Implementation and study of user strategies within a custom video game environment

Nick, Iliadis
Department of Informatics, Ionian University
p14ilia@ionio.gr

Stergios, M, Palamas
Department of Regional Development, Ionian University
spalamas@ionio.gr

Phivos, I., Mylonas∗[*]
Department of Informatics, Ionian University
fmylonas@ionio.gr

ABSTRACT

[*]The quality of the gameplay of a computer game is one of the most important factors for its success. Even the best game visuals (graphics) can not compensate for a bad gameplay. In this context, the difficulty of the game is an important factor for the overall gaming experience. Dynamically adjusting the difficulty to the player's skills can provide a more personalized and challenging playing environment. The present study suggests an implementation of a Dynamical Difficulty Adjustment mechanism, integrated in a First Person Shooter developed from scratch with Unreal Engine 4. The mechanism is based on an extensive set of user statistics/metrics collected during the game and processed with data-mining techniques, in order to rate user's skills and performance on every stage of the game. The rating is then used, to automatically adjust a set of gameplay parameters and behaviors for the next part of the game, providing a custom-tailored difficulty level that matches the player's skills.

CCS CONCEPTS

• **Interactive Games**; • **Personalization**; • **User centered design**;

KEYWORDS

Artificial Intelligence, video games, user behavior, skill level

ACM Reference Format:
Nick, Iliadis, Stergios, M, Palamas, and Phivos, I., Mylonas∗. 2021. Implementation and study of user strategies within a custom video game environment. In *25th Pan-Hellenic Conference on Informatics (PCI 2021), November 26–28, 2021, Volos, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3503823.3503832

1 INTRODUCTION

Modern game development tools, like the popular Unreal Engine, have made possible the development of modern, state-of-the art games, even by small working teams, lacking the resources and budget of big Game Studios. The gameplay is one of the most important parts of a digital game, judging in a large extend it's market success or failure. Even the best game visuals cannot compensate for a bad gameplay. In this context, game difficulty is an important factor of the gaming experience. Making a game too difficult to play will discourage many players, while in contrary, a very easy game will hardly provide any challenge for users to keep playing on. Csikszentmihalyi [1] first proposed that players, when kept away from the two opposite states of boredom and frustration, experience a "flow channel".

Difficulty adjustment is integrated on virtually every digital game, often in its most simple form: a predefined set of difficulty levels, adjusting some of the gameplay parameters without taking into account player's performance [2]. Dynamic Difficulty Adjustment (DDA) on the other hand, automatically and constantly modifies a game's difficulty level in order to match the player's in-game performance, providing thus a custom-tailored and challenging user experience.

There are several approaches to DDA implementation [3]: Probabilistic Methods, Single and multi-layered perceptions, Dynamic Scripting, Hamlet System, Reinforcement Learning, Upper Confidence Bound for Trees and AI Neural Networks and Self-organizing System and AI Neural Networks.

This study, building upon the principles of the Hamlet System [4] and Reinforcement Learning [5] [6] DDA approaches, suggests a method for dynamic difficulty adjustment of a First Person Shooter (FPS) game developed from scratch with Unreal Engine 4. The adjusting mechanism is based on the maintenance of an extensive set of user-statistics and metrics during the game, which, by utilizing data-mining techniques, are used to rate player's skills on every stage/part of the game and alter crucial gameplay parameters for the next stage of the game, matching the player's overall performance.

2 GAMEPLAY AND GAME ELEMENTS

The game was designed and implemented as a First Person Shooter (FPS), with additional characteristics of a survival game, adopting movement and shooting mechanics featured by popular games of this specific genre. Typical elements of the FPS game genre have been incorporated, such as a variety of guns (pistol, assault rifle, shotgun, grenade launcher and sniper), a health system, an energy system, a flashlight, the enemies and a dark environment. The player, throughout the game, must fight to maintain health, energy and battery (flashlight) levels, having a constant feeling of struggling to survive.

The player's health level can reach up to one hundred. The energy regains when the user is standing still and decays while the

[*]Corresponding Author.

