****

Software Engineering Department  
 Braude College of Engineering

Capstone Project Phase B – 61999

**Gravity shooter game using ML – Player behavior analysis**

[GitHub Repository](https://github.com/IbrahimDaoud5/Gravity-shooter.git)

**23-2-D-22**

Advisor: **PhD. Sulamy Moshe**

MosheSu@braude.ac.il

Students:

**Yosif Housen -** Yosif.Hosen@e.braude.ac.il

**Ibrahim Daoud -** Ibrahim.Daoud@e.braude.ac.il

[**Abstract** 2](#_Toc165886734)

[**1.** **Introduction** 3](#_Toc165886735)

[**2.** **Development process** 4](#_Toc165886736)

[**2.1. Diagrams** 5](#_Toc165886737)

[**2.1.1. AWS Architecture diagram** 5](#_Toc165886738)

[**2.1.2 Activity diagram** 6](#_Toc165886739)

[**2.2. Scenes and Flow** 8](#_Toc165886740)

[**2.3. Reinforcement Learning** 13](#_Toc165886741)

[**2.4. Challenges and Solutions** 17](#_Toc165886742)

[**2.4.1. Decisions** 18](#_Toc165886743)

[**2.5. Tools we used:** 19](#_Toc165886744)

[**2.6. Testing** 20](#_Toc165886745)

[**3. User Manual** 21](#_Toc165886746)

[**4. Operation & Maintenance Guide** 24](#_Toc165886747)

[**Server** 24](#_Toc165886748)

[**Client** 27](#_Toc165886749)

[**5. Results & Conclusions** 29](#_Toc165886750)

[**Further improvements** 29](#_Toc165886751)

[**6. References** 30](#_Toc165886752)

# **Abstract**

Gravity Shooter is an innovative multiplayer adventure game designed to immerse players in a fun, interactive learning experience with a primary focus on physics and machine learning (ML). The game features a dynamic 2D environment and employs elements like gravity manipulation to deliver educational content. As players navigate through a map densely populated with trees, rocks, and various other obstacles, they face the unique challenge of accurately aiming and hitting targets.

In a dedicated training screen, the ML algorithm analyzes the player’s current situation in real-time, including the target's distance and the player's position. Using this information, the algorithm suggests tips to help the player improve their aim. Players train their shooting skills with the help of ML on this dedicated screen before choosing to play in either solo or multiplayer modes.

A distinctive feature of multiplayer mode is the ability for players to manipulate their opponents' gravity or force, adding an extra layer of strategy and interaction. This manipulation allows for dynamic gameplay where players must not only master their own abilities but also anticipate and counteract the modifications imposed by their opponents.

The integration of ML within the game not only assists in enhancing the player's skills but also provides a method of learning and applying physics principles. As players become more familiar with the ML suggestions, they begin to intuitively understand and apply the principles of trajectory, force, and gravity.

The game is developed using various cutting-edge technologies, including new advancements in machine learning. It has been thoroughly tested and evaluated with a range of tools to ensure it fulfills its intended purpose and caters to a wide variety of interested individuals.

# **Introduction**

Gravity Shooter is a multiplayer adventure designed to captivate players while providing a unique, fun, and enjoyable approach to visualizing and learning school subjects, particularly in the field of physics.

Traditional methods of learning physics can sometimes feel disconnected from real-world applications, lacking excitement and engagement. By immersing players in a dynamic 2D environment filled with obstacles and interactive elements, Gravity Shooter brings physics to life. Through aiming, shooting, and strategizing, players gain hands-on experience with fundamental principles such as gravity, projectile motion, and trajectory.

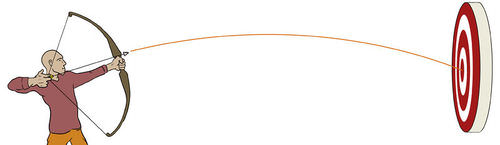
In Gravity Shooter, players have the unique opportunity to experience the concepts of changing gravity in a tangible way. Instead of just encountering equations on paper, they can actively alter the gravity within the game and witness its direct impact on specific objects. Through an engaging graphical interface, players can truly "feel" the effect of gravity changes, enhancing their understanding of these concepts in an interactive and entertaining manner.

During gameplay, players acquire the ability to make quick decisions to adapt to varying gravity values and maintain equilibrium.

Gravity Shooter goes beyond traditional gameplay by integrating cutting-edge machine learning in a dedicated training environment. This allows us to offer real-time guidance, assisting players in improving their shooting accuracy.

Gravity Shooter caters to students, enthusiasts, and anyone seeking an entertaining and educational gaming experience. Whether you're a high school student aiming to reinforce your physics knowledge or simply an avid learner with a passion for scientific concepts, this game is designed to fulfill your need for interactive and captivating learning. Gravity Shooter will not only provide hours of enjoyment but also unlock a deeper understanding of the fascinating world of physics.

All the mentioned above happens while the player navigates and explores unique maps filled with obstacles and objects, while racing against the clock and competing with multiple players to reach the predefined targets.



# **Development process**

Description:

In the development of "Gravity Shooter," a 2D Unity multiplayer game, we started by translating our initial ideas about "the requirements" into specifications that could be implemented in code. We quickly realized that developing a multiplayer game would require a server to manage user (player) credentials, such as signing in and signing up.

We chose AWS as our cloud platform because it is an industry standard for cloud applications and offers comprehensive tutorials for new developers, as well as a 1-year free-tier. Specifically, we set up an EC2 instance as our computational unit alongside RDS for the database, using MySQL. We selected the Stockholm region for our EC2 and RDS instances, as it is the nearest region to our target audience that is included in the free-tier.

Subsequently, we developed login and registration screens within Unity and connected these to our server in the cloud. For this "connection," we utilized REST APIs to enable the secure sending and receiving of information to and from the server. Information transmission uses HTTPS, combined with SSH for server control, ensuring high reliability and security in line with industry standards.

Next, we began constructing the game map for "Gravity Shooter," incorporating various elements such as the ground, obstacles (trees, rocks, wood, and fire), and targets. We also added more decorative objects to enhance the game's aesthetic appeal. However, we strategically placed numerous obstacles to ensure that finding and hitting targets was challenging rather than trivial.

Following the map design, we created the player character complete with necessary animations. We equipped the player with three movement capabilities: moving right, left, and jumping to navigate past obstacles. We also attached a bow to the player character, equipped with an endless supply of arrows for shooting targets. The bow's movement is controlled by the mouse position, allowing for a full range of motion up to 360 degrees. The force of each shot is determined by the time difference between the mouse release and the initial click, adding a dynamic element to the gameplay.

## **2.1. Diagrams**

### **2.1.1. AWS Architecture diagram**

The diagram represents the architecture of a cloud-based application deployed on AWS. At the core is an EC2 instance of type t3.micro, indicating a small, economical virtual server suitable for low to moderate traffic. This server, labeled with the Java logo, suggests that it's running a Java application. Elastic Block Store (EBS) is shown as the storage solution for the EC2 instance, providing persistent block storage.

