Interactively Prototyping Properties of Rigid Bodies for Physically-based Animation

Nicholas Pochinkov, supervised by John Dingliana

Trinity College Dublin
The University of Dublin

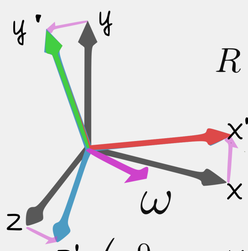
Introduction

The ultimate aim of the research project was to produce an interactive tool that can compute various physical aspects of a body, with a focus on computing and manipulating the tensor of inertia of a complex rigid body. This can be used to teach people about the tensors of inertia, and to advance physics simulation in modern live graphical simulations such as the growing video game and VR industry. To code this, I used the "Processing" programming language & development environment, a language favoured for its ease of use for producing graphical outputs and similarity to c, leading to a shorter time to get up and running.

Progress Reports

To begin in the first week, I focused on using the 3D engine implemented in the Processing language and created a basic class upon which I would build the rest of the project

Week 2, I implemented the Rotation matrix R



$$R = \begin{pmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{pmatrix}$$

$$\dot{R} = \begin{pmatrix} 0 & -\omega_z & +\omega_y \\ +\omega_z & 0 & -\omega_x \\ -\omega_y & +\omega_x & 0 \end{pmatrix} \begin{pmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{pmatrix}$$

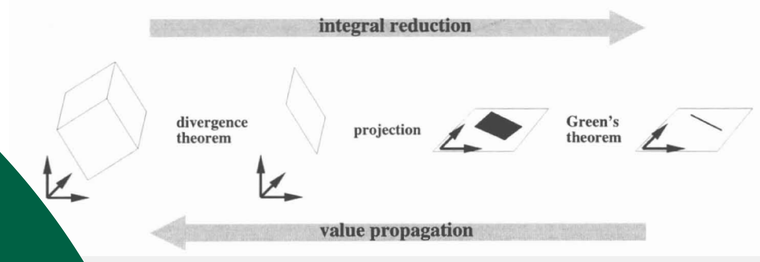
As well as the angular velocity vector to be able to procduce a spinning effect of the body used. Due to compounding numerical errors, It was also necessary to orthogonalise the rotation matrix after each step to avoid the object being deformed or oversized (which was achieved with Grahm-Shmidt orthogonalization)

Week 3, I implemented a force system into the tool that makes it such that the change in velocity, position, angular velocity, rotation matrix depended on a force by mass/tensor of inertia (for a cuboid), and the change was then integrated using a combnation of both the midpoint method and Euler method.

$$\vec{L} = \tilde{I}\vec{\omega} \Rightarrow \frac{d\vec{L}}{dt} = \tilde{I}\frac{d\vec{\omega}}{dt} \Rightarrow \frac{d\vec{\omega}}{dt} = \tilde{I}^{-1}\frac{d\vec{L}}{dt} = \tilde{I}^{-1}(\vec{r} \times \vec{f})$$

Week 4, I began to implement code that allows the import of any custom 3D .obj file, which required slight modifications to how other parts were processed. I also made code that sets the bounding box around the 3D object and sets the inertia tensor to the one for the bounding box for visual testing purposes, and tested it with a simple spring attatched to one of the vertices. This worked well and I then gave a seminar explaining the work I had done so far.

Week 5, I began making a function to do integrals to get the Tensor of Inertia by the method described in the paper by Brian Mirtich[1] which involves reducing the 3D integrals to 2D line integrals & propagating



Theorem 1 (Divergence)

$\mathcal{V}$ - region in space
$\partial\mathcal{V}$ - boundary of $\mathcal{V}$
$\hat{\mathbf{n}}$ - exterior normal

$$\int_{\mathcal{V}} \nabla \cdot \mathbf{F} dV = \int_{\partial\mathcal{V}} \mathbf{F} \cdot \hat{\mathbf{n}} dA$$

Theorem 2 (projection)

$\mathcal{F}$ a polygonal region
α-β-γ coordinate system
$\hat{n}$ surface normal
Π projection of $\mathcal{F}$ on α-β space

$$Given: \hat{n}_\alpha\alpha + \hat{n}_\beta\beta + \hat{n}_\gamma\gamma + w = 0$$

$$\int_{\mathcal{F}} f(\alpha,\beta,\gamma)dA = \frac{1}{|\hat{n}_\gamma|}\int_{\Pi} f(\alpha,\beta,h(\alpha,\beta))d\alpha d\beta$$

$$where: h(\alpha,\beta) = -\frac{1}{\hat{n}_\gamma}(\hat{n}_\alpha\alpha + \hat{n}_\beta\beta + w)$$

Theorem 3 (Green's)

Π - region in the plane
$\partial\Pi$ - single boundary
$\hat{m}$ - exterior normal along the boundary.
$\mathbf{H}$ - any vector field on Π
Line integral transverses anti-clockwise

$$\int_{\Pi} \nabla \cdot \mathbf{H} dA = \oint_{\partial\Pi} \mathbf{H} \cdot \hat{\mathbf{m}} ds$$

Theorem 4

L_e - length of the directed line segment from (α_e, β_e) to $(\alpha_{e+1}, \beta_{e+1})$
$(\alpha(s), \beta(s))$ - arc-length parameterization of line (s from start point)
For non-negative integers p and q:

$$\int_0^1 \alpha\,(L_e\lambda)^p\,\beta\,(L_e\lambda)^q\,d\lambda = \frac{1}{p+q+1}\sum_{i=0}^{p}\sum_{j=0}^{q}\frac{\binom{p}{i}\binom{q}{j}}{\binom{p+q}{i+j}}\alpha_{e+1}^{i}\alpha_e^{p-i}\beta_{e+1}^{j}\beta_e^{q-j}$$

To conclude, I would like to implement more ways of interacting with the imported object and add a basic collision detection mechanism which will allow the body to respond to surfaces and rebound in a realistic fashion, and this will conclude what I set out to achieve with this research project

A **Tool** which accurately **shows** how any imported **object** will **spin and move** in response to **forces & collisions** to explain the **Tensor of Inertia**

For the full report and the code, scan the QR code or visit my website at pesvut.netsoc.ie/researchProject2019.html

REFERENCES

1. Brian Mirtich. 1996. Fast and accurate computation of polyhedral mass properties. J. Graph. Tools 1, 2 (February 1996), 31-50. DOI= http://dx.doi.org/10.1080/10867651.1996.10487458