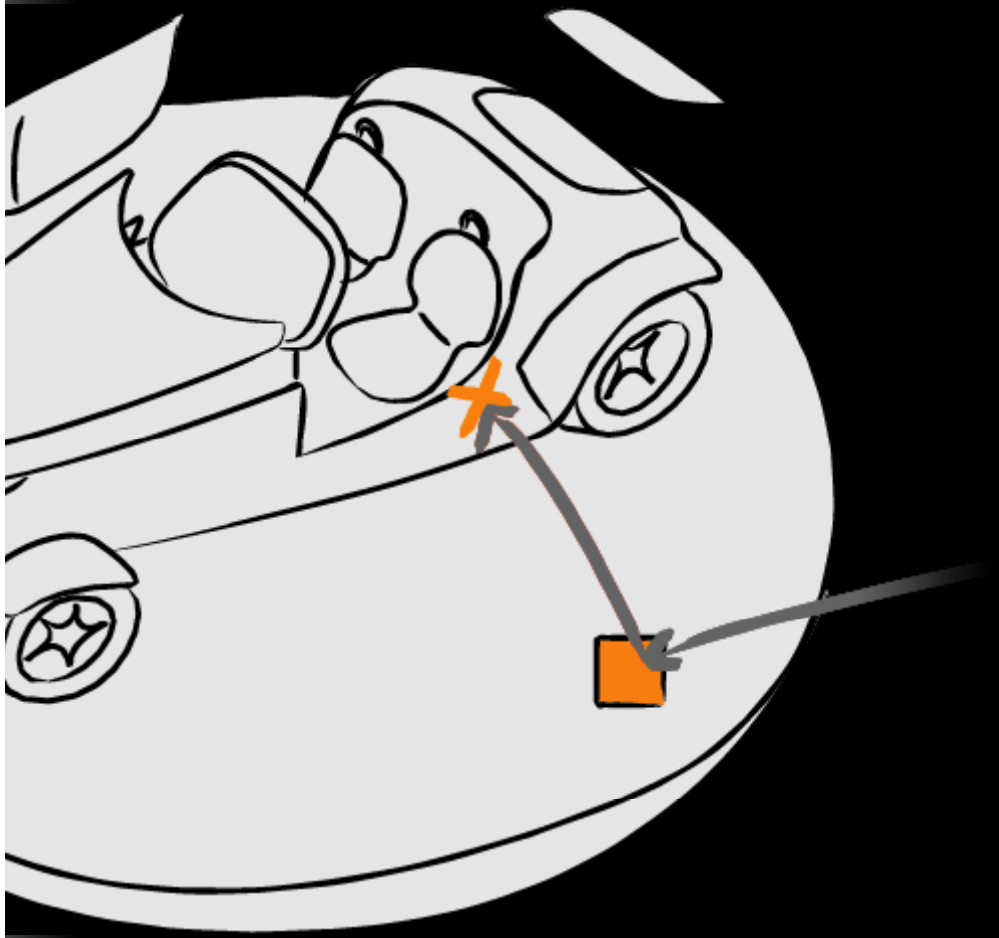
- Initial computation
  - When a object is picked to manipulate it
  - Reflections within the rest not considering the manipulated part

Rendering Approach (cont'd)



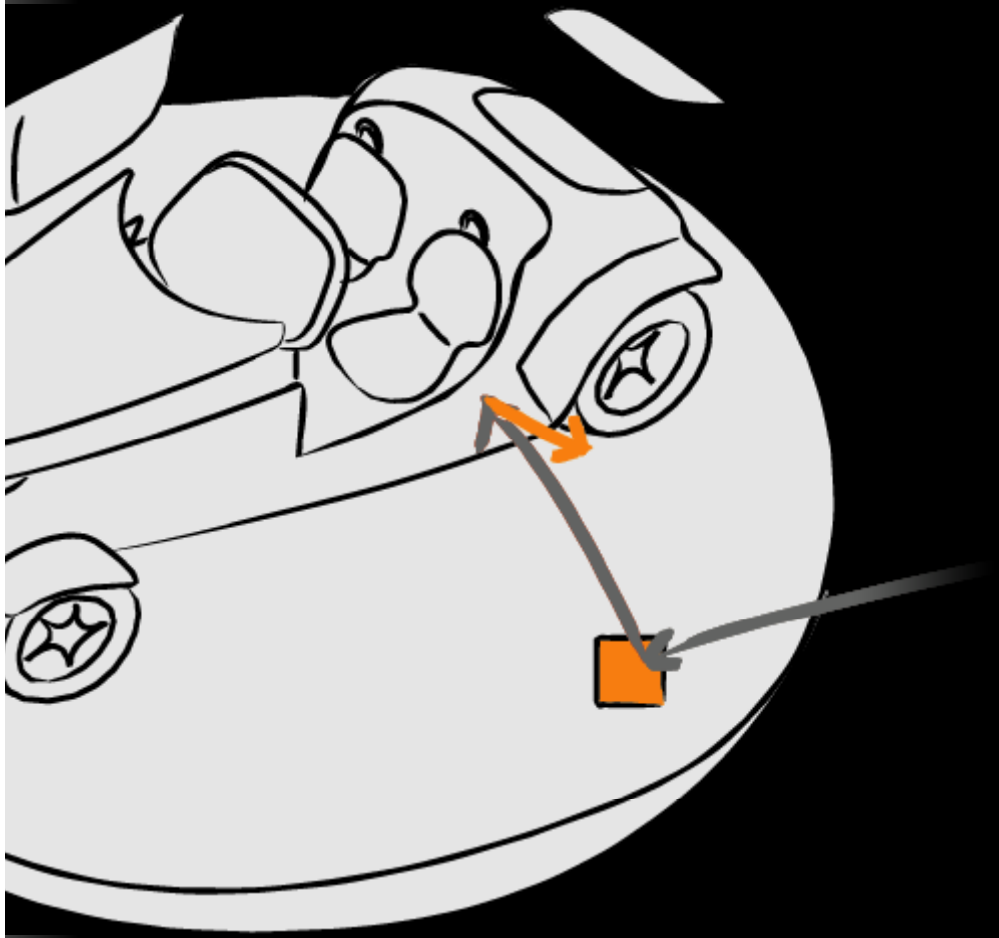
- Initial computation
  - When a object is picked to manipulate it
  - Reflections within the rest not considering the manipulated part
  - Properties of reflected surface stored in G-buffers