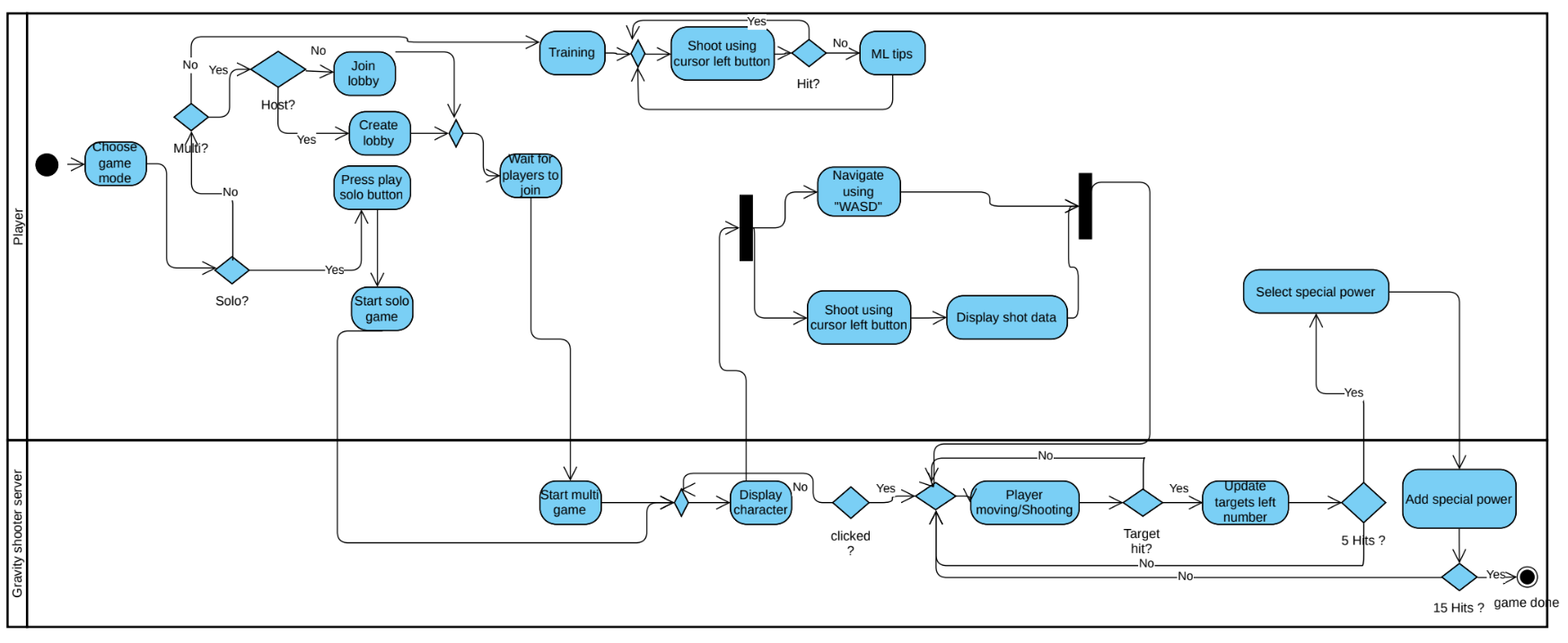
The EC2 server communicates with a database service, AWS RDS with MySQL, which means the server's persistent data is managed through a MySQL database on Amazon's Relational Database Service, ensuring data durability and facilitating easy scaling.

On the left, we see that the EC2 server is accessed via SSH to upload the server's JAR file, which is the Java application archive. This also implies that the server can be managed or updated remotely through secure shell access.

There's a secured root account, referring to AWS's root account with multi-factor authentication (MFA) enabled for additional security.

The SERVER is connected to a CLIENT through a REST API over the internet, using HTTPS for secure communication, which is encrypted with SSL/TLS. The client-side depicts a web interface that users interact with, possibly a web application or service.

### **2.1.2 Activity diagram**

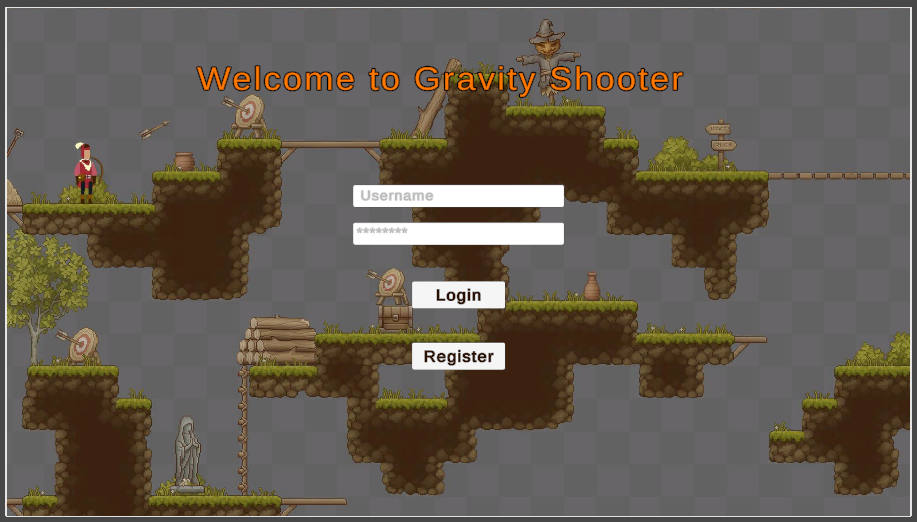
****

This activity diagram represents the flow of actions in Gravity Shooter Here's an overview of the diagram:

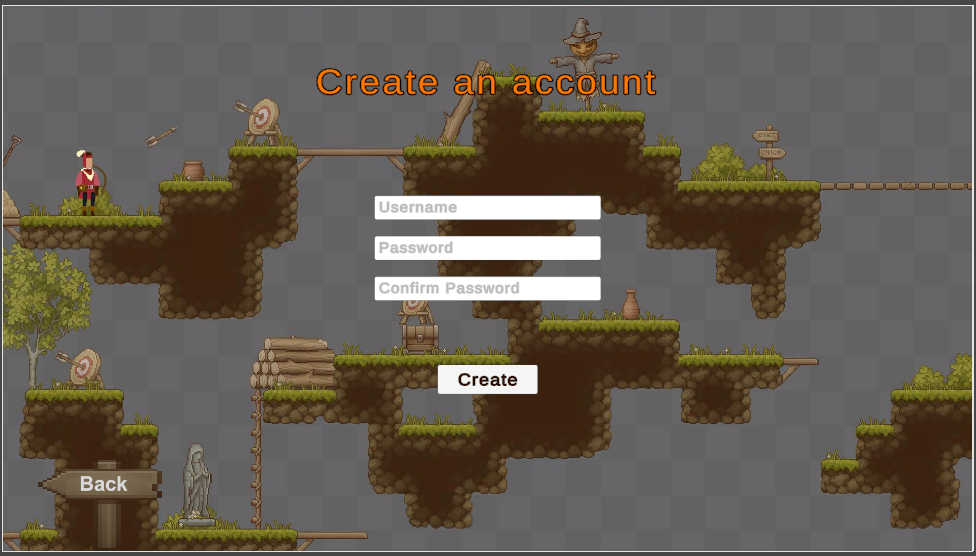
1. Choosing Game Mode: The player starts by selecting the game mode: multiplayer ("Multi?") or single-player ("Solo?").
2. Multiplayer Flow: If multiplayer is chosen, the player decides whether to host ("Host?") or join a game. If hosting, they create a lobby; if joining, they select a lobby to join. Once in the lobby, the player waits for others to join before starting the multiplayer game ("Start multi game").
3. Single-Player Flow: If single-player mode is selected, the player presses the play solo button to start a solo game.
4. Gameplay Mechanics: For training, the player shoots using the cursor's left button and may receive ML (Machine Learning) tips if the shot does not hit the target. General gameplay involves navigating using the "WASD" keys and shooting with the cursor's left button. Data from shots taken is displayed to the player.
5. Character Interaction: As gameplay continues, the diagram checks if the player has clicked on the display character. If not, the game continues normally.
6. Shooting Targets: When the player is moving or shooting, the game checks if the target is hit. If a target is hit, it updates the number of targets left. If a target is not hit, the game continues without update.
7. Special Powers:The player can select special powers after fulfilling certain conditions (not specified in the provided section of the diagram). Special powers are added after a series of successful hits ("5 Hits?"), and the game checks if the condition for "15 Hits?" is met to proceed with the endgame ("game done").

