



GRADUATE PROGRAMS INSTITUTE

**MEASURING THE IMPACT OF ADVANCED VIDEO GAME
PROGRAMMING ON PLAYER ENGAGEMENT AND
EXPERIENCE**

Omar Nameer Jasim ALQAYSI

MASTER'S THESIS

İstanbul, October 2024

MEASURING THE IMPACT OF ADVANCED VIDEO GAME PROGRAMMING ON PLAYER ENGAGEMENT AND EXPERIENCE

Omar Nameer Jasim ALQAYSI

MASTER'S THESIS

COMPUTER ENGINEERING DEPARTMENT

COMPUTER ENGINEERING MASTER'S PROGRAM WITH THESIS

MASTER'S THESIS ADVISOR

Asst. Prof. Duygu Demiray Akkaya

MASTER'S THESIS CO-ADVISOR

Prof. Dr. Shahzad Ahmed MEMON

MASTER'S THESIS JURY MEMBERS:

Asst. Prof. Dr. Burçin Külahçioğlu

Asst. Prof. Dr. Zeynep Behrin Güven Aydın

CONTENTS

ÖZET	5
ABSTRACT	6
ABBREVIATIONS	7
LIST OF FIGURES	8
1. INTRODUCTION	10
1.1 RESEARCH GAP AND OBJECTIVES	11
2. LITERATURE REVIEW	12
2.1 EVOLUTION OF VIDEO GAME PROGRAMMING	12
2.2 PLAYER ENGAGEMENT AND EXPERIENCE.....	13
Psychological and Emotional Dimensions:.....	13
Motivation and Rewards:.....	14
Social Interaction:	15
Emotional Resonance:.....	15
Customization and Personalization:.....	16
2.3 TECHNOLOGICAL INNOVATIONS IN VIDEO GAMES	16
Graphics and Rendering:.....	16
Artificial Intelligence:.....	17
Physics Simulation:.....	17
Virtual Reality and Augmented Reality:	17
Cloud Gaming and Streaming:.....	17
3. RESEARCH METHODOLOGY.....	18
3.1 GAME DEVELOPMENT:.....	19
3.1.1 Game design and implementation:.....	19
3.1.2 Development tools and languages:	21
3.1.3 Testing and debugging:	21
3.2 PARTICIPANT RECRUITMENT:	21

3.2.1 Sampling methods:.....	21
3.2.2 Demographic criteria:	21
3.3 SURVEY DESIGN:	22
3.3.1 Survey structure:	22
3.4 SURVEY DISTRUBTION:	22
3.5 DATA COLLECTION:.....	22
3.6 ETHICAL CONSIDERATIONS:	23
3.7 LIMITATIONS OF REASEARCH:	23
3.8 EXPECTED OUTCOMES:	23
4. ANALYSIS OF DATA AND RESULTS.....	24
4.1 KEY FINDINGS:	35
5. DISCUSSION	37
5.1 CONTRIBUTIONS ON FUTURE STUDIES:	38
6. CONCLUSION.....	40
7. APPENDIX.....	41
7.1 SURVEY STRUCTURE.....	64
8. REFERENCES	67
9. BIOGRAPHY	70

ÖZET

İLERİ VIDEO OYUN PROGRAMLAMANIN OYUNCU KATILIMI VE DENEYİMİ ÜZERİNDEKİ ETKİSİNİN ÖLÇÜLMESİ

Bu tez, gelişmiş video oyunu programlamanın geniş ve kapsamlı alanını ve oyunculara keyif ve etkileşim sağlamak için ortaya çıkardığı temel sonucu araştırmaktadır. Bir dizi anketle test edilmeden önce altı oyun tasarlandı ve inşa edildi; çok fazla genelleme yapmak istemeden, bulguların oyuncunun memnuniyetini ve oyuna dalma derecesini etkileyen en önemli faktörlerden bazılarını işaret etmeye yardımcı olması bekleniyor. Sonuçlar, görsel çekiciliğin, çeşitli oyun oynamanın ve deneyimi geliştirmek için özel yeteneğin önemli faktörler olduğunu göstermektedir. Tez ayrıca sanal gerçeklik gibi yeni ortaya çıkan teknolojilerin oyuncu katılımında devrimler yaratmak için nasıl uygulanabileceğini de yansıtmaktadır. Tez ayrıca tasarımda etik düşüncenin rolüne vurgu yapmaktadır. Bu tür araştırma içgörülerini, oyunlarda ilgi çekici ve sürükleyici deneyimler oluşturmak isteyen geliştiricilere rehberlik edecektir. Sonuç, sürekli yenilik ve uyarlanabilirliğe duyulan ihtiyacın uluslararası oyun topluluğunun dinamik tercihlerine yanıt olarak ortaya çıktığını göstermektedir.

Anahtar Kelimeler: Video Oyunları, Etkileşim, Deneyim, Oyun Tasarımı, Anket.

Date: Haziran 2024

ABSTRACT

MEASURING THE IMPACT OF ADVANCED VIDEO GAME PROGRAMMING ON PLAYER ENGAGEMENT AND EXPERIENCE

This thesis explores the wide and vast area of advanced video-game programming and the fundamental outcome it erupts to provide enjoyment and engagement to players. Six games were designed and constructed before being tested by a series of questionnaires; it is expected that without wanting to generalize too much, the findings will help to point towards some of the most imperative factors to influence satisfaction and the immersion degree of the player. The results show that visual appeal, varied gameplay, and custom ability to enhance the experience are important factors. The thesis also reflects on how such emerging technologies as virtual reality can be applied to make player engagement revolutions. The thesis has also emphasis on the role of ethical consideration in design. Such research insights would guide developers who wish to craft compelling and immersive experiences in gaming. The conclusion showcases that the need for continuous innovation and adaptability emerges in response to the dynamic preferences of the international gaming community.

Keywords: Video Games, Engagement, Experience, Game Design, Survey.

Date: June 2024

ABBREVIATIONS

PC	:	Personal Computer
AI	:	Artificial Intelligence
AAA	:	High-budget, High-profile Games
RPG	:	Role-Playing Game
FPS	:	First Person Shooter
VR	:	Virtual Reality
AR	:	Augmented Reality
PvE	:	Player versus Environment
PvP	:	Player versus Player

LIST OF FIGURES

Figure 3.1: Research methodology steps process	18
Figure 3.2: Interface of Snake game.....	20
Figure 3.3: Interface of Snake game 2.0.....	20
Figure 3.4: Interface of Pong game.....	20
Figure 3.5: Interface of Pong Shooter game.....	20
Figure 3.6: Interface of Space Defender game	20
Figure 3.7: Interface of Space Shooter game.....	20
Figure 4.1: The percentage of participation regarding gender.....	24
Figure 4.2: The percentage of participation regarding age range	25
Figure 4.3: The percentage of participation regarding overall game experience.....	25
Figure 4.4: The percentage of overall thought on Snake game.....	26
Figure 4.5: The percentage of improvement suggestions for Snake game	26
Figure 4.6: The percentage of overall thought on Pong game.....	27
Figure 4.7: The percentage of improvement suggestions for Pong game.....	27
Figure 4.8: The percentage of overall thought on Space Defender game.....	28
Figure 4.9: The percentage of improvement suggestions for Space Defender game.....	28
Figure 4.10: The percentage of overall thought on Snake game 2.0.....	29
Figure 4.11: The percentage of improvement noticed in Snake game 2.0.....	29
Figure 4.12: The percentage of overall thought on Pong Shooter game.....	30
Figure 4.13: The percentage of improvement noticed in Pong Shooter game.....	30
Figure 4.14: The percentage of overall thought on Space Shooter game	31
Figure 4.15: The percentage of improvement noticed in Space Shooter game	31
Figure 4.16: The percentage of participants answers regarding overall game enjoyment factors	32
Figure 4.17: The percentage of participants answers regarding preferred game genres.....	33
Figure 4.18: The percentage of participants answers on the influence for playing/buying a game	33
Figure 4.19: The percentage of participants answers regarding the importance of customization in video games.....	34
Figure 4.20: The percentage of participants answers regarding preferred innovations in future games	34
Figure 7.1: Code of Snake game in Python	41
Figure 7.2: Code of Snake game in Python	41
Figure 7.3: Code of Snake game in Python	42
Figure 7.4: Code of Snake game in Python	42
Figure 7.5: Code of Snake game in Python	43
Figure 7.6: Code of Snake game in Python	43
Figure 7.7: Code of Snake game 2.0 in Python	44

Figure 7.8: Code of Snake game 2.0 in Python	44
Figure 7.9: Code of Snake game 2.0 in Python	45
Figure 7.10: Code of Snake game 2.0 in Python	45
Figure 7.11: Code of Snake game 2.0 in Python.....	46
Figure 7.12: Code of Snake game 2.0 in Python	46
Figure 7.13: Code of Snake game 2.0 in Python	47
Figure 7.14: Code of Pong game in Python.....	47
Figure 7.15: Code of Pong game in Python.....	48
Figure 7.16: Code of Pong game in Python.....	48
Figure 7.17: Code of Pong game in Python.....	49
Figure 7.18: Code of Pong game in Python.....	49
Figure 7.19: Code of Pong game in Python.....	50
Figure 7.20: Code of Pong game in Python.....	50
Figure 7.21: Code of Pong game in Python.....	51
Figure 7.22: Code of Pong game in Python.....	51
Figure 7.23: Code of Pong game in Python.....	52
Figure 7.24: Code of Pong game in Python.....	52
Figure 7.25: Code of Pong game in Python.....	53
Figure 7.26: Code of Pong game in Python.....	53
Figure 7.27: Code of Pong Shooter game in Python.....	54
Figure 7.28: Code of Pong Shooter game in Python.....	54
Figure 7.29: Code of Pong Shooter game in Python.....	55
Figure 7.30: Code of Pong Shooter game in Python.....	55
Figure 7.31: Code of Pong Shooter game in Python.....	56
Figure 7.32: Code of Space Defender game in Python.....	56
Figure 7.33: Code of Space Defender game in Python.....	57
Figure 7.34: Code of Space Defender game in Python.....	57
Figure 7.35: Code of Space Defender game in Python.....	58
Figure 7.36: Code of Space Defender game in Python.....	58
Figure 7.37: Code of Space Shooter game in Python	59
Figure 7.38: Code of Space Shooter game in Python	59
Figure 7.39: Code of Space Shooter game in Python	60
Figure 7.40: Code of Space Shooter game in Python	60
Figure 7.41: Code of Space Shooter game in Python	61
Figure 7.42: Code of Space Shooter game in Python	61
Figure 7.43: Code of Space Shooter game in Python	62
Figure 7.44: Code of Space Shooter game in Python	62
Figure 7.45: Code of Space Shooter game in Python	63
Figure 7.46: Code of Space Shooter game in Python	63

1. INTRODUCTION

Video games have evolved from being just fun and entertaining over the last decades to really complex ecosystems that pull people in from all over the world. It's a revolution in the way interactive entertainment is being conducted, and with the seamless combination of superior technology and creative narration, it is driving the video game industry to levels unknown before. At no point in the history of video games has programming ever had the potential to influence the way that the gamer engages with video games more than it is currently doing; it is spiraling the search for experiences that carry gamers to greater depths.

This thesis gets an exciting vantage point within the world of advanced video game programming and revolutionizes how the gamer can relate to games. This research is going to delve deeply into modern game development methodologies to provide an insight into the tangled web of technology and player interaction, thereby opening up the dynamic relationship between programmers and players in the digital landscape of gaming.

This, at its core, is what the following research is going to try and uncover: how engagement with and within a game is constructed for a player. The paper tries to unearth how advanced programming techniques lead the way for redesigning game mechanics and presentation into helpful knowledge on how developers will be able to create many more engaging and immersive experiences that can step into the world of the players and change them to the very core.

Having learned about the evolution of video game programming, from its meager beginnings to the present high sophistication, I could trace the relevance of such advanced programming techniques in the broad context of the gaming industry. Consequently, moving further into an exploration of such critical technological novelties as advanced graphics rendering, artificial intelligence, and virtual reality, this study will only attempt to shed some light on how modern approaches to game development might be regarded as transformational for what lays ahead in the context of interactive entertainment.

1.1 RESEARCH GAP AND OBJECTIVES

While the existing literature offers valuable insights into the historical and technological aspects of video game programming, a clear gap in the research is one that fully explores the effect of advanced programming on player engagement. Therefore, this thesis aims to reach the following objectives:

1.1.1 Assess the progress of video game programming: Trace the development over time; key technological advances and the effects on player experience.

1.1.2 Analyze player engagement and experience: A critical review of the scholarly work done to highlight the psychological and experiential dimensions of player engagement.

1.1.3 Examine technological innovations: Learn how the emergence and rise of such advanced technologies as VR, AI, and photorealistic graphics affect player engagement and the overall gaming experience.

1.1.4 Explore the role of game development companies: Look into what both significant and indie game development companies do to boost player engagement with their marketing approaches, community-building initiatives, and monetization models.

As such, the thesis sets off with the aim of contributing to the academic thread tied around video game programming and the intellectual ripple effects such a field has on player engagement and experience. This study will focus exclusively on advanced programming interaction with video games, exploring its historical trajectory, recent technological developments, and company strategies.

2. LITERATURE REVIEW

This chapter discusses the background history of video game programming, explores its significance and impact on modern development through published literature. The literature has been collected through online libraries, published reports and websites. The importance of video games has been discussed and investigated by various researchers, teachers and even psychologists. The word "game" is used to connect both the concept of the game as an object and the game experience as an action (Deliyannis et al., 2023) (Vuorre et al., 2022). With that, we understand the sole purpose of video games, and irrespective of the type of games as objects or processes, the gameplay experience offers people multiple opportunities to learn and interact with the natural and artificial environment.

This chapter also investigates the sole factor that affects players engagement and experience. Furthermore, it goes in depth of the technological innovations in video game programming in detail.

2.1 EVOLUTION OF VIDEO GAME PROGRAMMING

The roots of video game programming trace back to the rudimentary days of electronic entertainment, where simple pixelated graphics and rather basic mechanics were the order of the day. As technology burgeoned, the industry went through a metamorphosis that allowed for sophistication in programming techniques, thereby birthing iconic titles and even genres. The literature review covers the historical development of video game creation (The Video Game DebaTe, n.d.), from its backward history through pivotal milestones; technological breakthroughs and the evolution of gaming platforms will be discussed.

The development of video game programming has been a journey that, in many respects, is very parallel to the rapid technological advances and constantly changing lands that we call the video gaming industry.

Video game programming goes back to the middle of the last century when computer scientists and engineers started to experiment with electronic interactive entertainment. These earlier games, such as 1962's "Spacewar!" were primitive compared to the standard set by games today but nonetheless paved the way. (Bowman et al., 2023; The Video Game DebaTe, n.d.)