Rendering Approach (cont'd)



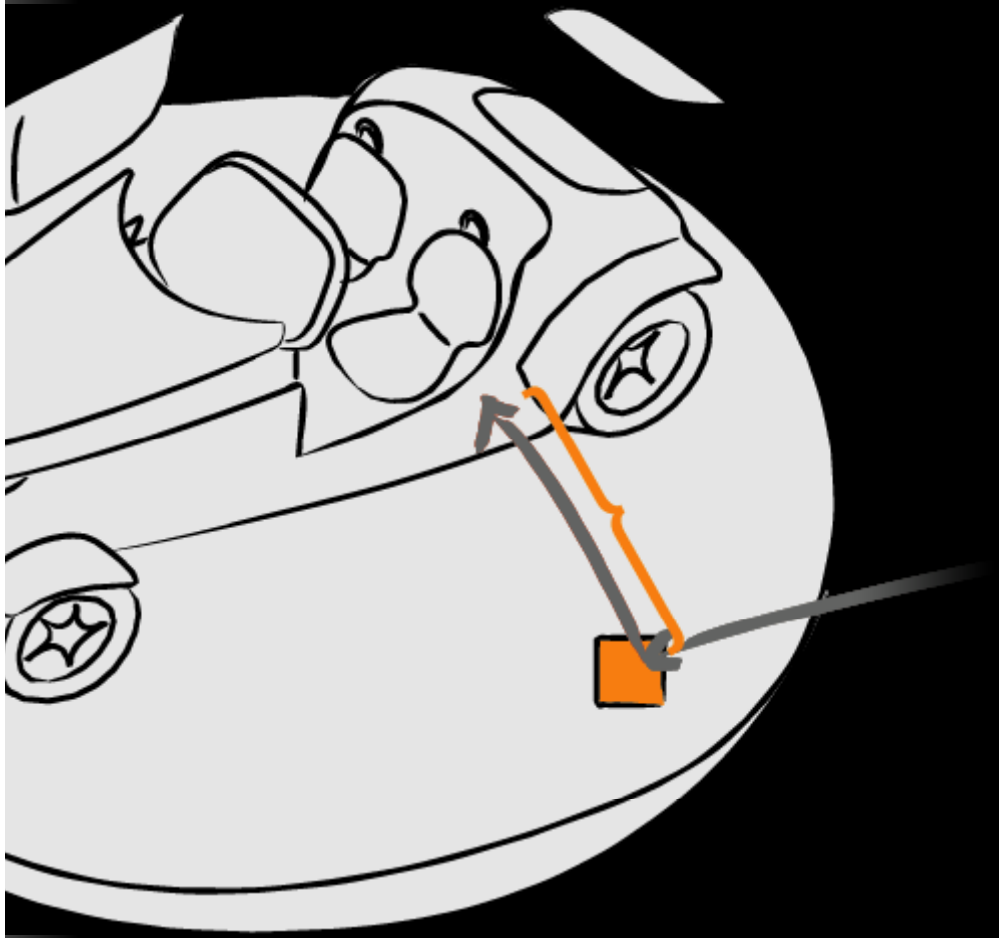
- Initial computation
  - When a object is picked to manipulate it
  - Reflections within the rest not considering the manipulated part
  - Properties of reflected surface stored in G-buffers
    - Position

Rendering Approach (cont'd)



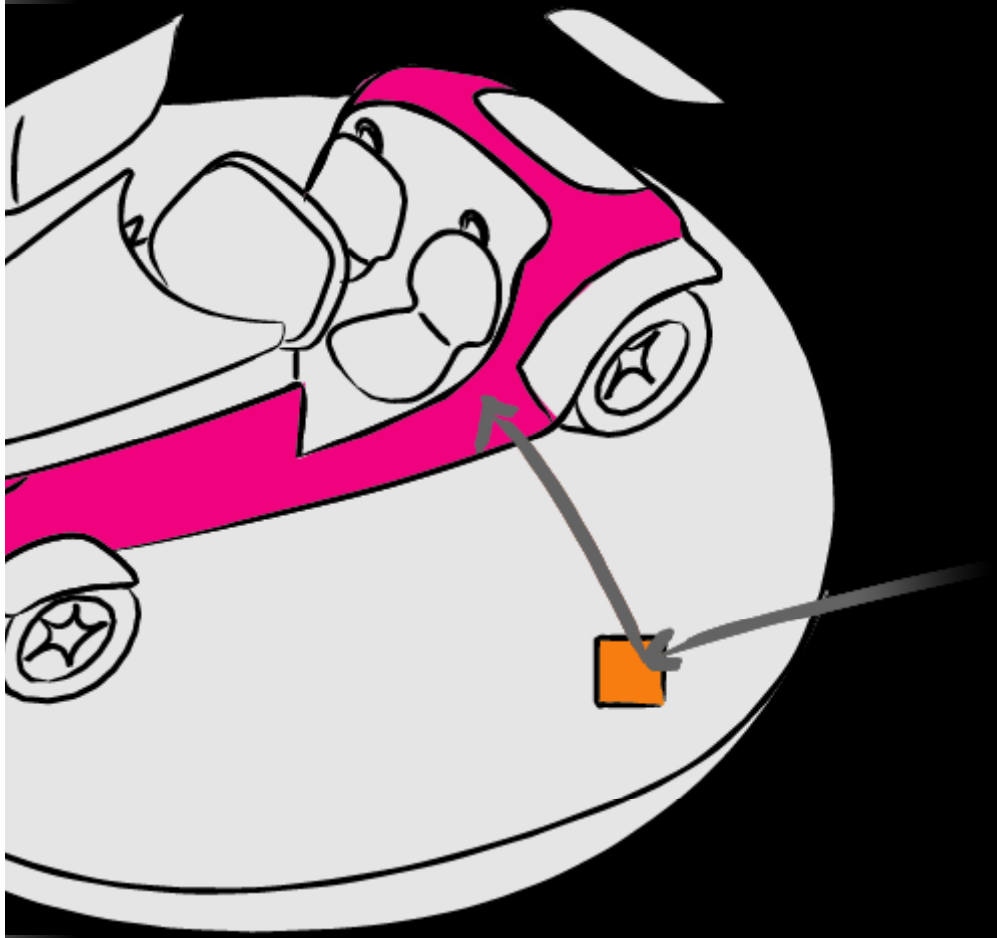
- Initial computation
  - When a object is picked to manipulate it
  - Reflections within the rest not considering the manipulated part
  - Properties of reflected surface stored in G-buffers
    - Position
    - Normal

