26/04/2017

[redacted]
Computer Games
Technology

PhySim Projectile Motion Library

Supervisor : Silvester Czanner

Why?

- Project Origin
  - How Game Engines manage physics?
  - Vectors?
  - Projectiles...
- PhySim Projectile Library
  - Do different game engines work differently?
  - Are Simulations Better?
  - Try to make both.
- Front end?

- Library
  - Game speed over accuracy
  - Simulation accuracy over speed
- Front End
  - OSG?
- What next?
  - Library development
  - Front End development
  - Result/Outcome
- Conclusion
  - Challenges and Improvements

Aims

Aim

- Create a library to update physics of projectiles/particles.
- Realistic projectiles
- Non realistic object manipulation

Objectives

- Identify / Investigate current solutions.
- Implement Physics library
  - Fundamental components
  - Configuration structure
  - Output Structure
  - GUI Demo
- Extensible.

Literature

Key Sources Engines.

- Cyclone Engine.
- Unity.

- Language Selection?
  - C++
  - Open GL

Mathematics

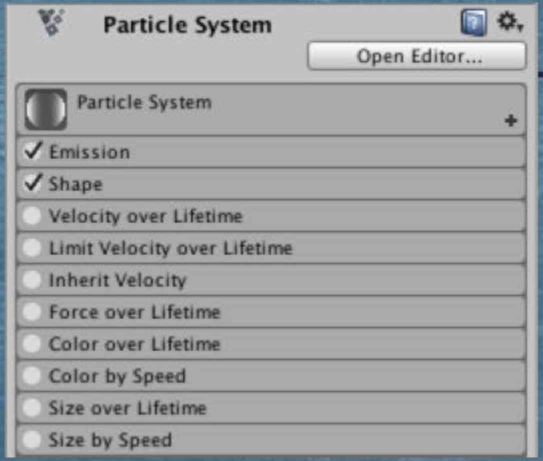
- Differentiation
- Integration
- Newtons 2nd Law
- D'Alemberts Principle
- Efficiency

What is Consistent between Cyclone and Unity?

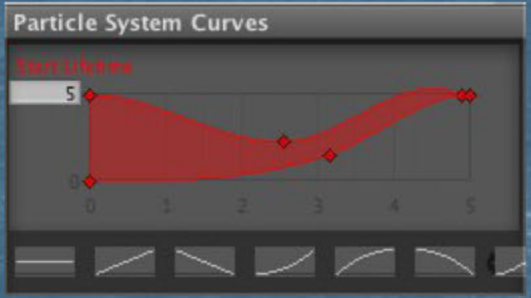
- Particle System
- Single
- Swarm

- Rigid Body Physics
  - A to B always = B to A

- Collision System
  - Detect Collision
  - Resolve Collision



✓ Emission
✓ Shape
○ Velocity over Lifetime
○ Limit Velocity over Lifetime
○ Inherit Velocity
○ Force over Lifetime
○ Color over Lifetime
○ Color by Speed
○ Size over Lifetime
○ Size by Speed



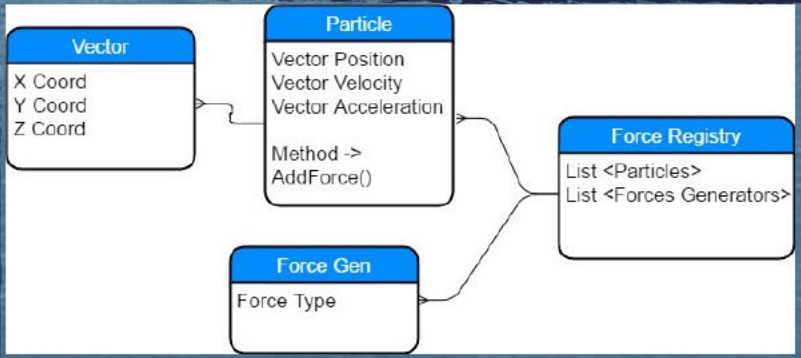
Core Procedure

- Understand good code practice.
  - OOP
  - Math
  - Memory

- Create Key Attributes
  - Vector
  - Particle
- Implement Vector functionality.
- Implement Particle functionality.

Design

- Vectors
- Particles
- Forces
- Management?