## **2.2. Scenes and Flow**

Home Screen:

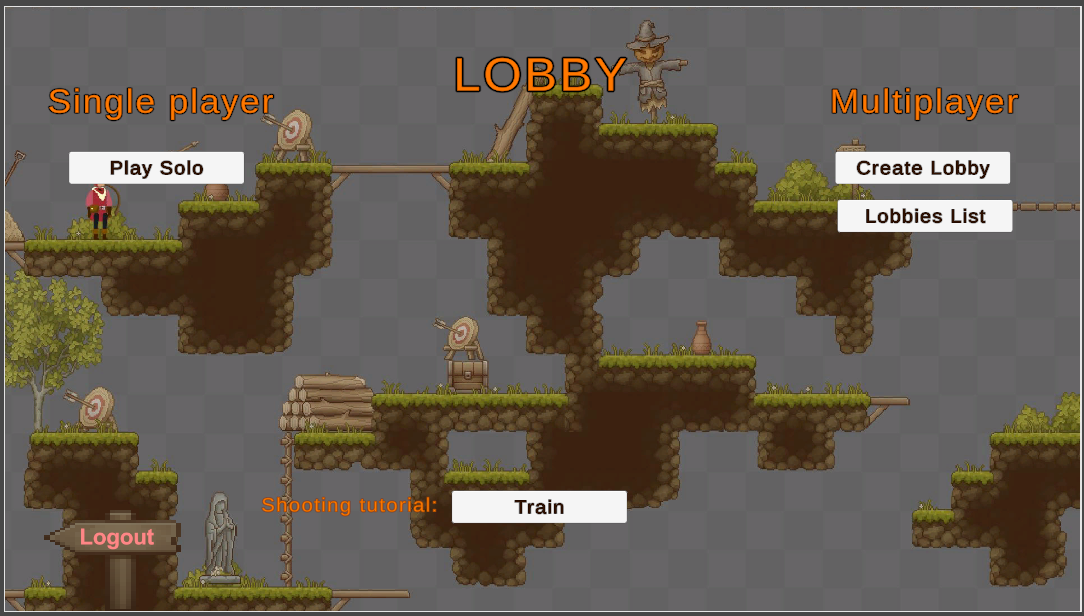
****

* This is the home screen of the Gravity Shooter game where you need to enter your credentials (username and password) if you have an account. If not, you can register by clicking the 'Register' button. After clicking on the 'Login' button, you will be directed to the game's lobby. To navigate between the input fields and the buttons, you can simply use the mouse or press 'Tab' to move forward and 'Shift + Tab' to move backward.

Register Screen:

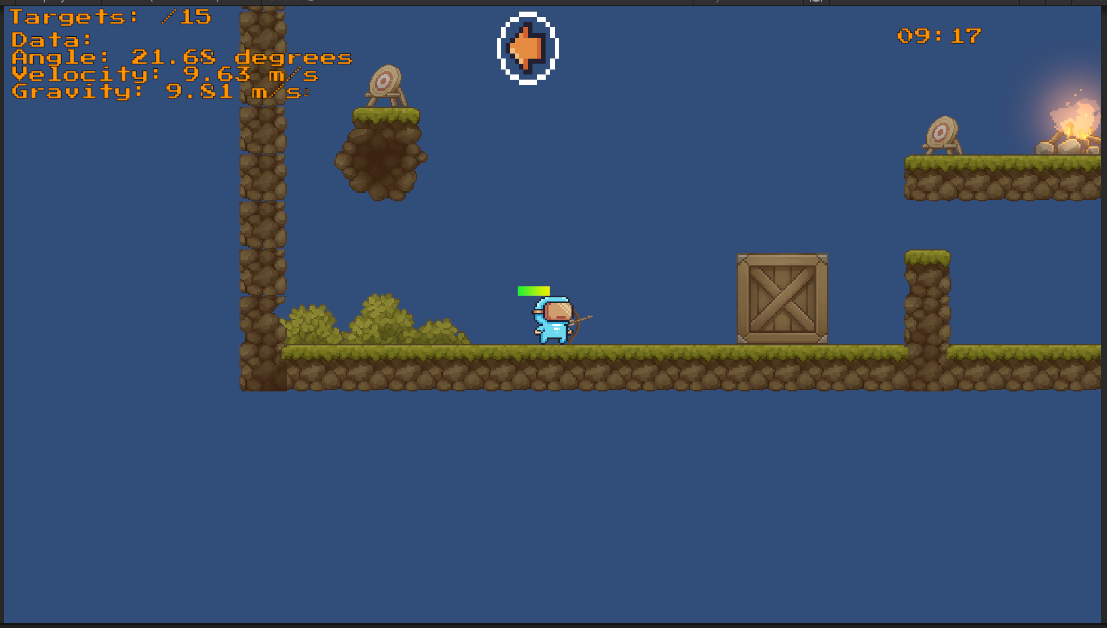
* Here, you can see the registration screen where you can create an account. You should enter a username and password. It requires you to type the password and then confirm it. After that, click 'Create.' You will always receive a response to show you if the action was successful or if there were any errors.

Lobby Screen:



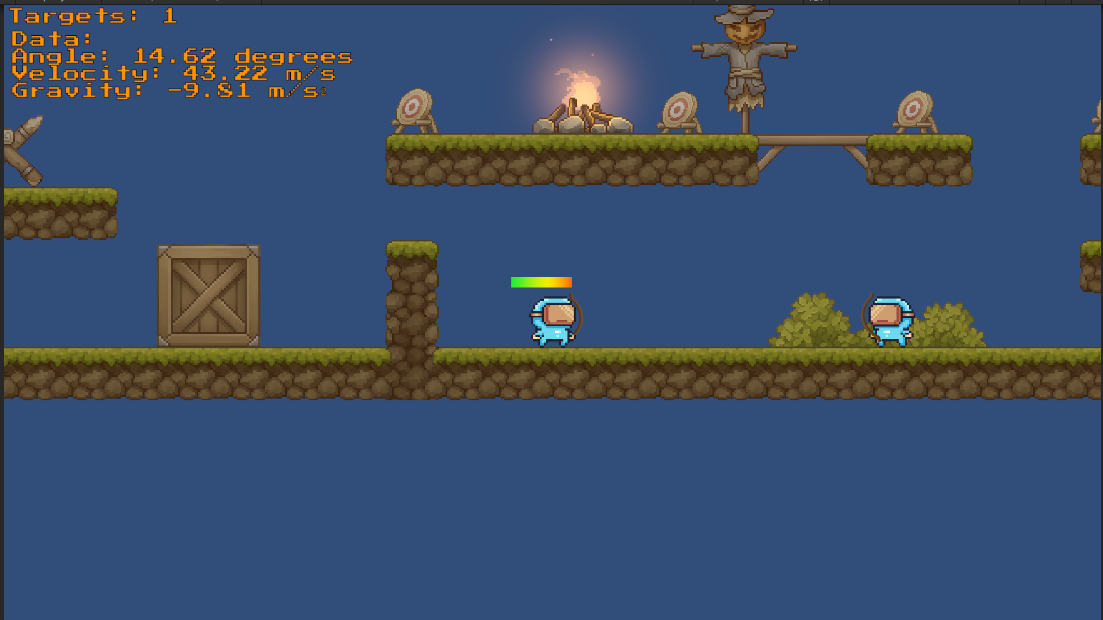
* Above is the lobby screen, which is mainly divided into three sections, each with a title. On the left side is the 'Single Player' section, where you can play solo by clicking on the 'Play Solo' button. In the middle section, there is the 'Shooting Tutorial.' Clicking on 'Train' will move you to the training screen. On the right side is the 'Multiplayer' section, where you have two options: 'Create Lobby,' where you can start a game as a host by clicking on the 'Create Lobby' button, and 'Lobbies List,' a button that takes you to a screen where you can view open lobbies and join them as a player.

Solo mode screen:

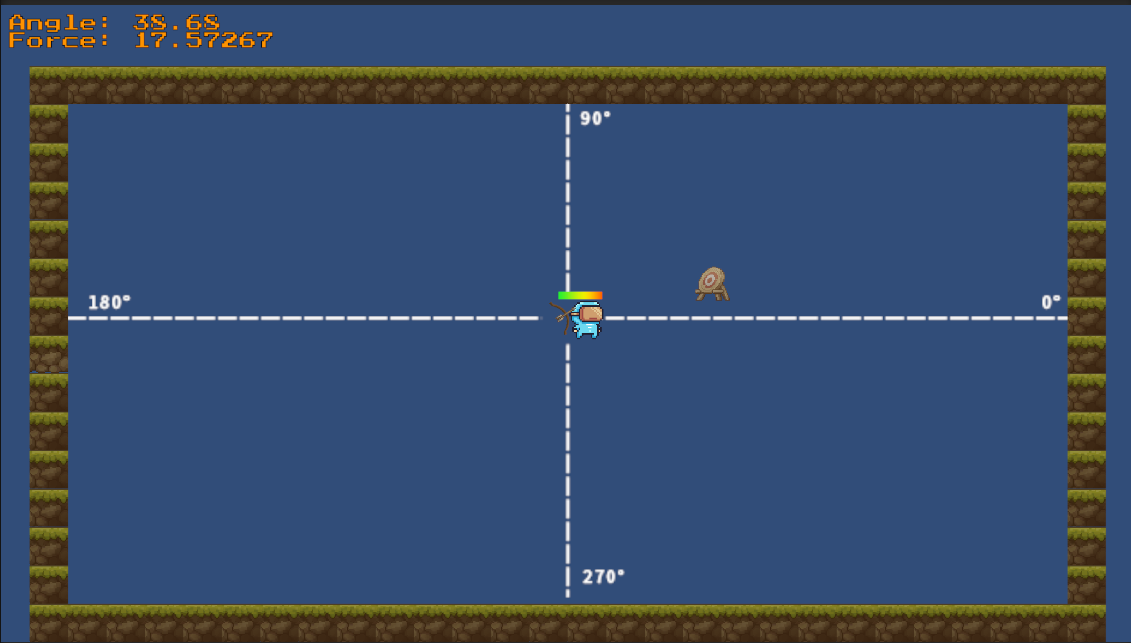


* In Solo mode, you play against time, with the remaining time displayed in the upper-right corner of the screen. In the middle upper part of the screen, you'll see an arrow indicating the wind resistance affecting the arrow you shoot. In the upper-left corner of the screen, you can find relevant data describing the shot and the number of targets you have successfully hit. Whenever you wish to resume the game, return to the lobby, or quit the game, you can press ESC on your keyboard and select from the available options:

