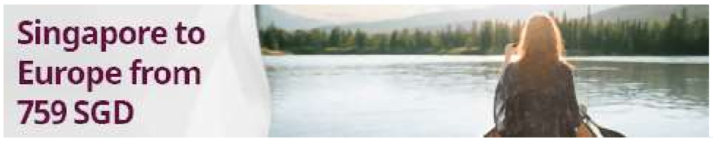
ASM Community



Idea : fast accurate Inertia Tensor for arbitrary meshes

Having scoured the internet for information on this topic, I narrowed by search down to two methods, which I will discuss, and then I'll describe my idea.
Please, feel free to comment - I have no idea what I'm doing :)

Note : both these methods take advantage of the fact that "moments of inertia are additive, since they are integrals".. also note that many variations exist for each of these methods, but they seem to be the backbone in general.

Method 1 : "Atomic Mass method"..
This method requires we "know" the Mass of the Body.
We think of our mesh as a collection of "point masses". More specifically, we distribute the Mass of the Body over the N Vertices of its Mesh.. each Vertex gets attributed to it a piece of the Body's total Mass.
We calculate the Moment of Inertia for each Vertex, adding the results to find the total MOI of the Body, and then build our Inertia Tensor (3x3 matrix) from the total MOI.

Method 2 : "Tetrahedral Integration method"..
This method presumes the Body has constant Density.
We find the Center of Mass of our Body, and immediately shift the Mesh Vertices so the Origin of the Body == Center of Mass (COM).
Next, we start iterating through all the Faces of the Mesh (which are presumed to be Triangles).
For each Triangle, we add a fourth point, being the COM, which creates a Tetrahedron pyramid which projects from the COM to the surface of the Body.
With a little work, we can accurately calculate the Volume of each Tetrahedron, and if we presume to know the Body Density, then we can easily calculate the Mass of each Tetrahedron.
With a bit more work, we can calculate the MOI of each Tetrahedron, and thus the MOI of the entire Body.
Far more accurate than Method 1, but much more work, and much slower.

OK, now for the good stuff.. my idea is based on a puzzle posted a little while ago on this board regarding two Balls of equal mass and size but of different density (ie different Material).
It occured to me that, at least as far as rigid body dynamics is concerned, these two Balls will react the same way to impulses - ie, they behave the same way when we push them around.
Basically, we can get the best of both methods, if we use Method 2 to calculate partial Masses, and use Method 1 to calculate the partial MOIs.. this greatly simplifies the math (making it a lot faster and easier to implement) while retaining ALL the accuracy.

Postscript : I neglected to mention one particular aspect of my idea.
I've discussed briefly how to calculate the mass of an arbitrary tetrahedron (ie chunk of body) - notice that for each tetrahedron, we have calculated a Mass.. this mass should be distributed among the three vertices of each Face, rather than the four vertices of the tetrahedron, such that "we are shoving all the mass to the Body's outer shell".. now it should make more sense why I mentioned the hollow and solid Balls earlier.
What I'm really trying to say is that we only use the tetrahedrons to calculate Volume (and thus Mass), and that once we have the Mass, we go back to an Atomic (point-based) way of thinking, and distribute Mass over each Face, and thus over the entire Volume.
What we're really achieving is A) accurate calculation of Volume and Mass and B) accurate distribution of Mass over Volume.

Posted on 2006-05-23 08:35:57 by Homer

The ideal formula for Volume of arbitrary tetrahedrons sharing common Origin

We don't need to mess around with Determinant math..or rather, we can lever one feature of my proposed method in order to take advantage of a mathematical anomoly, and simplify the fomula and the math involved in calculating arbitrary tetrahedron volume.

Recall that we are using the Body Origin (==COM) as one point in each Tetrahedron..
Lagrange showed that the tetrahedron formed by Origin(0,0,0) and the three points M(x,y,z), M'(x',y',z'), M"(x",y",z") has volume

1/6 .

