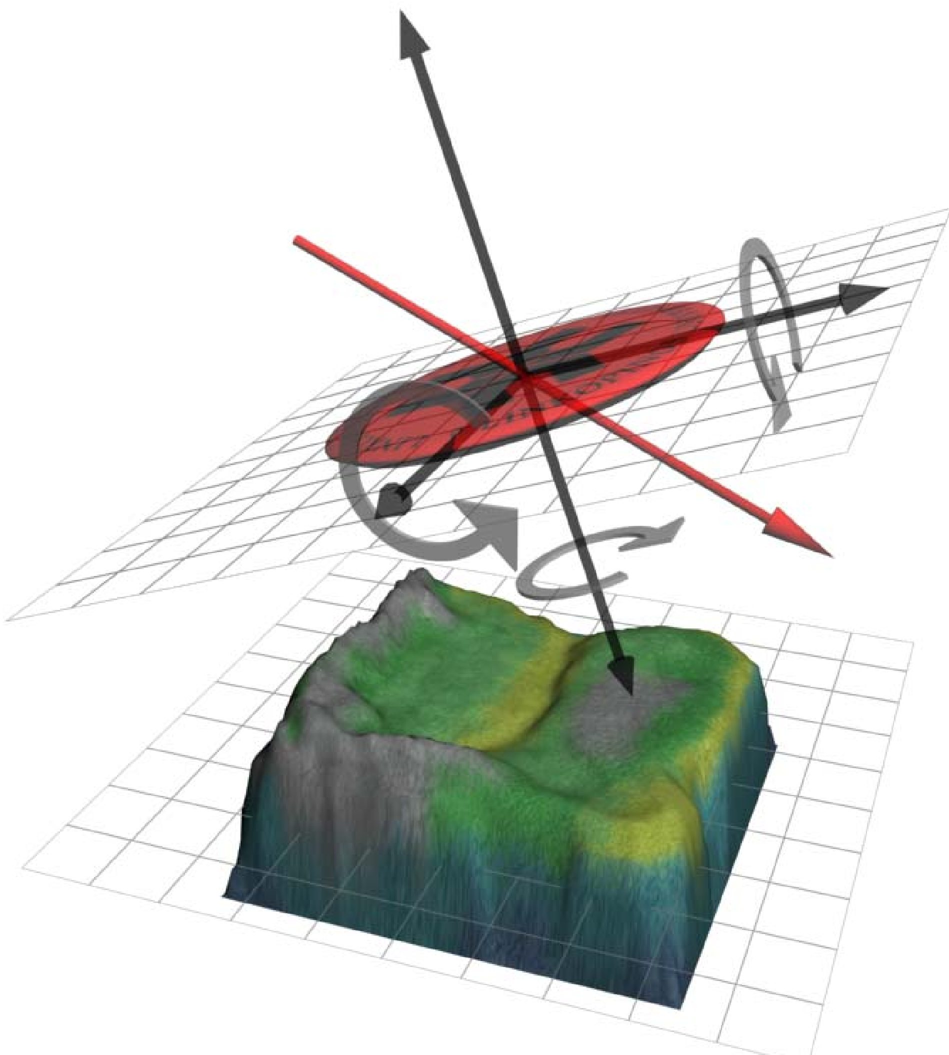
Simulation of a Frisbee

A report of the simulation project in TNM032



TNM032
Project group 3

Ann Holmberg
Tobias Johansson
Ebbe Strandell
Mikael Uddholm

Executive summary

When a Frisbee is thrown, forces act on its body and the flight depends on many different physical laws. In this report we will give some ideas of how to simulate the Frisbee flight. It is difficult to derive a exact model of the Frisbees flight because the forces that act on the Frisbee has complex relationships and has to be calculated in a three dimensional world. When we shall simulate the Frisbee are the forces from all dimensions greatly important, therefore we have to take ideas to the formulas from articles, written by professionals.

1 Introduction

1.1 Problem

A Frisbee can be thrown in many ways and the flight depends on many different physical laws. Therefore the Frisbee flight model is quite complex. Like a ball flying in the air, the initial velocity and angle, air resistance and the gravity will determine how far the ball will move. If we were to simulate a game of baseball we can't neglect lift, drag and momentum forces acting on the flying ball. If the ball moving through the air initially is set to spin these forces will twist the movement pattern of the ball. When simulating the movement of a flying ball these three latter forces often can be neglect because the effect on the ball is small. However, when simulating a Frisbee these forces are greatly important and must be calculated.

