ENHANCING TRAP SHOOTING PROFICIENCY THROUGH DYNAMIC HOME PRACTICE SIMULATION: A NOVEL APPROACH FOR PRECISION AND CONSISTENCY IMPROVEMENT

ZhiqiYuan[1] and John Morris[2]

[1]Seattle Academy, 1201 E Union St, Seattle, WA 98122
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

I noticed that most trap shooters practice dry firing towards a wall, pretending that there is a target on the wall, going right or left [1]. However, they are not aware if they have come close to or "hit" the target by just imagining it. It is very dull to lift a gun toward a wall with nothing on it. I decided to create a Python program to have a person practice and shoot as if at a range [2]. The program lets a person practice the correct technique which involves not moving the gun with their arms, but with their core. The shooters can practice peripheral vision, pulling the trigger at the right time and following the target. Results have shown that people were more engaged using the program while lifting a gun, than only lifting a gun towards the wall.

KEYWORDS

Trap Shooting, Trap Shooting practice, Clay Target Practice, Target Following Practice

1. INTRODUCTION

Trap shooting, a sport demanding precision and consistency, requires regular practice for athletes to maintain and improve their proficiency [3]. A crucial element in trap shooting is the proper elevation and handling of the firearm. It is imperative for shooters to regularly engage in exercises that promote muscle memory and form. However, regular visits to shooting ranges may not be possible for a lot of the lovers of the sport. My program will help trap shooters practice at home, without the need to go to the shooting range. Lifting one's gun is very important in shooting. It keeps a shooter in shape at home. Otherwise, a shooter's form will change during shooting. It's very hard to correct a shooter's form if there's no in-home practice. I found most trap shooters getting bored after only a few minutes of lifting a gun toward a wall or mirror. The program aims to simulate a dynamic shooting experience, as it helps shooters practice at home, become better, and is far more interactive, because one can visually see where to pull the trigger. Shooters only have to project this program on a wall and they can begin practicing at home.

As of today, there are no similar programs on the web. A lot of trapshooters who practice lifting a gun, place pieces of tape on a wall to mark where a target would be and they swing the gun to the tape to practice. During the development of the program, one of the challenges was to find the perfect distance between the projector that we have and our wall.

One of the other challenges was to find the perfect way for starting the shooting sequence. Voice Recognition and having a physical actuator are some of the ways that were put into consideration in development [4][5]. But at the end having a physical actuator was preferred because of the accidental issues that voice recognition caused.

My program lets the shooter practice dry firing at home, without having to go to the shooting range. It makes a target appear and proceed in a specific direction on the screen. Without this program, some shooters place tape on a wall pretending that a target is there. But this program allows a target to move instead of staying in one spot. One target will appear every three seconds. Five targets will fly in the same direction, in a row, and change. Ten targets will come out in a row, then there's a 30-second break. All repeats until a total of 90 targets appear. A shooter projects this on a wall and places a laser training cartridge in a gun to utilize this program.

My shooting teammates have tried my program several times, and I found out that everyone was more engaged and held the gun longer (than only lifting the gun towards a wall). Since most of us have trouble hitting targets flying to the right, I modified the program to cause the targets to only fly to the right, and everyone seemed to have improved.

The rest of the paper is organized as follows: Section 2 explains the challenges that I met designing the program and during the experiments; Section 3 focuses on our solutions to the challenges that I mentioned in Section 2; Section 4 describes the relevant information about the experiments I did, followed by offering the related work in Section 5. Finally, Section 6 offers concluding remarks and future forecasts for the project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The Timer

One of the biggest challenges I had making this program was the timer. I had to make it so that one target would come out every three seconds, and five targets would fly in the same direction in a row and change. And after 10 targets appear, there would be a 30-second break. All repeats until the number of targets reaches 90. The program will stop after that to give shooters a break. Unfortunately, our timer in this program kept coming up a few seconds early or late. To fix this, I rewrote the timer program and used a different approach to make it work.

2.2. Finding Appropriate Pictures

Another challenge was finding appropriate pictures. It was very difficult to find good transparent pictures for the target. Most of the pictures I found were either not transparent or not at the right angle. I also had to resize the pictures to make them big or small enough to fit the screen.