We can implement this formula 8)

Posted on 2006-05-23 12:15:06 by Homer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

This is not tested yet, but should work just fine.
If anyone cares to write an MMX version, I'm sure it would be faster.

```
;This procedure calculates the Volume of
;an arbitrary tetrahedron projected from
;the Origin, ie has one vertex at (0,0,0)
;The following solution was found by Lagrange:
;V = 1/6 .
;
;Input Params are three pointers to three vertices of the tetrahedron
;Output is returned on the FPU stack
.data
fp6 REAL4 6.0f
.code
Calc_Tetrahedron_Volume proc uses esi edi ecx pv1,pv2,pv3

mov esi,pv1
mov edi,pv2
mov ecx,pv3
```

Posted on 2006-05-24 11:44:01 by Homer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

after changing the fdiv with an appropriate fmul, the proc takes 30cycles here,
only optimizable to 29cycles when I remove the prologue/epilogue.
14 cycles go for just the proc, and 15 cycles go for the FPU part.
Can't optimize it with SSE, the data members are not used f2f (as SSE loves it).

```
align 16
f016 real4 0.16666666666666666666

option prologue:none
option epilogue:none
Calc_Tetrahedron_Volume proc pv1,pv2,pv3
push edi
        push esi
        push ecx
mov esi,
mov edi,
mov ecx,
fld .Vec3.x
fmul .Vec3.y
fld .Vec3.y
fmul .Vec3.x
fsub
```

Posted on 2006-05-24 16:10:38 by Ultrano

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

Cool 8)

I'll be interested in comparing the output of this procedure to a known regular geometry, such as a Cube (2 triangles per flat surface * 6 surfaces = 12 tetrahedral primitives projected from the cube's center of mass / body origin).
That will tell me immediately whether this formula is garbage or not :)

Unfortunately, I just switched motherboards and reinstalled everything, and DebugCenter is no longer working for me.. so I'll have to figure out what's wrong there first (I'm too lazy to write output code for a simple test).

Posted on 2006-05-24 23:40:12 by Homer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

> after changing the fdiv with an appropriate fmul, the proc takes 30cycles here,
> only optimizable to 29cycles when I remove the prologue/epilogue.
> 14 cycles go for just the proc, and 15 cycles go for the FPU part.
> Can't optimize it with SSE, the data members are not used f2f (as SSE loves it).

why not? the data structs shouildnt be your master
customized data, until all physics etc is done and final output to customvertex format
an MMX version requires different data anyway
ok if all masscalculation is done only with this formula for all object, if everymass is 6 times bigger or not does it matter?

Posted on 2006-05-25 02:53:50 by daydreamer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

daydreamer,
hmm, prolly my SSE experience isn't enough, but here're my thoughts on it:
z0,z1 and z2 should be in one 128-bit unit/struct for max-speed. Then, we're doing cross-multiplications of x and y members of the 3 vertices. Enough arguments to make SSE optimization seem useless - I doubt it can be done in 15 cycles.

Homer,
about DebugCenter - maybe you lost some Registry data - this happened to me when I changed my mobo+cpu last week. (but I also had trouble with a bad HDD partition cluster just after the bootsector...).

Posted on 2006-05-25 06:21:49 by Ultrano

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

Updated the fpu code a bit,

```
align 16
f016 real4 0.16666666666666666666666666666667


option prologue:none
option epilogue:none
Calc_Tetrahedron_Volume proc pv1,pv2,pv3
pushi ecx,esi,edi
mov esi,
mov edi,
mov ecx,
fld .Vec3.x
fmul .Vec3.y
```

```
fld .Vec3.y
fmul .Vec3.x
fsub
```

Tested it with a 2x2x2 box, and yes - it works :) - gives a volume=8

```
.data
align 16
t0 Vec3 <-1.0,-1.0,-1.0>
t1 Vec3 < 1.0,-1.0,-1.0>
t2 Vec3 < 1.0, 1.0,-1.0>
t3 Vec3 <-1.0, 1.0,-1.0>
t4 Vec3 <-1.0,-1.0,1.0>
t5 Vec3 < 1.0,-1.0,1.0>
t6 Vec3 < 1.0, 1.0,1.0>
t7 Vec3 <-1.0, 1.0,1.0>



.code


adx macro p0,p1,p2,p3
```

Posted on 2006-05-25 06:48:19 by Ultrano

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

OK sweet, it WORKS, and thats always good :)
As for my registry entries, well, I reinstalled oa32, so registry entries shouldn't be an issue.. I think the installer writes them now.