Vector

X Coord
Y Coord
Z Coord

Particle

Vector Position
Vector Velocity
Vector Acceleration

Method ->
AddForce()

Force Registry

List <Particles>
List <Forces Generators>

Force Gen

Force Type

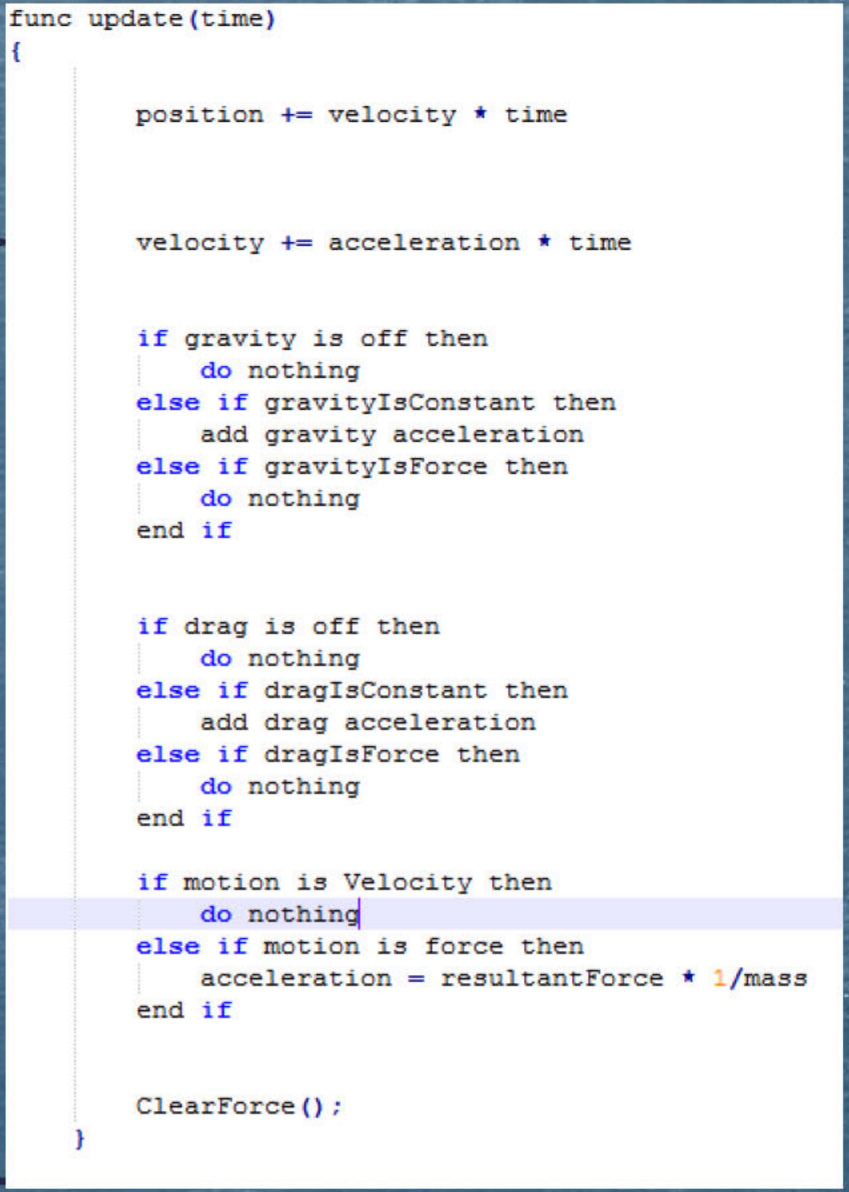
Control System

- Enums
  - Gravity
  - Drag
  - Motion

- Gravity
  - Off
  - Constant
  - Force
- Drag
  - Off
  - Constant
  - Force
- Motion
  - Constant Vector
  - Force Based

Physics

- How do the settings work together?
- Update Function.



```
func update(time)
{
        position += velocity * time


        velocity += acceleration * time

        if gravity is off then
            do nothing
        else if gravityIsConstant then
            add gravity acceleration
        else if gravityIsForce then
            do nothing
        end if


        if drag is off then
            do nothing
        else if dragIsConstant then
            add drag acceleration
        else if dragIsForce then
            do nothing
        end if

        if motion is Velocity then
            do nothing
        else if motion is force then
            acceleration = resultantForce * 1/mass
        end if


        ClearForce();
    }
```

Forces

- Interface Force.
  - Update
- Instance Force
  - Linear force
  - Drag force
  - Gravity force