Multiplayer mode screen:



* In the picture above, you can see two players after one has created a multiplayer game and the other has joined. The screen also displays data for each player, including details about their recent shot, gravity, and the number of targets they have hit. Scattered around the map are various targets and obstacles. Above the player character, you can see the power bar, which visually indicates the power of the shot. The power changes by clicking and holding the mouse button; it increases until the player releases the button.
* The picture above shows a popup screen presented to a player in a multiplayer game after hitting a certain number of targets. On the popup screen, there are two buttons that affect the game's behavior to benefit the player. The first button, labeled "Force," increases the maximum force a player can apply to a shot, resulting in more powerful shots. The second button, labeled "Gravity," allows a player to manipulate the gravity affecting all other players, making the game more challenging for them. This applies to all players in the game except the one who activated it. The player must decide between one of these options.

Training Screen:

* This is the training screen where you can practice hitting targets and receive tips/suggestions from the ML model whenever you miss; these tips will appear as popup messages. After shooting, you can immediately see the angle and force of the shot displayed in the upper-left corner. When you've trained sufficiently and wish to return to the lobby, simply press 'Esc' and select 'Menu' from the available options.

## **2.3. Reinforcement Learning**

MLAgents:

Unity's ML-Agents (Machine Learning Agents) is an open-source toolkit developed by Unity Technologies that allows researchers and developers to train intelligent agents using reinforcement learning (RL) algorithms in Unity games and simulations. ML-Agents integrates with Unity's game engine, providing a powerful environment for training and testing AI agents.

ML-Agents significantly simplified our development process by abstracting the complexities of the model, algorithm, and neural network infrastructure. It achieves this by establishing a communication pipeline between Unity (the environment) and the Python code (the neural network) using port number 5005. This setup allows us to directly pass observations—the inputs for the neural network—straight from the Unity Editor.

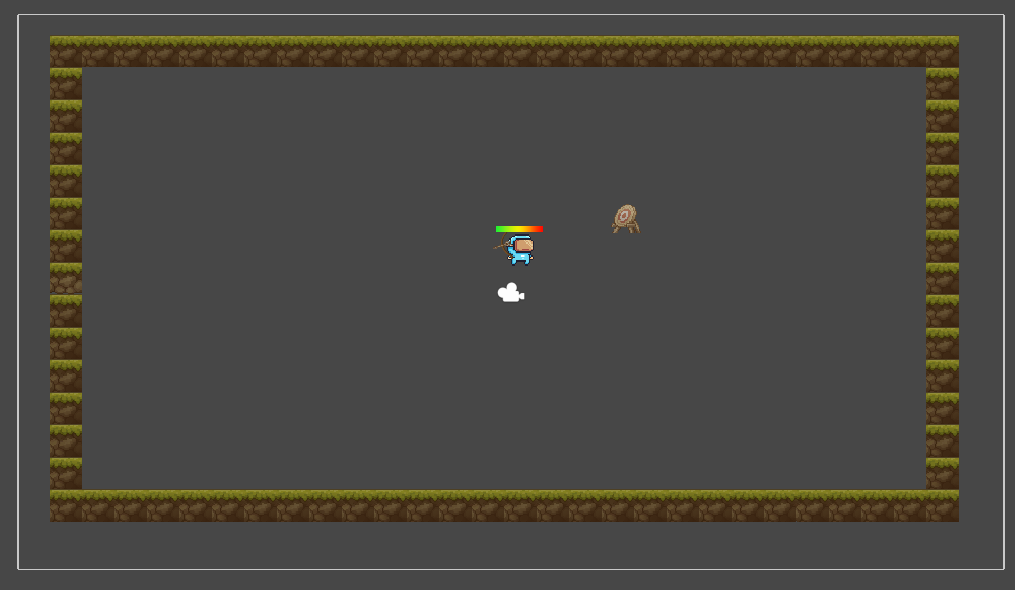
PPO:

Proximal Policy Optimization (PPO) is a reinforcement learning algorithm known for its efficiency and ease of use. Developed by OpenAI, PPO addresses stability and efficiency issues in earlier methods through a clipped objective function that ensures controlled policy updates. Widely used in diverse applications from gaming to robotics, PPO is favored in Unity's ML-Agents for its robust performance in simulating intelligent agents. This makes it ideal for developing complex AI models in varied virtual environments.

We utilized the PPO algorithm, and the agent’s **observations** (inputs for the neural network) were as follows:

1. The difference vector between the player (agent) and the target.
2. The magnitude of this vector.
3. The x-coordinate of the player.
4. The y-coordinate of the player.
5. The x-coordinate of the target.
6. The y-coordinate of the target.

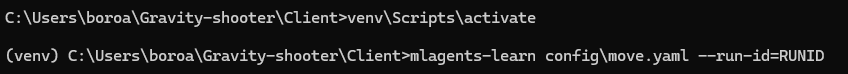
This results in a total of 7 observations. The first item, being a 2D vector, accounts for two observations.

****

(a)

Training:

We began the setup by installing a virtual environment (venv) along with all the necessary packages, which are listed on the official ML-Agents GitHub page (https://github.com/Unity-Technologies/ml-agents). To facilitate training with a GPU, we installed CUDA, a parallel computing platform and application programming interface (API) model created by Nvidia. CUDA allows for dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU). In our case, we're utilizing an NVIDIA GTX 1050 Ti.

After completing all the installations, we navigated to the virtual environment folder (venv), activated it from the command prompt, and executed the following command to start training:

We passed a configuration file along with a RUNID that specifies a unique identifier for the current training session. The mlagents-learn command can receive a variety of arguments depending on the developer's needs. For example, if you want to resume training from a specific checkpoint, you simply pass the --resume flag. If you wish to override an existing run ID, you can use the --force flag.

The observations were taken from 192 environments identical to the one shown in picture (A).

The agent (player) learns the correct angle and force through trial and error by randomly shooting at a target in the environment, which changes position randomly in each episode, to learn to shoot accurately regardless of the target’s position. When the arrow hits the target—a successful shot—the agent receives a reward valued at one. Conversely, missing the target results in a penalty, which is a negative reward of minus one.

The total number of training steps was approximately 49 million, the mean reward increased to 0.6 from an initial value of -1, and the total duration of the training was roughly 67 hours, which is equivalent to about 2.79 days.

The model's output is the agent's decision based on the earlier mentioned observations (inputs) from the environment. This decision consists of the angle and force that lead to a successful shot (target hit).