2.3. Setting the Speed of the Targets

The last challenge I had was setting the speed of the targets. This was difficult, as I didn't have a projector at the time, so I didn't know if the targets were too fast or slow. There is a big difference between viewing the program on a computer screen on a wall projection. I had to repeatedly change the speed the restart the program to find the correct speed.

3. SOLUTION

This program will be on a computer, and it will be projected onto a wall or screen for the best results, as most computer screens are too small. A target will appear and fly out every three seconds. Five targets will fly in the same direction in a row and change. After 10 targets appear, there will be a 30-second break (for the shooters to rest). When 90 targets have appeared, the program will shut itself off to give the shooters a break.

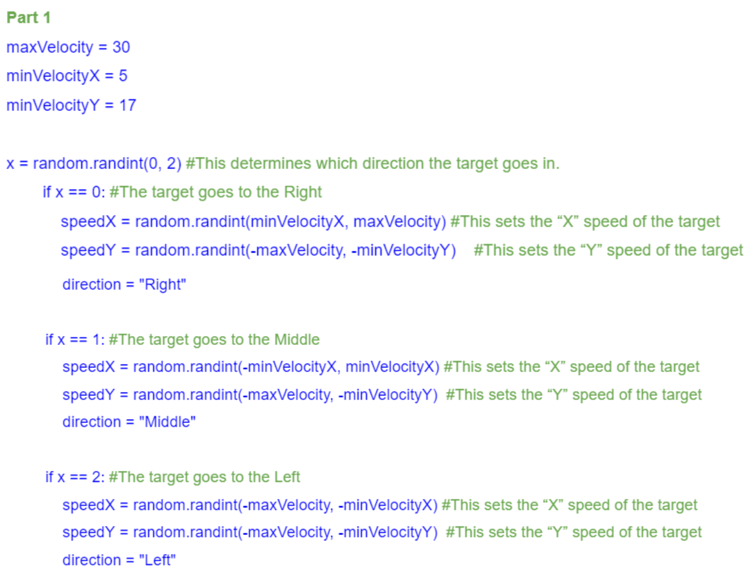
The first step of the program is to create a timer, and a target will be thrown out every three seconds. There will be another counter variable, which keeps track of how many targets have been thrown out. Once the counter reaches a multiple of five, a new direction will be chosen for the next five targets. When the counter reaches ten, shooters will get a 30-second break to rest.

A shooter would connect their computer to a projector and project their screen onto a wall. They would then start the program, which will take up the entire screen, and get ready to lift their gun and follow the target. The laser cartridge would be placed in the gun and will allow the coach to see where each shooter is pointing and their movement.

External components: Projector, Wall, Gun, Laser Cartridge [6]

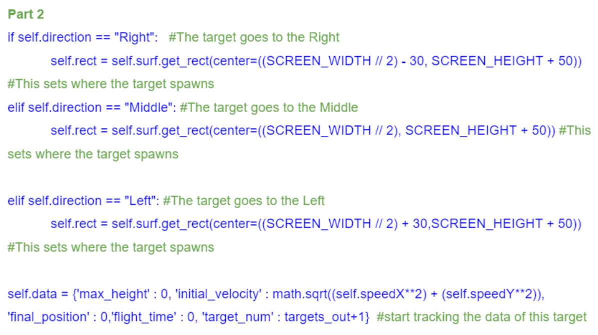
Internal components: Firing Control, Target

When the program starts, a timer and counter variable will be initiated. They control when a target will be generated and then moves it across the screen. The resulting image will be sent to a projector, which displays it on a wall or screen. Shooters would then point their guns and laser toward the wall to follow the target.