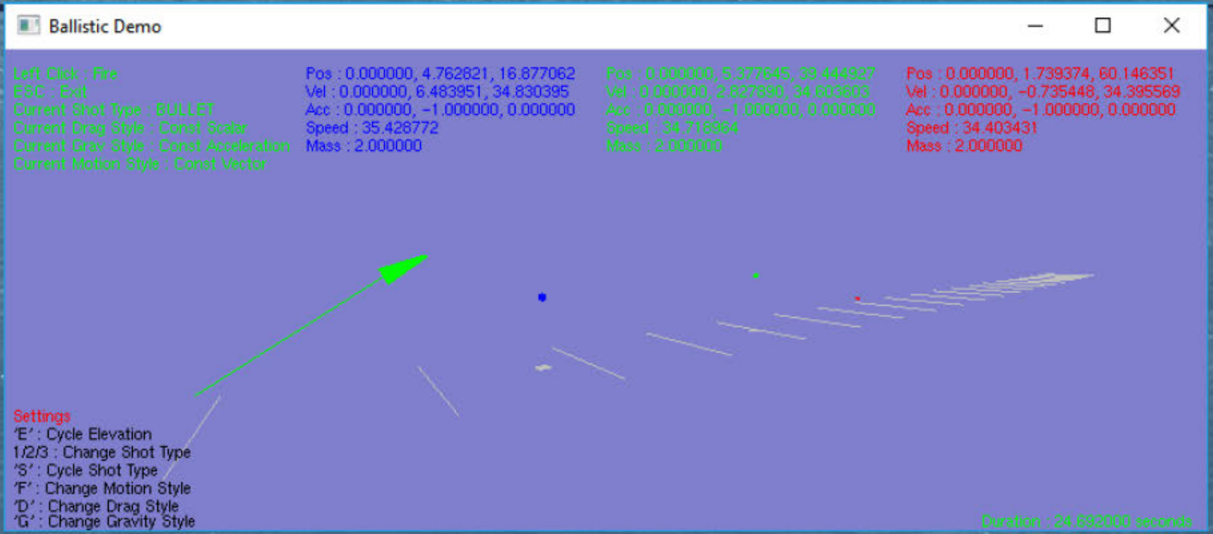
- Force Register
  - Force Instance
  - Particle (Pointer)

World

- Control
  - Liaison to PhySim
- Contained
  - Force Register.
  - Particle Register
  - Shot Structure/Register
- Shot?
  - Particle
  - Object type
  - Color
  - Status
- Update
  - Force Update
  - Particle Update

GUI

- Open GL
  - Display
  - Update
  - Fire



```
Left Click : Fire
ESC : Exit
Current Shot Type : BULLET
Current Drag Style : Const Scalar
Current Grav Style : Const Acceleration
Current Motion Style : Const Vector
```

```
Pos : 0.000000, 4.762821, 16.877062
Vel : 0.000000, 6.483951, 34.830395
Acc : 0.000000, -1.000000, 0.000000
Speed : 35.428772
Mass : 2.000000
```

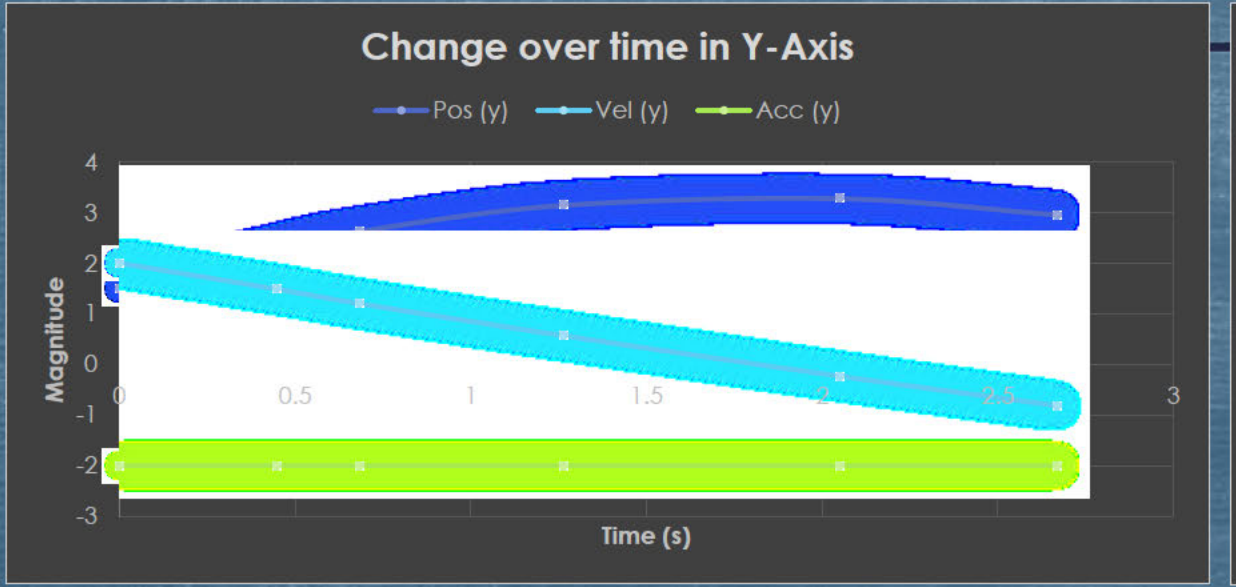
```
Pos : 0.000000, 5.377645, 39.444927
Vel : 0.000000, 2.827890, 34.603603
Acc : 0.000000, -1.000000, 0.000000
Speed : 34.718964
Mass : 2.000000
```