PCI 2021, November 26–28, 2021, Volos, Greece
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9555-7/21/11…$15.00
https://doi.org/10.1145/3503823.3503832

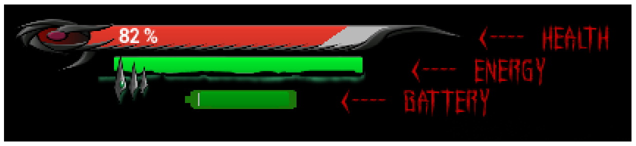


Figure 1: Health, energy, battery bars

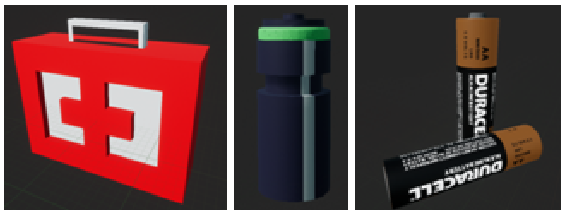


Figure 2: Health, energy, battery collectable items

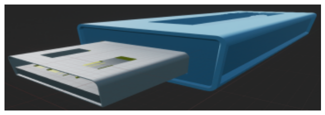


Figure 3: Flash Drive

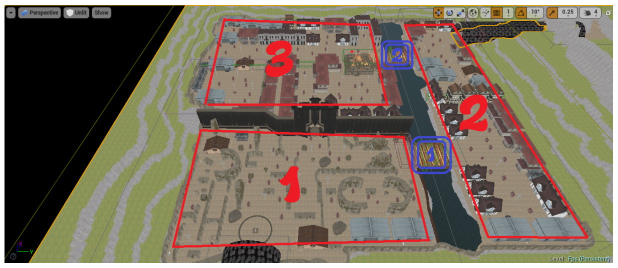


Figure 4: Structure of the game - map

player is moving. Running decreases the energy twice as fast. The battery (used by the flashlight) decreases while the player keeps the flashlight on and recharges (very slowly) when the flashlight is off. There are health, energy and battery packs, scattered across the game scene, which, when collected by the player, instantly increase the corresponding levels of resources.

The player's main mission is to survive throughout the game, eliminating enemies (bots), while trying to collect 20 flash drives. The later are the key item of the game and finding them all, is the only way to open the main gate to game completion.

The game is structured in a way that ensures the player cannot miss a track or use fewer resources, hindering the validity of the data collected throughout the game and described later. The player must kill the enemies while collecting the flash drives, while keeping an eye on his energy, health and battery levels. Equally important for surviving the game, is the player's bullet stock. The structure of the game-map (stage) is shown in Figure 4

The game scene is divided into 3 areas, and features two checkpoints (appearing in blue in Figure 4). User cannot proceed from area to area and finish the game, without first collecting the appropriate number of flash drives on each section.

While playing on each area, a set of variables (described later), is used to collect user-statistics and determine user's skills and tactics, adjusting accordingly the behavior of next area's enemies/bots in order to provide a more challenging level of difficulty.

When it comes to guns, there are five different types of weapons players can use.

Each weapon induces different amount of damage on enemies. Specifically:

1. The pistol does 40 units basic damage.
2. The assault rifle does 30 units basic damage.
3. The shotgun does 60 units basic damage.
4. The grenade launcher does 100 units basic damage.
5. The sniper does 100 units basic damage.

There are two different types of enemies that can hurt the player and both of them can kill the player if they aren't eliminated. The artificial intelligence (AI) of the game uses two different attacking mechanisms:

- The first one focuses on running towards player's location (short-range attack).
- The second one locates the player's position and deals damage by firing against him with a weapon (long-range attack).

Both types of bots can "sense" the player's presence, if the player fires a weapon close to them, and can "see" the player up to 82 ft away (2500 units in Unreal Engine 4) within a 90° angle view area. All enemies start with 100 health points and a basic health regeneration rate of 0.4 health points/second when being hurt by the player. The walking speed of the bots is 4 km/h (400 units/h in Unreal Engine 4).

