= Fast inverse square root =

Fast inverse square root (sometimes referred to as Fast InvSqrt ( ) or by the hexadecimal constant 0x5f3759df) is a method of calculating x ? ½ , the reciprocal (or multiplicative inverse) of a square root for a 32 @-@ bit floating point number in IEEE 754 floating point format . The algorithm was probably developed at Silicon Graphics in the early 1990s , and an implementation appeared in 1999 in the Quake III Arena source code , but the method did not appear on public forums such as Usenet until 2002 or 2003 . (There is a discussion on the Chinese developer forum CSDN back in 2000 .) At the time , the primary advantage of the algorithm came from avoiding computationally expensive floating point operations in favor of integer operations . Inverse square roots are used to compute angles of incidence and reflection for lighting and shading in computer graphics .

The algorithm accepts a 32 @-@ bit floating point number as the input and stores a halved value for later use . Then , treating the bits representing the floating point number as a 32 @-@ bit integer , a logical shift right of one bit is performed and the result subtracted from the magic number 0x5f3759df . This is the first approximation of the inverse square root of the input . Treating the bits again as floating point it runs one iteration of Newton 's method to return a more precise approximation . This computes an approximation of the inverse square root of a floating point number approximately four times faster than floating point division .

The algorithm was originally attributed to John Carmack , but an investigation showed that the code had deeper roots in both the hardware and software side of computer graphics . Adjustments and alterations passed through both Silicon Graphics and 3dfx Interactive , with Gary Tarolli 's implementation for the SGI Indigo as the earliest known use . It is not known how the constant was originally derived , though investigation has shed some light on possible methods .

= = Motivation = =

The inverse square root of a floating point number is used in calculating a normalized vector . Since a 3D graphics program uses these normalized vectors to determine lighting and reflection , millions of these calculations must be done per second . Before the creation of specialized hardware to handle transform and lighting , software computations could be slow . Specifically , when the code was developed in the early 1990s , most floating point processing power lagged behind the speed of integer processing .

To normalize a vector , the length of the vector is determined by calculating its Euclidean norm : the square root of the sum of squares of the vector components . When each component of the vector is divided by that length , the new vector will be a unit vector pointing in the same direction .

<formula> is the Euclidean norm of the vector , analogous to the calculation of the Euclidean distance between two points in Euclidean space .

<formula> is the normalized (unit) vector . Using <formula> to represent <formula> ,

<formula> , which relates the unit vector to the inverse square root of the distance components .

Quake III Arena used the fast inverse square root algorithm to speed graphics processing unit computation , but the algorithm has since been implemented in some dedicated hardware vertex shaders using field @-@ programmable gate arrays (FPGA) .

= = Overview of the code = =

The following code is the fast inverse square root implementation from Quake III Arena , stripped of C preprocessor directives , but including the exact original comment text :

In order to determine the inverse square root , an approximation for <formula> would be determined by the software , then some numerical method would revise that approximation until it came within an acceptable error range of the actual result . Common software methods in the early 1990s drew a first approximation from a lookup table . This bit of code proved faster than table lookups and approximately four times faster than regular floating point division . Some loss of precision occurred , but was offset by the significant gains in performance . The algorithm was designed with the IEEE

754 @-@ 1985 32 @-@ bit floating point specification in mind , but investigation from Chris Lomont and later Charles McEniry showed that it could be implemented in other floating point specifications .

The advantages in speed offered by the fast inverse square root kludge came from treating the longword containing the floating point number as an integer then subtracting it from a specific constant , 0x5f3759df . The purpose of the constant is not immediately clear to someone viewing the code , so , like other such constants found in code , it is often called a magic number . This integer subtraction and bit shift results in a longword which when treated as a floating point number is a rough approximation for the inverse square root of the input number . One iteration of Newton 's method is performed to gain some accuracy , and the code is finished . The algorithm generates reasonably accurate results using a unique first approximation for Newton 's method ; however , it is much slower and less accurate than using the SSE instruction rsqrtss on x86 processors also released in 1999 .

= = = A worked example = = =