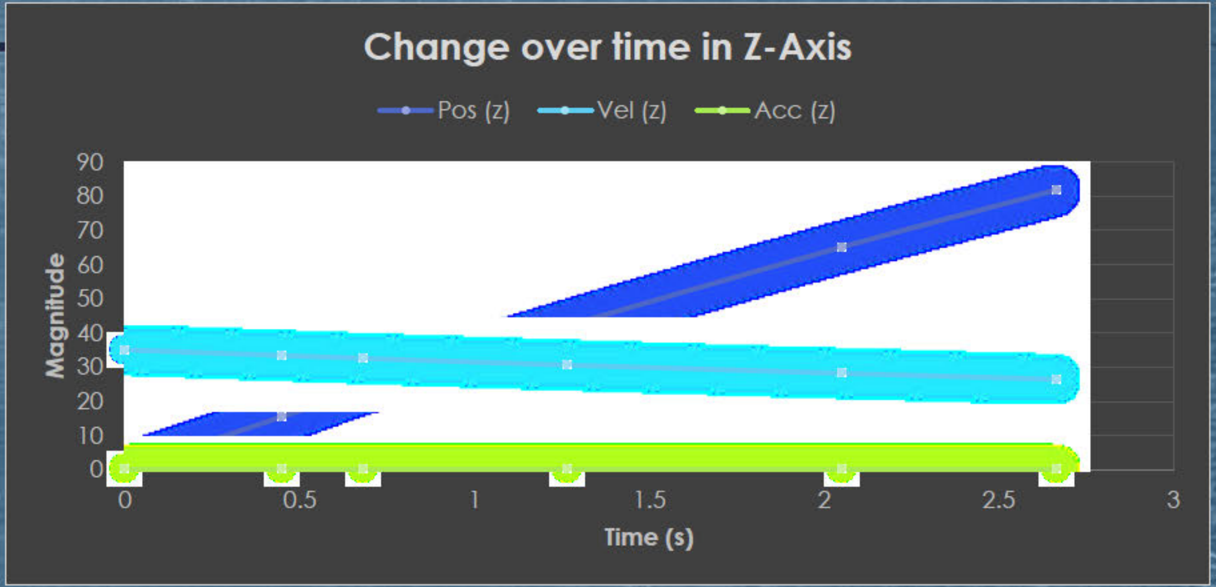
```
Pos : 0.000000, 1.739374, 60.146351
Vel : 0.000000, -0.735448, 34.395569
Acc : 0.000000, -1.000000, 0.000000
Speed : 34.403431
Mass : 2.000000
```