The goal of this project is to make a simulation in three dimensions and this involve dividing all forces and momentum into six degrees of freedom. This because the Frisbee has to be able to move in three dimensions and to rotate around its own three axes.

1.2 Project aim

The aim of this project is to simulate the flight model of a frisbee as close to the reality as possible. This is done by calculation of the forces acting on the body. As always, in simulation, some conditions are simplifications of the reality that may effect the result.

The simulation will be done in C/C++ and the flight model visualised through OpenGL libraries.

2 Background

2.1 The idea

The course, TNM032, in modelling and simulation at Linköping University aims at the students are to carry out a project where a physical situation is to be simulated. The first thoughts of this project came then a few of us in the group were out playing a game of Frisbee golf. The thought of simulating this excited all members in the group and we decided to give it a try.

During the discussion about the subject we also decided to try to write the simulation in such way the base of the program could be used in other projects, development of a Frisbee game for example. Therefore the program had to be well structured.

2.2 Brief History of the Frisbee

About 400BC we could see the flying discs in the sky for the first time. The Greeks was throwing discs in the Olympic Games, nowadays we call that discus. The discs they threw were different than the Frisbees we are used to see in modern time.

After the Second World War more peoples were interested about UFO's and a man, who called Fred Morrison saw his opportunity to make some money. He sold the first plastic Frisbee, called Pipco Crash after Morrisons Pipco Company.

In 1948 a toy company Wham-O and Morrison started to corporate to find a new better model. The second model was called the Pluto Platter, which were sold with success.

In 1959 the name Frisbee became a registered trademark After the Pluto Platter, the Sailing Satellite, the Sputnik and the Flying Saucer, Wham-O produced the worlds first Frisbee. [6]

3 Method

3.1 Analyse the flight dynamics

Many people are quite familiar with the movement of a flying Frisbee but even if one plays the game he or she might not know details about velocity and angular velocity of the disc. Knowledge about these conditions had to be measured and experiments on this subject were made.

Two thin lines, normally used for fishing, were attached to the centre of a Frisbee and the disc was thrown a certain distance. For every revolution of the disc the two lines were crossed two times. By measuring the distance, number of revolutions and the time of flight initial velocity and angular velocity could be calculated.

The maximum length and height for the disc, of course, is dependent of the how hard the thrower throws the disc and in what angle he or she aims. Although a decent understanding of theses limits are necessary when testing the simulation. Such understanding on how the disc moves through the air is also important.

These experiments paid of greatly during the simulations of the Frisbee although the formulas used in the end are way too complex to derive on such basic experiments. A limitation from our side was to not derive these equations by our selves. The equations used here are derived by experts on this subject. Of course these formulas had to be changed to fit into our 3D world and sometimes we had to make some simplifications.

3.2 Using Euler equation

The result effect from a force acting on an object during a certain amount of time is normally calculated with integrals. A problem with integrals is that infinite small area or volume elements are used. Such continues calculations cannot be done with computers where all calculations are non-continues. Nevertheless, in computer science one often need to calculate these integrals and we are then forced to use numerical methods. One famous numerical method is Euler's method.

In this simulation Euler's method is used to calculate the effect of all the forces acting on the Frisbee. Here we integrate over time and as time-element the time between each frame is used.

3.3 Programming language and graphic presentation

As stated before this simulation is to be represented in 3D, more precisely OpenGL libraries will be used for the graphics and C/C++ is used as interface to OpenGL. The physic calculations, system calls etc are also in C/C++ which is an effective programming language (and maybe the best for accessing OpenGL).

The graphical user interface is, in many ways, designed as some computer games. This way its easy to change the initial conditions and to see the result of the simulation.

The graphical implementation involves many functions that have really nothing to do with the simulation but necessary for the visualization. However, explanation of these functions such as object and bitmap loaders are out of this scope.

