Representations of Rotations

Several methods may be used to represent rotations: rotation matrices, axis/angle (which may be represented as unit quaternions), and Euler angles. The table below highlights some pros and cons for these representations.

	Rotation Matrices	*Axis/Angle*	*Euler Angles*
Size	9 numbers	4 numbers	3 numbers
Composition	Easy (multiplication)	Easy in quaternion representation (multiplication)	?
Normalization after round-off errors in composition	Hard	Easy in quaternion representation (normalize length)	(?)
Interpolation	?	Visually well functioning methods exist in quaternion representation (slerp, squad)	Methods not visually pleasing
Intuitive?	No	Yes	Yes
Caveats		Negation of axis and angle gives same rotation	Non-uniqueness of representation, gimbal lock

Note that the above table discusses representations of rotations at the application programming level. For use on the GPU, all rotations must be expressed as a matrix in the end.

There exist formulas for converting between the various representations, i.e., axis/angle $\Leftrightarrow$ rotation matrix $\Leftrightarrow$ Euler angles. The book contains axis/angle (quaternion) $\Rightarrow$ rotation matrix (p. 279, 236, 229), and rotation matrix $\Leftarrow$ Euler angles is obvious (from the definition of Euler angles and pp. 222–224). The rest can be found in e.g. *Real Time Rendering* by Akenine-Möller, Haines, and Hoffman.