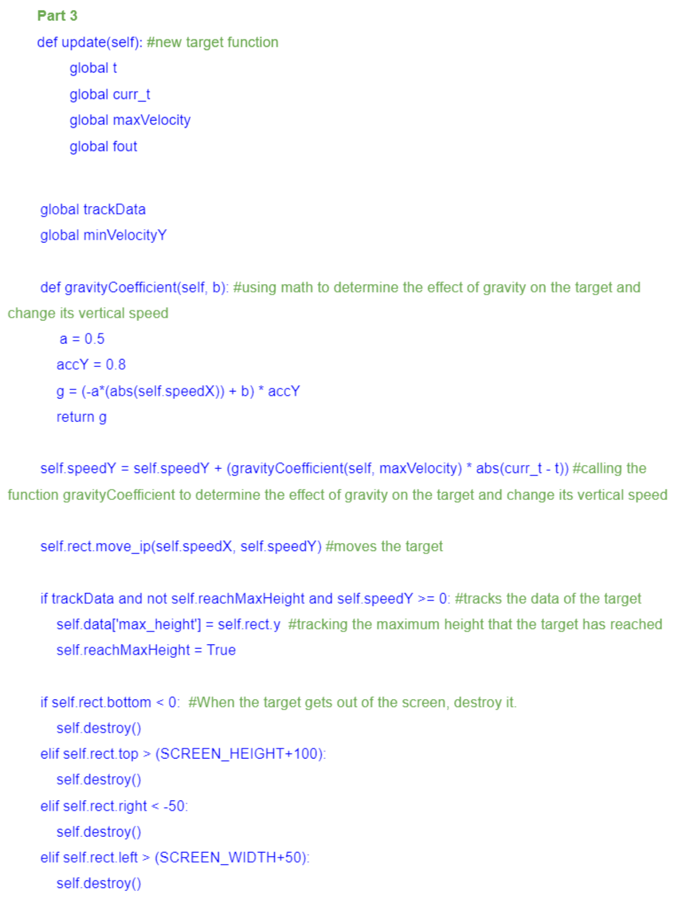
```
Part 1
maxVelocity = 30
minVelocityX = 5
minVelocityY = 17

x = random.randint(0, 2) #This determines which direction the target goes in.
    if x == 0: #The target goes to the Right
        speedX = random.randint(minVelocityX, maxVelocity) #This sets the "X" speed of the target
        speedY = random.randint(-maxVelocity, -minVelocityY)    #This sets the "Y" speed of the target
        direction = "Right"

    if x == 1: #The target goes to the Middle
        speedX = random.randint(-minVelocityX, minVelocityX) #This sets the "X" speed of the target
        speedY = random.randint(-maxVelocity, -minVelocityY)  #This sets the "Y" speed of the target
        direction = "Middle"

    if x == 2: #The target goes to the Left
        speedX = random.randint(-maxVelocity, -minVelocityX) #This sets the "X" speed of the target
        speedY = random.randint(-maxVelocity, -minVelocityY)  #This sets the "Y" speed of the target
        direction = "Left"
```

Figure 1. Screenshot of part 1 code



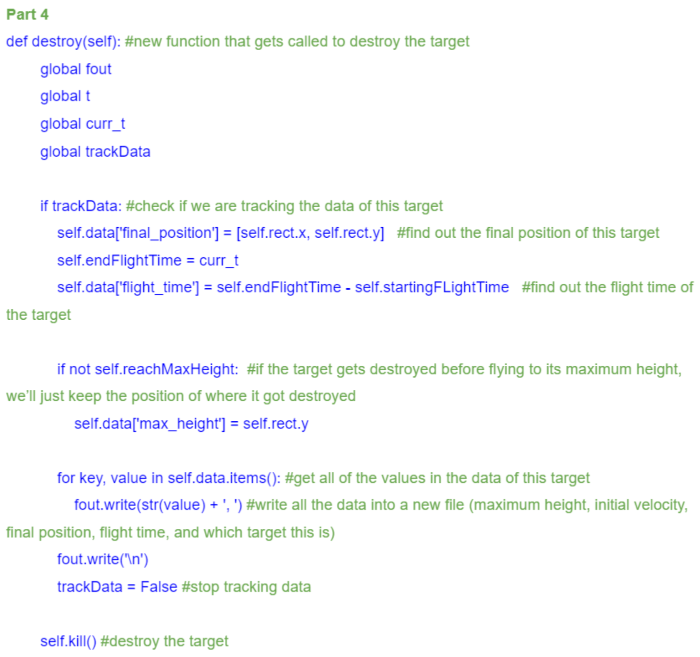
```
Part 2
if self.direction == "Right":   #The target goes to the Right
          self.rect = self.surf.get_rect(center=((SCREEN_WIDTH // 2) - 30, SCREEN_HEIGHT + 50))
#This sets where the target spawns
elif self.direction == "Middle": #The target goes to the Middle
          self.rect = self.surf.get_rect(center=((SCREEN_WIDTH // 2), SCREEN_HEIGHT + 50)) #This
sets where the target spawns

elif self.direction == "Left": #The target goes to the Left
          self.rect = self.surf.get_rect(center=((SCREEN_WIDTH // 2) + 30,SCREEN_HEIGHT + 50))
#This sets where the target spawns

self.data = {'max_height' : 0, 'initial_velocity' : math.sqrt((self.speedX**2) + (self.speedY**2)),
'final_position' : 0,'flight_time' : 0, 'target_num' : targets_out+1}  #start tracking the data of this target
```