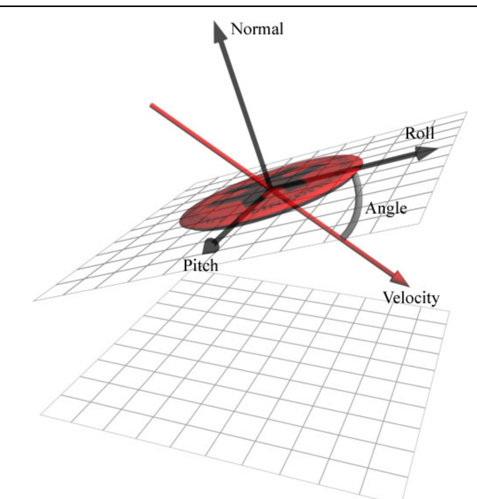
4 Simulation

4.1 Physical equations

As one can imagine the flying Frisbee has many counteracting forces acting on its body. The gravity will force the Frisbee towards the ground and the lift force will at the same time lift the disc towards the sky. The drag force will reduce the velocity and on the same time change the path of the disc. The rotation of the disc also creates momentums that will inflect the movement of the disc in certain ways.

As this simulation is supposed to be in three dimensions we must be able to see all forces as well as the velocity as vectors. One must understand that the direction of the velocity not have to be, seen as from the disc, straight forward. The lift force in a similar way isn't always pointing towards the sky.

When we calculate the physical proporties of the frisbee we use two different cordinate systems, here callde the *world* and the *discplane se f1*. The world is used to calculate the velocity, direction of movement and also used to draw the 3d grapics. The discplane is used to calculate the forces acting on te disc and the rotation. The forces actiong on the frisbee while it is in the air are depending on the magnitude of the velocity and the angle of attack. The shape of the disc determins how it reacts on the forces.

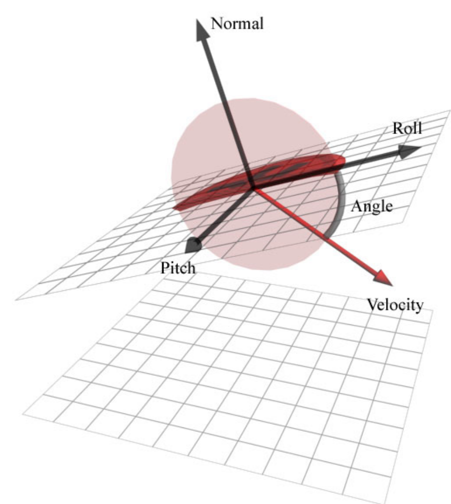


f1

4.1.1 Calculating the discplane

The magnitude of the velosity determine the pover of the flow of air around the frisbee. The angle of attack determines how the shape of the frisbee affects the flow and thereby the resulting forces on the frisbee.

First we need to know how the frisbee is orientated in the word. Since the frisbee is symetric around its normal we can neglect that rotation angle in the flow calculations. Therfore we only need to know the angle between the direction of the velocity and the rollvector. By calculating the cross product of the velocity and the normal wich booth are known we get the pitch vector [1]. Thereafter we use the crossproduct of the normal and pitch vector to get the roll vector, se f2. If the direction is paralell to the normal the lift force will be equal to zero and therfore we don't need to know the pitch or roll vector.



f2

To get the angle between the rollvector and the direction we use the definition of the dotproduct, it is also important to know if the angle is positive or negative and this can be done by comparing the normal and the direction.

Now that we know how the disc is orientated in the world and the angle of attack we can start to calculate the forces. The formulas used to calculate how the shape of the disc affects and are affected by the airflow are wery complex and has to be determined in a windtunnel. [4] The formulas we use are based on the formulas decribed in " Simulation of Frisbee flight" [2]

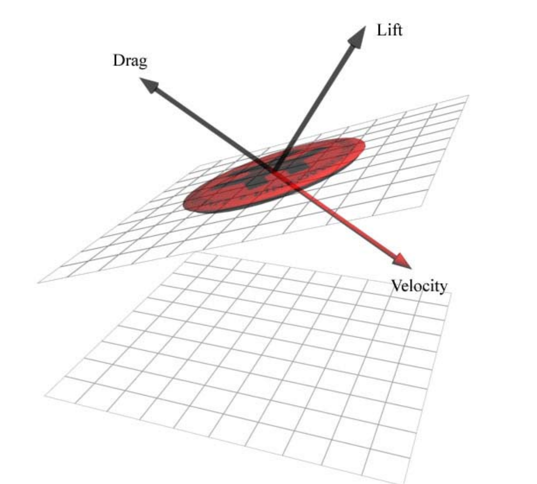
4.1.2 Lift and dragforce

The dragvector has the same direction as the air flow. The lift vector is othogonal to the dragvector and the pitchvector, se f3. We normalise the vectors and multiply them with the formulas below to get the force in 3d space.