Each bot, can deal a different amount of damage to the player. The normal bot (close range attack) can deal up to 20 units damage per attack (meaning that the player is capable to survive up to five consecutive attacks). On the other hand, the weapon bot (long range attack) can deal up to 3 units of damage per fired bullet, provided of course the bullet doesn't miss the target, and thus, the player can survive up to 34 successful shots.

3 DATA COLLECTION

When the game reached at the stage of development where we could collect the desired variables, we decided to put some users to play the game. The results were considered valid when the players were able to finish the first, the second track or the whole game. Failed attempts were excluded for the validity of our measurements. Measurements were made at the end of the game, at the two checkpoints or when the user chose (provided the user had reached at least the first checkpoint).

The variables/metrics collected during the testing phase of the game, are the following:

Weapon related variables:

- Total bullets used (total amount of bullets used by the player)
- Total bullets on target (total amount of bullets that hit the target/enemy)

Figure 5: Available weapons: pistol, assault rifle, shotgun, grenade launcher and sniper

Figure 6: Close-range attacking bots (2 Skins/types) and Long-range attacking bot (featuring weapon).

- Pistol bullets used (total amount of pistol bullets used by the player)
- Pistol bullets on target (total amount of pistol bullets that hit the target/enemy)
- Assault Rifle bullets used (total amount of assault rifle bullets used)
- Assault Rifle bullets on target (total amount of assault rifle bullets that hit the target/enemy)
- Shotgun bullets used (total amount of shotgun bullets used)
- Shotgun bullets on target (total amount of shotgun bullets that hit the target/enemy)
- Grenade Launcher bullets used (total amount of rockets of the grenade launcher used)
- Grenade Launcher bullets on target (total amount of grenade launcher rockets that hit the target/enemy)
- Sniper bullets used (total amount of sniper bullets used)
- Sniper bullets on target (total amount of sniper bullets that hit the target/enemy)

Time variables:

- Time Spent Aiming (aiming time)
- Time Spent Sprinting (time running)
- Time Spent Crouching (crouching time)
- Time Spent Playing (total playing time)
- Time to reach the First Checkpoint (First Bridge)
- Time to reach the Second Checkpoint (2nd Bridge)
- Time to reach the Third Checkpoint (End of Game)

General variables:

- Footsteps (steps made in the game)
- Bots Killed (enemies killed)
- Damage Taken (damage done by enemies)

Accuracy variables:

- Accuracy (General Bullets on target x 100 / Total Bullets Used)
- Accuracy Grade[1] (Bots killed / Total Bullets Used)
- Total Bullets Missed (how many bullets did not hit the enemy)
- Pistol Bullets Missed (how many bullets of the pistol have missed target)

[1]The Accuracy Grade variable was used to measure players' accuracy; fewer bullets per killed bot indicate more headshots.

- Assault Rifle Bullets Missed (how many assault rifle bullets have missed target)
- Shotgun Bullets Missed (how many bullets of the shotgun have missed target)
- Grenade Launcher Bullets Missed (how many rockets of the grenade launcher have missed target)
- Sniper Bullets Missed (how many sniper bullets have missed target)
- Total Headshots (how many times the player has hit the head of the enemy)
- Pistol Headshots (how many times the player has hit the head of the enemy with a pistol)
- Assault Rifle Headshots (how many times the player has hit the head of the enemy with an assault rifle)
- Grenade Launcher Headshots (how many times the player has hit the head of the enemy with a grenade launcher)
- Sniper Headshots (how many times the player has hit the head of the enemy with a sniper)

4 DATA ANALYSIS & VISUALIZATION

The data collected by the users playing the game were analyzed by utilizing the popular Weka tool[2]. Weka is a collection of machine learning algorithms for data mining operations. It includes tools for data pre-processing, sorting, regression, clustering, association rules, and visualization. In particular, version 3.8.2 was used to provide a tangible visualization of the collected data and to create a training dataset. Initially, the data were imported in Weka and were classified using the J48 algorithm. The classification resulted in a trained machine learning model (.model), outlining the variables with the highest impact, specifically:

- Accuracy
- Time Playing
- Damage Taken
- Accuracy Grade

Then we considered the model that came out as a test one and by utilizing the "supplied test set" and "cross-validation folds 10" Weka functionalities, we conducted a classification of the users who played our game in the following 5 categories/skill-levels:

- Very Bad (VB)
- Bad (B)
- Normal (N)
- Good (G)
- Very Good (VG)

The data collected by the players during the evaluation phase, were divided in two categories: those collected in the 1st part[3] of the game (1st area) and those collected in the 2nd part[4] of the game (2nd area).