The '70s and '80s were the era of arcade gaming, the golden days of games like "Pac-Man", "Space Invaders" and "Donkey Kong". Developers squeezed the most out of existing hardware. They used clever programming techniques on top to deliver fast-paced action and addictive gameplay within the constraints of the machines that housed them. (Wulf et al., 2018)

The video game on a home console was first introduced to the public during the late 1970s and early 1980s by "Atari", and soon "Nintendo" and "Sega", when gaming found a way into the living room. Professionals polished their skills at each new hardware generation through software development, making it possible for hardware to become ever more powerful. This would constitute a complex game with a visually appealing output.

Going on in parallel to the console market, the personal computer was becoming a platform in itself for gaming. The versatility of the PC allowed for the great flexibility of game development, thus encouraging a lively indie scene alongside the major AAA releases from established studios. Programming languages like BASIC, C, and later on specialized engines like Unity and Unreal Engine opened up new opportunities for developers to experiment by making games from various genres.

The gaming landscape has been changing with digital distribution platforms, mobile gaming, and ever-changing new technologies such as virtual reality and cloud gaming. This generation in gaming is characterized by extensive access to a huge library of games, new and innovative business models, democratization in the tools for game development, and therefore, bringing the ease of making their dreams come true in what they produce to aspiring developers.

2.2 PLAYER ENGAGEMENT AND EXPERIENCE

Advanced video game programming is an exploration based on grasping player experience and engagement with games. The extent literature review would be focused on different scholarly works related to psychological, cognitive, and emotional player engagement. In addition, it will analyze existing research on the player experience, the satisfaction and immersion of players in gaming, and factors that relate to a satisfying gaming experience.

Player engagement and experience are really at the core of the game industry, shaping the ways we interact with that genre's fans and video games. With all of that said, to have the ability to create engaging gameplay and memorable experiences, developers need to fully understand the basic principles behind it and then just make great games.

Psychological and Emotional Dimensions: Player engagement is a broad construct that comprises many psychological and emotional factors that influence the perception and interaction with video games. A large slice of engaging with the game immersion can be defined as being fundamentally connected to the game world. At the same time, little distinction remains between reality and fantasy. The sweet spot between plot and audiovisual performance combined with gameplay mechanics necessitates sufficient intensity to plunge the player into the experience. An Example is the game (The Witcher 3: Wild Hunt). It's an open world action RPG located in a dark fantasy universe. Players assume the role of (Geralt of Rivia), a traveling monster slayer for hire

on the goal of finding his lost adopted daughter. Every corner is filled with enriched narrative, complex world-building, and unique fight mechanics. Gamers get caught in the stream of detailed surroundings and complex storylines regularly, with an absolute imperative to choose through which branches the story should unfold.

Another example is the game (Dark Souls). A very challenging action RPG game with a high level of difficulty and good atmospheric world design, the game became a genre by itself for being the iconic most hard type of game of all time. The player travels through dim and dangerous locations, fighting strong enemies of all sizes and each has their own fighting styles, and the player must adapt to them, while the most exciting thing in the game is found through exploration and discovery. The Dark Souls series is complicated, and its lore goes deep. and shallow emotions are tied up within the difficulty, going from frustration to triumph. A lot of unique feelings of accomplishment are created within a player after beating a demanding boss, but at the same time, emotional investment in the dark, mysterious world remains high.

Motivation and Rewards: Attention can also be channeled by motivating players. There are numerous types of reward systems, be it in terms of achievements, unlockable content, or progression mechanics, that encourage desirable player behavior and give the player a sense of accomplishment. Anticipation of rewards and satisfaction from goal achievement are used as very strong motivators that maintain player interest in the game experience(Rodrigues & Brancher, n.d.). As an example of a reward system (Overwatch). An FPS game where players choose heroes from a wide hero roster, each hero has their own individual abilities and roles that defines their impact on the battlefield, in the game, players play in a team and face off another team in objective based modes, like capturing control points and escorting a payload, a team attacks and the other defend, and players must foster teamwork and strategic playstyle to achieve victory. (Overwatch) uses several reward systems, including loot boxes, achievements, and seasonal in-game events. Such unlockable skins, sprays, and other cosmetics operate, at the very least, as a reward so that the players are motivated or compelled to play more games by winning matches and completing achievements.

Another example regarding progression mechanics is the game (Fortnite). A popular battle royale game in which players skydive onto an island and battle in a free-for-all playstyle, The players collect resources and weapons, build structures with them, and utilize them as a cover from enemies' incoming attacks. The players must make use of every technique, strategy and most importantly teamwork to become the last person or team standing. The Battle Pass in Fortnite acts as a structured form of progression between players and unlockable tiers of content, pushing them to get challenges done to open more rewards and thus develop some form of anticipation by unlocking new items and cosmetics.

Social Interaction: The social dimension boosts the already complex nature of gaming to engage players. Multiplayer games, online community features, and other social features come alive on these platforms in a way that gives the player a sense of belonging and camaraderie. Whether competing against one another within a multiplayer match or together with strangers within an online raid, the social layer raises the base of experience and increases engagement over the long term for every game (Jin, 2024; Ma et al., 2023). The game (World of Warcraft) is a good example for multiplayer interaction. It's a massive multiplayer online RPG set in the fantasy world of Azeroth. All players can create characters from various races and classes, go on quests, and explore huge landscapes while performing PvE and PvP actions with other players. One of the greatest appeals of World of Warcraft lies in the importance placed on social interaction. Players can form guilds, participate in raids, and engage in player-versus-player (PvP) combat serving as a virtual meeting ground for the fostering of community and teamwork.

Speaking of Online Community, the game (Among Us) can be an example. It is a multiplayer party game. The players are crew members on a spaceship and must work together to complete several tasks while trying to identify the impostors in which they try to thwart the completion of the functions and vote all the crew members out. The game became so popular due to the significant social orientation it had. For some reason, players were expected to join hands to identify imposters and vote them out, which added the element of communication and strategy through the game's essence.

Emotional Resonance: Of course, outside of the gameplay mechanics, fundamentally crucial to storytelling are emotive elements. Memorable characters, great stories, powerful themes. All these evoke a range of feelings from joy and excitement to sadness and empathy and a range of feelings. Games that draw out emotions create long-lasting memories (Koçak et al., 2023), making significant connections other than in virtual reality. The best representation on storytelling and characters is the infamous game (Red Dead Redemption 2) An open world action-adventure game set in the American Wild West. The player will play as (Arthur Morgan), an outlaw and member of the (Van der Linde) gang, with his gang in between he faces enemies and authorities amidst misadventures packed with crime, betrayal, and redemption. One of the most captivating stories and complex sets of characters in this game. It delves so incredibly deep into the notions of loyalty, morality, and redemption, which helps in making an emotional connection with players. Memorable moments and character interactions leave a lasting impact.

Customization and Personalization: These features should make the game feel like a player's own by letting them, depending on their whims, take care of character building, gameplay settings, or difficulty levels of the game. This way, affording the player the agency will then make them create their own stories within the game world and further focus on engaging more with its contents. For Character Development we can take (Cyberpunk 2077) as an example. An open world action-adventure game set in a dystopian future where players take on the role of a mercenary named (V), exploring the dangerous streets of Night City in order to find salvation and fight through the corruption of both street gangs and corporate militias. Within Cyberpunk 2077, there are several changes that one can build into the character. This offers players a pool of opportunities to bring out a unique appearance, varying abilities, and style of play for each character. This player engagement became intense because of personalization, allowing everyone to feel unique game experiences.

2.3 TECHNOLOGICAL INNOVATIONS IN VIDEO GAMES

Advanced technology is one of the hallmarks that unlocks the transformational power in video game programming. This section will discuss technological innovations ranging from 3D graphics, which have paved the way for virtual reality and artificial intelligence, to reshaping the entire landscape of video game development. It will emphasize specifically the impact of advances on player involvement and how it contributes to this immersive experience. (Passos et al., n.d.)

The history of programming video games is a cardinal example of the marching progress of technology, where each innovation always seemed to reach new horizons in possible interactive entertainment.

Graphics and Rendering: Graphics and Rendering: Graphics technology has advanced in creating truly immersive, graphically powerful game environments. From the simple 2D sprites of olden days up to the complex 3D rendering techniques that have been developed up to this very moment, developers have truly come a long way in creating environments and characters that almost seem lifelike. Techniques such as ray tracing, photorealistic rendering, and dynamic lighting have created a sea change in the outlook and feel of games, somewhat blurring the difference between fantasy and reality.(Caroux, 2023)

Artificial Intelligence: AIs define the behavior of non-player characters and construct dynamic, responsive game worlds. Initially, AI algorithms focused on essential decision-making and pathfinding. Still, the latter advances in machine learning and neural networks made it possible to create more sophisticated AI systems which adapt and learn from player behavior in real-time. All this way, developing the gameplay even more comprehensive, compelling, and challenging might be realized—one where NPCs possessing human-level intelligence react adequately to the player's action.(Filipović, 2023)

Physics Simulation: Game elements are the constituents of an individual game. The game element requires essential interaction between itself, whereas the internal physics technology of the game is needed to support them. From the movement of a projectile to environmental physics, a physics engine like Havok or PhysX allows developers to craft immersive and dynamic gameplay. Credibility in the game world, emerging gameplay opportunities, immersion: in fact, all these things grow from realistic evolution in these environments.

Virtual Reality and Augmented Reality: VR and AR are the subsequent evolutions in rendering gaming experiences to be more immersive. Virtual reality promises an unmatched new world of virtual experience where gamers become deeply immersed in entirely created worlds and ecosystems far beyond just a representation or model. AR, however, superimposes real-world digital content onto the actual real world to merge this virtual and physical world into a new experience. These technologies would possibly redesign the way we currently play games and offer unparalleled levels of immersion and interactivity.

Cloud Gaming and Streaming: Cloud gaming and streaming services bring into play the enormous computational powers presented by cloud computing, delivering rich, quality gaming experiences to players over various devices. Such devices are vital in both processing and rendering operations that a player can undertake without any expensive hardware, thanks to cloud gaming platforms such as Google Stadia and GeForce Now. This will democratize gaming, making access broader and reaching a larger audience worldwide.

3. RESEARCH METHODOLOGY

This chapter discusses how the elaborated methodology was considered to collect and analyze player feedback on six Python games in this study. This research attempts to approach an understanding of players' experiences and preferences in a way that can provide insights valued by developers into whatever it is that players seek in a video game. The following figure shows the steps that were taken in the process of the research and furthermore will be explained in detail.

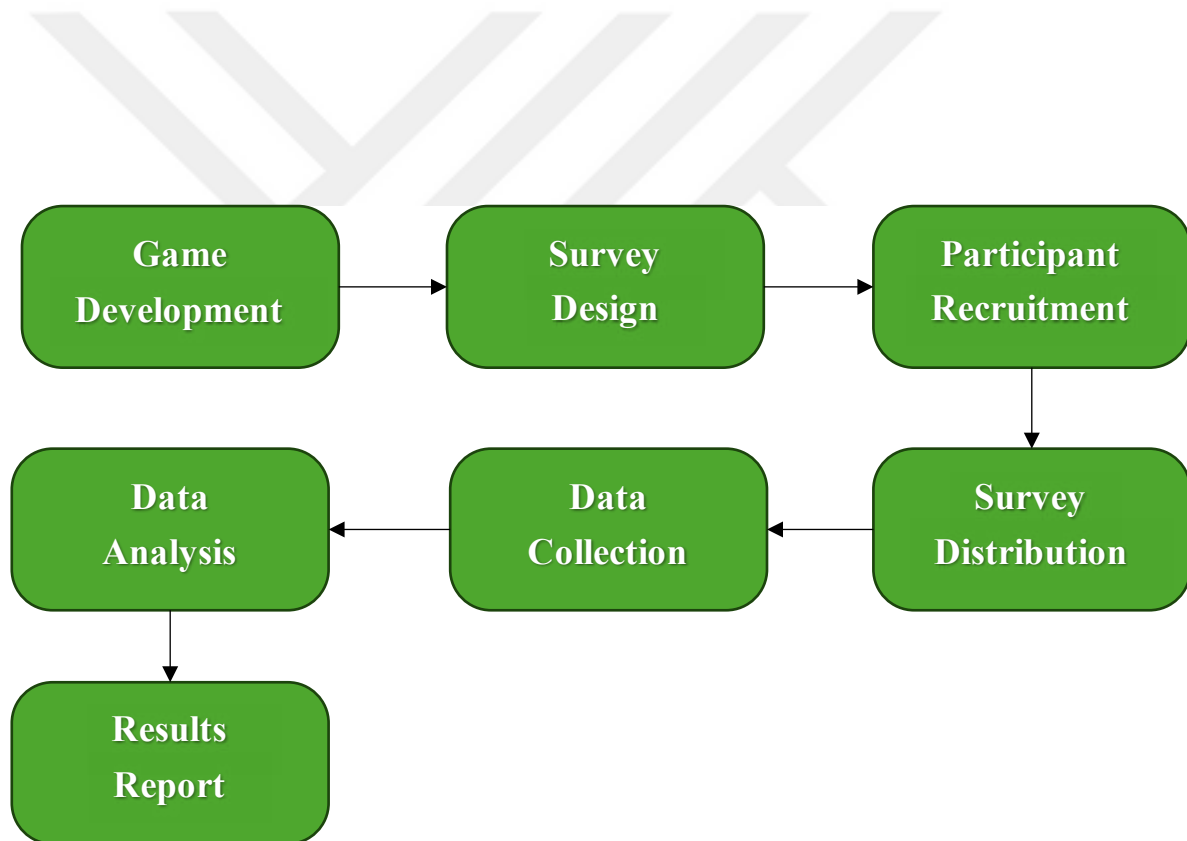


Figure 3.1: Research methodology steps process

3.1 GAME DEVELOPMENT:

3.1.1 Game design and implementation:

- **Snake Game:** This basic version features a simple snake that moves around the board, controlled by the player using the W, A, S, and D keys. The objective is to collect food while avoiding collisions with the screen borders and the snake's own body. As the snake eats, it grows larger. The game ends if the snake collides with itself or the walls.
- **Snake Game 2.0:** The advanced version of the Snake game introduces enhanced graphics regarding visuals in the background and the snake itself alongside the food the snake eats. It includes sound effects that queues when the snake consumes food, adding to the overall player experience.
- **Pong Game:** The classic two-player Pong game allows players to control paddles to hit a bouncing ball. A power-up effect is activated when the ball is struck in the center of the paddle or at the edge of it, propelling it at high speed toward the opponent.
- **Pong Shooter Game:** In this advanced version, two players control spaceships that shoot at each other in a galaxy-like background. Each ship has a unique design representing the player's side.
- **Space Defender Game:** This simple game features a spaceship at the center of the screen that rotates using the arrow buttons and shoots incoming enemy objects.
- **Space Shooter Game:** The advanced version of the previous game, with levels and improved graphics, provides a more immersive gaming experience.