$L = C_l A\rho v^2 / 2$

$D = C_d A\rho v^2 / 2$

The constants used in the formulas ar decribed in apendix 1



f3

4.1.3 Roll, pitch and spinn moment

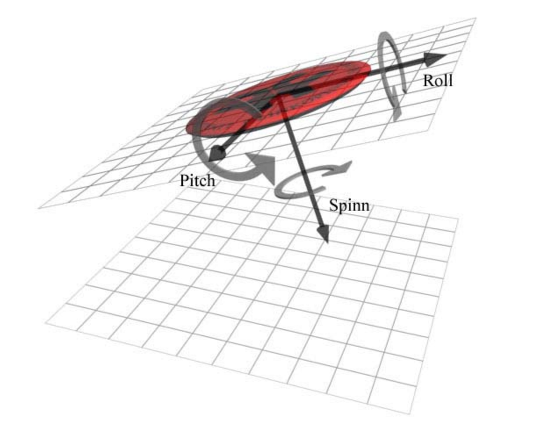
The frisbee is very dependent on rotation. It is the rotation around the normal here called spinn that stabilizes the frisbee in tha air, the fast rotation about the normal also give rize to a gyroeffect. The rotation vectors are orientated as shown in f4.

$R = Ad\rho v^2 (C_{Rr}\, r + C_{Rp}\, p) / 2$

$P = Ad\rho v^2 (C_{Mo} + C_{Ma} a + C_{Mq} q) / 2$

$S = Adv^2 C_{Nr}\, r / 2$

Each angular accelaration are multiplied with its inertia and vector and then sumed to a rotation vector. [3]



f4

4.2 Implementing simulation in C++

With all forces, acting on the Frisbee body, explained its time to explain how everything can be combined into a simulation. In this chapter we discuss how to translate all forces into three dimensions, how to use the Euler equation and how this can be visualised through 3D graphics.

4.2.1 Drawing in 3 dimensions

To visualise the disc we only need its position and rotation but to calculate this we need all information explained in chapter 4.

Due to some graphical problems with the limitations of the z-buffer we had too scale everything by a factor of 0.1.

4.2.2 The use of Euler equation

The forces acting on the disc must be calculated continuously. It is not possible to calculate all forces initially because a noise such as a wind may effect the disc and change its path. Instead each force has to be integrated over time. Since the simulation should run with the same speed irrespective of what computer is used the time element in the integral is preferably set to the time between each frame.

With Euler's equation the changes in velocity and position of the disc can be calculated. The mass of the disc is easy to measure and the force is the resulting force of all the forces calculated above.

A = F / m	Acceleration (A) of the disc at current frame, calculated with the forces (F) acting on the disc and the mass of the disc.
V = V + A * dt	Velocity (V) of the disc calculated with the velocity from the frame before current and acceleration times time different between each frame (dt).
P = P + V * dt	Position (P) of the disc at current frame, calculated with the position from frame before current and the calculated velocity at current frame times different between each frame.

4.2.3 The graphics

Graphs are in many cases important tools to mediate facts to other people but graphs can also in many cases bore the audience. If the grapgh was surrounded by a world the audience could rely to instead of numbers and lines maybe more people would appreciate the facts presented by the graph. When simulating a physical condition, in our case the frisbee, it can also be appreciated if the user can interact with the system. Our program is aimed to combine the best of them both, the ability to interact with the system and at the same time get a feeling for how the physics act together.

As said before the visualisation of the simulation is made in OpenGL, a fast and reliable API (aplication programming interface) for graphics. OpenGL is widely used in games and other graphical applications and therefor it is well documented and tutorials are easy to find. The frisbee is made in 3d studio max. It is a quite simple model, only 92 vertexs and 180 triangular faces. The frisbee is exported from 3d studio max as an ASE-file which is handy when it comes to import models to programming projects. A very efficient way to render objects with many vertexes is to load them into vertex arrays which OpenGL greatly supports. Most of our objects are rendered this way. The objects are loaded into vertex arrays directly in the ASE file-loader. This is done by scanning the ASE-file for keywords and through them locate the data about vertexes, normal and texture co-ordinates.

To make it easier to understand how the frisbee flies a tail was implemented. Each iteration a new node is added to a single-linked list containing the position of the two parts of the frisbee edge that is perpendicular to the roll-vector and the normal. These vertexes are linked together as polygons and a transparent texture is applied to for the visual effect.