Evaluating the model:

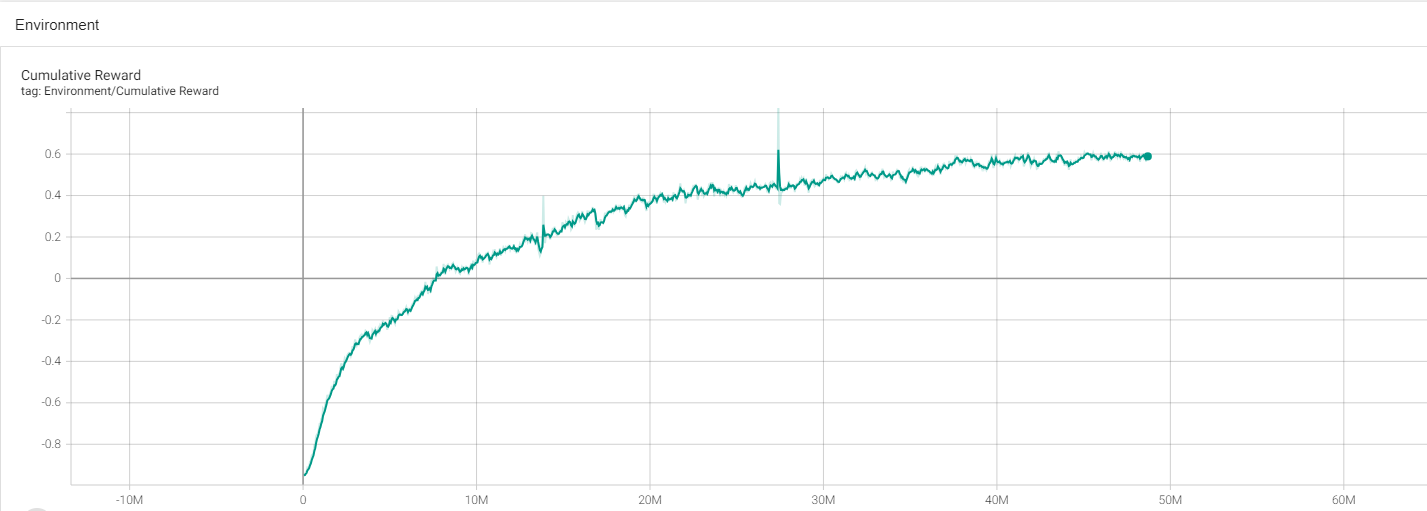
When evaluating reinforcement learning (RL) models, two key performance metrics often considered are the cumulative reward and episode length. The cumulative reward is a total sum of rewards that the agent accumulates over time, providing insight into how well it's learning from its environment to achieve its goal. An increasing cumulative reward trend typically indicates that the agent is improving its policy to maximize rewards. Episode length, on the other hand, reflects the duration or the number of steps an agent takes to reach a terminal state in an environment. Depending on the task, shorter episodes can either mean the agent has learned to reach its goal more efficiently or, conversely, is failing early.

Looking at graph (b) - "Cumulative Reward Graph," we observe a generally increasing trend over time, with the x-axis representing training steps or episodes. The agent’s performance appears to improve as it accumulates more experience, suggesting that the agent is learning a more effective policy over time. There are some fluctuations, including a spike around the 28 million step mark, but the overall trend stabilizes toward the end.

The "Episode Length Graph" - graph (c)- shows a decreasing trend, which generally suggests that the agent is learning to achieve its goal in fewer steps, becoming more efficient. Initially, there is a sharp decrease in the length of episodes, followed by a more gradual decline. The length of the episodes seems to stabilize after about 20 million steps, running parallel to the cumulative reward graph suggesting that the agent has reached a certain level of proficiency.

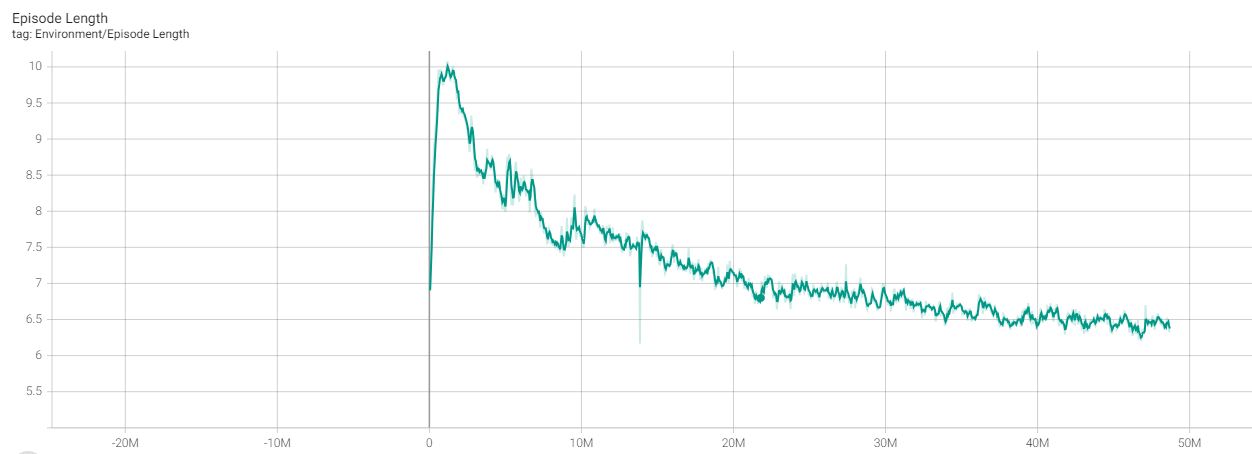
Both graphs together suggest a positive outcome where the agent is not only maximizing rewards but also becoming more efficient in doing so over time.

**Cumulative Reward Graph**

****

(b)

**Episode Length Graph**

****

(c)

Model usage:

After the training session was completed, we obtained a .onnx model that can be integrated with Unity and run in inference mode. However, instead of providing the exact angle and force needed to hit the target, we aimed to offer the player tips regarding their shot. To achieve this, we compared the actual force and angle chosen by the player with the agent's decisions (the model's output), which are very accurate and reliable. By not using the model outputs directly to perform the actual shot, and instead comparing the angles and forces, we achieved better accuracy. This approach was beneficial since the model was not always 100% accurate in executing a perfect shot.

## **2.4. Challenges and Solutions**

Problem: Selecting a cloud environment to deploy the game server for our multiplayer 2D Unity game, "Gravity Shooter," which required management of user credentials.

Solution: We chose AWS, an industry-standard platform that offers one year of free-tier usage for most of its services. This choice provided us with a cost-effective and reliable infrastructure.

Problem: Deciding on the optimal AWS region from the available options for deploying our server (EC2 instance) and database (RDS instance).

Solution: We selected the Stockholm (eu-north-1) region because it is the closest to our target audience, ensuring lower latency and better performance.

Problem: Handling a race condition when two users attempt to register simultaneously.

Solution: We successfully synchronized the two requests without introducing any time delays, ensuring that the users experienced no noticeable disruptions.

Problem: Our lack of extensive knowledge in Reinforcement Learning for building a robust infrastructure that could manage observations in the Unity environment and interface with the neural network (Python).

Solution: We utilized the MLAgents framework, which integrates a Python process with the Unity editor. This setup captures the necessary input for the neural network (observations) and relays the agent's decisions back to the Unity environment, streamlining the development process.

Problem: We encountered challenges in syncing objects across the cloud due to the complex understanding required for manually transmitting each object. This process, while offering better performance, is time-consuming and demands a deep technical comprehension.

Solution: To address this, we utilized built-in features available in Netcode, specifically employing the network transform component to efficiently manage and synchronize data across the cloud.

Problem: Managing object authority in a Unity multiplayer environment posed significant challenges. Ensuring that the right player has control over specific objects at the appropriate times requires a robust system, particularly in a networked game where multiple users interact with the same environment.

Solution: We addressed these challenges by leveraging Netcode for Unity, which provides sophisticated tools for authority management. By implementing Remote Procedure Calls (RPCs), we could efficiently designate control of objects to different players based on game logic and user actions. This approach ensured that authority was dynamically and securely managed across the network, enhancing both gameplay integrity and user experience.