Figure 2. Screenshot of the part 2 code



```
Part 3
def update(self): #new target function
        global t
        global curr_t
        global maxVelocity
        global fout


 global trackData
 global minVelocityY

 def gravityCoefficient(self, b): #using math to determine the effect of gravity on the target and
change its vertical speed
     a = 0.5
     accY = 0.8
     g = (-a*(abs(self.speedX)) + b) * accY
     return g

 self.speedY = self.speedY + (gravityCoefficient(self, maxVelocity) * abs(curr_t - t)) #calling the
function gravityCoefficient to determine the effect of gravity on the target and change its vertical speed

 self.rect.move_ip(self.speedX, self.speedY) #moves the target

 if trackData and not self.reachMaxHeight and self.speedY >= 0: #tracks the data of the target
     self.data['max_height'] = self.rect.y  #tracking the maximum height that the target has reached
     self.reachMaxHeight = True

 if self.rect.bottom < 0:  #When the target gets out of the screen, destroy it.
     self.destroy()
 elif self.rect.top > (SCREEN_HEIGHT+100):
     self.destroy()
 elif self.rect.right < -50:
     self.destroy()
 elif self.rect.left > (SCREEN_WIDTH+50):
     self.destroy()
```

Figure 3. Screenshot of part 3 code



```
Part 4
def destroy(self): #new function that gets called to destroy the target
    global fout
    global t
    global curr_t
    global trackData

    if trackData: #check if we are tracking the data of this target
        self.data['final_position'] = [self.rect.x, self.rect.y]   #find out the final position of this target
        self.endFlightTime = curr_t
        self.data['flight_time'] = self.endFlightTime - self.startingFLightTime   #find out the flight time of
the target

        if not self.reachMaxHeight:  #if the target gets destroyed before flying to its maximum height,
we'll just keep the position of where it got destroyed
            self.data['max_height'] = self.rect.y

        for key, value in self.data.items(): #get all of the values in the data of this target
            fout.write(str(value) + ', ') #write all the data into a new file (maximum height, initial velocity,
final position, flight time, and which target this is)
        fout.write('\n')
        trackData = False #stop tracking data

    self.kill() #destroy the target
```

Figure 4. Screenshot of part 4 code

Part 1: This part explains how the direction of the target is determined. Once every five targets, a new direction is set. The computer will first choose a random direction (right, left, middle). Then it will choose a random speed for the target based on which direction it's going. If the target goes to the middle, it might still go to the side, just not as much.

Part 2: This section takes the information from Part 1 and determines the spawn point of the target, as each direction has a different spawn point. At the end of this section, there is a dictionary that keeps track of all the data of the target (maximum height, initial velocity, final position, flight time, and which target this is). We're tracking this data to analyze the performance in order to improve the program.

Part 3: This is the target's update function that gets called 60 times per second. It first calls a function, gravityCoefficient(), which uses the speedX to determine the effect of gravity on the target and change its vertical speed. As speedX increases, so does the effect of gravity and vice versa. I created this to simulate the target's flight path. When the target flies near the middle, it has more time to be affected by gravity, which is why it seems to fall slower. The next line moves the target. The next few lines update the maximum height of this target. The last part checks if the target flies out of the screen, and if it does, call the destroy() function.

Part 4: This is the destroy() function. It gets called once the target flies out of the screen. This function will first update the data of this target, adding the final position and flight time. It also checks if the target gets destroyed before flying to its maximum height, if yes, then the computer will just keep the position of where it got destroyed. Next, it writes all the data in a file. Lastly, it destroys the target.