Rendering Approach (cont'd)



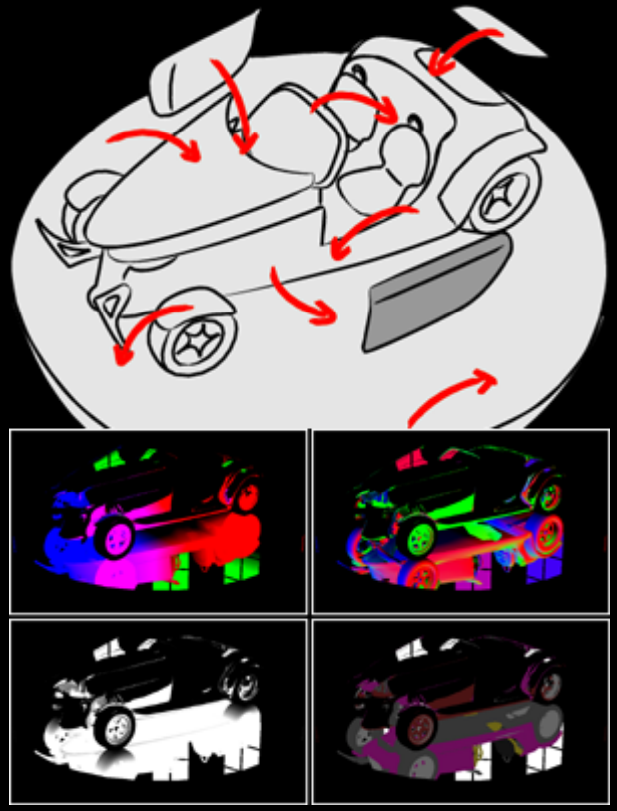
- Initial computation
  - When a object is picked to manipulate it
  - Reflections within the rest not considering the manipulated part
  - Properties of reflected surface stored in G-buffers
    - Position
    - Normal
    - Distance

Rendering Approach (cont'd)



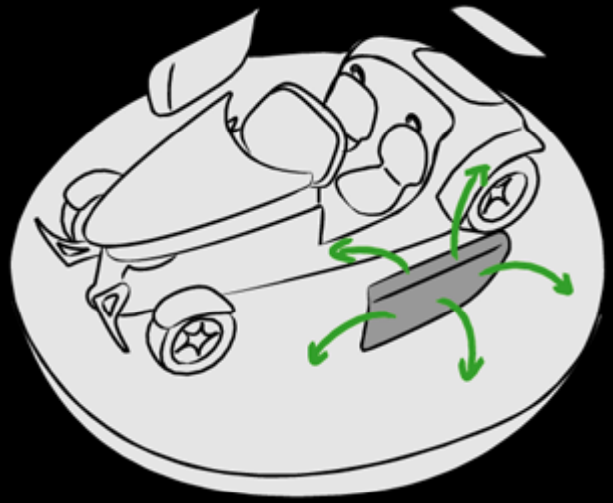
- Initial computation
  - When a object is picked to manipulate it
  - Reflections within the rest not considering the manipulated part
  - Properties of reflected surface stored in G-buffers
    - Position
    - Normal
    - Distance
    - Material

Rendering Approach (cont'd)



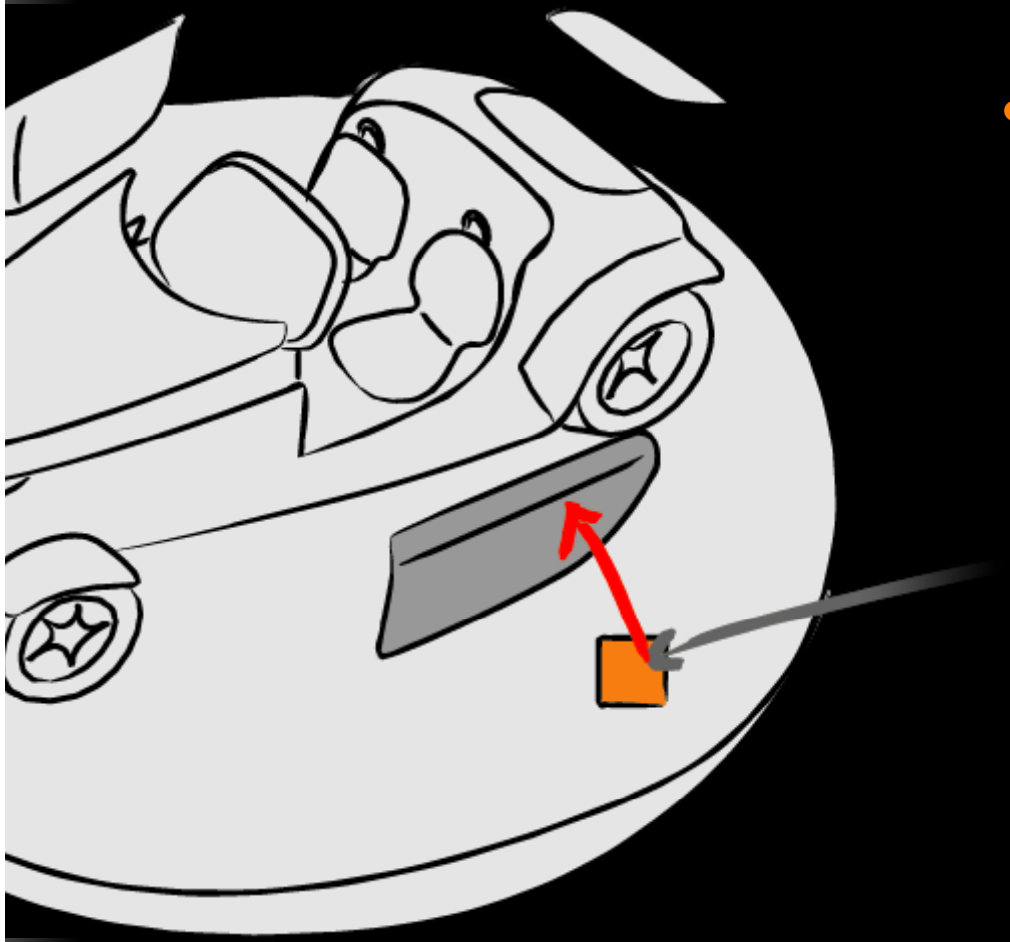
- Initial computation
  - When a object is picked to manipulate it
  - Reflections within the rest not considering the manipulated part
  - Properties of reflected surface stored in G-buffers
    - Position
    - Normal
    - Distance
    - Material