I will ask Biterider when I see him what he thinks, I know other users have had problems getting DebugCenter working too.. I have not in the past..

Anyway, YAY, it WORKS, great !! I'll go ahead and write code for simple (single mesh) objects which does the following:
1 - Calculate CenterOfMass by averaging the Vertices
2 - Shift the Mesh origin to coincide with COM by shifting all the Vertices explicitly
3 - Calculate Volume using Lagrange's method

All that we need to do after that is to decide what Density the material of the Body, and then calculate Mass = Volume * Density

At that point, I can start to think about calculating MOI for each tetra..

This is all seeming too easy :)

Posted on 2006-05-25 07:28:19 by Homer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

> **daydreamer,**
> hmm, prolly my SSE experience isn't enough, but here're my thoughts on it:
> z0,z1 and z2 should be in one 128-bit unit/struct for max-speed. Then, we're doing cross-multiplications of x and y members of the 3 vertices. Enough arguments to make SSE optimization seem useless - I doubt it can be done in 15 cycles.

you must think different
smallest denominator 3 arguments x 4 packed operations =12 packed operations
you make one proc that unrolled several calculations
MMX solution: inside parenteses :psub psub pmadd pmadd
outside parenteses:pmadd pmadd
remove final multiplication and work with densities that are stored 1/6
because dont wanna lose presicion and speed when working with integers a mul 1/6 is impossible anyway

Posted on 2006-05-25 08:21:10 by daydreamer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

uhm a code solution, instead of vague description of the idea would be more helpful imho :)
but anyway, at this point we don't know the higher-level usage of the proc (what kind of code calls Calc_Tetrahedron_Volume), so we'll leave the further optimization for later ;)

Posted on 2006-05-25 08:47:59 by Ultrano

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

Example Pseudocode to calculate total Volume and total Mass of arbitrary Mesh:

```
Calc_Object_Mass proc pFaces,pVertices,dwVertexStride,fDensity:REAL4
local fVolumeAccumulator:REAL8
mov fVolumeAccumulator,0
;Faces are triple word-sized Indices into Vertex array
foreach Face:
- grab indices
- multiply by Stride
- add base of vertex array
(such that we have pV1, pV2, pV3)
-invoke Calc_Tetreahedron_Volume,pV1,pV2,pV3
```

-fadd fVolumeAccumulator
-fstp fVolumeAccumulator
endfor
fld fVolumeAccumulator
fmul fDensity
ret ;return total Mass = total Volume * Density
Calc_Object_Mass endp

So yes we can see that its not necessary to wrap the tetra-volume code as a proc, and really, the same goes for this proc.. not shown here is code for calculating the partial MOI of each tetra and the accumulation of them to find total MOI.

I'm in bed sick with the flu so I hope this post made sense :)

Posted on 2006-05-26 01:14:14 by Homer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

Got my problem with DebugCenter fixed (thanks Biterider).
I performed my own small test of Lagrange's tetravolume formula and discovered that it can return negative values, so the formula SHOULD be:

fabs (1/6 *)

as implied by Ultrano's demo code (there's no such thing as negative volume in a 3 dimensional universe).
If we simply accepted the negative results, I think we'd arrive at a total volume of zero :P

Two thumbs up to Ultrano 8)

Posted on 2006-05-26 03:42:26 by Homer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

