Anti-Anti Air System

Ozan Cengiz
Electrics And Electronics Engineering
Ankara University
ozancengiz00@gmail.com

I. INTRODUCTION

This report is prepared by Ozan Cengiz, for the Electrics And Electronics Engineering Department of Ankara University. This report is to introduce a system and its simulation, called Anti-Anti Air System, designed in MATLAB. Anti-Anti Air System aims to find a solution or fina a way about avoidances of the Anti Air Systems. Yet it has no professional usages, or works perfecly, or performs %100 success rate, it can be optimized to reach much higher performances. Despite everything, this simulation has proven that it is promising and it can find a work place for itself.

II. REQUIREMENTS

This simulation is designed as a MATLAB source code, so MATLAB program should be installed (preferably version R2020B) and a license is required. For the user side, various mathematics, MATLAB and physics topics' knowledge is required. User should be familiar with measurement errors, Monte Carlo Simulations, Gaussian Distribution, Standard Error Distribution, projectile motion, trigonometry, 2D coordinate system, basic MATLAB knowledge and so on. User also can work on the simulation previously, and try different inputs to see the different effects since the simulation does not work %100 success rate and also, there is no another proper sample of this project on the internet.

III. A BRIEF SUMMARY OF PROJECT AND SIMULATION

As mentioned, this project covers the topics mentioned at previous part. One level deeeper expression of this project is there are two objects i.e. red object and blue object, blue object tries to hit red object in the before red object lands.

IV. EQUATIONS AND FORMULAS

Position at time t: $x = (v_0 \cos\theta_0)t$ (1)

Position at time t: $y = (v_0 \sin\theta_0)t - 1/2\ gt^2$ (2)

Time of flight: $2t_m = 2(v_0 \sin\theta_0/g)$ (3)

Maximum height of projectile: $h_m = (v_0 \sin\theta_0)^2/2g$ (4)

Horizontal range of projectile: $R = v_0^2 \sin 2\theta_0/g$ (5)

Distance between points:

$$d=\sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)} \quad (6)$$

V. COMPLETE EXPLANATION

There are two objects, one is for the red side and one for the blue side. While red side tries to hit ground that is close to blue, blue side tries to stop red side in the mid-air, by colliding it. So far, one of the simple and fundamental objective is completed that is expected from this project. Which is: "An object tries to hit another object, which makes projectile motion.". But after this simple objective, this is where the tricky part starts for both objects, also for the user. Red side has an air radar embedded in their object, but at the other side, blue has some imaginary displaying. Blue has some erroneous observations by the air radar side.

Radar scans for another objects(blue), if there will not be any object scanned, red keeps continuing its projectile motion and goes to its target(case 1). If radar scans blue, that is where the blue's approaching angle is important by the red's next move. As vertical as blue approachs red, red will want to stay as far as possible from blue, mostly it will travel upwards and increase its speed at the same time. Then the red will make a sharp peek and start to fall faster and sharper against the target. Since it is faster now than before detecting blue, it can also travel further and miss the target(case 2 and case 3). If blue tries to pass over red, red will start to fall immidiately, at the same time, increasing its horizontal speed to increase the chance to hit the target But this increase is more than its vertical speed, so it can cause shortening of its range thus it can fall at a shorter distance and can miss the target(case 4). Of course these are the most observed cases, there can be sub-cases that are rare to be observed, since the simulation is real-timed, it is open to surprises. Another point that has to be highlighted is, radar system does not alert the red depending on red's and blue's current location or tangent of their current positions. The alertion count depends on how much space that the radar line covers of blue's area and how much time has passed during that scene. For instance, if blue enters radar's range for a very short time, red's tracjectory will only change slightly.

VI. Code Explanation, Line by Line

A. Line 1: To clear previous runs and work, i.e., CLI, previously used variable values, previous graphs.

B. Line 3-27: IMAGES AND ENVIRONMENT ATTRIBUTES

a) Line 5: Maximum time for simulation.

b) Line 7: Default gravity acceleration.

c) Line 10-11: For defining the horizontal boundary of the simulation. It equals to 1.5x of the red's traveling distance.

