|  |
| --- |
| PHYS1521  **Math and Physics for Games**  Realistic Projectile  Simulation Report  Digital Media and IT  School of Applied Sciences and Technology |

|  |  |
| --- | --- |
| **Name(s):** | Alex Kinnear and Pierre Gravelle |
| **Section:** | A01 |
| **Date:** | March 22nd 2017 |
| **Instructor:** | Allan Anderson |



*Fig. 1.* Screenshot of Projectile Motion Flash Simulation.

From “Projectile Motion” by Splung.com at <http://www.splung.com/content/sid/2/page/projectiles>

Table of Contents

[Introduction 1](#_Toc478982600)

[Concept 1](#_Toc478982601)

[Drag and Lift 2](#_Toc478982602)

[What Is Drag? 2](#_Toc478982603)

[How do We Calculate Drag? 2](#_Toc478982604)

[What is Lift? 3](#_Toc478982605)

[How do we Calculate Lift? 3](#_Toc478982606)

[How Are These Relevant to Game Programming? 4](#_Toc478982607)

[Torque 4](#_Toc478982608)

[What Is Torque? 4](#_Toc478982609)

[How Is It Relevant to Game Programming? 4](#_Toc478982610)

[Conclusion 4](#_Toc478982611)

[References 5](#_Toc478982612)

**Table of Figures**

[Fig. 1. Screenshot of Projectile Motion Flash Simulation. i](#_Toc478982430)

[Fig. 2. Table of Drag Coefficients 2](file:///C:\Users\Pierre\Documents\GitHub\PhysicsResearchandSimulationProject\A01%20-%20Alex%20and%20Pierre%20-%20Report.docx#_Toc478982431)

[Fig. 3. Drag Force Formula 2](file:///C:\Users\Pierre\Documents\GitHub\PhysicsResearchandSimulationProject\A01%20-%20Alex%20and%20Pierre%20-%20Report.docx#_Toc478982432)

[Fig. 4. Air Density Formula 2](file:///C:\Users\Pierre\Documents\GitHub\PhysicsResearchandSimulationProject\A01%20-%20Alex%20and%20Pierre%20-%20Report.docx#_Toc478982433)

[Fig. 5. Air Pressure Formula 2](file:///C:\Users\Pierre\Documents\GitHub\PhysicsResearchandSimulationProject\A01%20-%20Alex%20and%20Pierre%20-%20Report.docx#_Toc478982434)

[Fig. 6. Lift, Drag, and Velocity’s Relation 3](#_Toc478982435)

[Fig. 7. Lift Formula 4](file:///C:\Users\Pierre\Documents\GitHub\PhysicsResearchandSimulationProject\A01%20-%20Alex%20and%20Pierre%20-%20Report.docx#_Toc478982436)

# Introduction

Projectiles are moving objects that have plenty of physics associated with them. Our project is to create a virtual and realistic simulation involving launching projectiles at several angles. We chose this topic because it is familiar to us due to our Math and Physics for Games class at NAIT. We also see this as an excellent opportunity to translate our calculations into visual results.

There are plenty of video games that use projectile-like assets. The physics behind these assets are a lot more complicated than they may seem at first. For example, the game “Angry Birds” uses birds as projectiles and they are affected by multiple forces. All these forces must be programmed in, otherwise the projectile may act differently than a user may expect. For example, a projectile may slow down faster than a user expected, hence ruining their attempt at the shot.

This report will highlight our efforts towards creating a realistic projectile simulation and will go into detail on how each physics concept involved affects the projectile.

# Concept

The simulation will involve two different scenarios. One scenario will have the projectile being launched from a cannon, and the other scenario having a ball being flung after multiple circular rotations around a center point. Our simulation will allow the user to change some of the variables in the simulation using a menu, this will demonstrate how well our mathematical calculations will react to the changes instead of being hard-coded values.

In our work, gravity’s acceleration will always be considered as -9.81 m/s2 as this is the constant we have been using in our physics class.

We’ll be using multiple concepts in our simulation. The concepts that we’ll be using that we’ve already learnt about in our Math and Physics for Games class include:

* Momentum Conservation between two colliding objects
* Linear Projectile Motion
* Rotational Projectile Motion

The following concepts are new to our group and will be the key points in our report. They will be explored in detail later:

* Torque
* Drag Force and Lift

## Torque

### What Is Torque?

Torque is a force that causes an object to rotate.