4.1 DECISION TABLES

Decision tables are a visual representation of actions to be performed according to given parameters. In our case, two decision tables were created, one for each area/part of the game. The two decision tables (Table 1, Table 2) illustrate how the combined range of values from the 4 variables/metrics, are used to classify the player in one of the five defined skill-levels.

4.2 UNREAL ENGINE VISUALIZATION

Once the rules that would define the behavior were produced, they had to be displayed within the Unreal Engine. We first implemented the rules in pseudocode and then passed them to the graphics engine. In other words, the transfer of data from the decision table to the actual code was first implemented in pseudocode and then transferred to the unreal engine.

Above pseudocode was then transferred to the Unreal Engine. As mentioned above we have a categorization of the player who plays the game into four categories. Regarding the first category, i.e., that of the very bad player we have the following implementation in code in the Unreal Engine graphics machine:

5 EVALUATION

The variables were weightened, so players can not fall into more than one categories/ratings at the same time:

- “Accuracy” got a weight of 1, as the most important variable.
- “Damage Taken” got a weight of 0.75.
- “Time Playing” got a weight of 0.2, and finally
- “Accuracy Grade” got a weight of 0.1.

In this way, e.g., if a player performs very well on Damage Taken, Time Playing, and Accuracy Grade can be rated above a player who has very good Accuracy. Players having a total score greater than 1, are then classified in one of the 5 skill-levels (Very Bad, Bad, Medium, Good, Very Good).

By combining the collected user-statistics and the decision tables on the Weka tool, the following (Table3 and Table 4) clustering of the test-users has emerged for each game area.

The player’s total skill-rating (Very Bad, Bad, Medium, Good, Very Good), is finally used by the game, in order to fine-tune specific gameplay parameters/variables, adjusting the overall game difficulty to user’s skills and providing a more challenging/personalized experience. The adjusted game variables/parameters are:

Player-related parameters:

1. Health Range of the player.
2. Energy Decay (how fast the player is getting tired)
3. Energy Regeneration (how fast the player’s energy is regenerating when resting)
4. Flashlight Decrease Rate (how fast the batteries discharge)
5. Flashlight Increase Rate (how fast the batteries recharge when flash-light is off)

Weapons-related parameters:

1. Pistol Damage (How much damage the gun does)
2. Assault Rifle Damage (How much damage the rifle does)
3. Damage to hunting weapons (how much damage the shotgun does)
4. Grenade Launcher Damage (How much damage the grenade launcher does)
5. Sniper Damage (How much damage the player’s sniper weapon does)

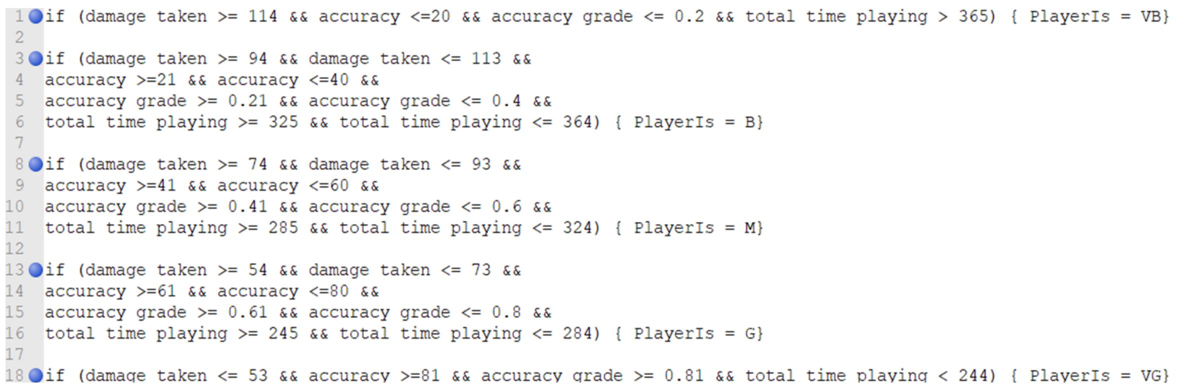
[2]https://www.cs.waikato.ac.nz/ml/weka/
[3]https://docs.google.com/spreadsheets/d/1wHjbzMx5OrI1tk3rCXWR7fqMnJcT48vcVp3hR-uz1P4/edit?usp=sharing
[4]https://docs.google.com/spreadsheets/d/1AE03hzFVZSm0brs1527lO83vCtdQpmi-Lei46XF7BYs/edit?usp=sharing