*d) Line 14-15: Time variables. dt is chosen 0.012977 to get approximate results of the real time. This reality is acquired by (count of iteration of while loop in 1 second) * dt = 1 second.*

e) Line 16-17: To display the current location of red's at 1, 2, 3, 4 and 5^{th} seconds.

f) Line 20: String variable for warning message of the radar.

g) Line 23-25: Image variables for the red, blue and explosion.

C. Line 30-39: TRAILS ATTRIBUTES

a) Line 35-37: This part of the code ensures that leave only one trail at every second of simulation despite the unusual value of dt.

D. Line 42-76: RED OBJECT ATTRIBUTES

a) Line 44-48: Definition of speed and angle limits for random pickings. The value of v0RedMinValue(114.39 m/s) is obtained to ensure that for red object to stay in air for at least 15 seconds. (See also Part IV., Formula (5))

b) Line 50-61: This piece of code includes some default initial values of red's motion.

c) Line 63-65: To assign a random values to v0Red and thetaRed.

d) Line 68-71: To pack the initial values of red in a better shape.

e) Line 72-73: To determine the red's range, so for the both horizontal limit of simulation and initial position of blue.

f) Line 74: This line ensures of observation of error values for red object to not exceed 1 or 2 times in a second, later will be used in an if condition.

E. Line 78-98: BLUE OBJECT ATTRIBUTES

a) Line 82-83: Initialization of speed and angle variables of blue.

b) Line 85-86: A default value set for speed and angle variables of blue.

c) Line 88-90: To pack the initial values of blue in a better shape.

d) Line 92-94: Initial position of blue.

e) Line 96: This line ensures of observation of error values for blue object to not exceed 1 or 2 times in a second, later will be used in an if condition.

F. Line 101-116 BACKGROUND

a) Line 102-104: Reading of BG.jpg file and definition of its borders.

b) Line 107: Creating a new figure i.e. background.

c) Line 109-111: Code piece to display the background image.

d) Line 114: For displaying continuously of the background image during in any loop.

G. Line 118-132: RADAR ATTRIBUTES

a) Line 119-122: Radar's initial angle, radar's rotating speed, radar's detection radius, respectively. Radar offset is to cut radar's length from the center of the red object to prevent overlapping of radar line and red object's image.

b) Line 124: A matrix to create radar stick.

c) Line 125-130: Some boolean and counters to control detection and time if there is any detection.

H. Line 137

Beginning of the loop.

I. Line 138

Definition of error values, randomly picked between -150 and 150.

J. Line 141-146: DISPLAY CURRENT TIME

a) Line 142-144: Deletes the previous displayed time.

b) Line 145: Defines the next timestamp and chooses its location on the graph, top-right corner.

K. Line 149-172: DISPLAY RED OBJECT STATS

a) Line 151-153: Deletes the previous displayed coordinates of the red object.

b) Line 154: Defines the next coordinates of the red object and chooses its location on the graph, top-right corner.

c) Line 156-158: Deletes the previous displayed speed of the red object.

d) Line 159: Defines the next speed of the red object and chooses its location on the graph, top-right corner.

e) Line 162-166: Defines the next speed of the red object n^{th} second, until 5^{th} second. Displays them at the top-left corner of the graph.

L. Line 175-184: DISPLAY BLUE OBJECT STATS

a) Line 176-178: Deletes the previous displayed coordinates of the blue object.

b) Line 179-182: Defines the next coordinates of the blue object and chooses its location on the graph, top-right corner.

c) Line 183: Defines the next speed of the blue object and chooses its location on the graph, top-right corner.

M. Line 187-245: BLUE OBJECT DATA INPUT

a) Line 189-192: Deletes the previous images of the blue object.

b) Line 194-196: Displays the blue object.

c) Line 199-203: Updates the coordinates of the blue ball with respect to dt, since blue ball will be thrown at the 5th second, if statement points to the end of 5th second.

d) Line 204-209: If time has finished 5th second, user is asked to enter the speed and angle values for the blue object. At line 209, angle is reversed around y-axis since it will be thrown from right side of the graph to left side.

e) Line 211-226: Several default values of blue object's attributes.

f) Line 229-231: Separation of vertical and horizontal speed values of blue object to ease performing the operations.

g) Line 232-234: Prevents an error to occur at the end of the simulation.

h) line 237-242: Draws the error values for blue object and ensures that they are drawn two times most in a second.