All games are shown from Figure 3.2 to Figure 3.7

These games were designed to represent 3 difficulty levels in both gameplay mechanics and development techniques, which are easy, medium and hard. In detail, both the Snake game and its advanced version is considered an easy difficulty game for its ease to understand and to reach higher scores with somewhat average or casual gaming experience. Meanwhile, Both Pong and Pong Shooter games fall under the medium difficulty category, these games require reflexes and strategic thinking to maneuver the enemy attacks since they are both played with another player/friend, and it also requires some aiming skills to achieve victory. Lastly, Space Defender and Space Shooter games are categorized as hard difficulty games, it is due the games has more challenging and aggressive environment, the player will need to have sharp precision to maneuver, defend and attack precisely to go to the next level, the aim of these games is to reach the highest level or score on the leaderboard since both games are designed to be in endless mode while the more levels are being progressed the harder the game becomes.

As it's noticed, these games are designed to be played sequentially, allowing the players to try the simple version of a game, give their feedback, then try the modified and enhanced version of the game they played, and see their feedback if the new updated version of the game has met their expectations and are they willing to see it grow into a bigger and better project.



Figure 3.2: Interface of Snake game

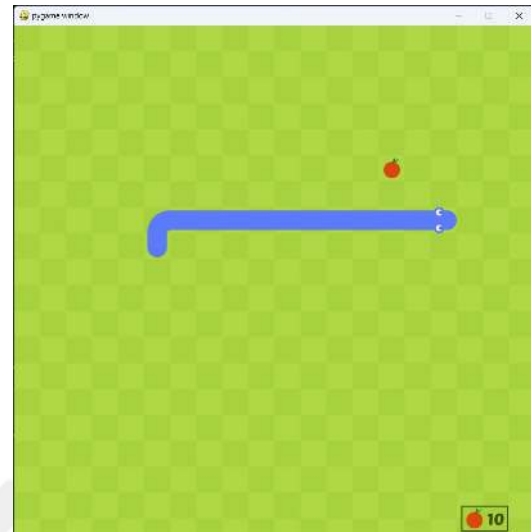


Figure 3.3: Interface of Snake game 2.0



Figure 3.4: Interface of Pong game



Figure 3.5: Interface of Pong Shooter game

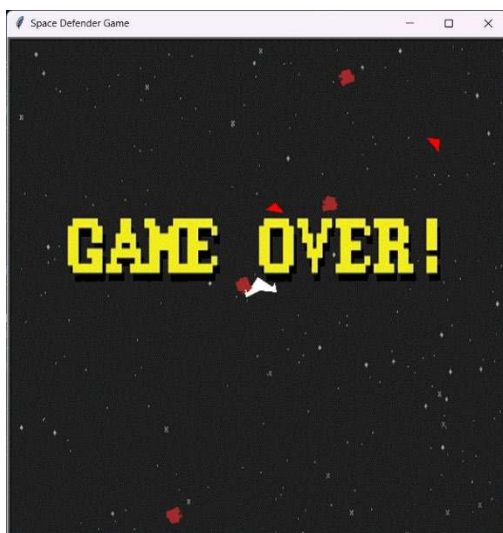


Figure 3.6: Interface of Space Defender game



Figure 3.7: Interface of Space Shooter game

3.1.2 Development tools and languages:

All games were developed using Python with the Pygame library, it helped in easing graphics handling and event processing in the development of the games.

Developments using Python are usually chosen due to its simplicity, accessibility, and suitability for developing small-scale, 2D games, which align with a given project's focus which is mostly for understanding fundamental game mechanics. Python's straightforward syntax and Pygame's robust library enable rapid prototyping, making it suitable for educational and experimental purposes.

In the other hand, modern game engines like Unity and Unreal Engine offer advanced capabilities, such as high-definition 3D graphics, physics, and integrated AI and multiplayer support. These engines are designed for complex, AAA game development but come with a steeper learning curve and require more resources.

The choice of development with Python reflects the study's emphasis on foundational gameplay elements and player engagement, rather than highly exaggerated graphics or advanced features. By using Pygame, the focus remains on comparing basic and advanced versions of the games, offering clear insights into the impact of improvements and implementations such as visuals, sound, and gameplay mechanics on player experience.

3.1.3 Testing and debugging:

Testing was done to ensure that the games worked correctly and that they gave users precisely the same experience each time they used them. This saw the debugging of gameplay issues and performance optimization for smoother playing.

3.2 PARTICIPANT RECRUITMENT:

3.2.1 Sampling methods:

Participants were recruited via social media platforms and gaming forums. An invitational thread was used for those who wish to participate and help this study.

3.2.2 Demographic criteria:

The study targeted individuals aged between 18-35, in which they have variety of levels in video game experience, from casual players to hardcore. This was meant to capture a wide spectrum of perspectives.

3.3 SURVEY DESIGN:

3.3.1 Survey structure:

The survey contains 5 sections to gather comprehensive feedback, and they are as the following:

1. **Demographics:** This section contains questions that collect demographic data of player including:
 - Age
 - Gender
 - Gaming Experience
2. **First User Feedback:** Participants provide feedback based on their experience with the simple draft of the games used in this study, which are Snake Game, Pong game and Space Defender Game, and they will be asked the following questions for each game:
 - What are your overall thoughts on the game?
 - How could the games be improved?
 - Would you recommend the game in this state to a friend?
3. **Second User Feedback:** Following their previous experience, they are asked to try the enhanced and modified draft of the games and see if further improvements to a game would meet their expectations, they will be asked the following questions for each game:
 - What are your overall thoughts on the game?
 - How could the games be improved?
 - Would you recommend the game in this state to a friend?
4. **General Thoughts on Video Games:** The following section of thesis will explore the opinion of participants on game development through a series of general questions:
 - What do you enjoy most about playing video games?
 - Which genres of video games do you prefer?
 - What factors influence your decision to purchase or play a video game?
 - How important is player choice and customization in video games to you?
 - What features or innovations would you like to see more of in future video games?
5. **Additional Comments:** This section is for participants to elaborate on any idea not explicitly addressed in the survey.

3.4 SURVEY DISTRIBUTION:

To encourage a high response rate, the survey was distributed via easy-to-access link created on Google Forms. Also, reminders were sent to encourage completion of surveys.

3.5 DATA COLLECTION:

All participants were requested to play each game for at least 5 min to secure a good level of exposure and interaction time with the game's mechanics, after that, an online survey link was attached, participants are asked to give their feedback on the game they tried and then move on to

try the advanced version of it, similarly, they will be asked about their feedback to the advanced version as well.

In this structure, the games and the survey, it is estimated that the overall experience with this study won't take longer than 45 minutes to be completed and submitted.

3.6 ETHICAL CONSIDERATIONS:

The survey used in this study was reviewed and approved by the Ethics Committee of Beykoz University, ensuring compliance with ethical guidelines.

Informed consent was obtained. The purpose and procedure of the study were explained to the participants, who were informed about their right to withdraw at any point in time without access to negative consequences affecting them. Consent was obtained digitally to confirm that participants had read and understood the scale of their involvement before it began.

Data confidentiality measures were considered to ensure privacy for the participants. Only a minimal number of data was collected, which included age, gender, and experience in gaming, while all personal identifiers in the data had been anonymized. With regard to data storage, the data were well kept according to protocol, with all information stored in password-protected systems where only the researchers had access. Data archiving shall be done upon completion of the study according to the requirements of the institution and the law.

By elaborating on these steps, the study underpins its commitment to ethical standards, protection of participants, and integrity in the research process. This careful procedure guarantees the credibility of the findings, and the trust of all participants concerned.

3.7 LIMITATIONS OF REASEARCH:

While this study contributes to the knowledge about player engagement and experience, some limitations need to be addressed. Such as the difficulty to get the player's attention to try and participate in this study, it took a lot of time to gather the number of participants that would seem fit for this study to ensure a valid idea that can be represented and understood.

3.8 EXPECTED OUTCOMES:

This study gives insights into basic implementable elements that increase or decrease player enjoyment and engagement. There will also be specific game mechanics and simple areas of design that could be improved, with ideas to help casual developers. Game development will contribute to much knowledge regarding the design of games. In contrast, understanding the preferences and the experiences of the gamers will support making the gaming experience more appealing and satisfying.

4. ANALYSIS OF DATA AND RESULTS

This section will view the participation results in the survey in which the study was implemented, the survey collected 50 responses as the maximum number of participants required for the study, and the responses were diverse yet simple to understand what players like to see in a future game project or studies.

Demographics:

The gender composition for the respondents in the study is shown in Figure 4.1. Most, 61%, reported being males and are 49 participants. These were followed by 28% representing those who reported their gender to be female and add up to 22 participants while 11% representing 9 respondents preferred not to disclose their gender. Figure 4.2 shows the results of the age group distribution: 61% of the total respondents were between ages 18-25 years (9 participants), 23% between the ages of 25-30 years (18 participants), and the remaining 16% in the 30+ age group (13 participants). Also in that respect, Figure 4.3 provides an overview of the experience level for playing games among the participants: 56% described themselves as casual gamers, 32% as moderate gamers, and 11% as hardcore gamers. This variation in gender, age, and experience with games offers a good diversity in the responses to extend the insight into player preferences for different demographics.

Gender: Please select your gender identity.

80 responses

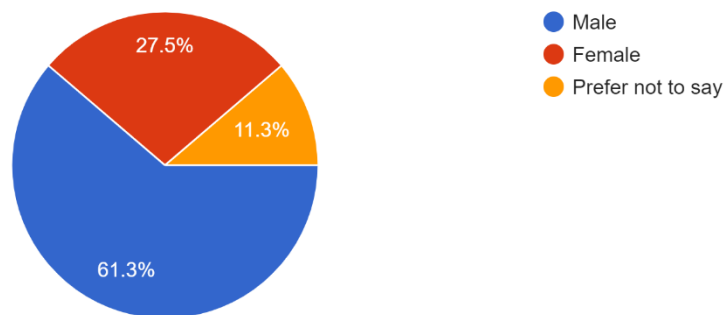


Figure 4.1: The percentage of participation regarding gender

Age: Please select your age range.

80 responses

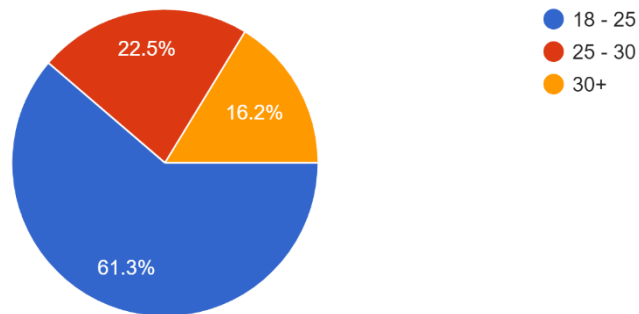


Figure 4.2: The percentage of participation regarding age range

Gaming Experience: How would you describe your level of gaming experience?

80 responses

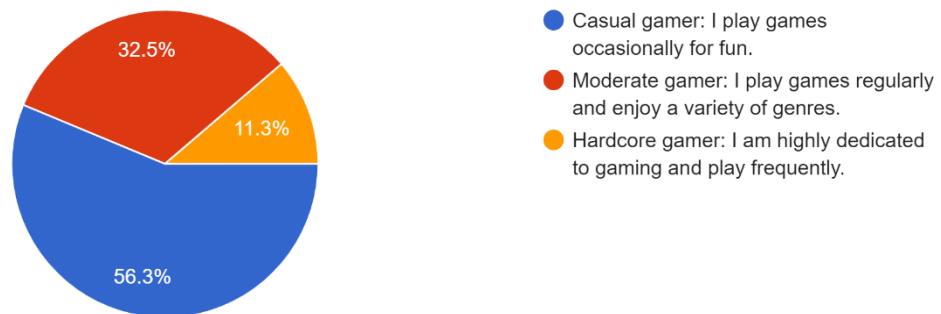


Figure 4.3: The percentage of participation regarding overall game experience

Snake Game Feedback:

Feedback on the simple version of the Snake Game is represented in Figure 4.4. A striking 83% of participants (66) rated the game as "Bad," while 17% (14) found it to be "Average," with no participants rating it as "Good." Figure 4.5 highlights suggested improvements, where 54% of participants (43) recommended better graphics, 25% (20) called for sound effects, and 21% (17) suggested smoother gameplay. And it has been made clear that none of the participants would recommend the game in its current state.

What are your overall thoughts on the Snake game?

80 responses

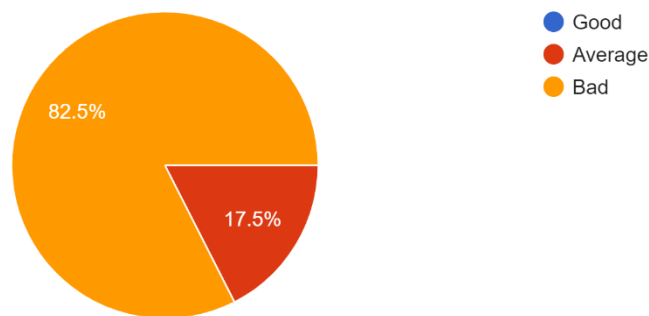


Figure 4.4: The percentage of overall thought on Snake game

How could the Snake Game be improved?

80 responses

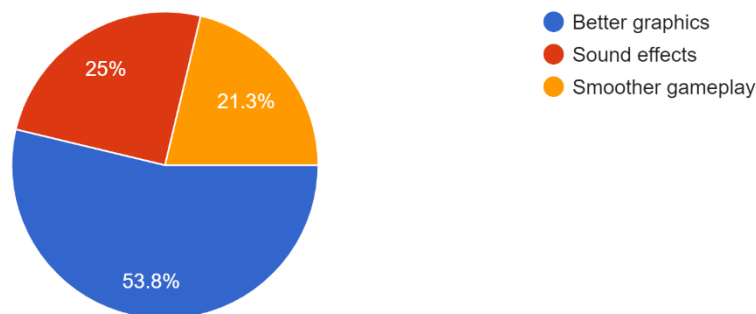


Figure 4.5: The percentage of improvement suggestions for Snake game

Pong Game Feedback:

As it's shown in Figure 4.6, 78% of participants (62) rated the simple version of the Pong Game as "Bad" while 22% (18) rated it as "Average" None rated it as "Good" Figure 4.7 reveals that 65% of participants (52) felt the game required better graphics, while 31% (25) desired smoother gameplay. Only 4% (3) suggested adding sound effects. Similarly, none of the participants would recommend this version of the Pong Game to others. The majority's feedback indicated a need for enhanced visual design with refined and more improved gameplay mechanics and elements.