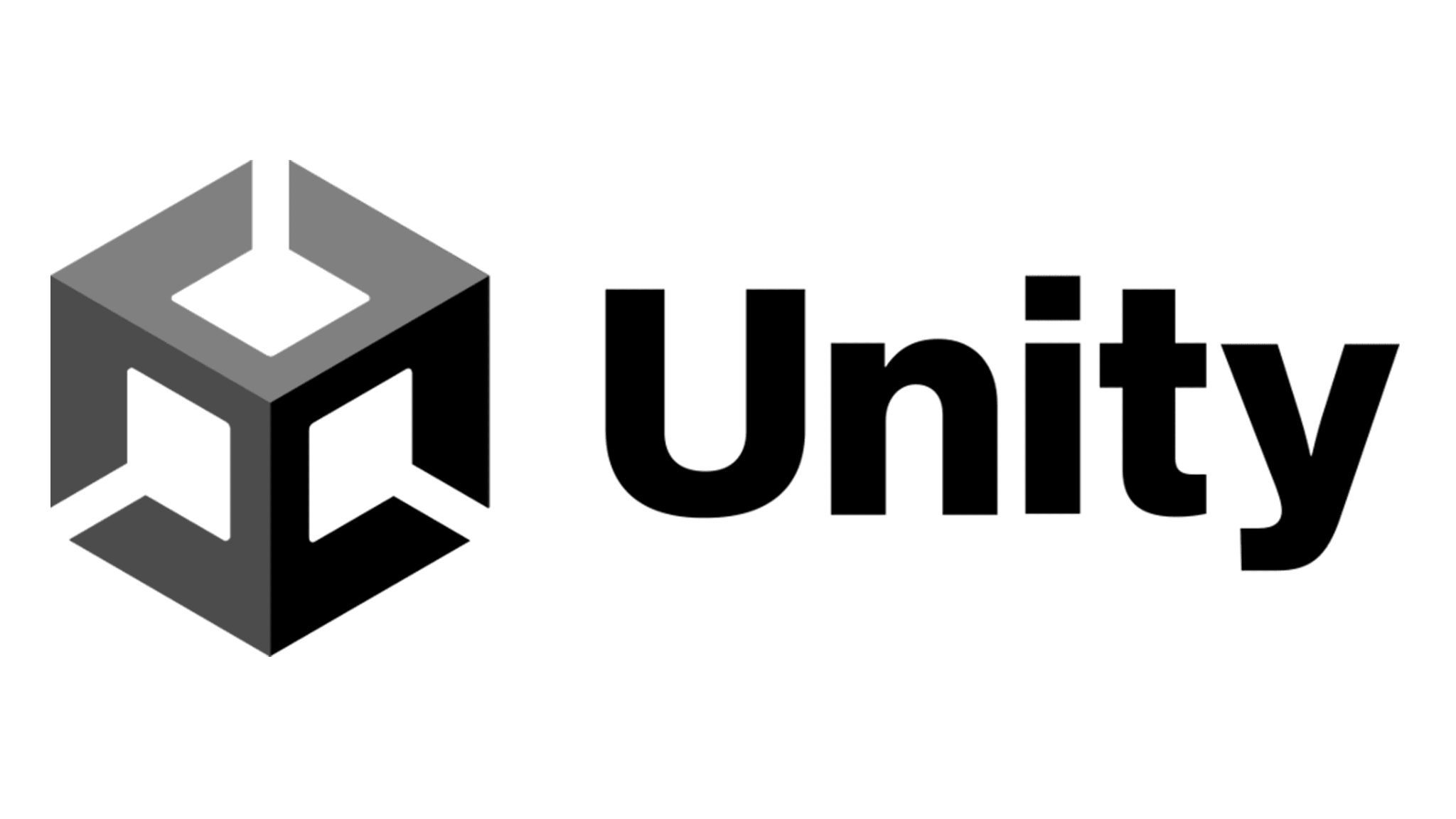
### **2.4.1. Decisions**

* Training an RL model is not a straightforward process; it took us weeks to determine the necessary observations, parameters, and configurations. After succeeding in training our model, which predicts the correct angle and force necessary to hit a target, we faced challenges. The training took place in a separate environment and required a considerable amount of time to complete. We realized that we had not considered obstacles such as trees and rocks during the training. If we had, it would have been very challenging for the agent to learn. After consulting with our advisor, we concluded that the model's predictions would not be applicable to the real game, which includes trees, rocks, and other elements that make the game more engaging and fun. Consequently, we decided to let players train in a different scene—one without obstacles or wind. This setup allows them to focus solely on improving their shooting skills and gaining a fundamental understanding of physics. Once they become proficient, they can advance to the real game and experiment with what they have learned in the training scene.
* Another decision we made was not to rely on the AWS server that manages user credentials for us to synchronize the necessary objects in multiplayer games. This decision was based on our need for a server located closer than Stockholm to ensure better performance. Consequently, we opted for Netcode, which addresses this issue by designating one player as both the host (serving as server and client simultaneously) and the others as clients. Netcode simplifies networking for game developers by handling complex scenarios and reducing latency, which is crucial for real-time multiplayer experiences.

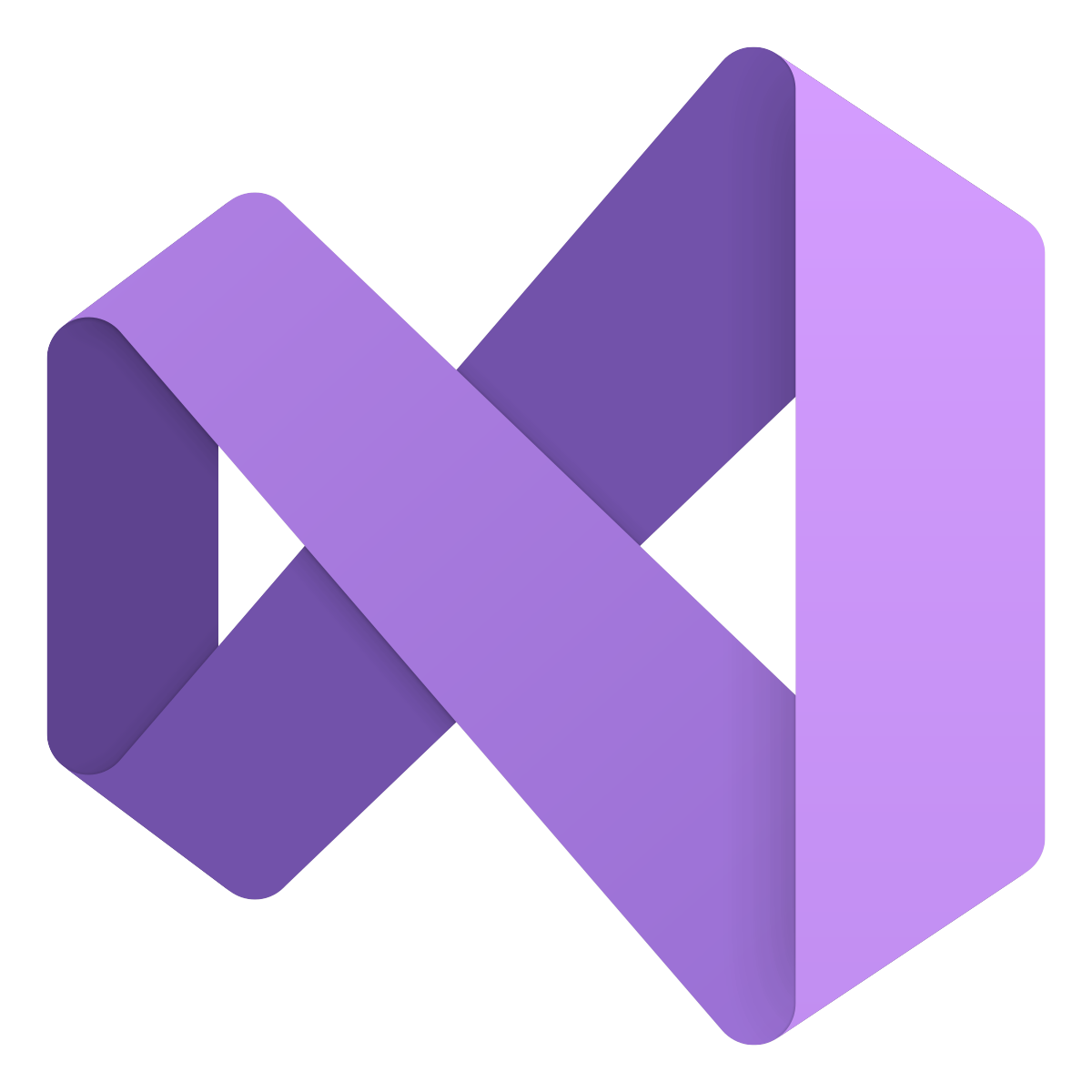
## **2.5. Tools we used:**

For the development of the game, we utilized Unity, leveraging its robust physics engine to create dynamic interactions. To manage and host our server, we opted for AWS. The server itself is developed using Java and Spring Boot, and for database management, we employed MySQL Workbench in conjunction with Amazon RDS. We chose Maven as our build tool within the IntelliJ IDE. For scripting within the game, we used C# due to its seamless integration with Unity, working within the Visual Studio 2022 IDE. For version control, we used Git, coupled with GitHub. To test the REST API endpoints of our server, we utilized Postman.