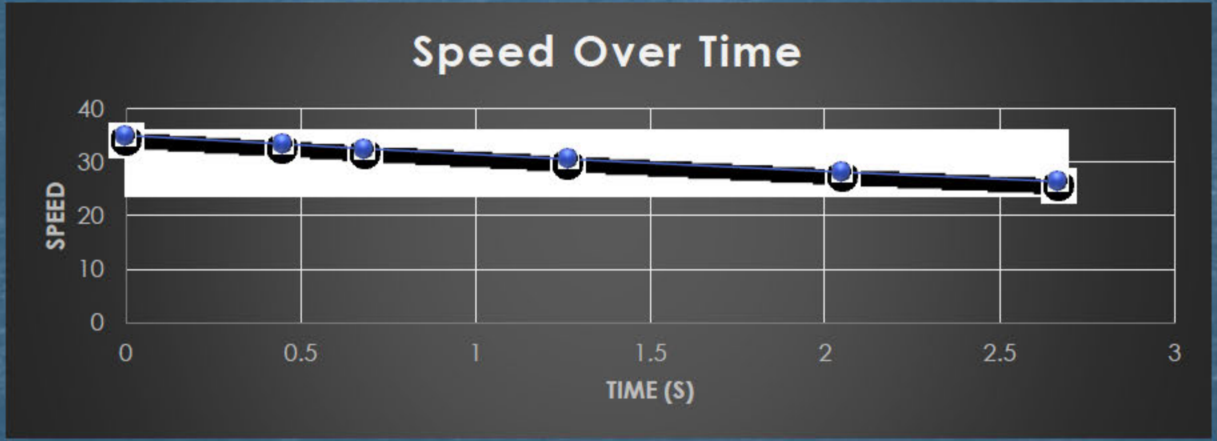
```
Settings
'E' : Cycle Elevation
1/2/3 : Change Shot Type
'S' : Cycle Shot Type
'F' : Change Motion Style
'D' : Change Drag Style
'G' : Change Gravity Style
```

```
Duration : 24.692000 seconds
```