What are your overall thoughts on the Pong game?

80 responses

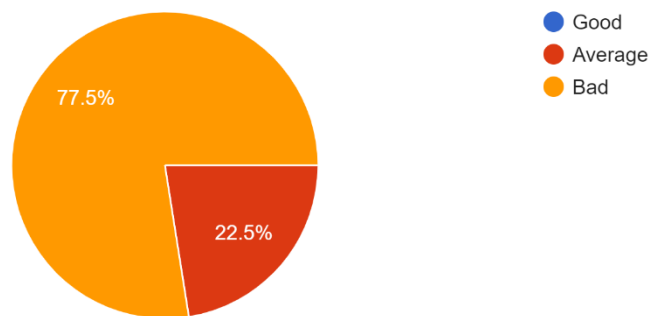


Figure 4.6: The percentage of overall thought on Pong game

How could the Pong Game be improved?

80 responses

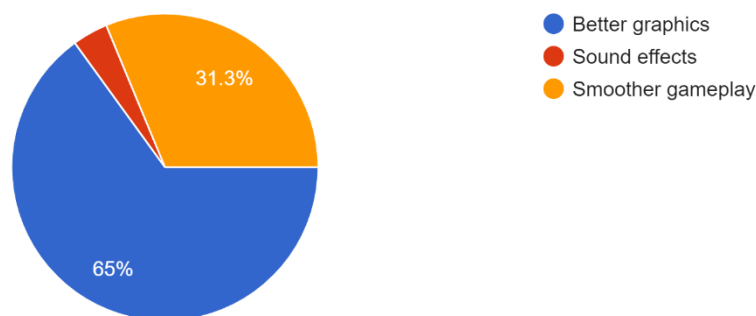


Figure 4.7: The percentage of improvement suggestions for Pong game

Space Defender Game Feedback:

Similarly, Figure 4.8 shows that 73% of participants (58) rated the simple version of Space Defender as "Bad," while 27% (22) rated it as "Average." Like the other simple games, no one rated it as "Good." Figure 4.9 shows that 79% of participants (63) wanted better graphics, 19% (15) suggested smoother gameplay, and only 2% (2) recommended sound effects. A small 3% of participants (2) indicated they would recommend the game, while 97% (78) said they would not.

What are your overall thoughts on the Space Defender Game?

80 responses

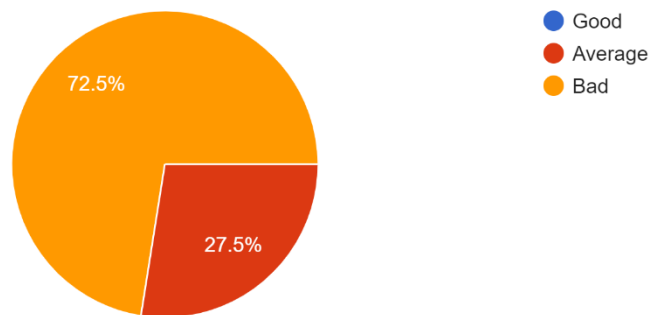


Figure 4.8: The percentage of overall thought on Space Defender game

How could the Space Defender Game be improved?

80 responses

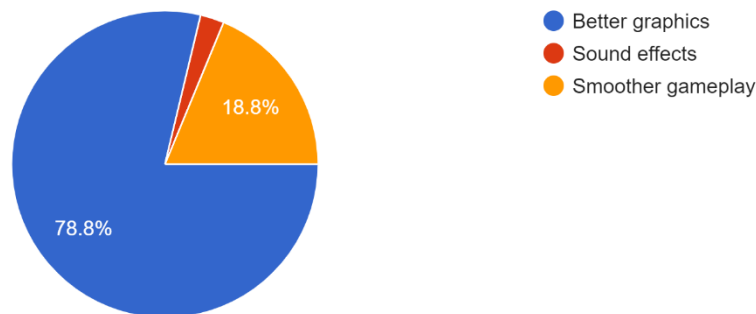


Figure 4.9: The percentage of improvement suggestions for Space Defender game

Snake Game 2.0 Feedback

The second, more advanced version of the Snake Game received more favorable responses. As seen in Figure 4.10, 33% of participants (26) rated the game as "Good," 56% (45) rated it as "Average," and 11% (9) rated it as "Bad." Figure 4.11 shows that 83% of participants (66) noticed improvements in the game's graphics, 10% (8) appreciated the addition of sound effects, and 8% (6) noted smoother gameplay. It was also revealed that 81% of participants (65) would recommend this version of the game, while 19% (15) would not.

What are your overall thoughts on the Snake game 2.0?

80 responses

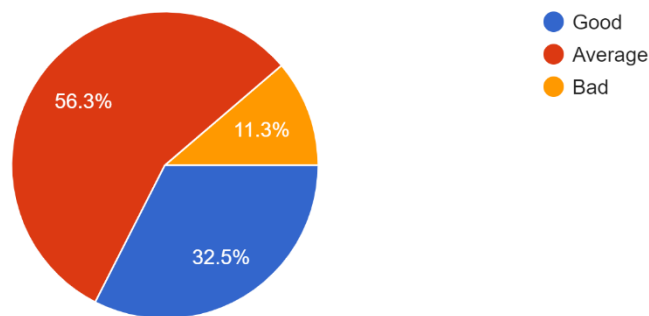


Figure 4.10: The percentage of overall thought on Snake game 2.0

What elements have you noticed that got improved in Snake game 2.0?

80 responses

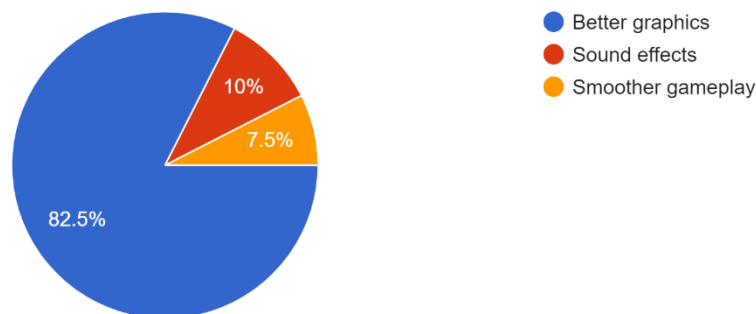


Figure 4.11: The percentage of improvement noticed in Snake game 2.0

Pong Shooter Game Feedback

For the Pong Shooter Game, Figure 4.12 shows that 29% of participants (23) rated it as "Good," 53% (42) rated it as "Average," and 19% (15) rated it as "Bad." In terms of improvements, Figure 4.13 shows that 85% of participants (68) noticed better graphics, while 15% (12) noted smoother gameplay. No participants observed improvements in sound effects. It was noted that 71% of participants (57) said they would recommend the game, while 29% (23) said they would not.

What are your overall thoughts on the Pong Shooter Game?

80 responses

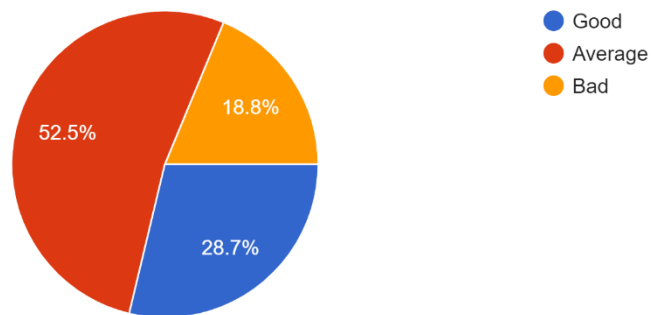


Figure 4.12: The percentage of overall thought on Pong Shooter game

What elements have you noticed that got improved in Pong Shooter Game?

80 responses

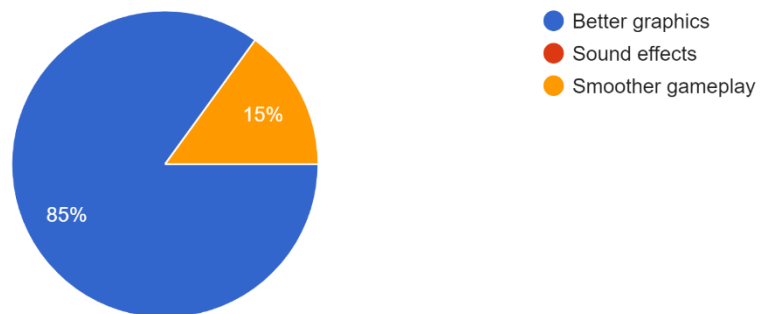


Figure 4.13: The percentage of improvement noticed in Pong Shooter game

Space Shooter Game Feedback

The advanced version of Space Shooter received the most positive feedback. As seen in Figure 4.14, 41% of participants (33) rated the game as "Good," 48% (38) rated it as "Average," and only 11% (9) rated it as "Bad." Figure 4.15 highlights that 76% of participants (61) noticed improved graphics, while 24% (19) noticed smoother gameplay. It was shown that 85% of participants (68) would recommend the game, while 15% (12) would not. The strong feedback for these games highlights the effectiveness of graphical improvements and gameplay enhancements.

What are your overall thoughts on the Space Shooter Game?

80 responses

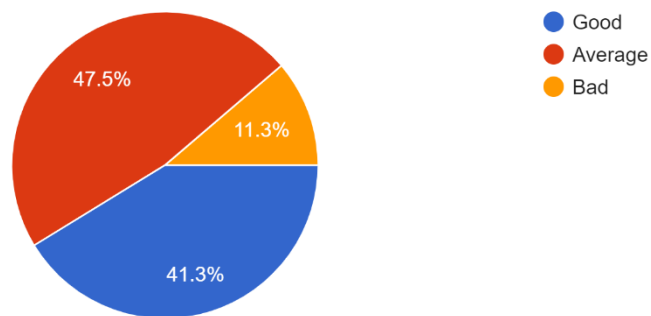


Figure 4.14: The percentage of overall thought on Space Shooter game

What elements have you noticed that got improved in Space Shooter Game?

80 responses

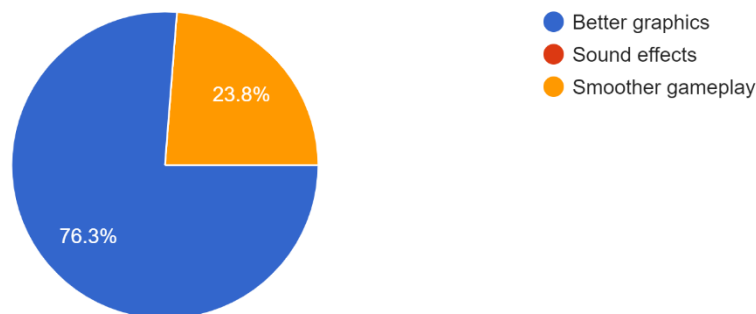


Figure 4.15: The percentage of improvement noticed in Space Shooter game

General Thoughts on Video Games:

The participants' general thoughts on video games are shown in Figures 4.16 to 4.20. In Figure 4.16, the top reasons for enjoying video games are Immersive storytelling (39% of participants, 31), Relaxation and escapism (28%, 22), and Challenging gameplay (16%, 13). In Figure 4.17, participants' preferred genres are led by Action (46%, 37), followed by Adventure (15%, 12) and Role-Playing Games (RPG) (11%, 9). Graphics and visual presentations were the top influencing factor for buying games, as shown in Figure 4.18, noted by 36% (29 participants), followed by Story and narrative depth at 28% (22 participants), and Gameplay mechanics and controls at 20% (16 participants).

In Figure 4.19, 79% of participants (63) said that player choice and customization are important to them, with only 9% (7 participants) feeling that they are not. Finally, Figure 4.20 highlights that Advanced graphics and realistic physics (29%, 23 participants) and Virtual reality and immersive technologies (30%, 24 participants) are the most desired innovations in future games, followed by Deep and branching narratives (19%, 19 participants) and Cross-platform compatibility (9%, 7 participants).

What do you enjoy most about playing video games?

80 responses

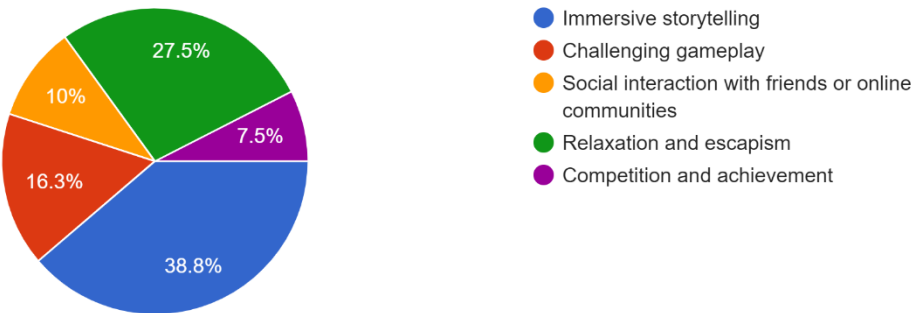


Figure 4.16: The percentage of participants answers regarding overall game enjoyment factors

Which genres of video games do you prefer?

80 responses

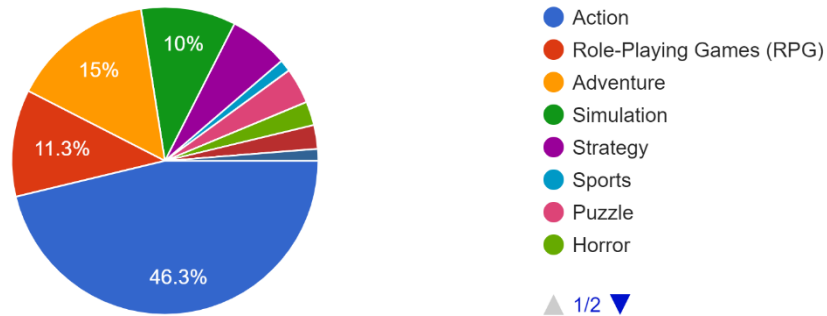


Figure 4.17: The percentage of participants answers regarding preferred game genres

What factors influence your decision to purchase or play a video game?

80 responses

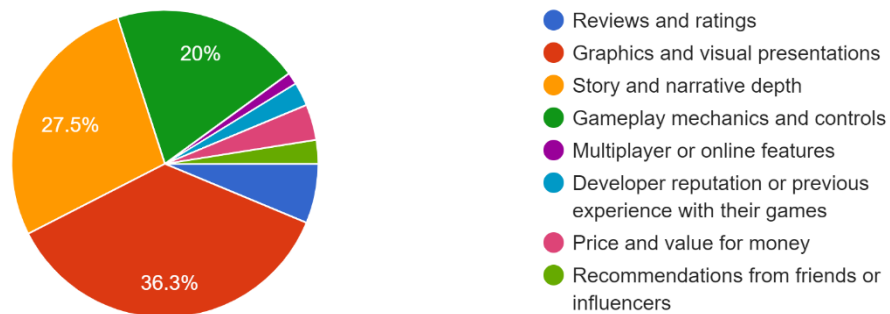


Figure 4.18: The percentage of participants answers on the influence for playing/buying a game

Player choice and customization in video games, is it important to you?