N. Line 248-279: RED OBJECT

a) Line 250-253: Deletes the previous images of the red object.

b) Line 255-257: Displays the blue object.

c) Line 259- 265: If no detection occurred by the radar, ensures that red object will continue its projectile motion.

d) Line 266-270: One of the most important part of the source code. If there is a detection by radar, red object will increase its speed by given factors: 2x at horizontal, 20x at vertical, that it tries to escape from blue object and its next possible locations.

e) Line 272-277: Draws the error values for red object and ensures that they are drawn two times most in a second.

O. Line 282-300: DISPLAY TRAILS

a) Line 284: This if statement ensures that trails displayed once in every second.

b) Line 285-287: Stores the current position of the trails, to prepare to be displayed at the next lines of the code.

c) Line 289-298: Draws the trails, red colored of the red object, blue colored of the blue object, and updates the most recently putted trails of the objects to ensure put trails once in a second.

P. Line 303-331: DISPLAY RADAR LINE

a) Line 305-308: Deletes the previous radar line.

b) Line 311: Keeps the track of distances between two objects in every iteration of loop. But not uses blue's actual location, uses its error-calculated form.

c) Line 313-317: Keep updates and rotates the radar line.

d) Line 320-325: Draws the radar line if and only if there has no collision.

e) Line 328: Draws the radar line.

Q. Line 334-368: RADAR'S DETECTION SYSTEM

a) Line 337: Checks if the blue object is within the detection radius of the radar.

b) Line 339: Checks if the radar line intersects with the blue object, 3 is just a factor here to increase the size blue object imaginarily since its real size is smaller than its .jpg image.

c) Line 340-341: If there is a detection, set detected flag to true, and wait for a while (0.2 seconds) for detecting another detection since dt is very small.

d) Line 342-345: Delete the previous warning message after 0.2 seconds.

e) Line 348: Display a warning message at the top of the red object.

f) Line 349: By setting onceDetected to true, program opens a way of using it to keep updated red object's speed values after detection, at line 266.

g) Line 351-352: Updates the last detection and warning times to ensure they are not shown up too many times since dt is very small.

h) Line 355-359: Ensures detection kept at false in any other situations.

i) Line 361-366: Keeps warning message 0.3 seconds at the screen at every time it need to be shown.

R. Line 371-390: COLLISION CHECKER

a) Line 373-380: Checks if there is any mid-air collision or red object falls to the ground, if there is, change red object's image with an explosion image to give the effect of explosion, and delete the warning message if it is still active.

b) Line 382: Sets the collisionOccured flag to true, used in different parts of the code previously.

c) Line 383-385: Stops displaying the radar line if collision has occurred.

d) Line 386: Displays the last status of the simulation for 5 seconds by pausing it.

S. Line 393-396:

At line 393, updates the current time by adding dt to it, at line 394, draws the graph in every iteration, at line 395, determines the size of the simulation window, it is from 0 to 1.5x of the red object's range's at x-axis and from -150 to 2000 at y-axis. At line 396, deletes the graph in every dt, later it will be drawn in a very short amount of time so it will never be observed.

VII. OUTPUTS

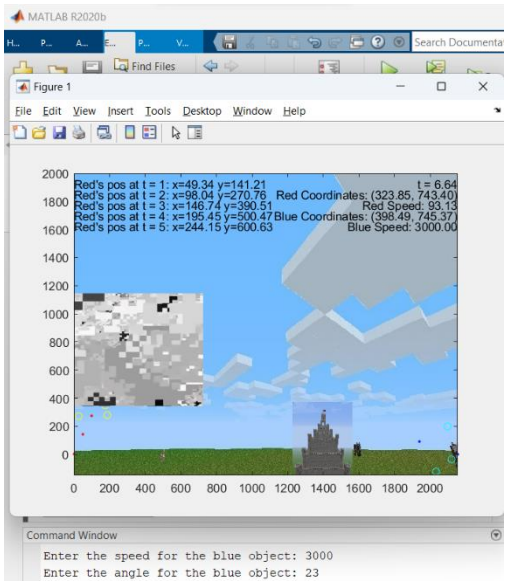

Fig. 1. A mid-air collision, caused an explosion.