Rendering Approach (cont'd)



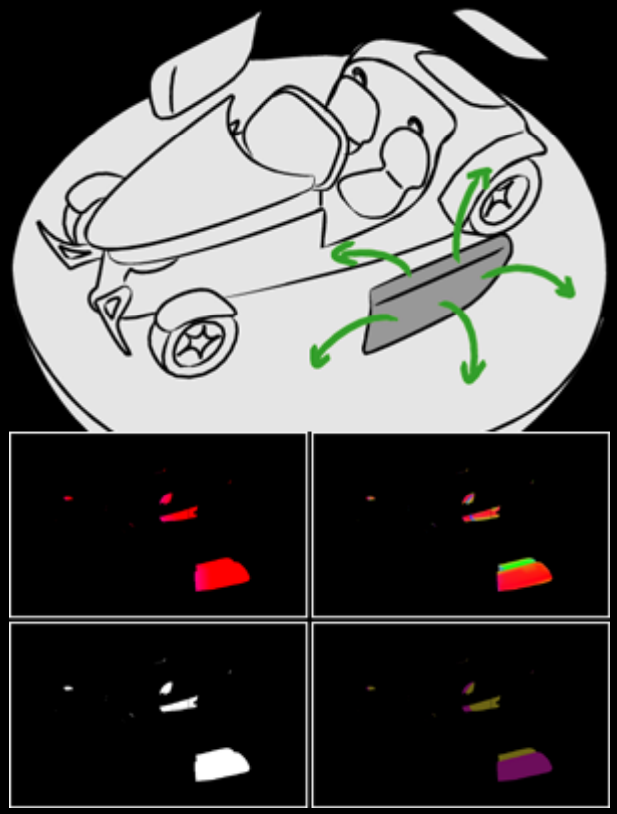
- Updating reflections of the door in the rest
  - Ray intersection test with door only
  - Much cheaper than testing against the whole scene!

Rendering Approach (cont'd)



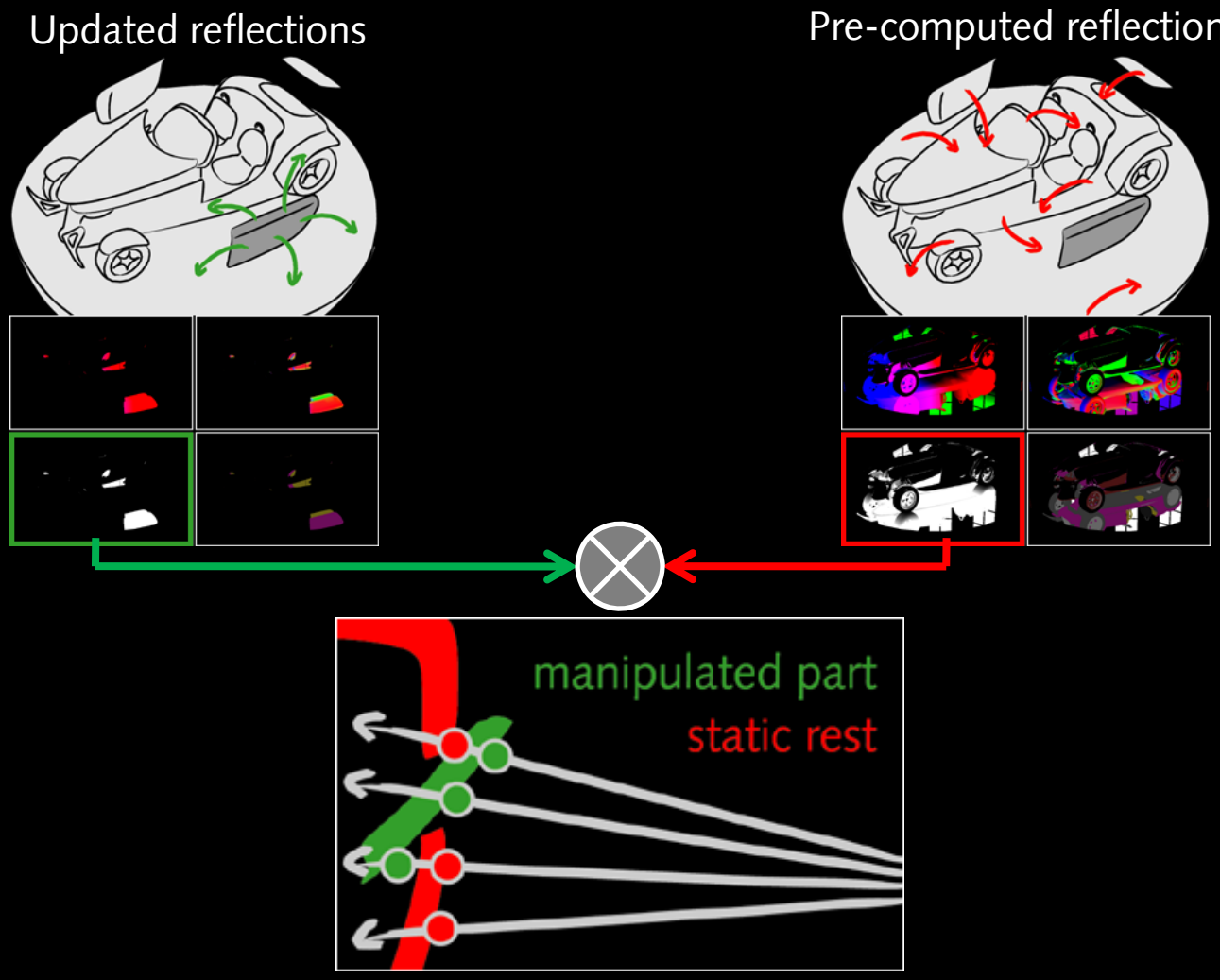
- Updating reflections of the door in the rest
  - Ray intersection test with door only
  - Much cheaper than testing against the whole scene!
  - Intersection point properties stored in G-buffers