The landscape is loaded into the world in the same way as the Frisbee. This model is more complexed and it is put together by 2601 vertexes and 5000 faces. The texture on the landscape is only 512x512 pixels and without the multitexturing each pixel in the texture is too big and too easy to spot. Instead the landscape has two texture maps, one map is streached all over the landscape and this map gives different color to different spots. It, for example, makes the grass green and the mountins grey. The other map, that makes the ground look bumpy and scratched, is repeated 400 times over the area. This way the landscape looks like it has a higher resolution then it really has.

In this project there is only one moving object and therefor no shadows are implemented. Nevertheless shadows are very important to make the world look real. In this project a shortcut to this was used instead. The world was made from a heightmap. This model, in 3d studio max, was othogonally rendered from above after a light was put in the scene. The mountains now produced shadows, the rendered image is used as texture and the landscape therefor has shadows. The background image and the water are simply a box and a halfsphere with textures that are loaded into the project with a bmp-reader. As well as the ASE-file the BMP-file is scanned, here numbers beween 0 and 255 are of interest and defines all possible colors in a 24 bit world.

4.2.4 To run the program

The program is tested under windows 2000, it is tested on computers with 2.4GHz Intel Pentium processors, 512 Mb DDR RAM (266MHz bus) and GeForce IV graphics cards. We can not guarantee that the program will run smoothly on computers with less power although it is likely that the program will run well on not too outdated computers.

The simulation is run with frisbee.exe. The user is thrown into the world at once, no intro needed. At first the user can walk around in the world. When spacebar is pressed a disc is displayed. Now the user can rotate the disc and set the initial speeds. See button schedule below. Space is pressed again and the disc is thrown.

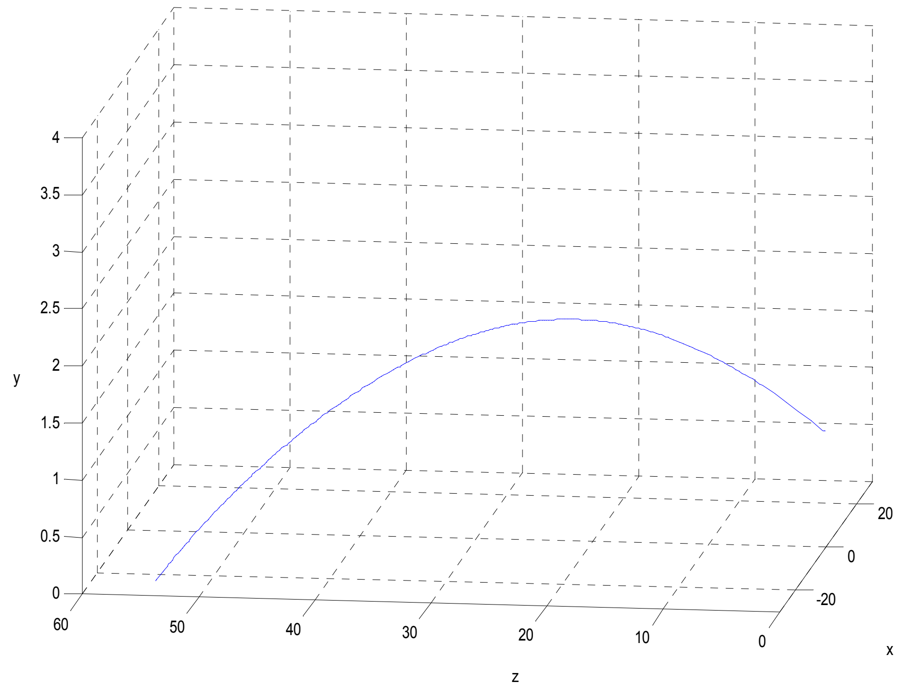
Please note that best results, closest to reality, is achieved when the disc is thrown slightly downwards or straightforward. Be sure to give the disc sufficient initial angular velocity and don't rotate the disc too much in any direction or else the outcome will turn out bad.