Table 1: Decision table Area One

Damage Taken	Accuracy	Accuracy Grade	Total Time Playing	PlayerIS
114 - ∞	0 - 20	0 - 0.2	365 - ∞	VB
94 - 113	21 - 40	0.21 - 0.4	325 - 364	B
74 - 93	41 - 60	0.41 - 0.6	285 - 324	M
54 - 73	61 - 80	0.61 - 0.8	245 - 284	G
0 – 53	81 - 100	0.81 - 1	0 – 244	VG

Table 2: Decision table Area Two

Damage Taken	Accuracy	Accuracy Grade	Total Time Playing	PlayerIS
232 - ∞	0 - 20	0 - 0.2	903 - ∞	VB
212 - 231	21 - 40	0.21 - 0.4	803- 902	B
192 - 211	41 - 60	0.41 - 0.6	703 - 802	M
172 - 191	61 - 80	0.61 - 0.8	603 - 702	G
0 - 171	81 - 100	0.81 - 1	0 – 602	VG



```
if (damage taken >= 114 && accuracy <=20 && accuracy grade <= 0.2 && total time playing > 365) { PlayerIs = VB}

if (damage taken >= 94 && damage taken <= 113 &&
accuracy >=21 && accuracy <=40 &&
accuracy grade >= 0.21 && accuracy grade <= 0.4 &&
total time playing >= 325 && total time playing <= 364) { PlayerIs = B}

if (damage taken >= 74 && damage taken <= 93 &&
accuracy >=41 && accuracy <=60 &&
accuracy grade >= 0.41 && accuracy grade <= 0.6 &&
total time playing >= 285 && total time playing <= 324) { PlayerIs = M}

if (damage taken >= 54 && damage taken <= 73 &&
accuracy >=61 && accuracy <=80 &&
accuracy grade >= 0.61 && accuracy grade <= 0.8 &&
total time playing >= 245 && total time playing <= 284) { PlayerIs = G}

if (damage taken <= 53 && accuracy >=81 && accuracy grade >= 0.81 && total time playing < 244) { PlayerIs = VG}
```

Figure 7: Pseudocode used for the first area (first player categorization).

Table 3: Clustering of the users on Area One

Damage Taken	Accuracy	Accuracy Grade	Total Time Playing	PlayerIS
57	26	0.1185	431	B
102	77	0.5185	326	G
130	35	0.2903	277	B
171	39	0.2708	524	B
56	65	0.4	281	G
73	36	0.2608	195	B
58	29	0.2702	197	B
93	37	0.2888	271	B
9	61	0.3888	117	G
94	72	0.4827	249	G
99	72	0.5454	252	G
117	74	0.5185	321	G
42	36	0.3469	397	B
150	29	0.1392	539	VB
68	71	0.5714	304	G

