CT5 MEGA Report

Word Count: 1501

# Background

Sniper Ballistic systems in games are systems which effect the projectiles based on factors such as wind rain and gravity as well as mass of object, these features are rarely done as a major feature, the two most notable games that have them as a major feature are Sniper Elite 4 (Rebellion Developments, 2017) and Sniper: Ghost Warrior 3 (CI Games, 2017). The problem is creating a system which effects Projectiles based on physical inputs in a realistic way this Projectile must go through a 3D environment checking for collision and receiving new forces regularly from gravity or wind. Sniper Elite 4 (Rebellion Developments, 2017) system is based on wind and gravity the player can set the range of their scope, and then the Projectile will go into a parabolic arc, these systems will use vector 3s and floats behind the scenes to cause the effect to work and may even use quaternions. For my system my chosen system is to have a vector 3 represent the force of this object, forces like gravity and wind will be added to the vector 3. Using delta time I will then use a semi implicit physics system to apply these forces to the object while making it realistic, dynamic fluid simulation including atmosphere and air Resistance is outside my scope. I am very satisfied with the requirements to build a ballistic system using vector 3s as it enables me to have a modular system which I can add and remove components to.

# Analysis

## Requirement 1 (Moving an object)

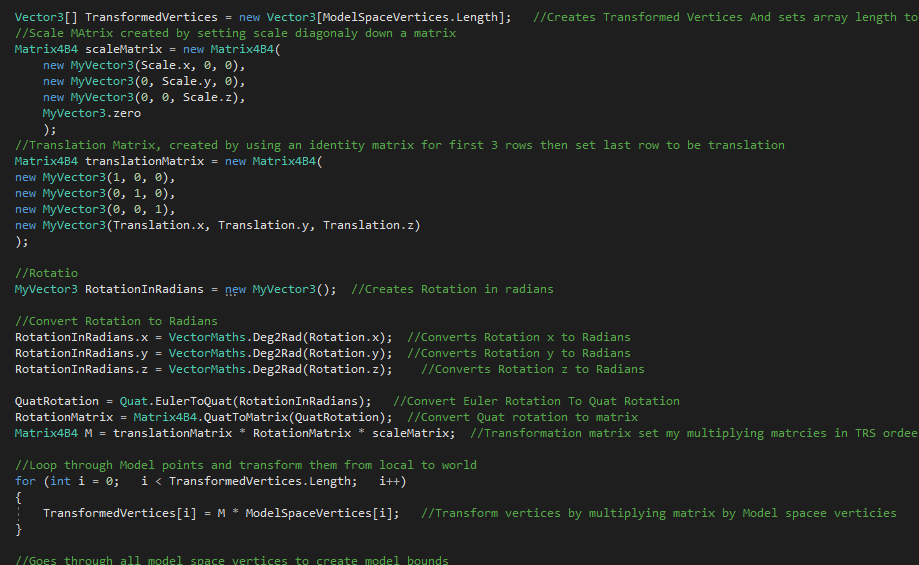
There are two ways in which a Projectile simulation could be done the first is as a Quaternion using SLERP the second is as a Vector 3 which can be added or subtracted to move the Projectile. For movement you only need basic operators such as Addition, subtraction, Multiplication and division you would not need dot products or cross products. I have chosen to do a Vector 3 as it is easier to edit in code and is more compatible with the rest of the project, a Quaternion will take more processing power to handle with its operators doing more maths a vector 3 is more efficient at representing movement. To move a player you must first work out their forward direction, this is done using by converting the rotation to a direction using Euler angles to direction however this function needs to be used with care as the code is dependent on what co-ordinate system is used. To increase computational power I have overloaded the function to work with a vector 2 which returns x and z as the player is most likely on the ground. I then used the cross product of up and forward direction to get the right direction. This will make sure that when I moving right it is from the rotation not just X as default. The right cross product has limitations and needs to be normalised otherwise when you like directly up or down moving left or right is slow/negligible.

