Hierarchical and Object-Oriented GPU Programming

Matthias Raspe, Guido Lorenz, Stephan Palmer

Computer Graphics Working Group, Institute for Computational Visualistics
University of Koblenz-Landau, Campus Koblenz, Germany

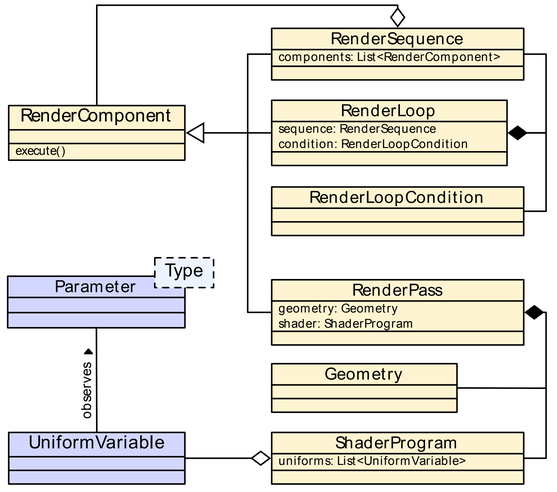
Motivation

Using commodity graphics hardware (GPU) for **general purpose computations** has become commonplace [1] and is beneficial for many applications.[2] However, implementing **algorithms and workflows on GPUs for real-world scenarios** using only shading languages like GLSL or low-level APIs as CUDA is **intricate** and often **lacks flexibility**.

Thus, we propose the software framework "**Cascada**" that allows for:

- **straightforward** implementation of complex workflows
- **re-use** of individual components
- **combination of GPU and CPU** implementations, and
- **simple integration** into host applications

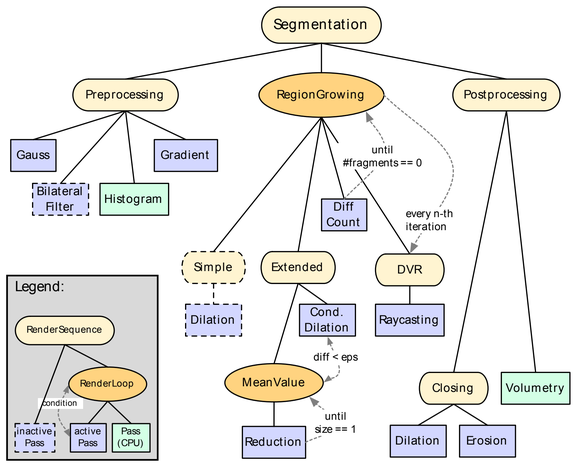
Our Approach



- **Hierarchical representation** of rendering operations
- **Atomic components** (*RenderPass*) implement GPU or CPU computations
- **Complex algorithms** can then be created **by concatenation** (*RenderSequence*) or **loops** (*RenderLoop*)
- Parameter sets provide **connectivity between host** and individual **components**

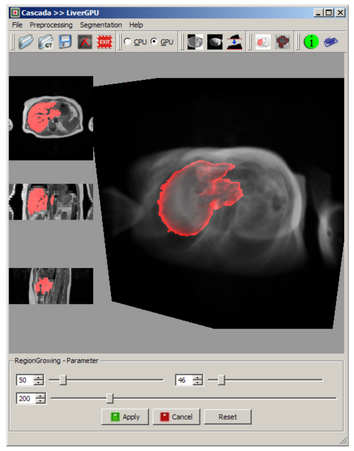
Example

Cascada has been developed in the context of **medical volume processing and direct visualization**. Some example workflow for the **segmentation of volume data** is depicted as graph and screenshot:



Some steps are standard CPU implementations to show the **interchangeability of components**

Direct volume rendering (DVR) can be integrated to **visualize results during computation**, with **negligible overhead** for GPU computations



Results and Conclusion

- Run-time overhead of **less than 10%** is outweighed by simplified implementation and maintenance
- Our approach is **not limited to computer graphics applications** as previous work [3,4]
- Cascada's **workflow abstraction** allows for an easy **integration** of state of the art low-level systems such as **CUDA or CTM**
- For the future, a **more general support for different data types** is of main interest

References

[1] Owens J.D., Luebke D., Govindaraju N., Harris M., Krueger J., Lefohn A.E., Purcell T.: *A Survey of General-Purpose Computation on Graphics Hardware,* Computer Graphics Forum, 2007; 26(1); pp. 80-113
[2] Raspe M., Lorenz G., Müller S.: *Evaluating the Performance of Processing Medical Volume Data on Graphics Hardware,* Bildverarbeitung für die Medizin 2008, Springer; pp. 427-431
[3] Trapp M., Döllner J.: *Automated Combination of Real-Time Shader Programs*, Proceedings of Eurographics, 2007, ACM; pp. 53-56
[4] McGuire M., Stathis G., Pfister H., Krishnamurthi S.: *Abstract Shade Trees,* Proceedings of the 2006 symposium on Interactive 3D graphics and games, ACM, 2006, pp. 79-86