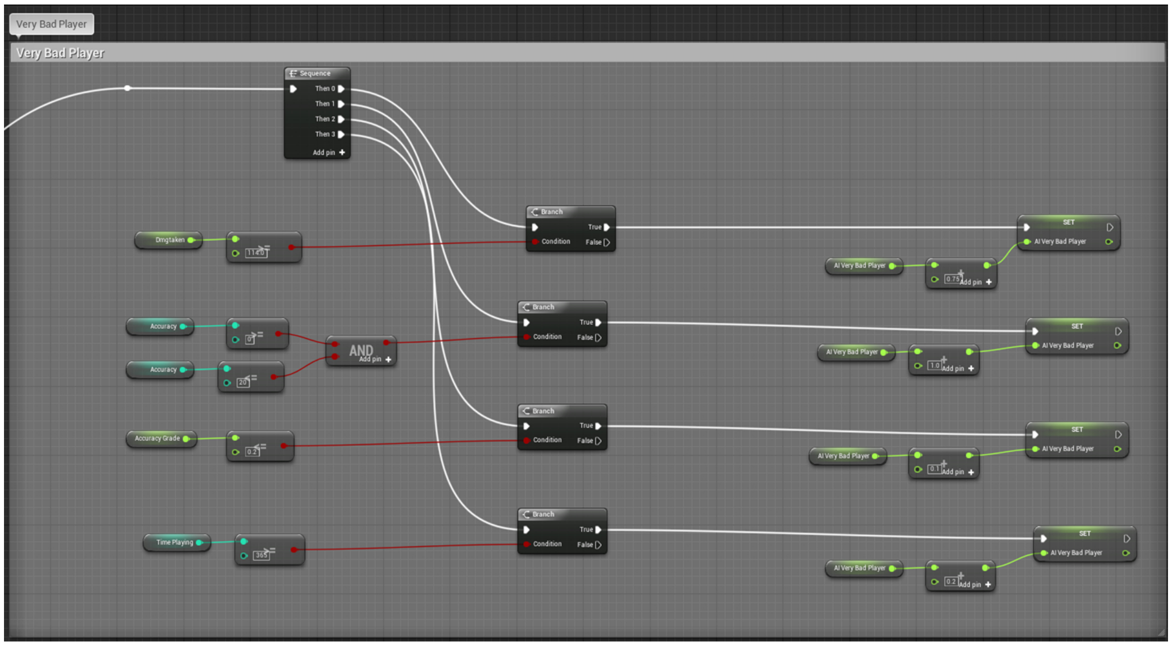


Figure 8: Very bad player categorization visualization within the Unreal Engine environment.

Table 4: Clustering of the users on Area Two

Damage Taken	Accuracy	Accuracy Grade	Total Time Playing	PlayerIS
216	28	0.1361	1004	B
242	64	0.4561	686	G
132	47	0.3595	773	M
221	71	0.5869	551	G
216	28	0.1361	1004	B

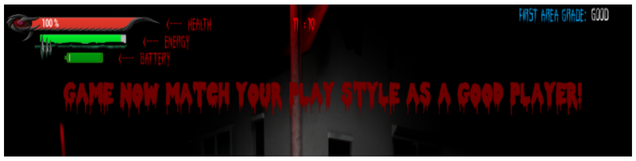
Artificial Intelligence - related variables:

1. LOSHearing Threshold (the range that that bots are "listening" for player's presence)
2. Hearing Threshold (bots' "hearing" ability)
3. Sight Radius (how long the bots can visually detect the player)
4. Damage to Player by Normal Bot (how much damage the normal bots induce to player per attack)
5. Damage to Player by Weapon Bot (how much damage the bots with weapons induce to player per attack)
6. Moving Speed (how fast the bots move)
7. Bot Health (how much total life the bots have)
8. Bot Health Regenerate (how fast do the bots regain health when not attacked by the player)

Track-related variables:
Light Control (changes the intensity of lights on the track)

6 SCOREBOARDING

Whenever the player finishes a section (area) of the game, or the whole game, information is provided on the skill/rating and the



GAME NOW MATCH YOUR PLAY STYLE AS A GOOD PLAYER!

Figure 9: Player's grade on the first area

resulting adjustment of the game's difficulty level for the rest of the game.

Specifically, when the player has completed the 1st part of the game (area 1), a message appears on the screen informing the player that the game has been customized to match player's skills, while the calculated grade is displayed on the upper right corner (Figure 9).