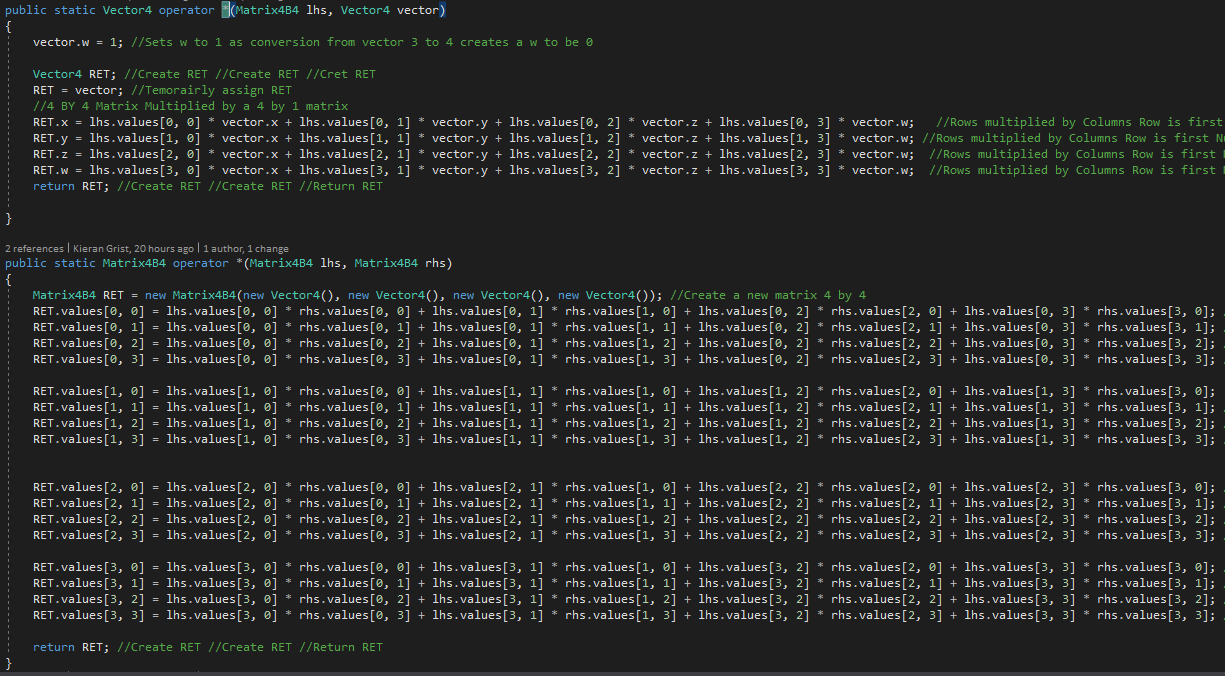
To constantly move an object you need to use addition and make sure that if it’s controlled a forward direction is constantly being calculated for that object otherwise mistakes can occur.

You then need to apply this position to a matrix to do so you need to create a 4 by 4 matrix, this matrix will then be filled with an identity matrix, with the last row being the translation matrix. Using TRS Order you then multiple the matrices together.

