MDU113.2 - Projectile Motion

**Sethiel Jann Cosio**

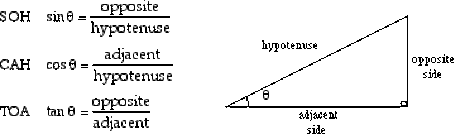
**DUB11735**

To make the projectile motion possible, the projectile’s overall horizontal and vertical velocities had to be calculated using the equations:

F = m \* a **(The Zooland Education, n.d.)**

vf2 = vi2 + 2 \* a \* S **(Garber, G., n.d.)**

As well as Pythagoras’ theorem on finding the horizontal and vertical velocities with the angles in triangular form:



SOHCAHTOA **(Pennysu's Blog, n.d.)**

Rearranging these formulae allowed me to find velocity by rooting the force and adding it to the sine and cosine of the launch angle to find the overall velocities. Within the same function, I used Quaternion.lookrotation to make sure the projectile doesn’t rotate erratically while in flight.

To launch the projectile, I used an if statement to check the position of the ball in the y and z planes to allow the ball to fire but stop moving once it no longer meets the conditions. Within this same if statement, I used the myProjectile.Translate to project the ball using the velocities the were calculated and finding their distance using the distance equation:

S = V \* T (**Velocity Time and Distance Ultra Calculator, n.d.)**

This multiplies the Velocity with Time (Time.deltaTime) and translate the object accordingly. X is not considered as the horizontal plane, so it will remain as 0, instead, Z is considered as the horizontal and is calculated normally since the horizontal velocity remains constant throughout the whole projection. For the Y axis, since it constantly changes from initial to final location, I had to reduce from the initial velocity by multiplying the set downwards acceleration (gravity) with the time that passed since the start of the projection, allowing the projectile to travel in a curved manner instead of just straight up.

To calculate the damage done on the enemy prefab, I used if statements to check the distance between the projectile and the enemy. The enemy would receive damage based on that distance using the inverse-square distance law where the overall damage done is equal to the actual damage divided by the square of the distance between the enemy and projectile.

**References**

**The Zooland Education. (n.d.). The Mighty F = ma. Retrieved April 07, 2018, from** [**http://zonalandeducation.com/mstm/physics/mechanics/forces/newton/mightyFEqMA/mightyFEqMA.html**](http://zonalandeducation.com/mstm/physics/mechanics/forces/newton/mightyFEqMA/mightyFEqMA.html)

**Garber, G. (n.d.). Gary Garber's Blog. Retrieved April 07, 2018, from** [**http://blogs.bu.edu/ggarber/archive/bua-physics/kinematics-equations-and-constant-acceleration/**](http://blogs.bu.edu/ggarber/archive/bua-physics/kinematics-equations-and-constant-acceleration/)

**Pennysu's Blog. (n.d.). TRIGONOMETRY. Retrieved April 07, 2018, from** [**http://pennysu.weebly.com/trigonometry.html**](http://pennysu.weebly.com/trigonometry.html)

**Velocity Time and Distance Ultra Calculator. (n.d.). Velocity Time and Distance Ultra Calculator. Retrieved April 07, 2018, from** [**http://www.1728.org/velocity.htm**](http://www.1728.org/velocity.htm)