80 responses

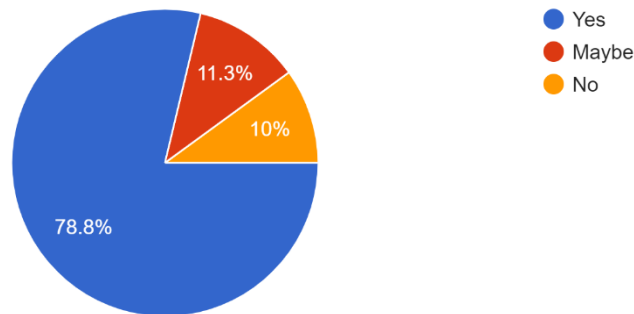


Figure 4.19: The percentage of participants answers regarding the importance of customization in video games

What features or innovations would you like to see more of in future video games?

80 responses

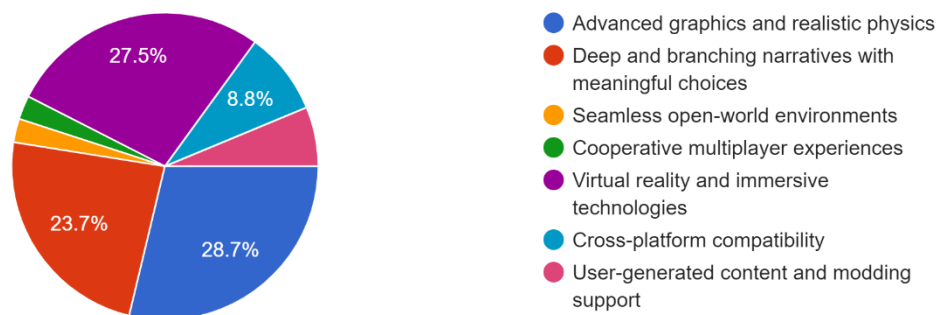


Figure 4.20: The percentage of participants answers regarding preferred innovations in future games

Additional Comments:

Participants provided various insights and suggestions. The common thread that wove through the recommendations for developing video games was better AI, better story integration, and better social features. What many of them touched upon was the balance of getting right the visual appeal against the mechanical depth in making more immersive and appealing games.

4.1 KEY FINDINGS:

The survey captured a relatively representative demographic of gamers with a balanced age group distribution across genders. Participants in the survey presented a wide and varied experience in gaming, from casual to hardcore, thus assuring that the point-of-view, given their preferences for or experiences with games, is as wide-angled as possible.

All this data collected gives an idea of the players' perception of different levels of game development and elements that are most important to them. Probably most striking was the strong contrast in responses between the original, more basic versions of the games and their more advanced versions. The first sets of games included the basic versions of Snake, Pong, and Space Defender. Most of those games received negative or lukewarm responses. The players said these games did not have any attractive features in them. Most common complaints were poor graphics, lack or poor quality of sound, and most importantly, not very polished gameplay. That means the basics-smooth controls or basic gameplay-aren't enough anymore to keep players interested in today's gaming environment. They want more immersive and visually magnetic games, even from the smallest-sized and simplest ones.

More importantly, it was the change in perception when players actually got down to playing the advanced versions of these games. The reaction of players during the second round of testing, which was of Snake Game 2.0, Pong Shooter, and Space Shooter, was overall much more positive. This was partially because of the major increase in graphical quality, but even with this being a very obvious main difference, this wasn't the only difference that did cause a game to be more enjoyable. Adding sound effects, better game elements, and overall smoother user experience made quite a big difference for this round of testing. This suggests that, while visuals are an important factor, it's only one piece of the overall puzzle that constitutes a fun game. The more finalized versions of those games proved a better design with more interesting features and smoother mechanics would result in a more positive reception of the game, even with the same core gameplay.

Evidently, from the different game versions created in this research, the importance of visual effects for the players' gaming experience can be understood. First, the player's attention is taken by the visual elements and then held by added immersive quality to gameplay. These would include simpler video games, where the visual effects of the original Snake Game and Pong Game are at a minimum, and due to that, the reception might be lukewarm. Participants in the experiment said that the graphics were too bland and lacked any feeling of depth or realism that a modern gamer would want. For these versions, there is not much to say about visual appeal, which, as Players' responses pointed out, makes the gameplay repetitive and uninteresting.

Contrarily, Snake Game 2.0 and Pong Shooter Game, both final versions, employed diversified graphic technologies: the creation of detailed backgrounds and characters' designs significantly raised the intensity of responses deeply. The participants felt that through better graphics, these advanced versions amplified the level of involvement in the game and made them continue playing longer. For instance, the cartoonish graphics and colorful field in Snake Game 2.0 brought life to

the game environment, whereas in Pong Shooter Game, there was a galaxy-themed background and uniquely designed spaceships that created a dynamic field of gameplay. These graphic updates have shown how graphics can make simple mechanics into more interesting experiences, simply with adding depth to the aesthetics.

These results show that high-quality, visually captivating images are crucial in keeping the gamer of today, for whom the visual experience is paramount. This further helps to identify wider trends within the gaming industry, which reveal how players are increasingly drawn to games with polished graphics and effects. Ideographically, the developers should not fall behind the times and try to raise the level of visuals in their games, because visuals scream to a modern gamer's senses—no less than emotion, feeling, and fun.

Given the general perceptions that players hold about video games, some interesting preferences stood out on what drew them into games and further kept them playing. Immersive storytelling, game challenges, and relaxation were some reasons many players said they played video games. This suggests that game design represents a point of balance, where individuals desire experiences that will challenge them yet allow them to relax and temporarily escape. The players, in particular, highlighted storytelling as one of the enjoyable factors, which underlined the developing expectation for in-depth narratives—even in genres traditionally known for action or mechanics over plot.

Another important finding of this research was the emphasis on all facets of customization and player choice that the players placed great value upon. While personalization may not be a crucial factor in all games, the importance it holds for most participants underlines a wider trend in gaming toward personalization. Players increasingly expect games to grant them some form of flexibility in how they engage with the game world through customization of their character, different paths in the telling of a story, or adjusting settings within the game. This desire for personalization speaks towards the trend of shifting preferences towards games that allow players the room for their agency within a hardcoded experience.

5. DISCUSSION

The findings of the survey research indicate different factors affecting player engagement and experience with a game. The next section advances these findings by discussing them further and suggesting conclusions that would imply future game development, while indicating areas that need improvement and innovation.

Most players pointed out the development of better graphics and visual effects in developing games. This goes to show that, in reality, developing a game to engage players comes back to what visually looks appealing to the player. Good visuals attract a player not only to the game itself but also add much to the in-game immersive factor. Therefore, players consider each contemporary game as of high-quality visuals developing appropriate gaming experience. Equally important will be the fact that people wanted more varied gameplay elements, which gives reason to believe that depth and variety mean a lot for them. The mechanics can get stale pretty fast; therefore, different challenges and complex gameplays keep things interesting. This corroborates the finding that there is an inclination toward action-adventure and RPG games, which often provide live and multi-faceted experiences.

Moreover, the participants emphasized freedom of choice to be given to a player in order to allow them to create their type of game style. Once players have options to personalize parts of a game, it usually makes them more interested and may even contribute to spending more time in the game world. Such instances include character customization, different modes of game play, and challenge levels.

The aspect that developers need to pay most attention to is what really attracts the gamers: great graphics, lots of gameplay, and variability in everything. Thus, games developed on this aspect would have a better chance of attracting and sustaining their player base. Although games in this research do not focus on the social dimension, it was clear from the feedback provided that features regarding social aspects are an important component in developing a superior gaming experience. Features related to multiplayer, online communities, and competitive venues gave players a high level of engagement. It's up to the developers to see how social interaction can be naturally introduced into the gameplay so as not to disrupt the natural balance it takes for such features to enhance it instead of harming it. For instance, such multidimensional aspects of designing video games and the realities of player experiences, if considered, will yield a far deeper understanding of the issues at play in today's gaming culture. The continuous evolution of technology and cultural values regarding gaming promptly requires critical discourse and reflection so that games remain in positive expression and a source of inspiration for gamers around the world.

5.1 CONTRIBUTIONS ON FUTURE STUDIES:

The findings from this thesis offer several pathways for future research to deepen the understanding of player engagement and experience in video games, such as expanding demographic research in which future research should aim to include a wider demographic range, encompassing different ages, cultural backgrounds(Dankov, 2023; Shliakhovchuk, 2023; Stevano H. Tobing, 2024), and socioeconomic statuses. This can help identify varied player preferences and engagement factors.

In other way, future research should target specific kinds of people and see what interests them and what can be beneficial for them, elementary students as an example, learning new ways that can make their video game experience not just joyful, but rather educational, by adding their school curriculum(Tunnel & Norbistrath, 2023) in a game that they progress with it and it would help them be more understanding to the different subjects at a young age, or even learn a new skill that will be developed with them as they grow up. Games can be the reason for many students to be speaking English better(HLADONIK & VÁRADI, 2023; Sinar et al., 2024), learn historical facts(Spring, 2015; Stevano H. Tobing, 2024), explore and understand exactly how complex real-life physics works by simplifying the formulas and giving visual yet interactive examples(Rodrigues & Brancher, n.d.), or even learn how to code at a young age. These methods could be really beneficial for students at any age not only the young ones, as harder and more complex and in depth the education can be, developers can take the challenges to the next level and make educational subjects easier to understand. (Laine & Lindberg, 2020; Zhang et al., 2021)

Even in higher education, video games can be implemented to create an environment to learn economic and business strategies, the value would be to learn the principles of economics, business management, financial literacy and strategicplannings(Castro et al., 2021). A game like that would feature real-life economic models, market analysis tools, dynamic supply and demand scenarios, and competition with AI or other players. This can develop the mindset to innovate and plan for future business ideas with the wide range of creativity the students would have. (Fernández-Raga et al., 2023; Pozo et al., 2022)

Health and medical fields are also can be interpreted and implemented as a video game to understand the fundamental knowledge needed for a freshman in medical school, such as anatomy and the complexity of the human body, public health concept, medical procedures, and physiology. It would be a safe environment to learn without the risk of errors or injuries for students to learn and with the right amount of experience they would be ready to get involved in the real-life difficulties of medical issues.(Almulla et al., 2024; Kuo et al., 2024)

Engineering can be another field that video games have immense potential to inspire and educate players about its wide and various fields, including mechanical, electrical and even robotic engineering. Such games would simulate real-world challenges, provide a hands-on experience that is both engaging and educational. Learning values from this method can be various depending on the field that is taken is being educated, for mechanical engineering, a student can learn the basics of mechanics from force, torque, gears, and linkages. same as experimenting with different

designs to observe how changes will affect the project at hand. Same could be said about electrical engineering where the value lies in learning basic circuits, sensors, and actuators. Electrical systems can be designed by students to evaluate their understanding of this matter. All could be used to learn coding skills and software engineering (Gordillo et al., 2022) where players write code to control a machine or a robot and order it to function in a given way or serve a given purpose.(Qosimov, n.d.; Zhang et al., 2021)

Future research should also be invested on how new technologies such as VR, AR, AI will be further developed toward building player experiences. Virtual Reality provides ultimate immersion, placing the user into a completely interactive 3D environment. For example, the implicitly physical games like (Beat Saber), where the use of VR creates engaging experiences through a combination of music and motion that truly enhance user participation and satisfaction. Future game creation could use VR in creating more interactive worlds with improved storytelling, whereby one feels part of the game narrative.

Augmented Reality fills the gap between the virtual and real worlds, allowing unique interaction. (Pokémon Go) represents an example of how AR can foster physical activity and socialization by superimposing digital information on the natural environment. Adding AR to new games in the future will enhance the player's experience through more interactive and contextual engagement and provide a better link among players and surroundings.

Artificial intelligence also plays an important role in realizing adaptive and responsive game environments. Using AI-driven NPCs would greatly enhance reasonable realism and challenge to games. For instance, a concrete example is using AI for customization of enemy behavior depending on player performance, such as in (Left 4 Dead), utilizing the Director AI that dynamically alters game difficulty to sustain player interest. Future research should be directed towards the aspect of how advanced AI will make gaming very personal: offering personalized challenges in a bid to make the games interesting and strong.

Another key direction for future research is the further development of multiplayer features. (Fortnite) and (Among Us) are examples of games that have downplayed the need to involve other players and build a community to retain interest in the game. Co-op missions, competitive leagues, and robust in-game communication features will further develop multiplayer capabilities and ultimately create stronger communities among players, increasing longevity in play. Cross-platform multiplayer can expand the player base even more and lead to even more complex social dynamics in games.

6. CONCLUSION

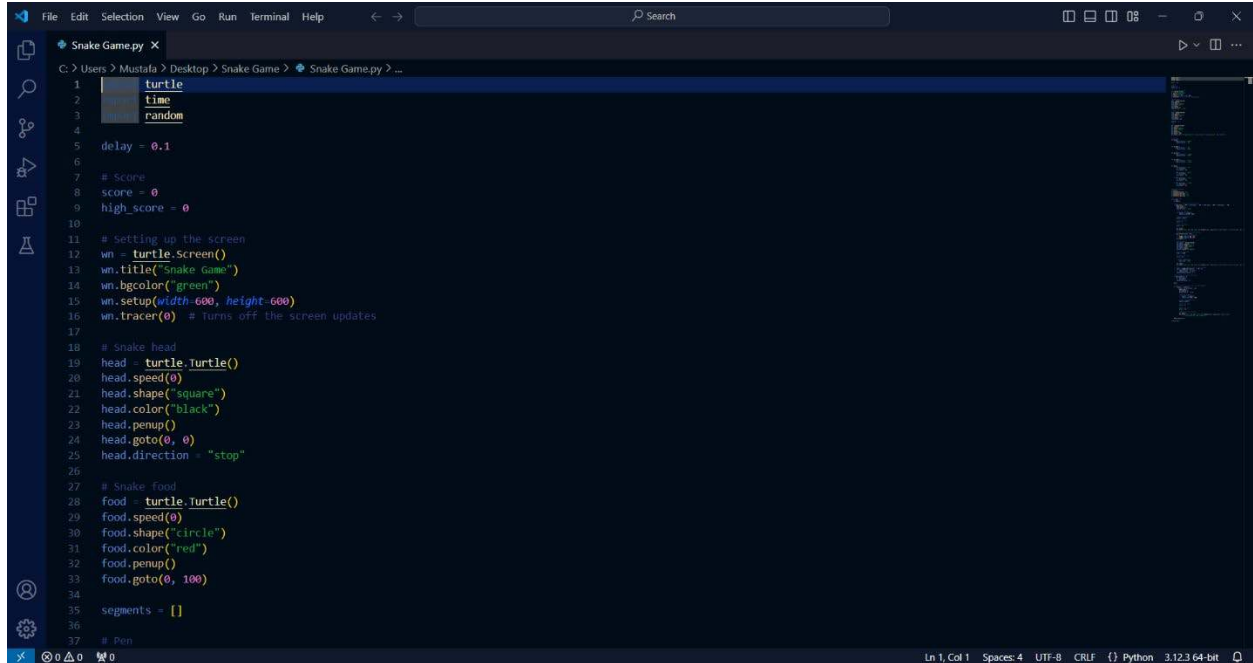
In the end of this study, it is certain that advanced video game programming is a rich tapestry of interwoven elements that creates excellent depth in impact on players. The overwhelming number of themes emerging across the landscape of this far-reaching analysis into industry trends, technological advancement, and player preference are fairly noted. Due to the dynamic nature of video game programming, creativeness and innovation become essential in a player's engagement and attention. More specifically, through visual aesthetics, storytelling, or gameplay mechanics that works as a way for players to escape reality (Wagener & Melzer, 2023) (Koçak et al., 2023; Parong & Green, n.d.; Vuorre et al., 2022) and invest more time into a game, developers are given an exceptional opportunity for giving the player that experience.