Now we can get down and dirty..
The following class encapsulates everything we've discussed so far, and a little more.
I have not tested this code heavily, and I'm not sure that I am calculating the MOI correctly, so let's check out the code and then talk about it :)

PS : Corrected a bug in calculation of PointMasses..

```
.data
fhalf REAL4 0.500f
.code

RBID equ 93945
Object RigidBody,RBID,Primer
StaticMethod Load, Pointer,REAL4
StaticMethod CorrectTheOrigin, Pointer
StaticMethod Calc_Tetra_MOI, REAL4,Pointer,Pointer,Pointer,Pointer
;
DefineVariable pMesh,Pointer,NULL
DefineVariable dwNumFaces,dword,NULL
DefineVariable dwNumVerts,dword,NULL
DefineVariable dwStride,dword,NULL
DefineVariable dwFVF,dword,NULL
ObjectEnd
```

Posted on 2006-05-28 02:07:30 by Homer

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

For those who are too lazy to study the code closely, here's a description of what's going on in laymans terms..

We load a static Mesh from an XFile.
We check how far 'out of center' the Mesh is, and correct the error, recentering the Mesh so that the Mesh Origin and the Center Of Mass are coincidental.. (unsure if I did this correctly).
We iterate through all the Faces in the Mesh... for each Face, we obtain pointers to its three Vertices, and then we hand those three pointers to our handy, dandy Tetrahedron Volume calculator (which assumes the fourth Vertex is at the Origin of the Body).
This tells us the Volume of each "chunk of flesh", from which we immediately obtain the Mass (since Density is a given).
The Mass of each Tetrahedron is then distributed across the three Vertices of the Face (imagine we're shoving all the Mass to the outer shell of the 3D Body represented by the Mesh), which basically gives us three PointMasses for each Tetrahedron.
We calculate the Moments of Inertia for each PointMass, and we sum them (add them up) to find the Total MOI of the Body.
Thanks for flying with Homer Brainwaves :D

Posted on 2006-05-28 02:26:55 by Homer

Re: quick FPU to SSE conversion

;;Method RigidBody.Calc_Tetra_MOI
;;Step 1 : Calculate three ray lengths
mov eax, pV1
mov ecx, pV2
mov edx, pV3
movdqa xmm0,

```
movdqa xmm1,
movdqa xmm2,
mulps xmm0, xmm0
mulps xmm1, xmm1
mulps xmm2, xmm2
movhlps xmm3, xmm0
movhlps xmm4, xmm1
movhlps xmm5, xmm2
addps xmm0, xmm3
addps xmm1, xmm4
addps xmm2, xmm5
pshufd xmm3, xmm0, 1
pshufd xmm4, xmm1, 1
pshufd xmm5, xmm2, 1
addps xmm0, xmm3
addps xmm1, xmm4
addps xmm2, xmm5
sqrtss xmm0, xmm0
sqrtss xmm1, xmm1
sqrtss xmm2, xmm2
movss fRay1, xmm0
movss fRay2, xmm1
movss fRay3, xmm2
```

I'm assuming that the vectors are stored

```
align 16
dword x, y, z, 0.0f
```

If this ISN'T the case then 6 more instructions are required

```
movdqa xmm0,
movdqa xmm1,
movdqa xmm2,
;;then
pslldq xmm0, 4
pslldq xmm1, 4
pslldq xmm2, 4
por xmm0,
por xmm1,
por xmm2,
```

and global data

```
ScalarZero dd 0.0f, 0h, 0h, 0h
```

I didn't use HADDPS for compatibility issues, so the code can be run on systems with only SSE/2.

Posted on 2006-05-30 20:59:25 by r22

Re: Idea : fast accurate Inertia Tensor for arbitrary meshes

takes ~500 cycles
You mean SSE2 (on a quick look), movdqa and pshufd aren't present in SSE, I guess - masm doesn't recognize these 2 instructions.
My Sempron doesn't support them (and btw I am quite happy by its performance, no upgrading for 2-4 years more)

Posted on 2006-05-31 02:06:15 by Ultrano