At this point, the game-code adjusts accordingly all the gameplay parameters mentioned on section 6 for the next area. When the player successfully completes the second part of the game, the whole process is repeated and the player is informed accordingly on the new and the previous skill rating (Figure 10).

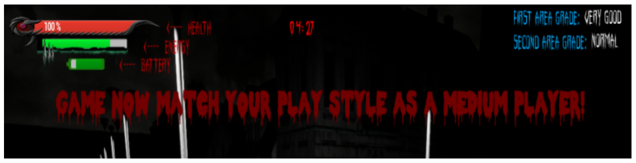
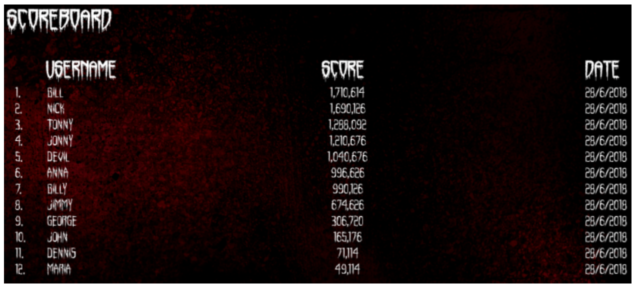

100 %
HEALTH
ENERGY
BATTERY
04:27
FIRST AREA GRADE: VERY GOOD
SECOND AREA GRADE: NORMAL
GAME NOW MATCH YOUR PLAY STYLE AS A MEDIUM PLAYER!

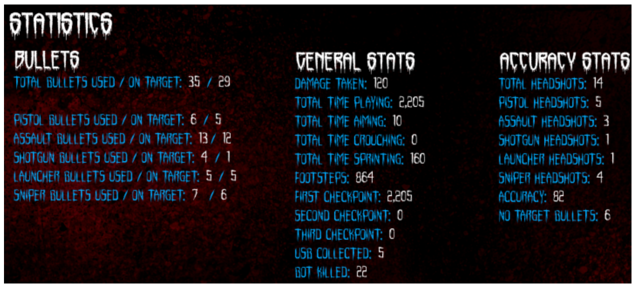
Figure 10: Player's grade on the second area



SCOREBOARD

	USERNAME	SCORE	DATE
1.	BILL	1,710,614	28/6/2018
2.	NICK	1,690,126	28/6/2018
3.	TONNY	1,288,092	28/6/2018
4.	JONNY	1,210,676	28/6/2018
5.	DEVIL	1,040,676	28/6/2018
6.	ANNA	996,626	28/6/2018
7.	BILLY	990,126	28/6/2018
8.	JIMMY	674,626	28/6/2018
9.	GEORGE	306,720	28/6/2018
10.	JOHN	165,176	28/6/2018
11.	DENNIS	71,114	28/6/2018
12.	MARIA	49,114	28/6/2018

Figure 11: Player's scoreboard



STATISTICS

BULLETS
TOTAL BULLETS USED / ON TARGET: 35 / 29

PISTOL BULLETS USED / ON TARGET: 6 / 5
ASSAULT BULLETS USED / ON TARGET: 13 / 12
SHOTGUN BULLETS USED / ON TARGET: 4 / 1
LAUNCHER BULLETS USED / ON TARGET: 5 / 5
SNIPER BULLETS USED / ON TARGET: 7 / 6

GENERAL STATS
DAMAGE TAKEN: 120
TOTAL TIME PLAYING: 2,205
TOTAL TIME AIMING: 10
TOTAL TIME CROUCHING: 0
TOTAL TIME SPRINTING: 160
FOOTSTEPS: 864
FIRST CHECKPOINT: 2,205
SECOND CHECKPOINT: 0
THIRD CHECKPOINT: 0
USB COLLECTED: 5
BOT KILLED: 22

ACCURACY STATS
TOTAL HEADSHOTS: 14
PISTOL HEADSHOTS: 5
ASSAULT HEADSHOTS: 3
SHOTGUN HEADSHOTS: 1
LAUNCHER HEADSHOTS: 1
SNIPER HEADSHOTS: 4
ACCURACY: 82
NO TARGET BULLETS: 6

Figure 12: Player's statistics

When a player finishes playing either by successfully completing the game or failing to do so (e.g. by getting killed), a scorecard database is maintained and displayed. The total score is calculated by the player's performance on Accuracy, Accuracy Grade, Playing Time and Damage Taken. The player has also access to the user statistics/metrics registered throughout the gaming time.