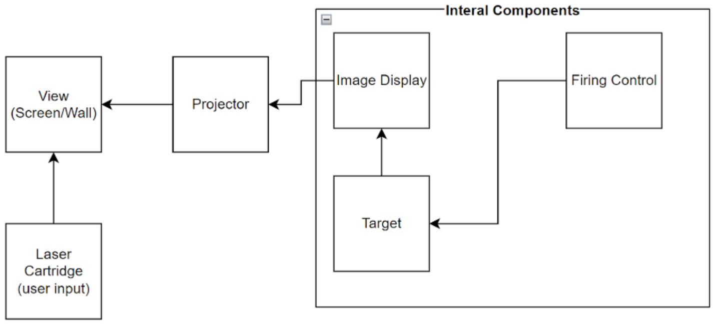


Figure 5. Overview of the project

In this program, there are 2 countdown timers and a “targets out” counter that are displayed on the screen. The main countdown timer is in the top left corner displaying when the next target will come out and the additional one is in the bottom left corner. The additional countdown timer allows the shooters to view the timer using their peripheral vision while preparing for the next target to come out. The “targets out” counter is in the top right corner, displaying the number of targets that have come out. In each of the four images below, one of the 3 possible firing directions is displayed. The dashed yellow line in each screenshot shows the path of the target. The target can go left, right, or straight up.

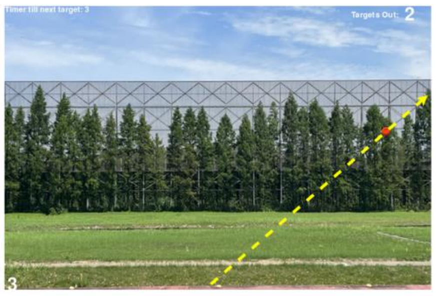


Figure 6. The path of the target 1

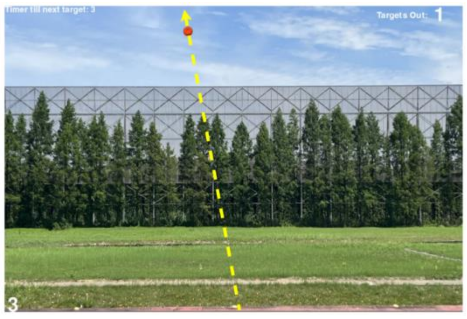


Figure 7. The path of the target 2

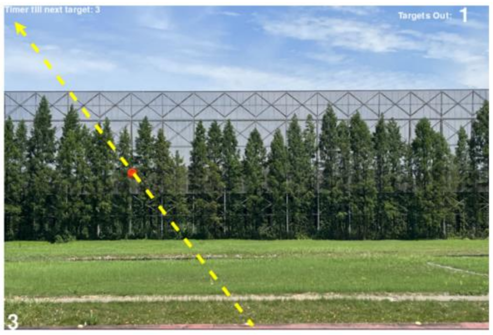


Figure 8. The path of the target 3

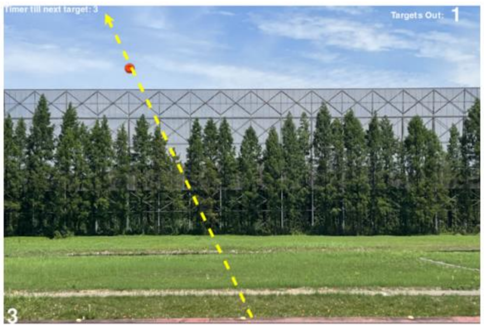


Figure 9. The path of the target 4

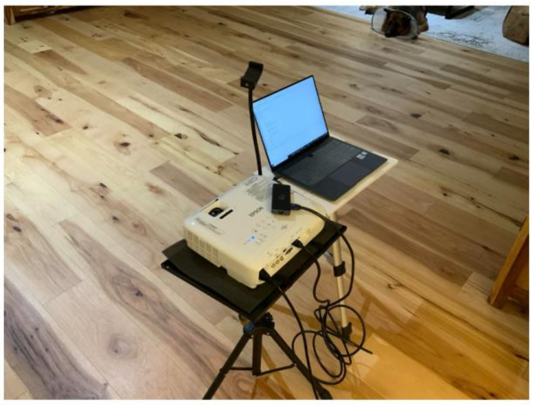


Figure 10. How to set up my program

This photo shows how my program would be set up. As one can see, it does not require much space or equipment at all, only needing a computer and projector.