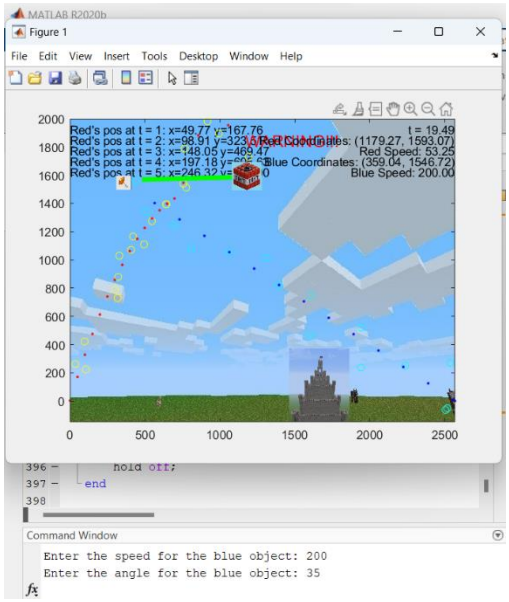

Fig. 2. Red perfectly dodges blue and keeps continuing on its way to target.

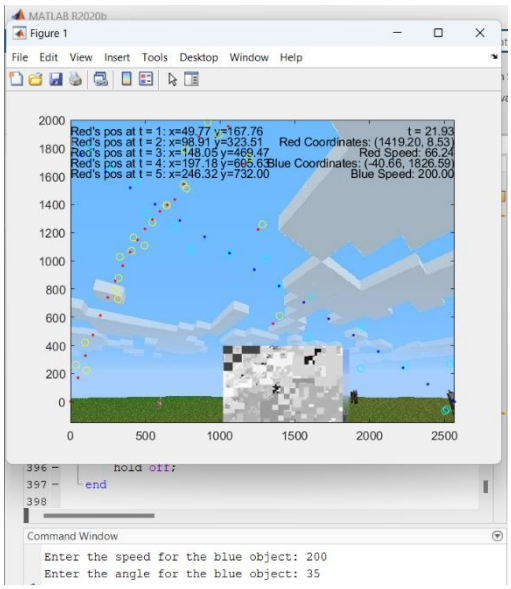

Fig. 3. After dodging, red hits the target perfectly.

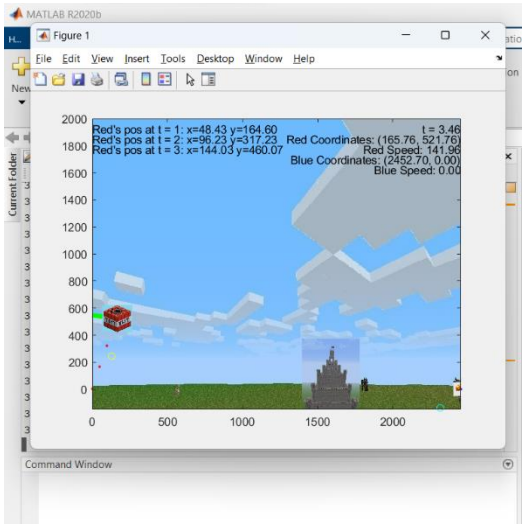

Fig. 4. A snapshot, at 4th second, just before input for blue object is asked.

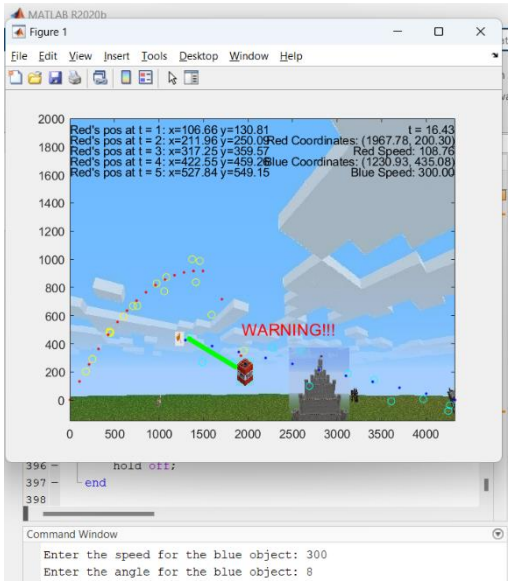

Fig. 5. Red perfectly dodges the blue object again.