Keyboard button	**State 1**	**State 2**	**State 3**
W	Walk forwards		
S	Walk backwards		
A	Float upwards		
D	Float downwards		
Up arrow	Look upwards	Rotate the disc from the camera	Increase velocity
Down arrow	Look downwards	Rotate the disc towards the camera	Decrease velocity
Left arrow	Look left	Rotate the disc to the left	Decrease angular velocity
Right arrow	Look right	Rotate the disc to the right	Increase angular velocity
Spacebar	Alter state	Alter state	Release the disc
C	Toggle cameramode	Toggle cameramode	Toggle cameramode
F1	Toggle fullscreen	Toggle fullscreen	Toggle fullscreen
Esc	Quit the program	Quit the program	Quit the program

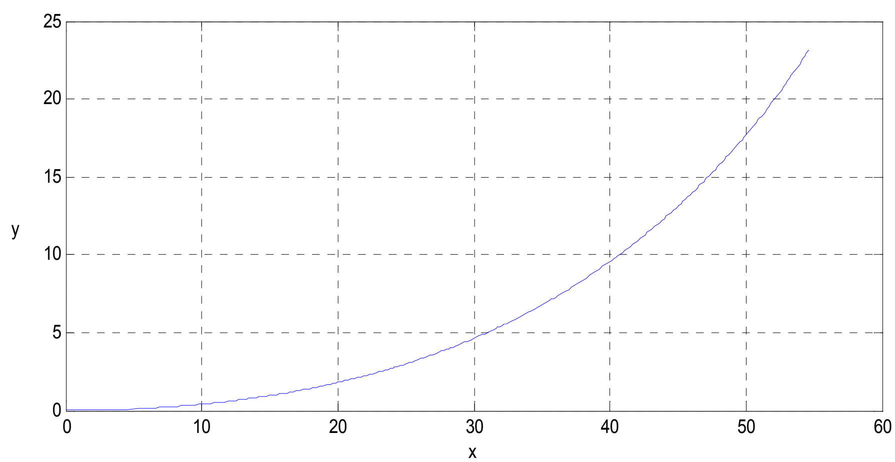
5 Result

5.1 Simulation

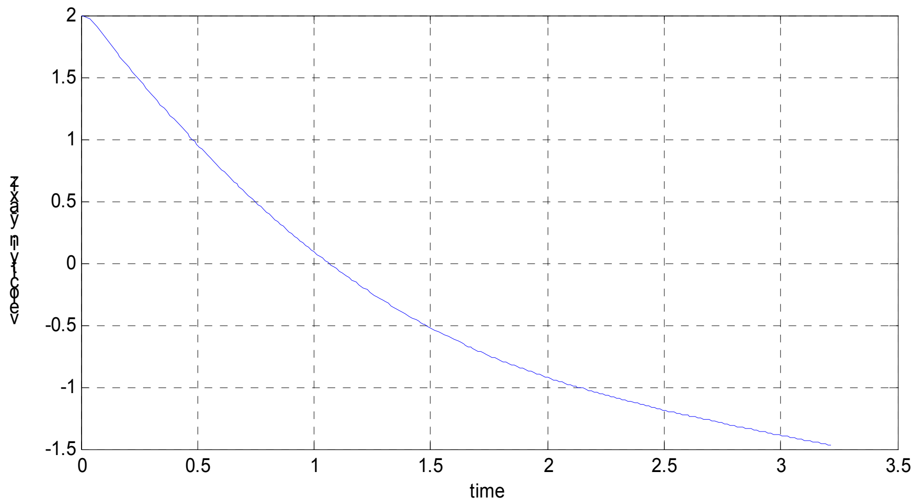
The result of the simulation turned out well. A simulation of the Frisbee flying in a 3D-space with all the forces acting continently on the body is the result. The movement of the Frisbee is calculated with Euler's formula (Appendix (ref)) and the drag and lift forces and spin, pitch and roll momentum is calculated and set together to rich as near the reality as possible.



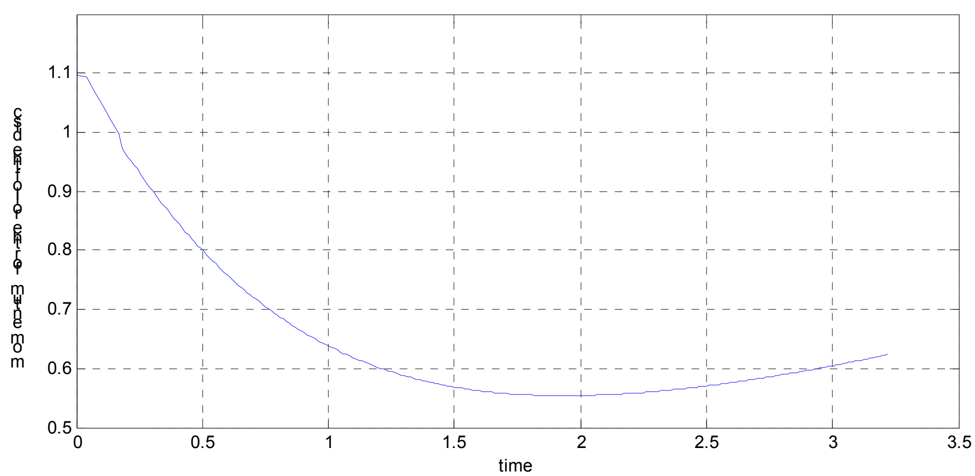
f5 *Frisbee moving thrown with initial velocity of 25 m/s with angle of 4.5 degrees.*