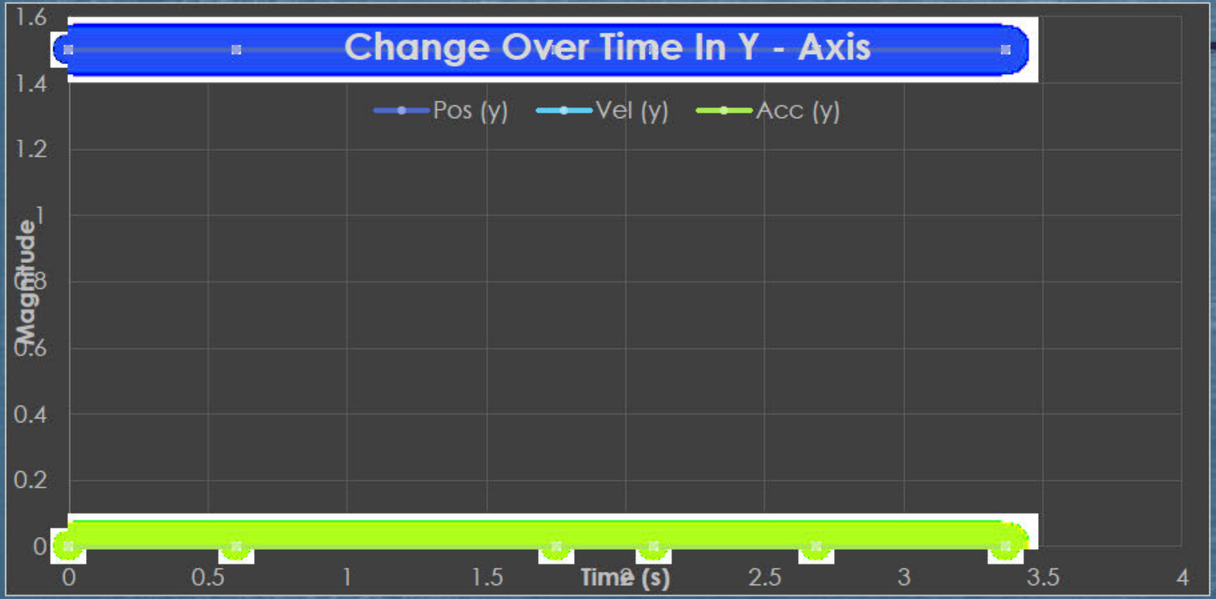
Constant Results

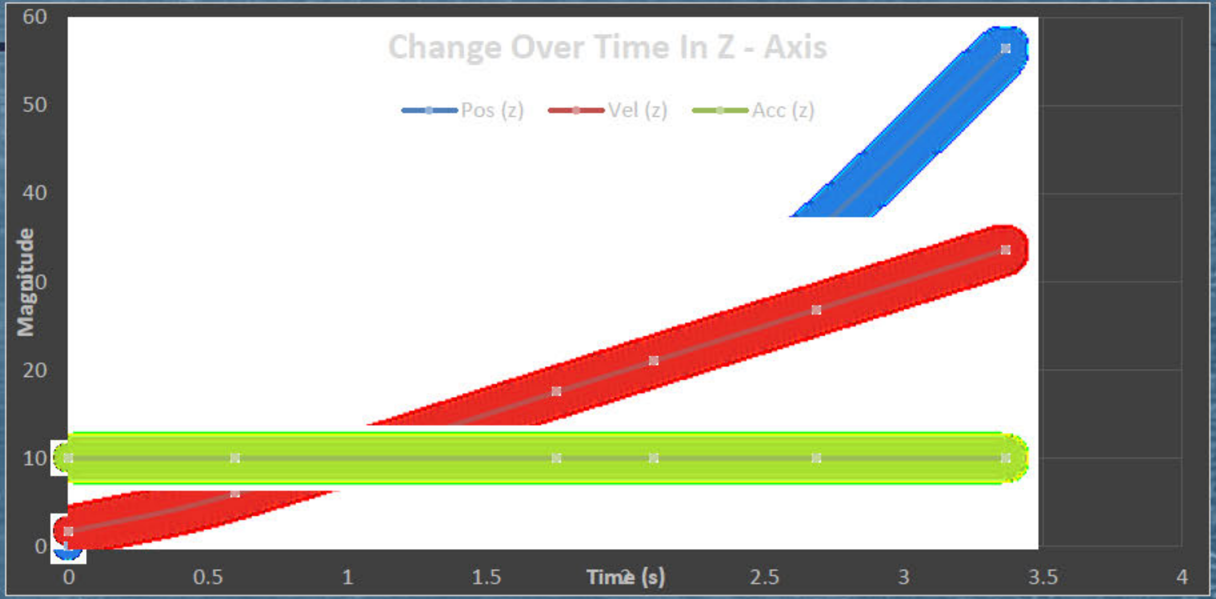


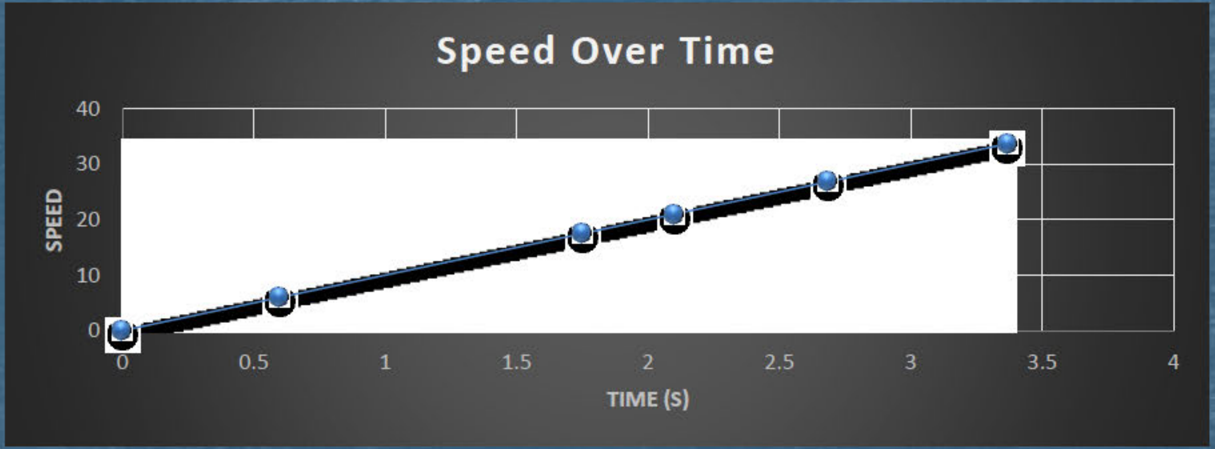




Vector Results







- Issue?
  - Acceleration over time
  - Drag proportionality

Challenges

- Code Walkthrough
- Test Results
- Implementation

- Syntax use
  - <Vector> class
  - Pointer Referencing
  - Open GL Syntax

Future Changes..

- Drag Equation
- Namespaces
- Rigid Bodies
- Collision detection

Conclusion

- Did I map projectile motion?
  - Yes
- Did I implement a stable control system?
  - Yes
- Did levels of control system reflect true accuracy?
  - No
- Could accuracy be added in the future?
  - yes

Questions?