Source Code



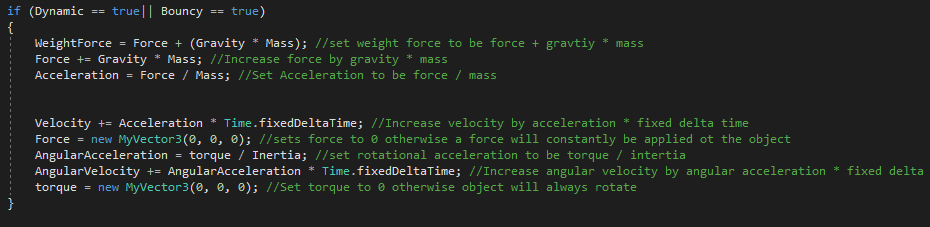


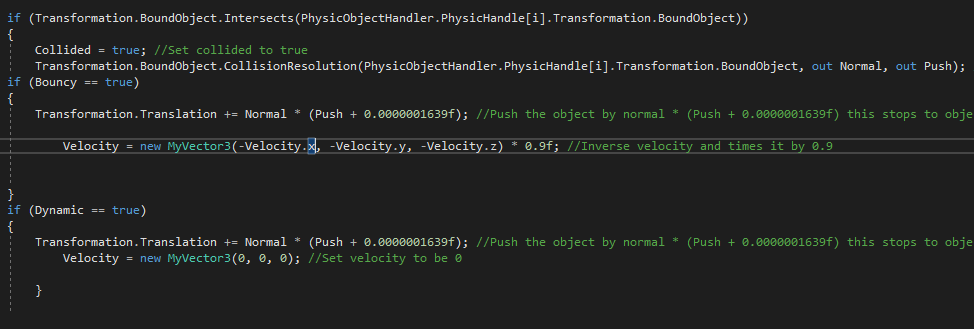


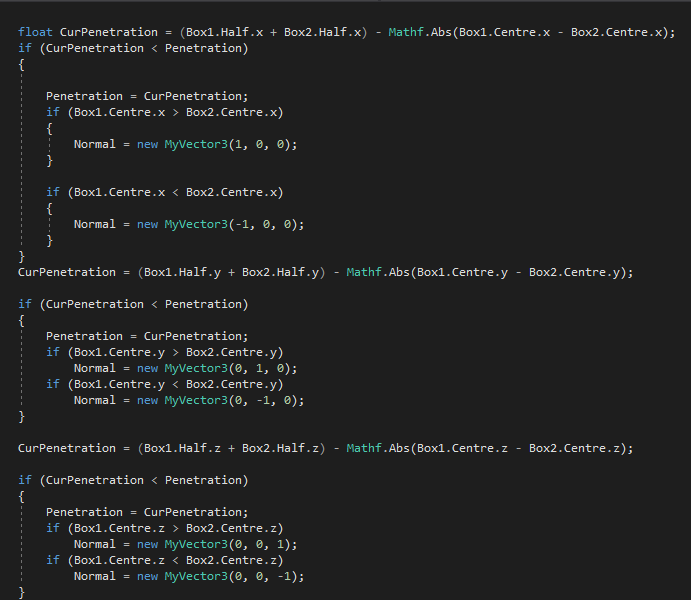
## Requirement 2 (Physical Simulation)

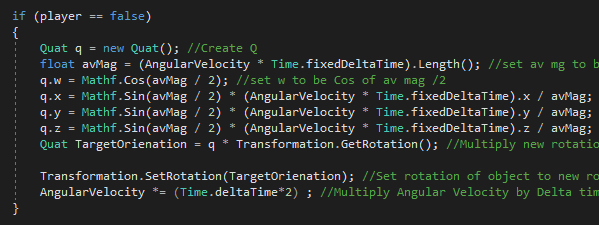
For physical simulation there are many potential systems based on how complex the simulation is going to be. The choice I had was between RK4 and Semi implicit Euler integration. I choose semi-implicitly because it is easier to implement for a small focused project like mine and does not take too much processing on the CPU this also worked well with my vector 3 transformation system. I used Euler angles to direction to get a forward direction to apply the force needed for the Projectiles using += will make sure that any external force like wind is applied correctly. To simulate physics you need to make sure that gravity is being applied correctly to do this you must make gravity a force which is multiplied by mass, without this calculation objects will be pulled at different speeds which is not accurate. As David Scott (BBC, 2017) proved on the moon without an atmosphere effecting an object two objects with differing mass fall at the same speed due to having no air resistance or fluid dynamics effecting them. I also used concepts such as inversing a vector 3 to achieve a bouncy ball. I then focused on rotational physics which uses advanced concepts like getting and setting rotation from matrices. For rotational physics I apply the angular velocity to a new quaternion by getting the magnitude of velocity multiplied by time and dividing it by two then applying the sin function to it. This is then times by Angular Velocity \* Time divided by mag which is repeated for y and z. To get the rotation in quaternions you need to perform the Euler to quaternion code, first the rotation needs to be in radians. The video explains how the function works. With the rotation now stored I need to multiply it by the new quaternion calculated before. Multiplying a Quaternion can be done multiple ways (my way can be seen in the source code). Setting the rotation is a complicated process using multiple concepts to do this you need to convert a quaternion to Euler. First you need to make sure that all values are numbers otherwise the function will break, you then need to convert the quaternion to a matrix. You then need to transpose the matrix which flips the order of rows and columns. Not transposing your matrix causing issues with the rotation with the values being in the wrong place. You then check if the values are singular or not by square root [0, 0] Squared + [1, 0] squared. Once this is done you then get the atan2 of the values once the vector is set you need to times it by rad 2 degrees to make sure it is correct.