How Is It Relevant to Game Programming?

## Drag and Lift

### What Is Drag?

**Drag** (also known as **Air Resistance**) is a force that pushes in the opposite direction of an object’s velocity. Air resistance is dependent on an object’s velocity (Rit.edu, n.d.). So, the faster an object is moving, the more air resistance it will have. For example, a cube being dropped from a 500m high building will start with very little air resistance. However, as the object falls it starts to accelerate and the air resistance becomes a lot stronger.

Since air resistance is a force moving against our velocity, this means that the object’s speed will eventually hit a maximum value. The velocity of the object will no longer increase, and the air resistance will stay constant as well.

Drag is also dependent on the shape of the object being used. A sphere for example, will have less air resistance than a cube would. The reason for this is the **Drag Coefficient**.

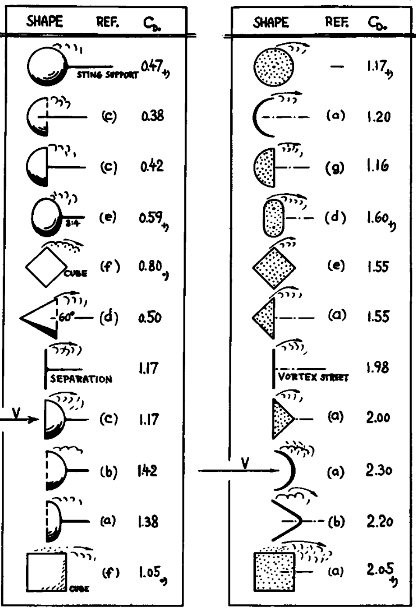
The Drag Coefficient is a number that represents how much drag an object will have; it is dependent on the shape of the object being used. (engineeringtoolbox.com, n.d.). See Fig. 2 for a table full of drag coefficients and the shapes they are associated to.

Fig. 2. Table of Drag Coefficients

From “Fluid-Dynamic Drag” by Sighard Hoerner (1965)

Modern vehicles take drag coefficient into serious consideration, that’s why they have a very smooth and aerodynamic shape. Patrick E. George from HowStuffWorks.com gives the example of the Toyota Prius. He writes:

*“Among other efficient characteristics, its Cd of .26 helps it achieve very high mileage. In fact, reducing the Cd of a car by just 0.01 can result in a 0.2 miles per gallon (.09 kilometers per liter) increase in fuel economy.”*   
- Patrick E. George (March 2009)

### How do We Calculate Drag?

To find the drag force, we must use the Drag Force formula (See Fig. 3.). The formula takes factors into account that we haven’t seen in class, such as **air density**, and the **frontal area** of the object.   
The density of the air that the object is travelling through is measured in kg/m3 (kilograms per cubic meter). It is calculated using the Air Density formula (See Fig. 4) However, as you may have seen from the figure, to find the density of the air we need to know the air pressure. Lucky for us, there’s another formula (See Fig. 5) to calculate this.

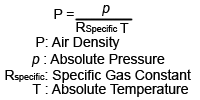
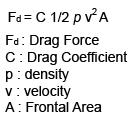
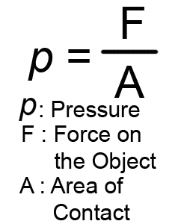


Fig. 3. Drag Force Formula

By Pierre G. (2017)

Fig. 5. Air Pressure Formula

By Pierre G. (2017)

Fig. 4. Air Density Formula

By Pierre G. (2017)

### What is Lift?

Lift is a force that carries an object upwards while travelling high speeds. Some may think it’s the opposite of drag, however that isn’t the case. Drag goes in the opposite direction of velocity, while Lift helps carry the velocity of the object upwards. In the case of a perfect sphere, it would have to rotate to have lift. The reason for this is that a perfect sphere would evenly distribute the air around it, making the pressure of the air above the sphere and the pressure of the air bellow the sphere equal. Equal pressure means there’s no lift force.

Lift’s direction is dependent on the velocity’s direction. You can see this in the diagram I created bellow (Fig. 6.). As velocity moves downwards, the lift of the object follows along with it. Drag also follows along because as stated before, drag is a force that moves in the opposite direction of velocity.