Rendering Approach (cont'd)



- Updating reflections of the door in the rest
  - Ray intersection test with door only
  - Much cheaper than testing against the whole scene!
  - Intersection point properties stored in G-buffers

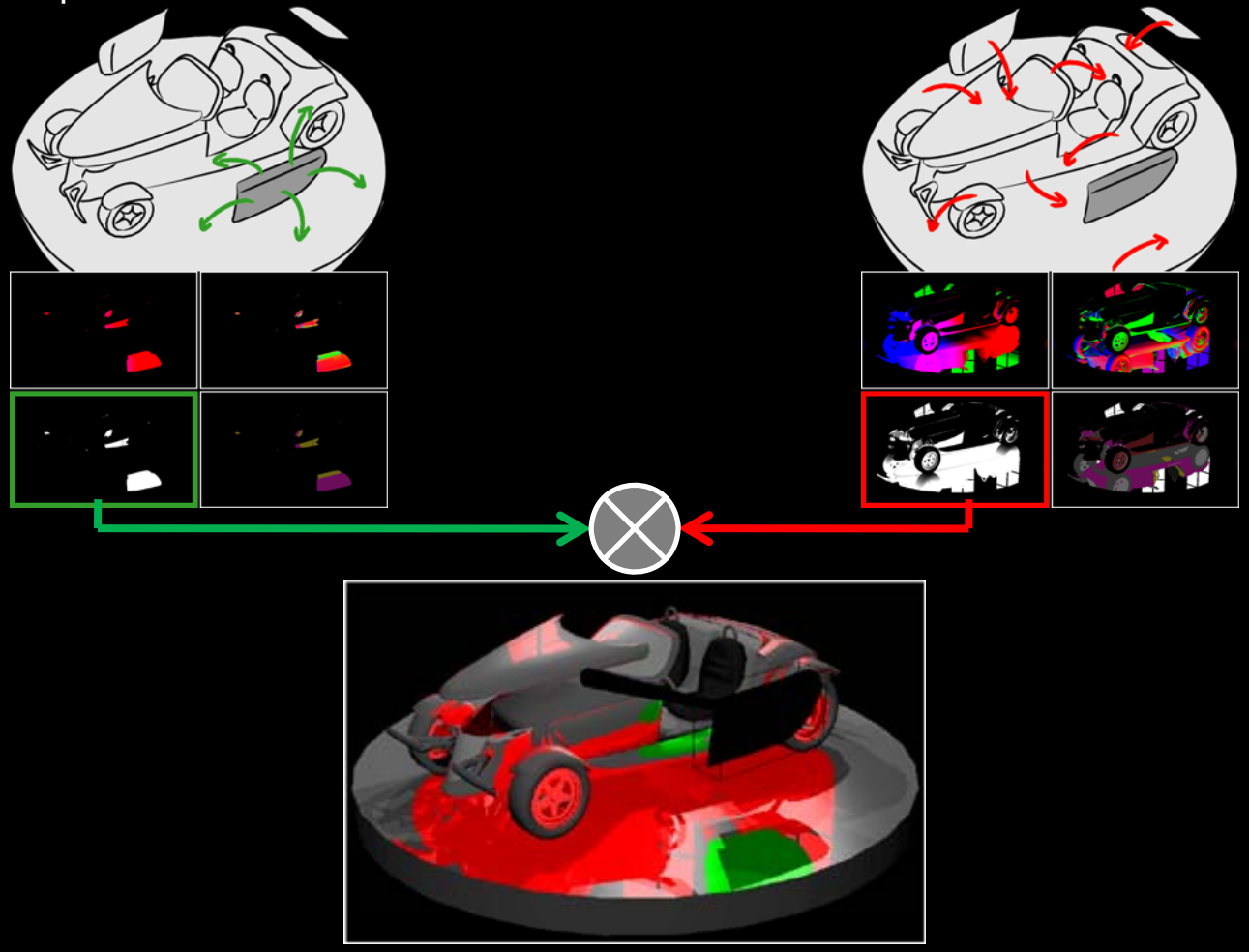
Rendering Approach (cont'd)



Rendering Approach (cont'd)

