Evaluation of rendering optimizations for virtual reality applications in Vulkan

Paul Preißner

Oct 11 2019

Kickoff: Master Informatics: Games Engineering

Supervisor(s): M.Sc. Sven Liedtke

in collaboration with RTG Echtzeitgraphik GmbH

Introduction / Motivation

- Collaboration with RTG Echtzeitgraphik GmbH (@gate)
- Hot topics: Vulkan and Virtual Reality
- High performance rendering required for many applications/games
- Available material on optimization in Vulkan rather scarce, material on optimization for VR often basic
- **Especially in enterprise**: custom solutions wanted, licensed engines may be a problem

Problem Description: Issues

- Basic premise: “how to make rendering fast?”
- many generic approaches exist, but few dive into Vulkan specifics
  → which potential approaches are suitable or even ideal for VR?
  → which can be combined for greater impact?
  → which are specifically meant for VR?
  → how do they impact performance for a large, complex scene?

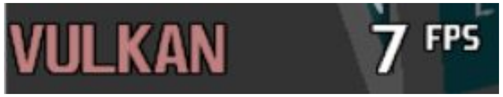
Related Work (selected)

- Vulkan community contributions
  - A. Kapoulkine. *Niagara*. 2018
  - S. Willems. *Vulkan C++ examples and demos*. 2015-2019
- N. Whiting. *Oculus Connect 4 | The Road to Shipping: Technical Postmortem for Robo Recall: Superfrustum culling*. 2017. (+ C. Everitt)
- C. Chandrasekaran, F. Giesen, K. Kuah. *Software Occlusion Culling*. 2013.
- J. Hasselgren, M. Andersson, T. Akenine-Möller. *Masked Software Occlusion Culling*. 2016.
- H. Shen. *Scientific Visualization: Systems & Techniques (Visibility Culling)*. 2001.
- R. Palandri, S. Green. *Hybrid Mono Rendering*. 2016.

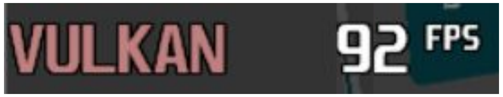
Goals of this Thesis

- in-house render engine as foundation
- implementation of several optimizations for rendering & VR
    - pre-render input reduction
    - render effort reduction
- gain insight into performance impact through benchmarking and evaluation
- recommendations based on these results

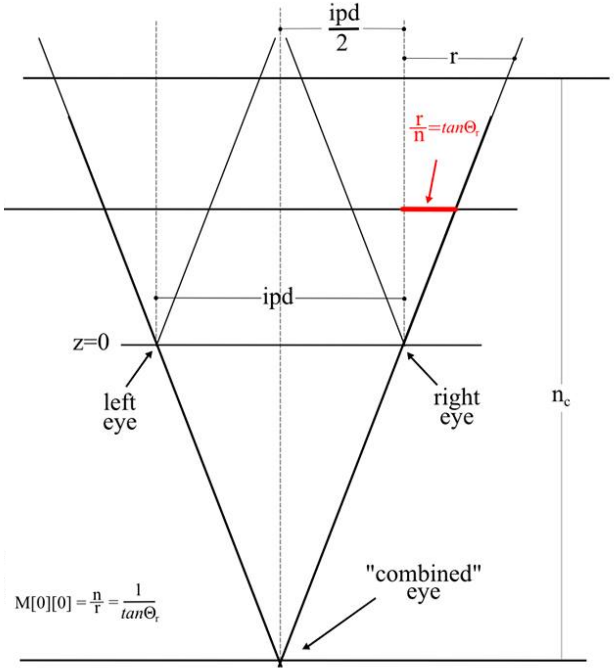
- ideal showcase: complex, high object & poly count scene goes from slideshow to realtime VR


VULKAN 7 FPS

→


VULKAN 92 FPS

Proposed Work / Approach

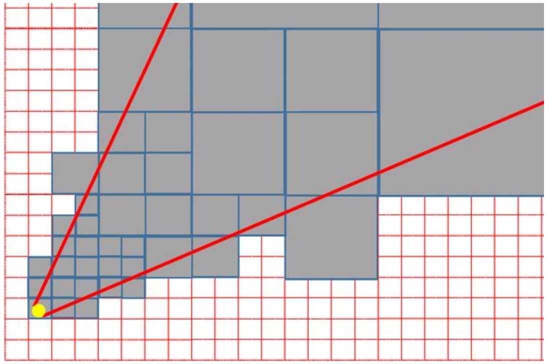


$\frac{ipd}{2}$
r
$\frac{r}{n}=tan\Theta_r$
ipd
z=0
left eye
right eye
n_c
"combined" eye
$M[0][0]=\frac{n}{r}=\frac{1}{tan\Theta_r}$