The other thing is that as technologies emerge, such as virtual reality, unprecedented opportunities exist to create new definitions of player engagement and immersion, mainly due to the level of interactivity that can be offered through VR over any other technology since players can now inhabit the virtual worlds and experiences like never before.

In conclusion, this thesis has covered the wealth of knowledge in advanced video game programming and its ramifications on the engagement and experience of players. Innovation through building on the community and ethical design enables developers to make games more than mere entertainment to connect and inspire players meaningfully. Since needs and preferences change constantly in the gaming industry worldwide with time, there is a need to be adaptive in responding to such changes.

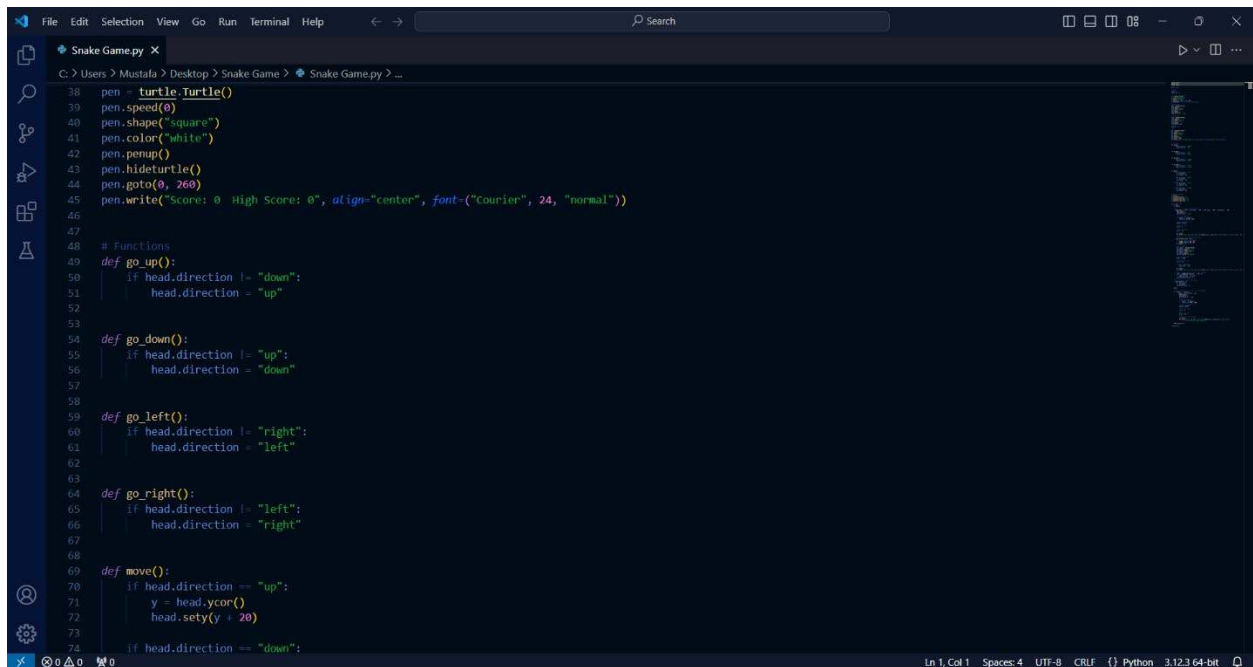
But amidst all the enthusiasm for this technological advancement, the developers should not overlook the ethical factors that should be considered during game development. The developers need to be responsible so that their player base receives their share of enjoyment, and nothing at any end deprives them of such an opportunity.

7. APPENDIX



```
1 import turtle
2 import time
3 import random
4
5 delay = 0.1
6
7 # Score
8 score = 0
9 high_score = 0
10
11 # Setting up the screen
12 wn = turtle.Screen()
13 wn.title("Snake Game")
14 wn.bgcolor("green")
15 wn.setup(width=600, height=600)
16 wn.tracer(0) # turns off the screen updates
17
18 # Snake head
19 head = turtle.Turtle()
20 head.speed(0)
21 head.shape("square")
22 head.color("black")
23 head.penup()
24 head.goto(0, 0)
25 head.direction = "stop"
26
27 # Snake food
28 food = turtle.Turtle()
29 food.speed(0)
30 food.shape("circle")
31 food.color("red")
32 food.penup()
33 food.goto(0, 100)
34
35 segments = []
36
37 # Pen
```

Figure 7.1: Code of Snake game in Python



```
38 pen = turtle.Turtle()
39 pen.speed(0)
40 pen.shape("square")
41 pen.color("white")
42 pen.penup()
43 pen.hideturtle()
44 pen.goto(0, 260)
45 pen.write("Score: 0 High Score: 0", align="center", font=("Courier", 24, "normal"))
46
47 # Functions
48 def go_up():
49     if head.direction != "down":
50         head.direction = "up"
51
52 def go_down():
53     if head.direction != "up":
54         head.direction = "down"
55
56 def go_left():
57     if head.direction != "right":
58         head.direction = "left"
59
60 def go_right():
61     if head.direction != "left":
62         head.direction = "right"
63
64 def move():
65     if head.direction == "up":
66         y = head.ycor()
67         head.sety(y + 20)
68     if head.direction == "down":
```

Figure 7.2: Code of Snake game in Python

```

73 def move():
74     if head.direction == "down":
75         y = head.ycor()
76         head.sety(y + 20)
77     if head.direction == "left":
78         x = head.xcor()
79         head.setx(x - 20)
80     if head.direction == "right":
81         x = head.xcor()
82         head.setx(x + 20)
83
84 # keyboard bindings
85 wn.listen()
86 wn.onkeypress(go_up, "w")
87 wn.onkeypress(go_down, "s")
88 wn.onkeypress(go_left, "a")
89 wn.onkeypress(go_right, "d")
90
91 # Main game loop
92 while True:
93     wn.update()
94
95     # Check for a collision with the border
96     if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor() < -290:
97         time.sleep(1)
98         head.goto(0, 0)
99         head.direction = "stop"
100
101     # Hide the segments
102     for segment in segments:
103         segment.goto(1000, 1000)
104     # Clear the segments list

```

Figure 7.3: Code of Snake game in Python

```

109 segments.clear()
110
111 # Reset the score
112 score = 0
113
114 # Reset the delay
115 delay = 0.1
116
117 pen.clear()
118 pen.write("Score: {} High Score: {}".format(score, high_score), align="center", font=("Courier", 24, "normal"))
119
120 # Check for a collision with the food
121 if head.distance(food) < 20:
122     # Move the food to a random spot
123     x = random.randint(290, 290)
124     y = random.randint(290, 290)
125     food.goto(x, y)
126
127     # Add a segment
128     new_segment = turtle.Turtle()
129     new_segment.speed(0)
130     new_segment.shape("square")
131     new_segment.color("grey")
132     new_segment.penup()
133     segments.append(new_segment)
134
135     # shorten the delay
136     delay -= 0.001
137
138     # Increase the score
139     score += 10
140
141     if score > high_score:
142         high_score = score
143
144     pen.clear()
145     pen.write("Score: {} High Score: {}".format(score, high_score), align="center", font=("Courier", 24, "normal"))

```

Figure 7.4: Code of Snake game in Python

```

146
147 # Move the end segments first in reverse order
148 for index in range(len(segments) - 1, 0, -1):
149     x = segments[index - 1].xcor()
150     y = segments[index - 1].ycor()
151     segments[index].goto(x, y)
152
153 # Move segment 0 to where the head is
154 if len(segments) > 0:
155     x = head.xcor()
156     y = head.ycor()
157     segments[0].goto(x, y)
158
159 move()
160
161 # Check for head collision with the body segments
162 for segment in segments:
163     if segment.distance(head) < 20:
164         time.sleep(1)
165         head.goto(0, 0)
166         head.direction = "stop"
167
168         # Hide the segments
169         for segment in segments:
170             segment.goto(1000, 1000)
171
172         # Clear the segments list
173         segments.clear()
174
175         # Reset the score
176         score = 0
177
178         # Reset the delay
179         delay = 0.1
180
181         # update the score display
182         pen.clear()

```

Figure 7.5: Code of Snake game in Python

```

154
155     x = head.xcor()
156     y = head.ycor()
157     segments[0].goto(x, y)
158
159 move()
160
161 # Check for head collision with the body segments
162 for segment in segments:
163     if segment.distance(head) < 20:
164         time.sleep(1)
165         head.goto(0, 0)
166         head.direction = "stop"
167
168         # Hide the segments
169         for segment in segments:
170             segment.goto(1000, 1000)
171
172         # Clear the segments list
173         segments.clear()
174
175         # Reset the score
176         score = 0
177
178         # Reset the delay
179         delay = 0.1
180
181         # update the score display
182         pen.clear()
183         pen.write("Score: {} High Score: {}".format(score, high_score), align="center",
184                 font=("Courier", 24, "normal"))
185
186     time.sleep(delay)
187
188 wn.mainloop()

```

Figure 7.6: Code of Snake game in Python

```

1  pygame.sys, random
2  from pygame.math import Vector2
3
4  class SNAKE:
5      def __init__(self):
6          self.body = [Vector2(5,10), Vector2(4,10), Vector2(3,10)]
7          self.direction = Vector2(0,0)
8          self.new_block = False
9
10         self.head_up = pygame.image.load('graphics/head_up.png').convert_alpha()
11         self.head_down = pygame.image.load('graphics/head_down.png').convert_alpha()
12         self.head_right = pygame.image.load('graphics/head_right.png').convert_alpha()
13         self.head_left = pygame.image.load('graphics/head_left.png').convert_alpha()
14
15         self.tail_up = pygame.image.load('graphics/tail_up.png').convert_alpha()
16         self.tail_down = pygame.image.load('graphics/tail_down.png').convert_alpha()
17         self.tail_right = pygame.image.load('graphics/tail_right.png').convert_alpha()
18         self.tail_left = pygame.image.load('graphics/tail_left.png').convert_alpha()
19
20         self.body_vertical = pygame.image.load('graphics/body_vertical.png').convert_alpha()
21         self.body_horizontal = pygame.image.load('graphics/body_horizontal.png').convert_alpha()
22
23         self.body_tr = pygame.image.load('graphics/body_tr.png').convert_alpha()
24         self.body_tl = pygame.image.load('graphics/body_tl.png').convert_alpha()
25         self.body_br = pygame.image.load('graphics/body_br.png').convert_alpha()
26         self.body_bl = pygame.image.load('graphics/body_bl.png').convert_alpha()
27         self.crunch_sound = pygame.mixer.Sound('sound/crunch.wav')
28
29     def draw_snake(self):
30         self.update_head_graphics()
31         self.update_tail_graphics()
32
33         for index, block in enumerate(self.body):
34             x_pos = int(block.x * cell_size)
35             y_pos = int(block.y * cell_size)
36             block_rect = pygame.Rect(x_pos, y_pos, cell_size, cell_size)
37

```

Figure 7.7: Code of Snake game 2.0 in Python

```

25     def draw_snake(self):
26
27         if index == 0:
28             screen.blit(self.head, block_rect)
29         elif index == len(self.body) - 1:
30             screen.blit(self.tail, block_rect)
31         else:
32             previous_block = self.body[index + 1] - block
33             next_block = self.body[index - 1] - block
34             if previous_block.x == next_block.x:
35                 screen.blit(self.body_vertical, block_rect)
36             elif previous_block.y == next_block.y:
37                 screen.blit(self.body_horizontal, block_rect)
38             else:
39                 if previous_block.x == -1 and next_block.y == -1 or previous_block.y == -1 and next_block.x == -1:
40                     screen.blit(self.body_tl, block_rect)
41                 elif previous_block.x == -1 and next_block.y == 1 or previous_block.y == 1 and next_block.x == -1:
42                     screen.blit(self.body_bl, block_rect)
43                 elif previous_block.x == 1 and next_block.y == -1 or previous_block.y == -1 and next_block.x == 1:
44                     screen.blit(self.body_tr, block_rect)
45                 elif previous_block.x == 1 and next_block.y == 1 or previous_block.y == 1 and next_block.x == 1:
46                     screen.blit(self.body_br, block_rect)
47
48     def update_head_graphics(self):
49         head_relation = self.body[1] - self.body[0]
50         if head_relation == Vector2(1,0): self.head = self.head_right
51         elif head_relation == Vector2(-1,0): self.head = self.head_left
52         elif head_relation == Vector2(0,1): self.head = self.head_up
53         elif head_relation == Vector2(0,-1): self.head = self.head_down
54
55     def update_tail_graphics(self):
56         tail_relation = self.body[-2] - self.body[-1]
57         if tail_relation == Vector2(1,0): self.tail = self.tail_right
58         elif tail_relation == Vector2(-1,0): self.tail = self.tail_left
59         elif tail_relation == Vector2(0,1): self.tail = self.tail_up
60         elif tail_relation == Vector2(0,-1): self.tail = self.tail_down
61