Updated reflections

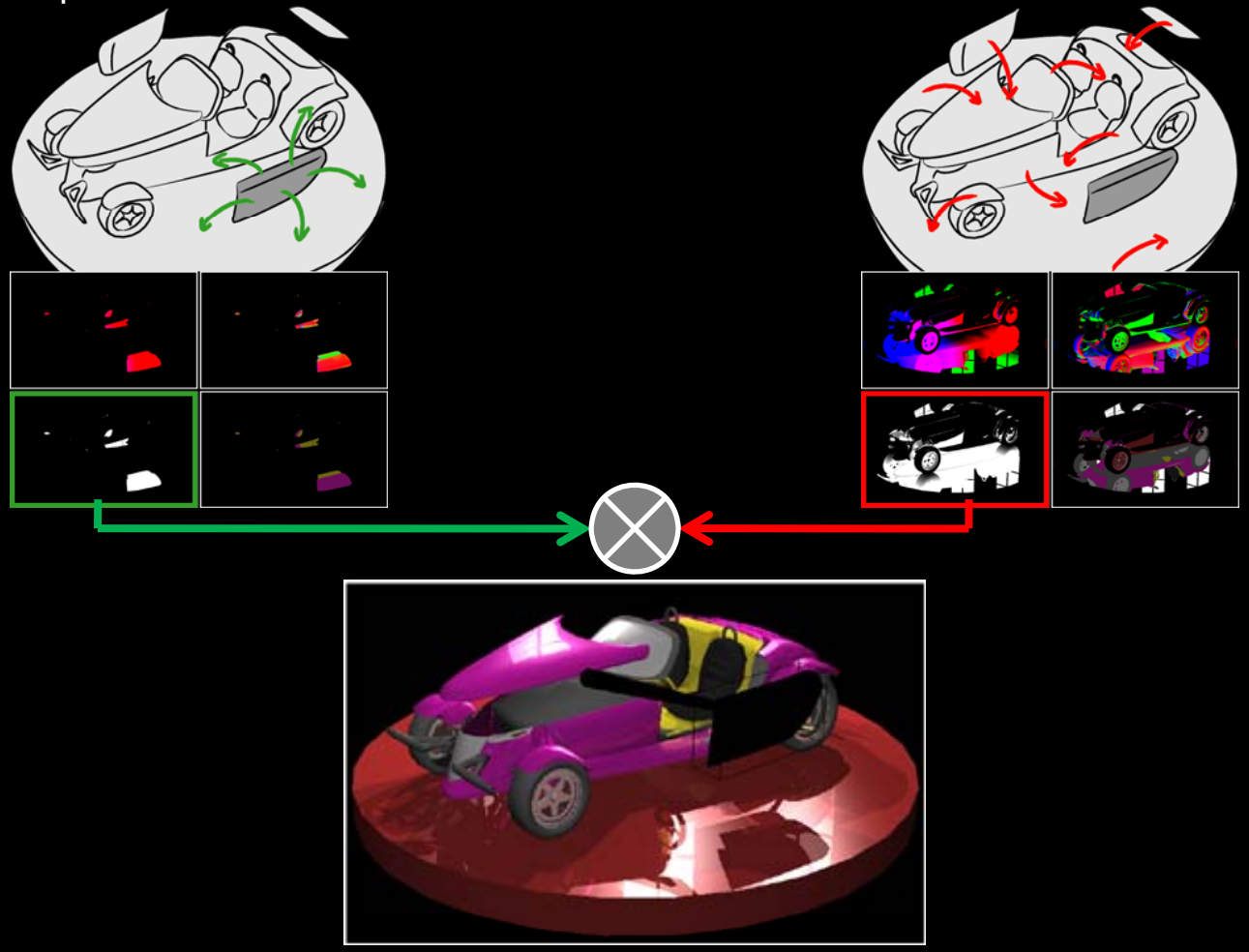
Pre-computed reflections



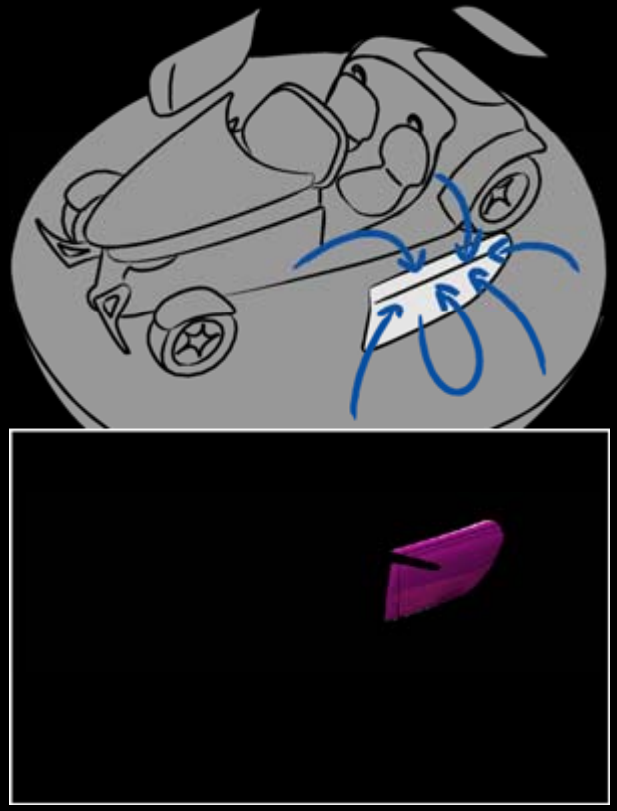
Rendering Approach (cont'd)