Source Code:









# Reflection

One of the issues I encountered was getting the physics correct while it works as expected the physics system does not simulate an atmosphere yet still has wind which is not a full accurate system. If I was going to do this project again I would look into physic systems more early and make sure I had an understanding earlier on so I could implement the harder to understand concepts.

The next issue I encountered was rotational physics for 2 weeks I was trying to implement this system trying different websites and solutions until I tried to transpose the matrix and found that the code I had worked. When copying code from books or websites it is important to make sure you keep all of the code in the function to insure it is working correctly and accurately.

On hindsight an approach I could of taken for the project was to put all my maths functions in the my vector 3 and vector 2 classes so I could call them more easily and use more of c# functionality such as using get and set functions to stop the need for () and create more modular code.

Another approach I should of taken was have rotation be a quaternion and if I needed to set it with an Euler angle I could do so with function calls instead of having rotation constantly be effected by a vector 3 creating issues down the line with rotation and quaternions.

To improve the project I would: Create fluid dynamics simulation, fix collision resolution, implement accurate rotational physics and collision geometry to mash, Stop the player from jumping if they are in there, fix the sniper rifle firing and deleting the Projectile at times, create a scope, allow for stances, add rain simulation, add weather simulation, add air pockets back in, add spatial partitioning to have more physic objects in game, have weapons with pre-set stats for Projectile weight and firing speed and interaction with the rigid bodies being able to move around the objects and for them to fall land rotate correctly.

This project is a good starter show of someone who knew nothing of the maths before but has its limitations and hacks, I have implemented functions for concepts like dot product but have failed to use them in my project and have not used as many advanced concepts as I would like. I have also failed to implement more advanced collision detection techniques like rotational boxes and convex hulls. I feel my performance was as best as it could be with the struggles I had during the project and the lack of understanding I implemented everything on my own with little to no help and found the project was rewarding and would like to do something similar with less code snippets.

# References

BBC. (2017, May 10). *Moon feather and hammer drop*. Retrieved from www.bbc.co.uk: https://www.bbc.co.uk/programmes/p052jk41

Bobic, N. (1998, July 5). *Rotating Objects Using Quaternions*. Retrieved from www.gamasutra.com: https://www.gamasutra.com/view/feature/131686/rotating\_objects\_using\_quaternions.php?page=2

CI Games. (2017, April 25). Sniper: Ghost Warrior 3. *Sniper: Ghost Warrior 3*. Warsaw, Poland: CI Games.

Epic Games. (2018, Novemeber 7). Unreal Engine 4. *Unreal Engine 4*. Cary, North Carolina, United States Of America: Epic Games.

Ericson, C. (2004). *Real-Time Collision Detection.* London: CRC Press.

Mallick, S. (2016, June 4). *Rotation Matrix To Euler Angles*. Retrieved from www.learnopencv.com/: https://www.learnopencv.com/rotation-matrix-to-euler-angles/

Rebellion Developments. (2017, Febuary 14). Sniper Elite 4. *Sniper Elite 4*. Oxford, England: Rebellion Developments.

user1423893. (2012, June 16). *Quaternion to Euler angles algorithm - How to convert to 'Y = Up' and between handedness?* Retrieved from www.stackoverflow.com: https://stackoverflow.com/questions/11492299/quaternion-to-euler-angles-algorithm-how-to-convert-to-y-up-and-between-ha/11505219