```

Figure 7.8: Code of Snake game 2.0 in Python

```

File Edit Selection View Go Run Terminal Help
Snake Game 2.0.py X
C:\Users\Mustafa\Desktop>Snake Game 2.0>Snake Game 2.0.py ...
4 class SNAKE:
5     def __init__(self):
6         self.tail_relation = Vector2(0, 1)
7         self.tail = self.tail_down
8
9     def move_snake(self):
10        if self.new_block == True:
11            body_copy = self.body[:]
12            body_copy.insert(0, body_copy[0] + self.direction)
13            self.body = body_copy[1:]
14            self.new_block = False
15        else:
16            body_copy = self.body[:-1]
17            body_copy.insert(0, body_copy[0] + self.direction)
18            self.body = body_copy[:]
19
20    def add_block(self):
21        self.new_block = True
22
23    def play_crunch_sound(self):
24        self.crunch_sound.play()
25
26    def reset(self):
27        self.body = [Vector2(5, 10), Vector2(4, 10), Vector2(3, 10)]
28        self.direction = Vector2(0, 0)
29
30    class FRUIT:
31    def __init__(self):
32        self.randomize()
33
34    def draw_fruit(self):
35        fruit_rect = pygame.Rect(int(self.pos.x * cell_size), int(self.pos.y * cell_size), cell_size, cell_size)
36        screen.blit(apple, fruit_rect)
37        pygame.draw.rect(screen, (126, 166, 114), fruit_rect)
38
39    def randomize(self):
40        self.x = random.randint(0, cell_number - 1)
41        self.y = random.randint(0, cell_number - 1)

```

Figure 7.9: Code of Snake game 2.0 in Python

```

File Edit Selection View Go Run Terminal Help
Snake Game 2.0.py X
C:\Users\Mustafa\Desktop>Snake Game 2.0>Snake Game 2.0.py ...
95 class FRUIT:
96     def randomize(self):
97         self.y = random.randint(0, cell_number - 1)
98         self.pos = Vector2(self.x, self.y)
99
100 class MAIN:
101     def __init__(self):
102         self.snake = SNAKE()
103         self.fruit = FRUIT()
104
105     def update(self):
106         self.snake.move_snake()
107         self.check_collision()
108         self.check_fail()
109
110     def draw_elements(self):
111         self.draw_grass()
112         self.fruit.draw_fruit()
113         self.snake.draw_snake()
114         self.draw_score()
115
116     def check_collision(self):
117         if self.fruit.pos == self.snake.body[0]:
118             self.fruit.randomize()
119             self.snake.add_block()
120             self.snake.play_crunch_sound()
121
122         for block in self.snake.body[1:]:
123             if block == self.fruit.pos:
124                 self.fruit.randomize()
125
126     def check_fail(self):
127         if not 0 <= self.snake.body[0].x < cell_number or not 0 <= self.snake.body[0].y < cell_number:
128             self.game_over()
129
130         for block in self.snake.body[1:]:
131             if block == self.snake.body[0]:

```

Figure 7.10: Code of Snake game 2.0 in Python

```

139 class MAIN:
140     def check_fail(self):
141         if block == self.snake.body[0]:
142             self.game_over()
143
144     def game_over(self):
145         self.snake.reset()
146
147     def draw_grass(self):
148         grass_color = (167,209,61)
149         for row in range(cell_number):
150             if row % 2 == 0:
151                 for col in range(cell_number):
152                     if col % 2 == 0:
153                         grass_rect = pygame.Rect(col * cell_size, row * cell_size, cell_size, cell_size)
154                         pygame.draw.rect(screen, grass_color, grass_rect)
155                     else:
156                         for col in range(cell_number):
157                             if col % 2 != 0:
158                                 grass_rect = pygame.Rect(col * cell_size, row * cell_size, cell_size, cell_size)
159                                 pygame.draw.rect(screen, grass_color, grass_rect)
160
161     def draw_score(self):
162         score_text = str(len(self.snake.body) - 3)
163         score_surface = game_font.render(score_text, True, (56,74,12))
164         score_x = int((cell_size * cell_number - 60))
165         score_y = int((cell_size * cell_number - 40))
166         score_rect = score_surface.get_rect(center = (score_x, score_y))
167         apple_rect = apple.get_rect(midnight = (score_rect.left, score_rect.centery))
168         bg_rect = pygame.Rect(apple_rect.left, apple_rect.top, apple_rect.width + score_rect.width + 6, apple_rect.height)
169
170         pygame.draw.rect(screen, (167,209,61), bg_rect)
171         screen.blit(score_surface, score_rect)
172         screen.blit(apple, apple_rect)
173         pygame.draw.rect(screen, (56,74,12), bg_rect, 2)
174
175 pygame.mixer.pre_init(44100, 16, 2, 512)

```

Figure 7.11: Code of Snake game 2.0 in Python

```

172         pygame.draw.rect(screen, (56,74,12), bg_rect, 2)
173
174     pygame.mixer.pre_init(44100, 16, 2, 512)
175     pygame.init()
176     cell_size = 40
177     cell_number = 20
178     screen = pygame.display.set_mode((cell_number * cell_size, cell_number * cell_size))
179     clock = pygame.time.Clock()
180     apple = pygame.image.load('graphics/apple.png').convert_alpha()
181     game_font = pygame.font.Font('font/PoetsenOne-Regular.ttf', 25)
182
183     SCREEN_UPDATE = pygame.USEREVENT
184     pygame.time.set_timer(SCREEN_UPDATE, 150)
185
186     main_game = MAIN()
187
188     while True:
189         for event in pygame.event.get():
190             if event.type == pygame.QUIT:
191                 pygame.quit()
192                 sys.exit()
193             if event.type == SCREEN_UPDATE:
194                 main_game.update()
195             if event.type == pygame.KEYDOWN:
196                 if event.key == pygame.K_UP:
197                     if main_game.snake.direction.y != 1:
198                         main_game.snake.direction = Vector2(0, 1)
199                 if event.key == pygame.K_RIGHT:
200                     if main_game.snake.direction.x != -1:
201                         main_game.snake.direction = Vector2(1, 0)
202                 if event.key == pygame.K_DOWN:
203                     if main_game.snake.direction.y != -1:
204                         main_game.snake.direction = Vector2(0, 1)
205                 if event.key == pygame.K_LEFT:
206                     if main_game.snake.direction.x != 1:
207                         main_game.snake.direction = Vector2(-1, 0)
208

```

Figure 7.12: Code of Snake game 2.0 in Python

```

177 cell_number = 20
178 screen = pygame.display.set_mode((cell_number * cell_size, cell_number * cell_size))
179 clock = pygame.time.Clock()
180 apple = pygame.image.load('graphics/apple.png').convert_alpha()
181 game_font = pygame.font.Font('font/PoetsenOne-Regular.ttf', 25)
182
183 SCREEN_UPDATE = pygame.USEREVENT
184 pygame.time.set_timer(SCREEN_UPDATE, 150)
185
186 main_game = MAIN()
187
188 while True:
189     for event in pygame.event.get():
190         if event.type == pygame.QUIT:
191             pygame.quit()
192             sys.exit()
193         if event.type == SCREEN_UPDATE:
194             main_game.update()
195         if event.type == pygame.KEYDOWN:
196             if event.key == pygame.K_UP:
197                 if main_game.snake.direction.y != 1:
198                     main_game.snake.direction = Vector2(0, -1)
199             if event.key == pygame.K_RIGHT:
200                 if main_game.snake.direction.x != -1:
201                     main_game.snake.direction = Vector2(1, 0)
202             if event.key == pygame.K_DOWN:
203                 if main_game.snake.direction.y != -1:
204                     main_game.snake.direction = Vector2(0, 1)
205             if event.key == pygame.K_LEFT:
206                 if main_game.snake.direction.x != 1:
207                     main_game.snake.direction = Vector2(-1, 0)
208
209     screen.fill((175, 215, 70))
210     main_game.draw_elements()
211     pygame.display.update()
212     clock.tick(60)

```

Figure 7.13: Code of Snake game 2.0 in Python

```

1
2 import time
3 import turtle
4 import keyboard
5 import random
6 import pickle
7 from playsound import playsound
8 import os.path
9 from os import path
10
11 #window specs & info
12 wn = turtle.Screen()
13 wn.title("Pong Game")
14 wn.bgcolor("black")
15 wn.setup(width=800, height=600)
16 wn.tracer(0)
17
18 #scores
19 score_a = 0
20 score_b = 0
21
22 #collision count
23 a_collisions = 0
24 b_collisions = 0
25
26 #Paddle A
27 paddle_a = turtle.Turtle()
28 paddle_a.speed(0)
29 paddle_a.shape("square")
30 paddle_a.color("red")
31 paddle_a.shapesize(stretch_wid=5, stretch_len=1)
32 paddle_a.penup()
33 paddle_a.goto(-350, 0)
34
35 #paddle B
36 paddle_b = turtle.Turtle()
37 paddle_b.speed(0)

```

Figure 7.14: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py > ...
38 paddle_b.shape("square")
39 paddle_b.color("orange")
40 paddle_b.shapesize(stretch_wid=5, stretch_len=1)
41 paddle_b.penup()
42 paddle_b.goto(350, 0)
43
44 #Ball
45 ball = turtle.Turtle()
46 ball.speed(0)
47 ball.shape("circle")
48 ball.color("white")
49 ball.penup()
50 ball.goto(0, 0)
51 ball.dx = 0.45 # <--zcontrol-a1-- Ball's LEFT/RIGHT, HORIZONTAL, X AXIS speed
52 ball.dy = 0.35 # <--zcontrol-a2-- Ball's UP/DOWN, VERTICAL, Y AXIS speed
53 ball_reset_dy = .35 # <--zcontrol-a3-- (Reset) Ball's LEFT/RIGHT, HORIZONTAL, X AXIS speed
54 ball_reset_dx_player_a = .42 # <--zcontrol-a4-- (Reset Player A) Ball's UP/DOWN, VERTICAL, Y AXIS speed
55 ball_reset_dx_player_b = -.42 # <--zcontrol-a5-- (Reset Player B) Ball's UP/DOWN, VERTICAL, Y AXIS speed ***Negative so if Player score it goes toward other player's goal
56
57 #Pen
58 pen = turtle.Turtle()
59 pen.speed(0)
60 pen.color("white")
61 pen.penup()
62 pen.hideturtle()
63 pen.goto(0, 260)
64 pen.write("Player A : 0 Player B : 0", align="center", font=("courier", 24, "normal"))
65
66 #Center court line
67 center_court = turtle.Turtle()
68 center_court.width(2)
69 center_court.color("white")
70 center_court.hideturtle()
71 point1 = (0, 400)
72 point2 = (0, 400)
73 center_court.penup()
74 center_court.goto(point1)

```

Figure 7.15: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py > ...
75 center_court.pendown()
76 center_court.goto(point2)
77
78 #Paddle A functions
79 def paddle_a_up():
80     y = paddle_a.ycor()
81     y += 42.8
82     paddle_a.sety(y)
83
84 def paddle_a_down():
85     y = paddle_a.ycor()
86     y -= 42.8
87     paddle_a.sety(y)
88
89 #Paddle A keyboard binding while True:
90 wn.listen()
91 wn.onkeypress(paddle_a_up, "W") # <--zcontrol-a6-- Player A's paddle move UP
92 wn.onkeypress(paddle_a_up, "w") # <--zcontrol-a6-- Player A's paddle move UP
93 wn.onkeypress(paddle_a_down, "S") # <--zcontrol-a7-- Player A's paddle move DOWN
94 wn.onkeypress(paddle_a_down, "s") # <--zcontrol-a7-- Player A's paddle move DOWN
95
96 #Paddle B functions
97 def paddle_b_up():
98     y = paddle_b.ycor()
99     y += 42.8
100     paddle_b.sety(y)
101
102 def paddle_b_down():
103     y = paddle_b.ycor()
104     y -= 42.8
105     paddle_b.sety(y)
106
107 #Paddle B keyboard binding while True:
108 wn.listen()
109 wn.onkeypress(paddle_b_up, "Up") # <--zcontrol-a8-- Player B's paddle move UP
110 wn.onkeypress(paddle_b_down, "Down") # <--zcontrol-a9-- Player B's paddle move DOWN
111

```

Figure 7.16: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py > ...
109 wn.onkeypress(paddle_b_up, "Up") # <-zcontrol-all- Player B's paddle move UP
110 wn.onkeypress(paddle_b_down, "Down") # <-zcontrol-a9- Player B's paddle move DOWN
111
112 while True:
113
114     #Startup screen -- Countdown and gameplay window prep
115     if ball.ycor() == 0: # <---- Probably not best but hasn't broke yet (if additional colision physics are added, could cause bug from the coordinates)
116         pen.goto(0, 0)
117         pen.clear()
118         pen.write("1", align="center", font=("lato", 95, "normal"))
119         pen.clear()
120         time.sleep(1)
121         pen.write("2", align="center", font=("lato", 95, "normal"))
122         pen.clear()
123         time.sleep(1)
124         pen.write("3", align="center", font=("lato", 95, "normal"))
125         pen.clear()
126         time.sleep(1)
127         pen.color("green")
128         pen.write("START", align="center", font=("lato", 100, "normal"))
129         pen.clear()
130         time.sleep(.4)
131         pen.color("white")
132         pen.goto(0, 260)
133         pen.write("Player A: 0 Player B: 0", align="center", font=("courier", 24, "normal"))
134         #random ball direction at startup
135         rand = random.randrange(0, 2)
136         if rand < 2:
137             ball.dy = 0.45
138             ball.dx = 0.45
139         if rand < 1:
140             ball.dy = -0.45
141             ball.dx = -0.45
142
143         #move the ball
144         ball.setx(ball.xcor() + ball.dx)
145         ball.sety(ball.ycor() + ball.dy)

```

Figure 7.17: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py > ...
146 wn.update()
147
148 if ball.ycor() > 500 or ball.y < 500:
149     point1 = (0, 400)
150     point2 = (0, 400)
151
152 #border checking
153 #stop ball from going off screen vertically
154 #top border for ball
155 if ball.ycor() > 290:
156     ball.sety(290)
157     ball.dy *= -1
158
159 #Bottom border for ball
160 if ball.ycor() < -290:
161     ball.sety(-290)
162     ball.dy *= -1
163
164 #stop paddles from going off screen
165 #player A's paddle border
166 if (paddle_a.ycor() > 245):
167     paddle_a.sety(245)
168 if (paddle_a.ycor() < -242):
169     paddle_a.sety(-242)
170
171 #Player B's paddle border
172 if (paddle_b.ycor() > 245):
173     paddle_b.sety(245)
174 if (paddle_b.ycor() < -242):
175     paddle_b.sety(-242)
176
177 #scoring and events triggered by a score
178 #Ball passes paddle on right - Player A scored, ball changes to their color, point added, speed adjusted, screen flash (if scored 3 times then winner screen is shown)
179 if ball.xcor() > 395:
180     ball.setx(0)
181     ball.dx *= -1
182     score_a += 1