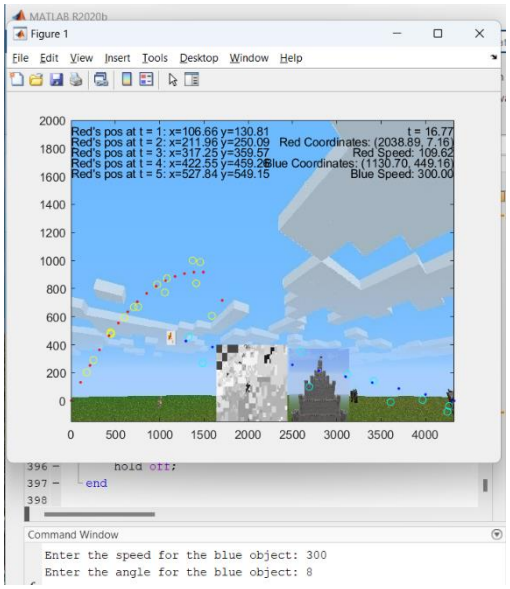

Fig. 6. But this time, it is unable to hit the target perfectly. Mentioned earlier, system does not work with %100 success rate.

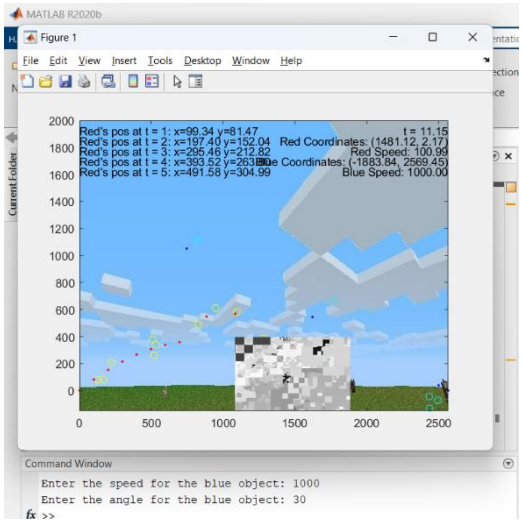

Fig. 7. A result when red hits the target.

VIII. CONCLUSION

After a long time of work, a product of matlab code that implements, projectile motion, and another object that tries to get it before it hits to target, an air radar and most importantly, an avoidance mechanism thus, an anti air system has released. But this simulation is very live and active, and includes so many features to find an exact solution in terms of Monte Carlo Simulations, but some verbal results can be reached.

If blue object is at the range of radar line, but the radar line is unable to reach blue's position again if blue is fast enough to reach red, this guarantees that blue can hit red before it lands to the target. Or if blue is too fast but not approaching red with a very narrow angle, red will easily detect the blue and will try to avoid with full speed, also at that script, if blue passes upper side of red, red will most likely to dodge blue and tries to hit target. Of course, this needs to be highlighted, radar will detect the blue object's errorous values.

Lastly, due to these factors, it is very hard to obtain a absolute result in terms of Monte Carlo Methods. But at some scenarios, results can be guessed with high success rate.

REFERENCES

[1] https://www.toppr.com/guides/physics/motion-in-a-plane/projectile-motion/

[2] Simulation Modeling and Analysis, Third Edition, Averill M. Law, W. David Kelton

[3] https://www.investopedia.com/terms/m/montecarlosimulation.asp#:~:text=A%20Monte%20Carlo%20simulation%20is%20a%20model%20used%20to%20predict,in%20prediction%20and%20forecasting%20models.

[4] https://en.wikipedia.org/wiki/Monte_Carlo_method

[5] https://www.ibm.com/topics/monte-carlo-simulation

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] https://study.com/skill/learn/calculating-the-standard-error-of-the-sampling-distribution-of-a-sample-mean-explanation.html#:~:text=Standard%20Error%3A%20The%20standard%20error,divided%20by%20the%20sample%20size

[8] https://www.mathworks.com/help/stats/prob.normaldistribution.random.html
[9] https://www.mathworks.com/help/matlab/ref/ishandle.html
[10] https://www.mathworks.com/help/matlab/ref/imagesc.html
[11] https://www.mathworks.com/help/matlab/ref/set.html