To produce compelling images, a real-time renderer is responsible for simulating many real-world visual effects. These effects range from modeling the material properties of surfaces to evaluating complex lighting conditions, to the animation of surfaces. Given the diversity of scenes in modern graphics applications such as games, Industrial real-time renderers contain hundreds of thousands of lines of code that define these shading effects. Like any complex software system, it is desirable for these code bases to be implemented in a flexible and extensible framework to enable productive use and development. Additionally, real-time renderers must meet extreme performance requirements, which requires using the GPU graphics pipeline efficiently for the drawing tasks. As a result, the core logic for simulating various visual appearances must be written as shader code that runs at different stages of the GPU graphics pipeline. While traditional object-oriented programming principles work well for abstracting shading system concepts into an extensible and flexible framework, their current implementations in existing programming languages are not sufficient for generating high-performance GPU code. Specifically, renderers must make different trade-offs between shader compilation time and execution efficiency by implementing dispatch of shading features differently – either via dynamic control flows in shader code, or via static shader specialization. Also, renderers must minimize and efficiently manage CPU-GPU communication. To meet this challenge, this thesis contributes the design of the Slang shading language and its compilation system, which adopts object-oriented programming concepts under GPU performance constraints. We address introduce a design pattern called shader components, which can be implemented with the enhanced language mechanisms in Slang, to decouple shading logic from the specific implementation choices of GPU code dispatching or CPU-GPU communication mechanisms. We demonstrate the effectiveness the Slang shader compilation system in a reference renderer implementation that is modular, flexible, extensible and fast. We further evaluate our work by adopting Slang into an existing research rendering framework to achieve higher performance with a more maintainable code base.