7 FUTURE WORKS

Ideas for future improvements are focused more on the game Artificial Intelligence rather than new game features:

- Optimizing how the Artificial Intelligence reacts to player's skills
- More extensive gameplay fine-tuning in response to user's rating
- Maintenance of user metrics/scores on the Cloud rather than locally.

Enemy bots could by programmed to hide away when the player is at distance and, depending on the player's active weapon and ranking, they could seek cover behind buildings or other objects/structures for protection. A behavior tree could also be utilized so bots behave accordingly to user's state, e.g. when the player is vulnerable (low healht/energy) the bots can attack, while search for a cover when player's resources are high. Ideally, when the player is ranked as very good, bots should behave like controlled by other human players.

Regarding gameplay fine-tuning in response to user's skills, side-levels could be created, transferring good players to challenging environments, while game-adjusted gates, could close specific paths to skilled players, forcing them to seek new ways to proceed.

Finally, cloud-based maintenance of the scores and the users' metrics, could create a more competitive and challenging environment for the players.

8 CONCLUSION

Many games utilize a common, fixed, mechanism on adjusting the overall difficulty level and how the game reacts to the player. This can often lead to an excessively difficult game that will discourage players, or a very easy one, that most players won't find challenging at all. By utilizing in-game player statistics/metrics, and analyzing them with data mining / machine learning techniques, the game can self-adjust a variety of gameplay parameters to match user's skills and performance on every part of the game, providing a far more custom-tailored and challenging experience to the player.

ACKNOWLEDGMENTS

This research was co-financed by the European Union and Greek national funds through the Competitiveness, Entrepreneurship and Innovation Operational Programme, under the Call "Special Actions "Aquaculture" - "Industrial materials" - "Open innovation in culture""; project title: "Strengthening User Experience & Cultural Innovation through Experiential Knowledge Enhancement with Enhanced Reality Technologies — MON REPO"; project code: T6ΥΒΠ - 00303; MIS code: 5066856

REFERENCES

[1] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience, Harper Row, NewYork, NY,USA, 2009.
[2] R. Koster, A theory of fun for game design. Sebastopol (Cali.), OReilly Media, 2014.
[3] Mohammad Zohaib, Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review, Advances in Human-Computer Interaction, Volume 2018, Article ID 5681652, 12 pages, November 2018
[4] R. Hunicke and V. Chapman, "AI for Dynamic Difficulty Adjustment in Games," in Proceedings of the Challenges in Game Artificial Intelligence AAAIWorkshop, pp. 91–96, San Jose,Calif, USA, 2004.
[5] J. Hagelback and S. J. Johansson, "Measuring player experience on runtime dynamic difficulty scaling in an RTS game," in Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG), pp. 46–52,Milano, Italy, September 2009.
[6] C. H. Tan, K. C. Tan, and A. Tay, "Dynamic game difficulty scaling using adaptive behavior-based AI," IEEE Transactions on Computational Intelligence and AI in Games, vol. 3, no. 4, pp.289–301, 2011.
[7] Unreal Engine Documentation, Retrieved October 04, 2021 from https://docs.unrealengine.com/en-us
[8] Weka, October 04, 2021 from https://www.cs.waikato.ac.nz/ml/weka/documentation.html
[9] Artificial Intelligence in Games A Survey of the State of the Art, Technical Memorandum DRDC Ottawa TM 2012-084, August 2012
[10] Russell, Stuart J. Norvig, Peter, Artificial Intelligence : A Modern Approach, Pearson Education Limited, Malaysia
[11] Ian Millington, John Funge, Artificial Intelligence for Games (2nd. ed.), Boca Raton.