Updated reflections

Pre-computed reflections

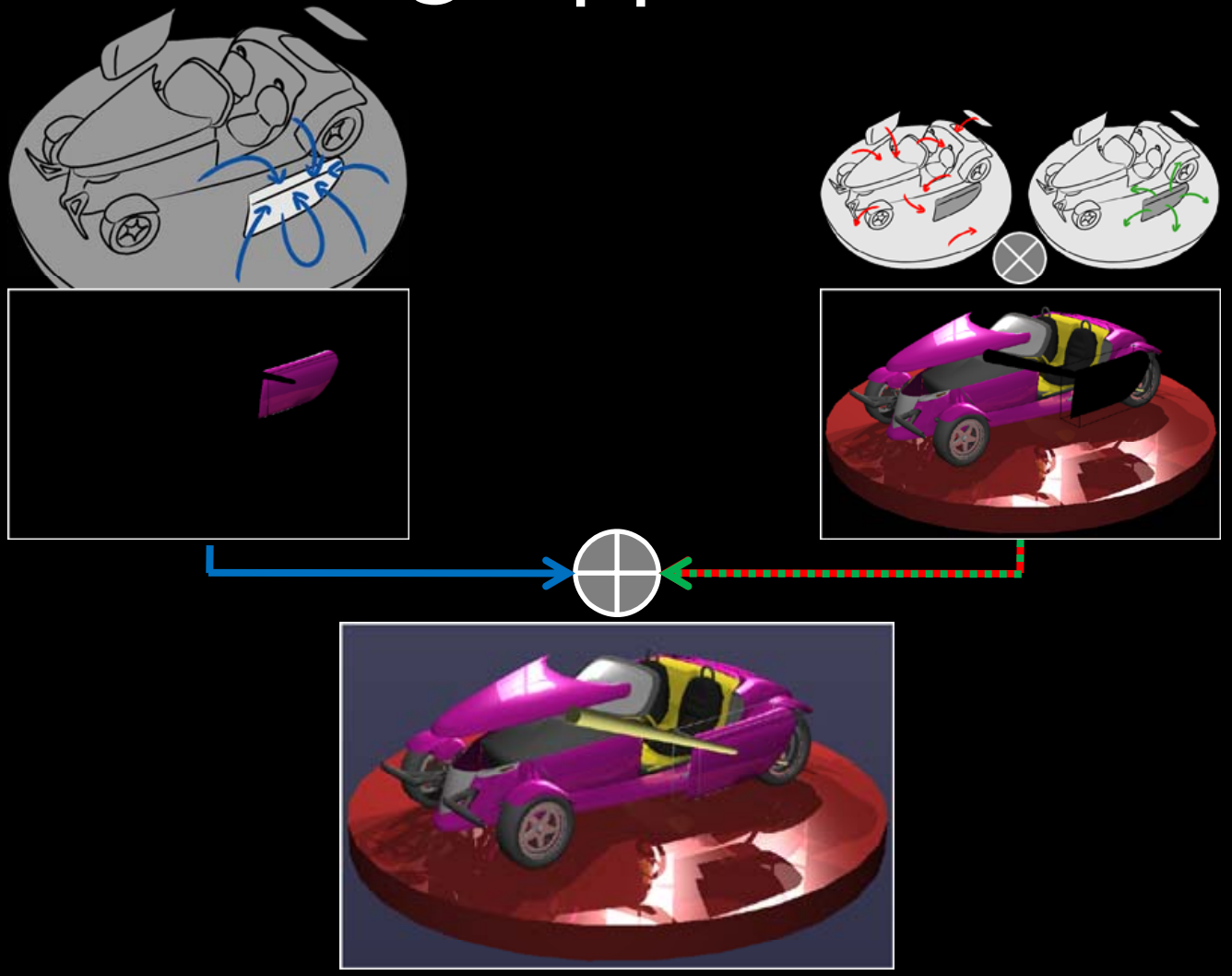


Rendering Approach (cont'd)

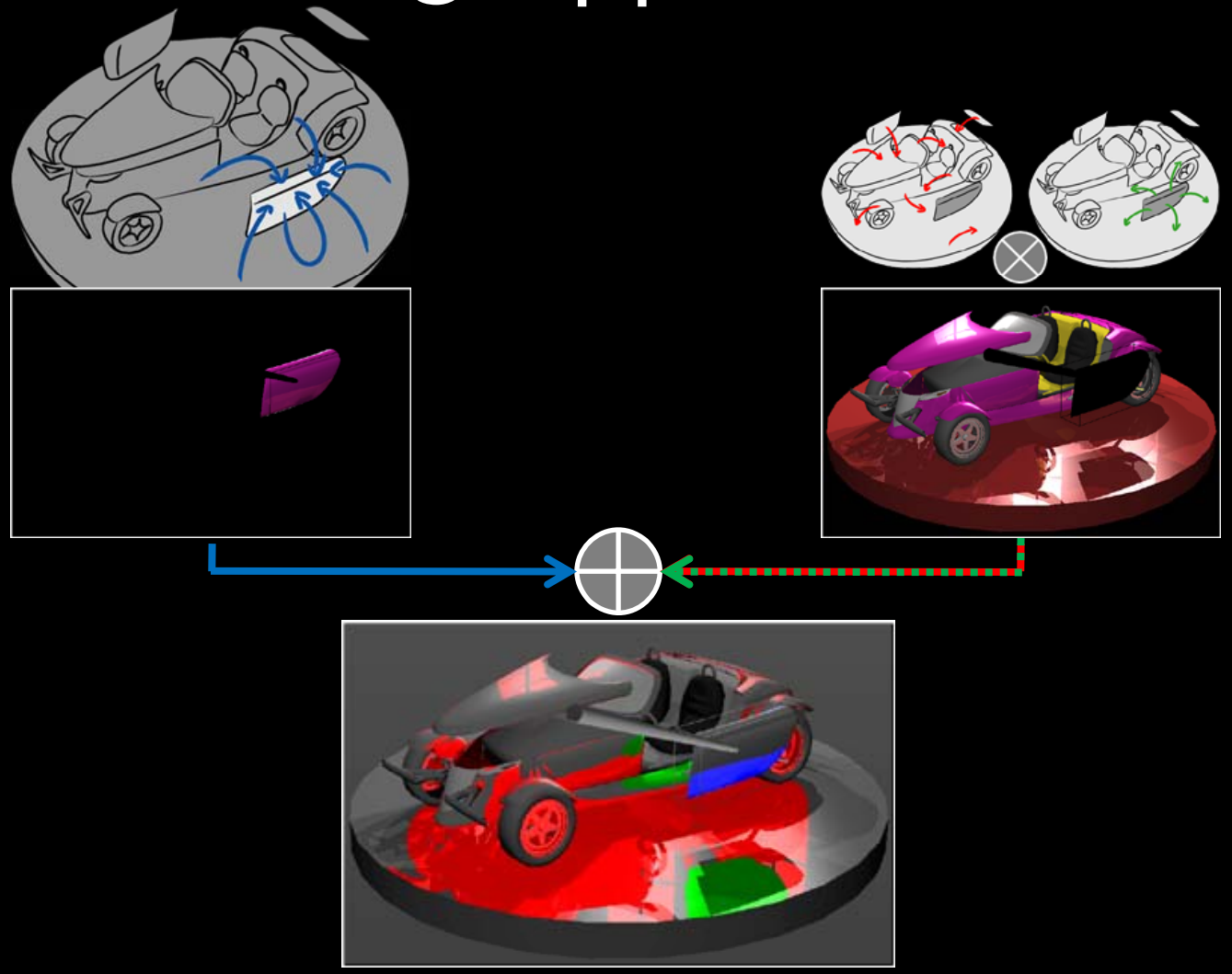


- Computation of reflections in the door
  - Regular ray tracing for the involved pixels
  - Reflections are added to local lighting
  - Rendered to a G-buffer