For this project, I mainly used Pygame to create this program [7]. Pygame helps create the window and screen which everything is displayed on. The timers used by my program are also from Pygame [8].

I also used “random.randint” to get a random place for the target should go to, and it determines whether the target should go left, right, or middle. It will also determine where the target will go and the speed of the target. I also made sure that when the target leaves the screen, it will be deleted to not cause lag.

4. EXPERIMENT

4.1. Experiment 1

My program helps shooters practice dry firing. Dry firing is the practice of lifting one's gun toward a wall or mirror [9]. Instead of merely lifting their guns in front of a wall or mirror, my program helps shooters feel like they are at a shooting range, improving their gun swing, and target approach, by providing them with a visual of the shooting range and a target moving across the screen in a specific direction. This program also helps them realize their shooting problems in order to fix them.

Ten shooters out of the fifteen from my shooting team used to practice regular dry firing, but now they practice simulated dry firing, which allowed them to practice with a moving target instead of dry firing toward a spot on a wall. They practiced at least one hour every week, and saw improvement in just a few weeks.

Some feedback that I've received was to change the background of the program, add gravity, and make each target go in the same direction five times to practice consistency.
The shooters that practiced with my program were able to produce much better results than before when they only practiced dry firing toward a wall or mirror. A few others were also able to achieve a new personal high score.

From the feedback, I've created a new version of the program. It has a new background, since with the previous background the target was hard to see when it first flew out, the new background provides more contrast between the target and the background. The new background is a picture of a giant field taken at station 3 of an international shooting range. The target now moves in the same direction five times to practice consistency. Lastly, gravity was added to the new version to make the target's movement much more realistic and creates more variation in the target's path across the screen, as it can either fly extremely high or low, simulating the motion of the target falling and flying away from the shooter [10].

New shooters were able to shoot much better right away, being able to hit more than 10 targets the very next day, while they were only able to hit less than 10 before. The more experienced shooters improved slower than the newer shooters, as they can already shoot extremely well, but just needs to practice consistency.

5. RELATED WORK

This book talks about how important dry firing is, as it helped them improve massively [11]. Dry firing is when shooters lift their guns towards a mirror or wall and practice their form and movement. Practicing dry firing can help shooters get rid of bad habits that can affect them during shooting, and helps correct a shooter's form, movement, gun positioning, etc. Doing this every day for over an hour can be extremely dull, which is why I made this program to help shooters improve further. My program provides shooters with a visual that simulates a target flying out of a trap house, stimulating variation in their movement.

This research shows that practicing dry firing has a big impact on shooters' scores [12]. It allows them to improve much faster than shooters who don't practice dry firing. This paper explains how they conducted an experiment where they tracked the scores of shooters who don't practice dry firing, then had the shooters practice for around one hour of dry firing every day, and in the end, compared their final results to their score on day 1. Every single shooter improved, as their scores were higher than before. Although the shooters were able to improve while dry firing without a specialized program, their scores would increase even more if they practiced with a program that simulated a target flying.

This research talks about the trap shooting sport, and why dry firing practice matters for trap shooting [13]. It also talks about a program, DryFire, and how it works and allows the shooter to also practice dry firing.

The DryFire program is extremely advanced, as it can simulate a target flying out of a trap house and detect if the shooters hit the simulated target. If the shooter didn't hit the target, the system can show them where their shot was, so they know how to improve next time. Although their program is highly developed, it takes a long time to set up, as it needs quite a lot of equipment to operate. For the DryFire program, the shooter needs a trigger switch, barrel clips, trigger control box, and muzzle insert and cable just for the gun. It also requires a lot of space, as shooters would need to set up two laptops to look at the feedback and four CCD cameras to utilize the DryFire program. This program also costs up to $6572.99. On the other hand, my program currently is completely free to use, and would not require much equipment and space. The only equipment required for my program is a projector, laptop, gun, and an optional laser cartridge.