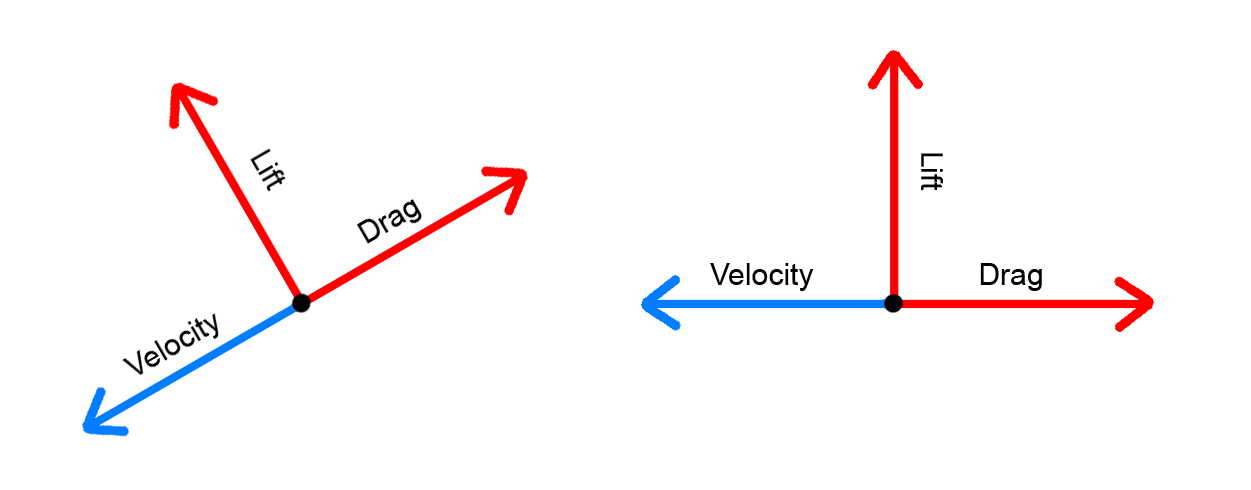


Fig. 6. Lift, Drag, and Velocity’s Relation

By Pierre G. (2017)

### How do we Calculate Lift?

In order to calculate Lift we’ll need to use multiple formulas from the Torque section seen of this report.

First, we need to find the Alpha that’s being applied to this object. We can find this by rearranging the formula for Sum of Torque seen in the torque section. Don’t forget that Alpha is a symbol representing Angular Acceleration and that it’s calculated in rads/s2.

Once we have calculated the Alpha of this object, we are now able to calculate the Omega of the rotation, which is the Angular Velocity (also known as rotation speed) in rads/s. All we need is to know the amount of time that the object is rotating for.

We need this Angular Velocity to find the revolutions that the object does around its pivot point per second. This is also known as Revolutions per Second (RPS). In order to convert our rads/s into rev/s we need to understand that a full revolution (rotation) in radians is equivalent to 2π. So we simply need to divide our Omega by 2π in order to find our RPS.

We’re almost done. Now all we need to do is use the Lift formula seen below in order to calculate lift!

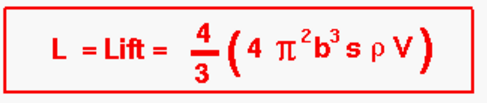


Fig. 7. Lift Formula

From “Ideal Lift of a Spinning Ball” By Nancy Hall

In this formula **b** is equal to the radius of the object, **s** is equal to the RPS that we just found, **p** is equal to the air density (which we can find using the formula shown in Fig. 4.), and finally **V** is the velocity that the object is travelling at.

### How Are These Relevant to Game Programming?

In video game programming, the developers may want the most realistic physics possible. Calculating the drag force and lift force increases the realism of the video game. Plenty of video games are dedicated to using realistic physics, such as the Battlefield series. Bullet trajectories are calculated realistically. A larger projectile (like a missile from a rocket launcher) falls quicker and doesn’t go as far as a small bullet does.

Developers may also base the concept of their game on physics as well. In games that use physics to solve puzzles, the player should know what to expect. Realistic physics due to drag force and lift can make a drastic difference between a “Game Over” screen and a “You Win!” screen.

# Conclusion

Summarize the report by restating the reason for this topic and how the key points (covered above) make this topic relevant to Game Programming.

# References

Put all your references here.