f6 *Frisbee moving from the thrower at the x-axis and turns left at the y-axis. The disc was thrown with initial velocity of 25 m/s and with angle of 4.5 degrees.*



f7 *The disc y-axis velocity against time. The disc was thrown with initial velocity of 25 m/s and with angle of 4.5 degrees.*



f8 *Figure views the changing in roll momentum for the disc in flight. The disc was thrown with initial velocity of 25 m/s and with angle of 4.5 degrees.*

6 Discussion

Now as before we started to simulate our Frisbee we agree that the Frisbee flight model is quite complex. The flight depends on many different physical laws and when we should simulate the Frisbee these forces were greatly important and had to be calculated in a good way.

After several test flights we know that our model have some limitations. This is most certainly due to the fact that we don't have the knowledge about aerodynamics to fully revise the formulas and that it is very hard to create a model of a flow system.

Further there are many things that can be done to expand the capabilities of our simulation. For example implementation of wind and the ability to test different Frisbee shapes.

The goal of this project was to simulate the flight model of a Frisbee as close as possible to the reality and that should be done in three dimensions. By calculating all forces acting on the body of the Frisbee in our three-dimensional world, we now have a quite good simulation of a flight with a Frisbee in the reality.

7 References

7.1 Books

[1] Linjär algebra Gunnar Sparr 1996

7.2 Articles

[2] Simulation of frisbee flight, M. Hubbard and S.A. Hummel, Department of Mechanical and Aeronautical Engineering, University of California, Davis CA 95616 USA, Presented at the 5th Conference on Mathematics and Computers in Sport, G. Cohen, Editor, University of Technology, Sydney, New South Wales, Australia, June 2000.

[3] Analysis of a Flying Disc, Brian Danowsky and Babak Cohanim, Spring 2002 Symposium, April 26, 2002, Iowa State University Ames, Iowa

7.3 Internet

7.3.1 Physics

[4] http://www.grc.nasa.gov/WWW/K-12/airplane/short.html 04-01-18

[5] http://www.ukultimate.com/history.asp 04-01-18

7.3.2 C++ and OpenGl references

[6] http://nehe.gamedev.net/ 04-01-18

[7] http://www.gametutorials.com/ 04-01-18

[8] http://www.cplusplus.com 04-01-18

[6], [7] and [8] are represented in the code and are not refereed to in the text.

Appendix 1

m	Frisbee mass
A	Frisbee planform area
d	Frisbee diameter
ρ	air density
I_d	diametrial inertia
I_a	axial inertia
$\boldsymbol{\omega}$	total angular velocity of disc
v	velocity of the center of mass
L	lift force
D	drag force
$C_l = C_{lo} + C_{la}a$	lift coefficient
$C_d = C_{do} + C_{da}(a-a_{eq})^2$	drag coefficient
$C_{lo} = 0.188$	lift coefficient at $a = 0$
$C_{la} = 2.37$	lift coefficient dependent on a
$C_{do} = 0.15$	drag coefficient at $a = a_{eq}$
$C_{da} = 1.24$	drag coefficient dependent on á
$a_{eq} = -C_{lo} / C_{la}$	alpha at zero lift and minimum drag
R, P, S	rolling, pitching, and yawing moments
$C_{Mo} = -0.06$	pitching moment coefficient at $a = 0$
$C_{Ma} = 0.38$	pitching moment coefficient dependent on alpha
$C_{Mq} = 0.0008$	pitching moment damping coefficient
$C_{Rr} = 0.0004$	roll moment coefficient due to spin r
$C_{Rp} = -0.013$	roll moment damping coefficient
$C_{Nr} = -0.000028$	spin moment damping coefficient