6. CONCLUSIONS

I made a program that helps shooters practice at home. I've noticed that most trap shooters practice dry firing toward a wall, pretending that there is a target on the wall that is going right or left. However, they never know if they come close to or "hit" the target through this method. It is very boring to lift a gun toward a wall with nothing on it. This Python program lets the shooter shoot as if at the range. It lets a shooter practice, staring at the target, pulling the trigger at the right time, and following the target [14]. The results show that shooters were more engaged when using this program while lifting a gun than only lifting a gun toward a wall.

When my shooting team practiced with my program, The program was set up and projected around 12 feet away from the wall. Shooters would stand behind the projector, lift their guns, and point their lasers at the screen projected onto the wall. The coach would stand behind the shooters, looking at where and how their gun was moving toward the target.

My coach noticed that shooters who practiced with my program were able to improve much faster than shooters who didn't, especially the new shooters.

There is no gravity in my program, which means the target will not fall down, and will only go up. This program will not have the same feeling as going to a shooting range, but it will help a shooter improve. The target will not break when the shooter presses the trigger, but it will keep going, so the shooter can practice a second shot and follow the target.

This program works best on a big screen or while being projected onto a screen onto a wall, as the target would be smaller and slower if it's on a small screen.

I will allow the targets to fall, appear on sound control, and have the shooter be able to change the speed of the target.

I plan to allow the targets to fall down, so I would add gravity. I also plan to have the program work on sound control, so that when the shooter calls for the target, a target appears [15]. I will add better backgrounds and get improved pictures of the target.

REFERENCES

[1] Rodríguez-Seijo, Andrés, et al. "Pb pollution in soils from a trap shooting range and the phytoremediation ability of Agrostis capillaris L." Environmental Science and Pollution Research 23 (2016): 1312-1323.

[2] Sanner, Michel F. "Python: a programming language for software integration and development." J Mol Graph Model 17.1 (1999): 57-61.

[3] Rodríguez-Seijo, Andrés, et al. "Pb pollution in soils from a trap shooting range and the phytoremediation ability of Agrostis capillaris L." Environmental Science and Pollution Research 23 (2016): 1312-1323.

[4] McGehee, Frances. "An experimental study of voice recognition." The Journal of General Psychology 31.1 (1944): 53-65.

[5] Hurst, Jonathan W., Joel E. Chestnutt, and Alfred A. Rizzi. "An actuator with physically variable stiffness for highly dynamic legged locomotion." IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004. Vol. 5. IEEE, 2004.

[6] Raskar, Ramesh, and Paul Beardsley. "A self-correcting projector." Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. Vol. 2. IEEE, 2001.

[7] Kelly, Sloan, and Sloan Kelly. "Basic introduction to pygame." Python, PyGame and Raspberry Pi Game Development (2016): 59-65.

[8] McGugan, Will. Beginning game development with Python and Pygame: from novice to professional. Apress, 2007.

[9] Oldford, Steven, et al. "Predicting slow-drying fire weather index fuel moisture codes with NOAA-AVHRR images in Canada's northern boreal forests." International Journal of Remote Sensing 27.18 (2006): 3881-3902.

[10] Gunstone, Richard F., and Richard T. White. "Understanding of gravity." Science education 65.3 (1981): 291-299.

[11] Heiple, King. Mastering Skeet: Fundamental Shooting Techniques for Hitting the Target in Championship Form. Stackpole Books, 2007.

[12] Saini, Pradip, T. Onima Reddy, and Vikram Singh. "Influence of Dry Fire Practice on Rifle Shooting Performance of School Going Students."

[13] Swanton, Alan. Development of a recording system to empirically analyse the shooting characteristics of olympic trap clay target shooters. Diss. Master’s thesis, University of Limerick, 2011.

[14] Konopka, Wieslaw, Piotr Zalewski, and Piotr Pietkiewicz. "Evaluation of transient and distortion product otoacoustic emissions before and after shooting practice." Noise and Health 3.10 (2001): 29.

[15] Elliott, S. J., and P. A. Nelson. "The active control of sound." Electronics & communication engineering journal 2.4 (1990): 127-136.

© 2023 By AIRCC Publishing Corporation. This article is published under the Creative Commons Attribution (CC BY) license.