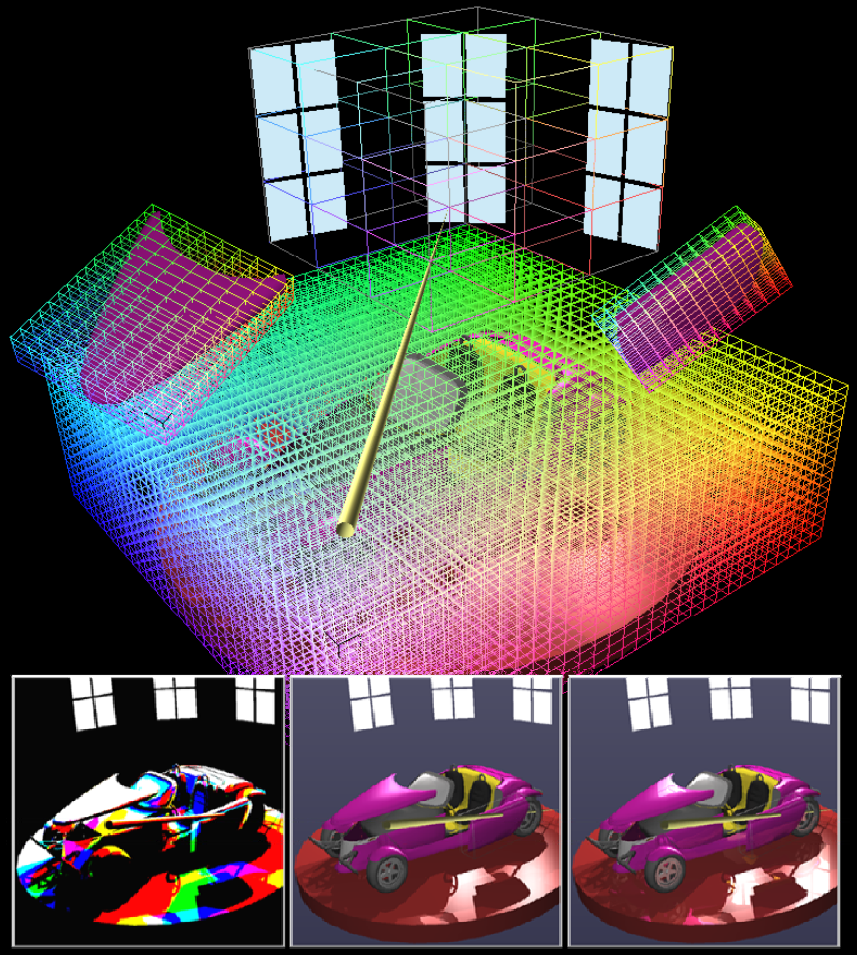
Rendering Approach (cont'd)



Rendering Approach (cont'd)

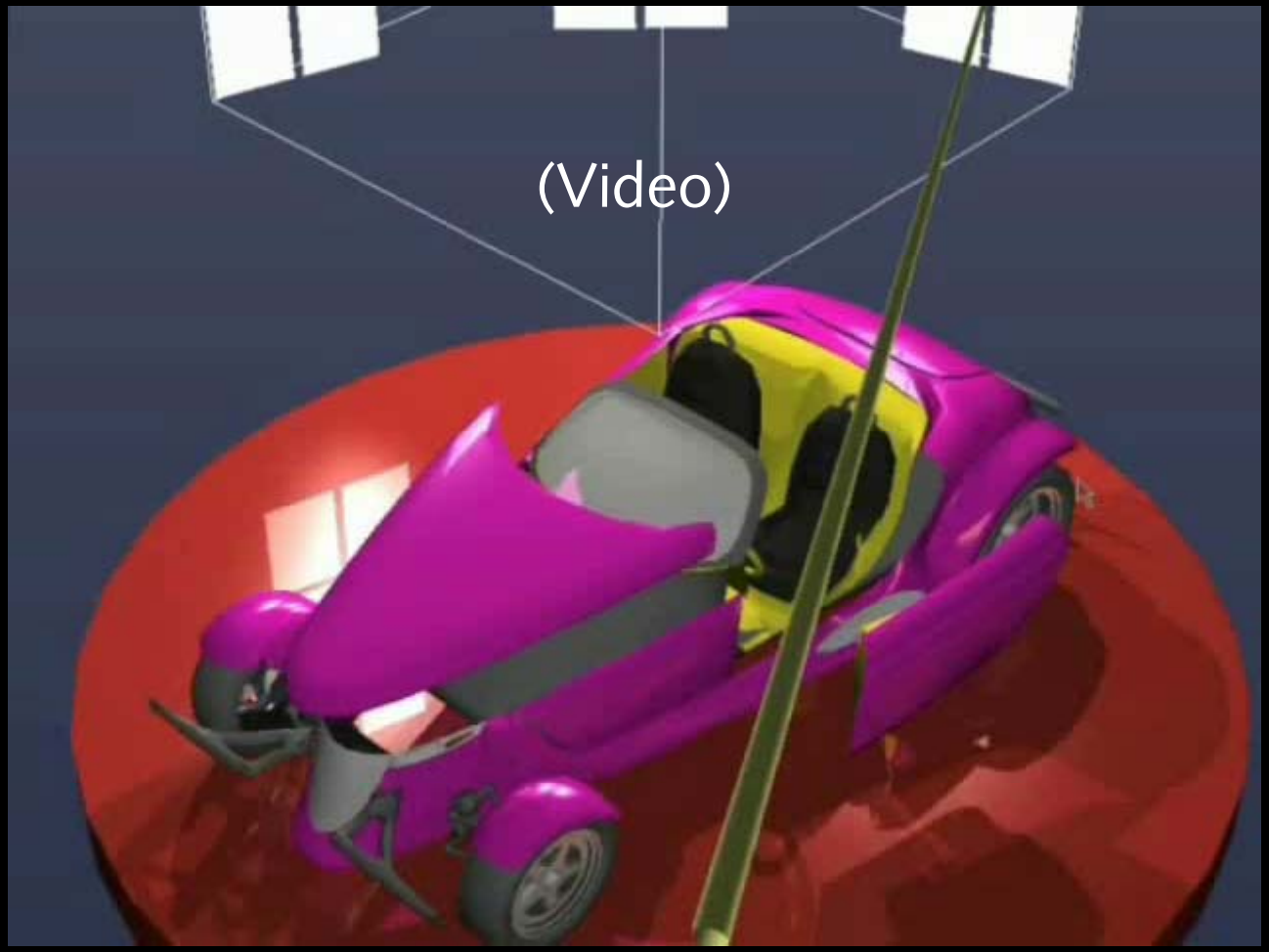


Example Implementation



- Integration into in-house developed framework
  - Navigation, selection and manipulation of objects
  - Uniform grid per object
  - Increasing performance
    - Shadow mapping
    - Rasterization of primary rays
    - GPU ray tracing for first order reflections
- Purpose
  - Proof of concept
  - Performance evaluation

Implementation (cont'd)



Results and Discussion

- Significantly higher frame rates during object manipulation
  - Speedup up to 10
- Independent of the spatial data structure
- No artifacts
- No decrease in visual quality

- General approach
  - Extends to refraction and shadow rays
  - Applies to further ray generations
    - Speedup probably less significant
    - Large memory overhead
  - Multi-frame rate ray tracing
- Future work!

Thanks for your attention!

Further information:

http://www.uni-weimar.de/medien/VR

E-mail:

daniel.kurz@medien.uni-weimar.de

christopher.lux@medien.uni-weimar.de

jan.springer@medien.uni-weimar.de

bernd.froehlich@medien.uni-weimar.de