Cass Everitt 2015

- in RTG's Tachyon renderer, integrate
  - hw-multiview stereo rendering ☑
  - hierarchical (super)frustum culling □
  - (masked) software occlusion culling □
  - monoscopic far-field rendering +
  - round robin VR occlusion +
- benchmark individual & combined performance & resource impact using two test environments
  - simple high primitive count scene
  - complex "real-world" showcase scene (provided from real enterprise project)
- outline unimplemented/further approaches for reference

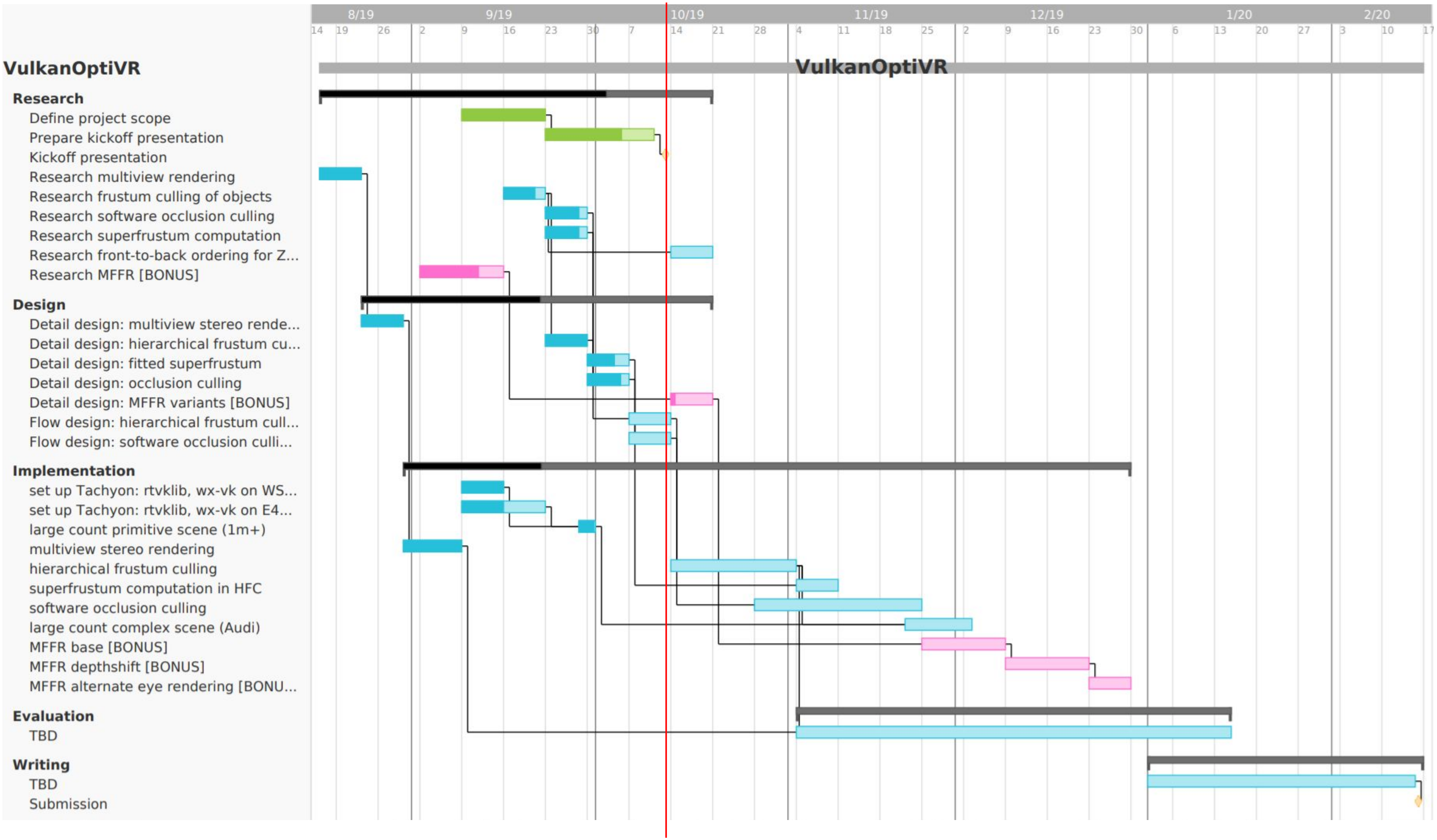
Oscar Martinez Rubi 2015



Discussion of Potential Issues

- Risk: implementation difficulty
  - not a Vulkan/graphics programming expert
- Risk: unexpected performance stagnation/regression due to unforeseen limitations (bad test scene, flawed implementation, API quirks, etc)
- Enterprise involvement, some code may be under NDA

Time Line (Aug 15 - Feb 15)



Q&A