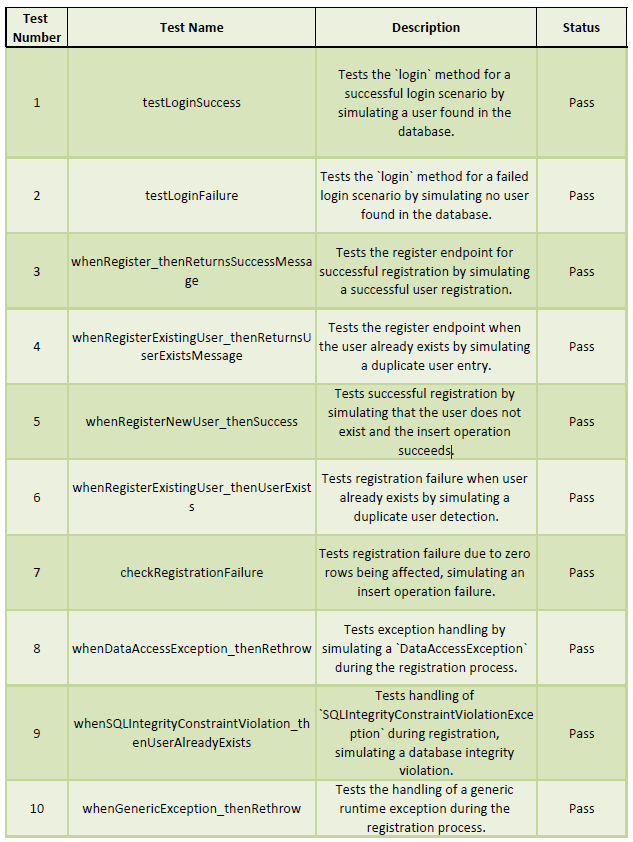






## **2.6. Testing**

Unit Tests:



# **3. User Manual**

**Welcome to Gravity Shooter**

Get ready to challenge the forces of gravity in this exciting shooter game! Whether you’re going solo or joining a team, Gravity Shooter offers a thrilling experience that tests your accuracy and quick thinking.

**System Requirements**

Before installing Gravity Shooter, ensure your system meets the minimum requirements:

System requirements:

* Memory: 4 GB
* Graphics Card: NVidia GTX 1050ti or newer
* CPU: Intel Pentium 4 2.20GHz
* File Size: 100 MB
* OS: Windows 7 or newer

**Getting Started**

**Installation**

1. Download Gravity Shooter installer file from: https://github.com/IbrahimDaoud5/Gravity-shooter.git.
2. Run the installer and follow the on-screen instructions.
3. Once installed, launch the executable file of the game, Gravity-Shooter.exe .

**Creating an Account**

1. Launch Gravity Shooter and select ‘Register’ on the main menu.
2. Enter a Username.
3. Create a Password and confirm it by entering it again.
4. Click ‘Create’ to register your account. You’ll receive immediate feedback on the status of your registration.

**Logging In**

1. Enter your Username and Password.
2. Click ‘Login’.
3. If the credentials are incorrect, a message will inform you of the issue.

**Navigating the Main Menu**

Upon successful login, the main menu will display your username at the top. From here, you can choose to play in Single Player or Multiplayer mode.

**Single Player**

* Play Solo: Test your skills against the clock and aim to hit all targets within the time limit.

**Multiplayer**

* Create Lobby: Set up a new game and wait for others to join.
* Lobbies List: Browse and join available games hosted by other players.

**Game Controls and Interface**

**Performing a Shot**

* To initiate a shot, press and hold the left mouse button. This will cause a power bar to appear above your character.
* The longer you hold the mouse button, the more powerful your shot will be. Release the mouse button to fire.
* Pay attention to the shot data displayed at the top center of the screen, which includes an arrow indicating wind direction—a crucial factor in aligning your shot.

**In-Game Display**

* **Targets:** The number of targets left to hit is displayed in the upper left corner of the screen.
* **Shot Data:** The left top corner and the center of the screen provides shot data. This includes an arrow that shows the wind direction, giving you the information needed to adjust your shot for environmental factors.
* **Power Bar:** Appears above the player when preparing a shot, indicating the shot's potential strength.
* **Timer:** Keep an eye on the time left for your challenge, shown in the upper right corner of the screen.

**Pausing the Game**

* Press the Esc key to access the pause menu with the options to ‘Resume’, go to the ‘Menu’, or ‘Quit’ the game.

**Troubleshooting and Support**

* For issues during gameplay or technical problems, Contact:

[Gravity-Shooter@e.braude]

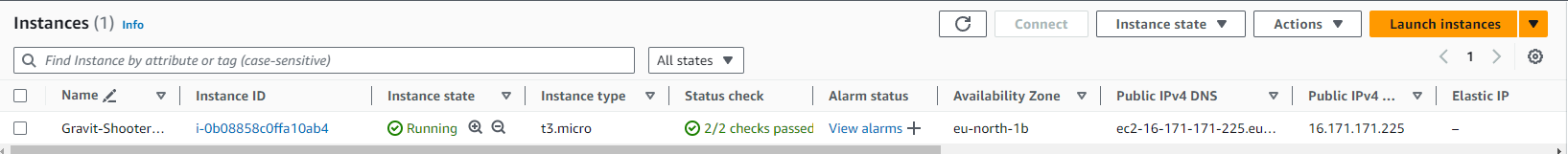
**Conclusion**

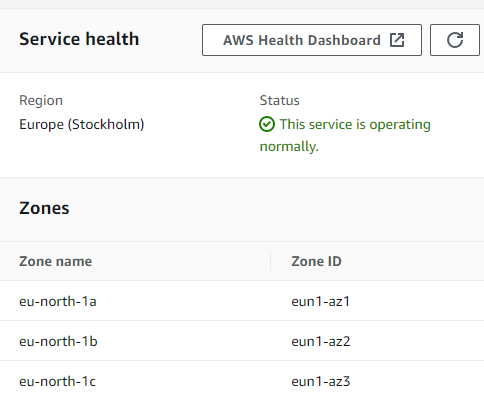
Gravity Shooter offers an engaging blend of fun and education, wrapping challenging gameplay with a lesson in basic physics. Through mastering the effects of gravity and wind resistance, players not only sharpen their gaming skills but also get hands-on experience with the principles that govern our physical world. We hope this manual provides a clear path to both your enjoyment and your learning. Jump in, and let the physics fun begin. Happy shooting!

# **4. Operation & Maintenance Guide**

## **Server**

We chose AWS because it is an industry-standard cloud service and allows us to use most of its products for free for one year within the free tier. We selected the Stockholm region (eu-north-1) because it is the closest destination for our target audience. Although there is a relatively new region in TLV, Israel, it is not included in the free tier.



Zones inside the region:

Accessing the server using SSH

SSH, or Secure Shell, is a protocol that provides a secure method for remote access and administration of devices over unsecured networks. It uses strong encryption to ensure both privacy and data integrity. For instance, when accessing an EC2 instance on AWS, SSH allows us to securely log in and manage the server from anywhere in the world. This capability is crucial for tasks like updating software, configuring services, and running scripts, making it an essential tool for managing cloud resources efficiently and safely.



we runned the command above in the command prompt, the first argument is the private key followed by the ec2 user and the instance public IP address.

