2.2 The Application Stage

The developer has full control over what happens in the application stage, since it usually executes on the CPU. Therefore, the developer can entirely determine the implementation and can later modify it in order to improve performance. Changes here can also affect the performance of subsequent stages. For example, an application stage algorithm or setting could decrease the number of triangles to be rendered.

All this said, some application work can be performed by the GPU, using a separate mode called a *compute shader*. This mode treats the GPU as a highly parallel general processor, ignoring its special functionality meant specifically for rendering graphics. At the end of the application stage, the geometry to be rendered is fed to the geometry processing stage. These are the *rendering primitives*, i.e., points, lines, and triangles, that might eventually end up on the screen (or whatever output device is being used). This is the most important task of the application stage.

A consequence of the software-based implementation of this stage is that it is not divided into substages, as are the geometry processing, rasterization, and pixel processing stages. [1] However, to increase performance, this stage is often executed in parallel on several processor cores. In CPU design, this is called a *superscalar* construction, since it is able to execute several processes at the same time in the same stage. Section 18.5 presents various methods for using multiple processor cores.

One process commonly implemented in this stage is *collision detection*. After a collision is detected between two objects, a response may be generated and sent back to the colliding objects, as well as to a force feedback device. The application stage is also the place to take care of input from other sources, such as the keyboard, the mouse, or a head-mounted display. Depending on this input, several different kinds of actions may be taken. Acceleration algorithms, such as particular culling algorithms (Chapter 19), are also implemented here, along with whatever else the rest of the pipeline cannot handle.