
Subject: "better" version of Iscale functions
Posted by [mdelfede](#) on Tue, 01 Apr 2008 21:10:55 GMT
[View Forum Message](#) <> [Reply to Message](#)

In file 'mathutils.cpp' the Iscale...() functions use some floating point math, when assembly is unavailable OR when it's not compatible with intel syntax (I.E. GCC and MinGW). That makes it slow and not showing divide-by-0 errors when third argument is 0. So, here an (IMHO) better version of such functions :

```
#include "Core.h"

// iscale: computes x * y / z.

#ifdef flagGCC
#define __USE_64BIT_MATH__
#endif

NAMESPACE_UPP

int iscale(int x, int y, int z)
{
#ifdef __NOASSEMBLY__
#ifdef __USE_64BIT_MATH__
    return int(x * (double)y / z);
#else
    int64_t res = x;
    res *= y;
    res /= z;
    return (int)res;
#endif
#else
    __asm
    {
        mov eax, [x]
        imul [y]
        idiv [z]
    }
#endif
}

// iscalefloor: computes x * y / z, rounded towards -infty.

int iscalefloor(int x, int y, int z)
{
#ifdef __NOASSEMBLY__
#ifdef __USE_64BIT_MATH__
    return (int)ffloor(x * (double)y / z);
#else
```

```

int64_t res = x;
int64_t mulres = res * y;
res = mulres / z;
if(res * z != mulres)
    res--;
return (int)res;
#endif
#else
__asm
{
    mov eax, [x]
    imul [y]
    idiv [z]
    and edx, edx
    jge __1
    dec eax
__1:
}
#endif
}

```

// iscaleceil: computes $x * y / z$, rounded towards +infy.

```

int iscaleceil(int x, int y, int z)
{
#ifdef __NOASSEMBLY__
#ifndef __USE_64BIT_MATH__
return fceil(x * (double)y / z);
#else
int64_t res = x;
int64_t mulres = res * y;
res = mulres / z;
if(res * z != mulres)
    res++;
return (int)res;
#endif
#else
__asm
{
    mov eax, [x]
    imul [y]
    idiv [z]
    and edx, edx
    jle __1
    inc eax
__1:
}
#endif
}

```

```
}
```

BTW, we could completely drop the assembly code, as-is it's not portable between compilers with greater integer width.

My version is also *not* portable on compilers with 64 bit wide integers, but can be made ok just changing function prototype :

```
int32_t iscale(int32_t x, int32_t y, int32_t z)
```

Leaving so to the compiler the integer width check and warnings.

Attached here the patched 'mathutil.cpp' (NO patched function prototype, as it'll require Core.h patch too).

Ciao

Max

File Attachments

1) [mathutil.cpp](#), downloaded 408 times