Uploading the Server

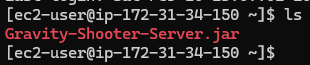
After writing and locally testing the server code, we created a JAR file using the Maven build tool in IntelliJ. We then deployed the JAR to AWS using this command:



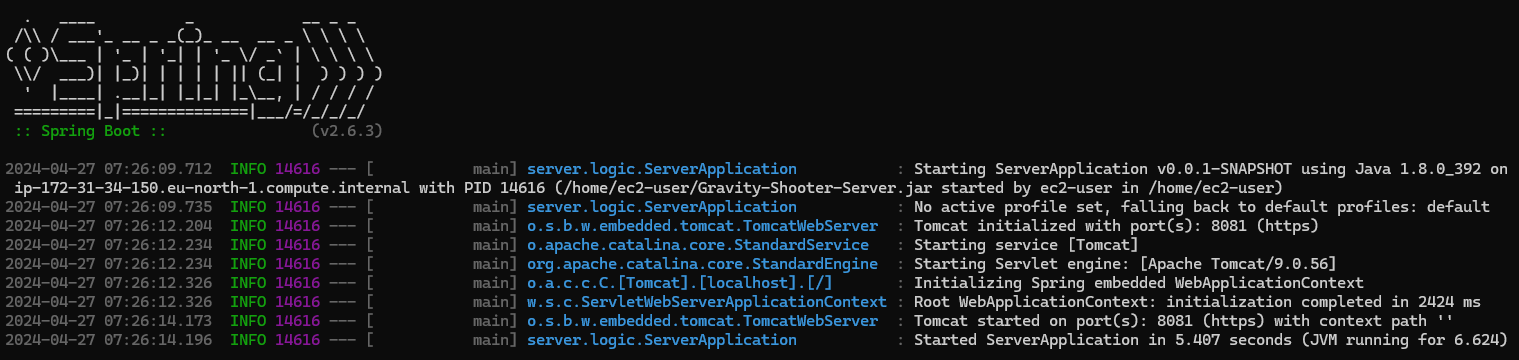
SCP (Secure Copy Protocol) is a network protocol that allows for the secure transfer of files between a local host and a remote host over an SSH connection. It uses the same authentication and provides the same security as SSH. The command takes several arguments: `-i /pathToPrivate-key.pem` specifies the SSH key for secure authentication, `/pathToFile.jar` is the path to the file to transfer, `ec2-user@your-instance-ip` is the user and IP address of the remote host, and `/pathToPut/file.jar` is the destination path on the remote server where the file will be placed.

Running the Server

After connecting to the remote machine (through SSH) we see the JAR file of the server:

  
and we can simply run it by writing this command:

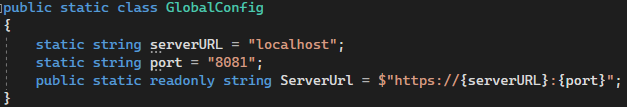
  
and we will see this screen indicating the the Spring Boot server is running on our EC2:



Database:

Our database is MySQL on an Amazon RDS instance. It has one table - users - with fields for username and password to store user credentials. We use the database only for logging in and creating accounts

## **Client**

To be able to connect to the server we need to update the serverURL in the GlobalConfig.cs script to be the public ip of the EC2 instance.

**Lobby and Relay Service Overview**

"Gravity Shooter" utilizes a lobby and relay service to manage multiplayer gameplay. The lobby system allows players to create, join, and manage game sessions, while the relay service facilitates the connection between players, ensuring a seamless multiplayer experience.

**Monitoring and Metrics**

The Unity cloud provides monitoring tools for tracking requests, bandwidth, and other critical metrics. Regular review of these metrics is essential to preemptively address issues such as server overload, bandwidth spikes, or performance bottlenecks. The key metrics to monitor include:

* **All Requests:** Total number of requests to the server.
* **Requests by Type:** Frequency of specific requests (Create, Heartbeat, Join by Code, etc.).
* **Bandwidth Consumption:** Total and average data transmitted.
* **Hosted Allocations:** Number of active server instances.

**Creating and Joining a Lobby**

CreateRelay and JoinRelay functions are utilized to facilitate the creation and joining of a lobby with relay server integration. The relay service uses allocation IDs and join codes to securely connect players.

**Function Maintenance Tips:**

* **CreateRelay:** Ensure that the allocation for the maximum number of players is accurately configured.
* **JoinRelay:** Verify that join codes are correctly retrieved and utilized.

**Troubleshooting Common Issues**

**Lobby Request Limits:**

* **Problem:** Exceeding the API rate limits can cause service disruptions.
* **Solution:** Implement request throttling and review game design to reduce unnecessary API calls.

**Connectivity and Synchronization:**

* **Problem:** Players with unstable internet connections may experience lag or desynchronization.
* **Solution:** Optimize netcode to handle varying latencies and implement a robust client-side prediction system.

**Lobby Creation Errors:**

* **Problem:** Errors during lobby creation can prevent players from starting or joining multiplayer games.
* **Solution:** Implement error handling in CreateLobby and provide informative feedback to the players. Monitor logs for patterns that could indicate deeper issues.

**Relay Service Exceptions:**

* **Problem:** RelayServiceException errors when attempting to create or join a relay.
* **Solution:** Ensure that error logging is detailed enough to diagnose. Consider fallback mechanisms for retrying connections and alerting administrators when service health degrades.

**Maintenance Procedures**

Regular maintenance tasks are critical for ensuring the multiplayer components of "Gravity Shooter" remain operational:

* **Update Services:** Apply updates to the lobby and relay services promptly.
* **Backup Data:** Regularly backup lobby configurations and player data.
* **Security Audits:** Perform security checks to prevent unauthorized access to the lobby system.

# **5. Results & Conclusions**

* We needed to further develop the RL model that demonstrates the educational impact of the game, including considerations for wind and varying gravity. However, training the model with a simple environment proved to be very time-consuming. Additionally, preparing the training environment would have taken a long time, preventing us from working on other parts of the game. Perhaps we should have planned our time more effectively before starting development to allocate the appropriate amount of time to each component.
* The output of the RL algorithm is an angle and force that lead to a target hit. However, in the real world, there are a number of these combinations (angle and force) that can lead to a hit, while the model is capable of learning only one combination.

## **Further improvements**

* Despite utilizing the free version of our networking tools, we encountered persistent latency issues, primarily due to the server's local setup and the uneven internet quality among players. Upgrading to a more robust networking tool and moving to a dedicated server in a central location could help distribute latency more evenly. Additionally, implementing advanced network management strategies could enhance game fairness and performance. Client Authority allows for high reactivity but poses sync risks. Action Anticipation and Prediction improve security and smooth gameplay but require careful handling of visual and synchronization discrepancies. Server Side Rewind enhances accuracy and favors the attacker but may cause issues like shots landing despite cover due to latency. Each of these methods can help refine gameplay, making it fairer and more enjoyable for all players.
* The RL model can generate the correct angle and force to hit a target, but it doesn't account for obstacles in the shot's trajectory. We plan to train the model to consider these obstacles as well as wind conditions. Although this training might take a significant amount of time, it could eventually work and integrate seamlessly into the game as intended.

# **6. References**

1. Chang, K. E., Chen, Y. L., Lin, H. Y., & Sung, Y. T. (2008). Effects of learning support in simulation-based physics learning. *Computers & Education*, *51*(4), 1486-1498.‏
2. Annetta, L. A. (2008). Video games in education: Why they should be used and how they are being used. Theory into practice, 47(3), 229-239.‏
3. Squire, K. (2003). Video games in education. Int. J. Intell. Games & Simulation, 2(1), 49-62.‏

3.1. Graesser, A., Chipman, P., & Leeming, F. (2009). Deep learning and emotion in serious games. In Serious games (pp. 105-124). Routledge.

1. <https://www.mathworks.com/discovery/machine-learning.html>
2. <https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292>
3. [https://www.analyticsvidhya.com/blog/2021/02/introduction-to-reinforcement-learning-for-beginners](https://www.analyticsvidhya.com/blog/2021/02/introduction-to-reinforcement-learning-for-beginners/)
4. <https://en.wikipedia.org/wiki/Amazon_Web_Services>
5. https://aws.amazon.com/gamelift/
6. <https://docs.aws.amazon.com/gamelift/latest/developerguide/gamelift-intro.html>

9.1. Lample, G., & Chaplot, D. S. (2017, February). Playing FPS games with deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).‏

1. <https://github.com/miyamotok0105/unity-ml-agents/blob/master/docs/Training-ML-Agents.md>
2. Ducheneaut, N., & Moore, R. J. (2005). More than just ‘XP’: Learning social skills in massively multiplayer online games. *Interactive technology and smart education*.‏
3. <https://flexbooks.ck12.org/cbook/ck-12-middle-school-physical-science-flexbook-2.0/section/10.10/primary/lesson/projectile-motion-ms-ps/>

12.1. Bowling, M., Fürnkranz, J., Graepel, T., & Musick, R. (2006). Machine learning and games. Machine learning, 63(3), 211.‏

1. <https://aws.amazon.com/products/databases/>
2. https://unity.com/how-to/unity-test-framework-video-game-development
3. https://nunit.org/
4. <https://www.headspin.io/blog/game-testing-a-complete-guide-to-its-types-and-processes>
5. Code Monkey channel: https://www.youtube.com/channel/UCFK6NCbuCIVzA6Yj1G\_ZqCg

‏