```

Figure 7.18: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py ...
180 ball.setx(0)
181 ball.dx *= -1
182 score_a += 1
183 ball.color("red")
184 ball.shape("circle")
185 wn.bgcolor("red")
186 pen.clear()
187 ball.shapesize(stretch_wid=1, stretch_len=1)
188 #reset ball speed if Player A scores
189 ball.dy = ball.reset_dy
190 ball.dx = ball.reset_dx_player_a # <-> control ad-- Ball's LEFT/RIGHT, HORIZONTAL, X AXIS speed
191 pen.write("Player A: {} Player B: {}".format(score_a, score_b), align="center", font=("courier", 24, "normal"))
192 time.sleep(.1)
193 wn.bgcolor("black")
194 #Finish game after X rounds and output Player A as winner
195 if score_a > 2: # <-> control bi-- Finish game after X rounds and output Player A as winner. ***Default --- score_b > 2:
196     pen.clear()
197     wn.clear()
198     wn.bgcolor("red")
199     playsound('sounds/game_finish.mp3', block=False)
200     pen.goto(0, 220)
201     pen.write("Player A Wins!" .format(score_a, a_collisions), align="center", font=("lato", 32, "normal"))
202     pen.goto(0, 120)
203     pen.write("With {} hits " .format(a_collisions), align="center", font=("lato", 32, "normal"))
204     print("Player A wins with {} points and {} collisions" .format(score_a, a_collisions))
205     #keep count of games won via pickling
206     player_a_scores_exists = path.exists("player_a_scores.pickle")
207     if not player_a_scores_exists:
208         pickle_out_a = open("player_a_scores.pickle", "wb")
209         player_a_scores = 0
210         pickle.dump(player_a_scores, pickle_out_a)
211         pickle_out_a.close()
212     player_b_scores_exists = path.exists("player_b_scores.pickle")
213     if not player_b_scores_exists:
214         pickle_out_b = open("player_b_scores.pickle", "wb")
215         player_b_scores = 0
216         pickle.dump(player_b_scores, pickle_out_b)

```

Figure 7.19: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py ...
217     pickle_out_b.close()
218     #adding in Player B's score for final game stats
219     pickle_in_b = open("player_b_scores.pickle", "rb")
220     player_b_scores = pickle.load(pickle_in_b)
221     #incrementing game wins by 1 for Player A
222     pickle_in_a = open("player_a_scores.pickle", "rb")
223     player_a_scores = pickle.load(pickle_in_a)
224     player_a_scores += 1
225     pickle_out_a = open("player_a_scores.pickle", "wb")
226     pickle.dump(player_a_scores, pickle_out_a)
227     pickle_out_a.close()
228     print(player_a_scores)
229     pen.goto(0, 220)
230     pen.write("Player A total wins: {}".format(player_a_scores), align="center", font=("lato", 32, "normal"))
231     pen.goto(0, 80)
232     pen.write("Player B total wins: {}".format(player_b_scores), align="center", font=("lato", 32, "normal"))
233     time.sleep(3)
234     break
235
236 #ball passes paddle on left - Player B scored, ball changes to their color, point added, speed adjusted. If score 3 times then winner screen is shown, and winner point
237 if ball.xcor() < -395:
238     ball.setx(0)
239     ball.dx *= -1
240     score_b += 1
241     wn.bgcolor("orange")
242     ball.shape("circle")
243     ball.color("orange")
244     pen.clear()
245     ball.shapesize(stretch_wid=1, stretch_len=1)
246     #reset ball speed if Player B scores
247     ball.dy = ball.reset_dy
248     ball.dx = ball.reset_dx_player_b
249     pen.write("Player A: {} Player B: {}".format(score_a, score_b), align="center", font=("courier", 24, "normal"))
250     time.sleep(.1)
251     wn.bgcolor("black")
252     #Finish game after X rounds and output Player B as winner
253     if score_b > 2: # <-> control bi-- Finish game after X rounds and output Player B as winner. ***Default --- score_b > 2:

```

Figure 7.20: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py ...

254 time.sleep(.14)
255 pen.clear()
256 wn.clear()
257 wn.bgcolor("orange")
258 playsound('sounds/game_finish.mp3', block=False)
259 pen.goto(0, 220)
260 pen.write("Player B Wins!" .format(score_b, b_collisions), align="center", font=("lato", 32, "normal"))
261 pen.goto(0, 120)
262 pen.write("With {} hit(s) " .format(b_collisions), align="center", font=("lato", 32, "normal"))
263 print("Player B wins with {} points and {} collisions" .format(score_b, b_collisions))
264 #keep count of games won via Pickling
265 player_a_scores_exists = path.exists("player_a_scores.pickle")
266 if not player_a_scores_exists:
267     pickle_out_a = open("player_a_scores.pickle", "wb")
268     player_a_scores = 0
269     pickle.dump(player_a_scores, pickle_out_a)
270     pickle_out_a.close()
271 player_b_scores_exists = path.exists("player_b_scores.pickle")
272 if not player_b_scores_exists:
273     pickle_out_b = open("player_b_scores.pickle", "wb")
274     player_b_scores = 0
275     pickle.dump(player_b_scores, pickle_out_b)
276     pickle_out_b.close()
277 #Being in Player A's score for final game stats
278 pickle_in_a = open("player_a_scores.pickle", "rb")
279 player_a_scores = pickle.load(pickle_in_a)
280 #Increment++ game wins by 1 for Player B
281 pickle_in_b = open("player_b_scores.pickle", "rb")
282 player_b_scores = pickle.load(pickle_in_b)
283 player_b_scores += 1
284 pickle_out_b = open("player_b_scores.pickle", "wb")
285 pickle.dump(player_b_scores, pickle_out_b)
286 pickle_out_b.close()
287 print(player_b_scores)
288 pen.goto(0, 20)
289 pen.write("Player A total wins: {}".format(player_a_scores), align="center", font=("lato", 32, "normal"))
290
291

```

Figure 7.21: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py ...

288 pen.goto(0, 20)
289 pen.write("Player A total wins: {}".format(player_a_scores), align="center", font=("lato", 32, "normal"))
290 pen.goto(0, 80)
291 pen.write("Player B total wins: {}".format(player_b_scores), align="center", font=("lato", 32, "normal"))
292 time.sleep(3)
293 break
294
295 #reset ball bounce effects
296 #Add bounce and speed effects to ball
297 #thinnet shape 1/10
298 if (ball.shape() == "circle" and ball.xcor() < 300 and ball.xcor() > -300):
299     ball.shapesize(stretch_wid=.8, stretch_len=.9)
300 if (ball.shape() == "triangle" and ball.xcor() < 210 and ball.xcor() > -210):
301     ball.shapesize(stretch_wid=.4, stretch_len=.2)
302 #regainng shape 2/10
303 if (ball.shape() == "circle" and ball.xcor() < 200 and ball.xcor() > -200):
304     ball.shapesize(stretch_wid=.56, stretch_len=.1)
305 #regainng shape 3/10
306 if (ball.shape() == "circle" and ball.xcor() < 190 and ball.xcor() > -200):
307     ball.shapesize(stretch_wid=.60, stretch_len=.1)
308 #regainng shape 4/10
309 if (ball.shape() == "circle" and ball.xcor() < 190 and ball.xcor() > -200):
310     ball.shapesize(stretch_wid=.64, stretch_len=.1)
311 #Regainng shape 5/10
312 if (ball.shape() == "circle" and ball.xcor() < 180 and ball.xcor() > -200):
313     ball.shapesize(stretch_wid=.67, stretch_len=.1)
314 #Regainng shape 6/10
315 if (ball.shape() == "circle" and ball.xcor() < 160 and ball.xcor() > -160):
316     ball.shapesize(stretch_wid=.70, stretch_len=.1)
317 #Regainng shape 7/10
318 if (ball.shape() == "circle" and ball.xcor() < 125 and ball.xcor() > -125):
319     ball.shapesize(stretch_wid=.73, stretch_len=.1)
320 #Regainng shape 8/10
321 if (ball.shape() == "circle" and ball.xcor() < 125 and ball.xcor() > -125):
322     ball.shapesize(stretch_wid=.86, stretch_len=.1)
323 #Regainng shape 9/10
324 if (ball.shape() == "circle" and ball.xcor() < 80 and ball.xcor() > -80):

```

Figure 7.22: Code of Pong game in Python

```

325 ball.shape((stretch_wid-1, stretch_len-1))
326 #back to full size and shape 10/10
327 if (ball.shape() == "triangle" and ball.xcor() < 210 and ball.xcor() > -210): #nuclear attack ball bounce finish
328     ball.shape((stretch_wid-4, stretch_len-4))
329
330 #Paddles and ball collisions
331 #Player A
332 if (ball.xcor() < -330 and ball.xcor() > -340) and (ball.ycor() < paddle_a.ycor() + 59 and ball.ycor() > paddle_a.ycor() - 59):
333     ball.dx *= -1
334     ball.dy *= -1.19
335     ball.sety(ball.ycor() + ball.dy + 5)
336     a_collisions += 1
337     playsound('sounds/player_a_hit.mp3', block=False)
338     ball.shape((stretch_wid-1, stretch_len-.5)) #squish ball/Bounce effect when hits paddle
339     #random "super attack"
340     rand = random.randrange(0, 15) #1/15 chance - update both player's rand variables for fair gameplay.
341     if rand < 2:
342         wn.bgcolor("white")
343         time.sleep(.1)
344         wn.bgcolor("red")
345         time.sleep(.1)
346         wn.bgcolor("white")
347         time.sleep(.1)
348         ball.dy = 0.12
349         ball.dx = 0.11
350         ball.dy *= 6.1
351         ball.dx *= 9.2
352         wn.bgcolor("black")
353         ball.color("red")
354     #nuclear Super attack
355     if keyboard.is_pressed("Space"):
356         playsound('sounds/pre_attack.mp3', block=False)
357         playsound('sounds/nuclear_attack.mp3', block=False)
358         ball.dy = ball_reset_dy
359         ball.dy = 20
360         ball.dx = 3.9

```

Figure 7.23: Code of Pong game in Python

```

359 ball.dy = ball_reset_dy
360 ball.dy = 20
361 ball.dx = 3.9
362 ball.setheading(0)
363 ball.color("red")
364 ball.shape("triangle")
365 ball.shape((stretch_wid-2, stretch_len-4)) #squish ball/Bounce effect when hits paddle
366 wn.bgcolor("white")
367 time.sleep(.1)
368 wn.bgcolor("red")
369 time.sleep(.1)
370 wn.bgcolor("white")
371 time.sleep(.2)
372 wn.bgcolor("red")
373 time.sleep(.2)
374 wn.bgcolor("black")
375 time.sleep(.1)
376 #when ball collides with paddle, ball's y axis is random
377 rand = random.randrange(0, 2)
378 if rand < 2:
379     ball.dy = 0.45
380 if rand < 1:
381     ball.dy = -0.45
382
383 #Player B
384 if (ball.xcor() > 330 and ball.xcor() < 340) and (ball.ycor() < paddle_b.ycor() + 59 and ball.ycor() > paddle_b.ycor() - 59):
385     ball.setx(330)
386     ball.dx *= -1
387     ball.dy *= -1.19
388     ball.sety(ball.ycor() + ball.dy + 5)
389     b_collisions += 1
390     playsound('sounds/player_b_hit.mp3', block=False)
391     ball.shape((stretch_wid-1, stretch_len-.5)) #squish ball/Bounce effect when hits paddle
392     #Auto Super attack
393     rand = random.randrange(0, 15) #1/15 chance - update both player's rand variables for fair gameplay. (or don't)
394     if rand < 2:
395         wn.bgcolor("white")

```

Figure 7.24: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py > ...
396     time.sleep(.1)
397     wn.bgcolor("orange")
398     time.sleep(.1)
399     wn.bgcolor("white")
400     time.sleep(.1)
401     ball.dy = 0.12
402     ball.dx = 0.11
403     ball.dy += 6.1
404     ball.dx += 0.2
405     wn.bgcolor("black")
406     ball.color("red")
407     #nuclear super attack
408     if keyboard.is_pressed('0'):
409         playsound('sounds/pre_attack.mp3', block=False)
410         playsound('sounds/nuclear_attack.mp3', block=False)
411         ball.dy = 3.9
412         ball.dx = -3.3
413         ball.setheading(180)
414         ball.shape("triangle")
415         ball.color("orange")
416         ball.shape("triangle")
417         ball.shapesize(stretch_wid=2, stretch_len=4) #squish ball/bounce effect when hits paddle
418         wn.bgcolor("white")
419         time.sleep(.1)
420         wn.bgcolor("orange")
421         time.sleep(.1)
422         wn.bgcolor("yellow")
423         time.sleep(.1)
424         wn.bgcolor("white")
425         time.sleep(.2)
426         wn.bgcolor("black")
427         time.sleep(.1)
428
429     #when ball collides with paddle, ball's y axis is random
430     rand = random.randrange(0, 3)
431     if rand < 2:
432         ball.dy *= -0.45
433
434
Ln 1, Col 1  Spaces: 4  UTF-8  LF  Python  3.12.3 64-bit

```

Figure 7.25: Code of Pong game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Pong Game > Pong Game.py > ...
404     ball.dx *= 0.2
405     wn.bgcolor("black")
406     ball.color("red")
407     #nuclear super attack
408     if keyboard.is_pressed('0'):
409         playsound('sounds/pre_attack.mp3', block=False)
410         playsound('sounds/nuclear_attack.mp3', block=False)
411         ball.dy = 3.9
412         ball.dx = -3.3
413         ball.setheading(180)
414         ball.shape("triangle")
415         ball.color("orange")
416         ball.shape("triangle")
417         ball.shapesize(stretch_wid=2, stretch_len=4) #squish ball/bounce effect when hits paddle
418         wn.bgcolor("white")
419         time.sleep(.1)
420         wn.bgcolor("orange")
421         time.sleep(.1)
422         wn.bgcolor("yellow")
423         time.sleep(.1)
424         wn.bgcolor("white")
425         time.sleep(.2)
426         wn.bgcolor("black")
427         time.sleep(.1)
428
429     #when ball collides with paddle, ball's y axis is random
430     rand = random.randrange(0, 3)
431     if rand < 2:
432         ball.dy *= -0.45
433     if rand > 2:
434         ball.dy *= 0.45
435
436
Ln 1, Col 1  Spaces: 4  UTF-8  LF  Python  3.12.3 64-bit

```

Figure 7.26: Code of Pong game in Python

```

1  pygame
2  os
3  pygame.font.init()
4  pygame.mixer.init()
5
6  WIDTH, HEIGHT = 900, 500
7  WIN = pygame.display.set_mode((WIDTH, HEIGHT))
8  pygame.display.set_caption("Pong Shooter Game")
9
10 WHITE = (255, 255, 255)
11 BLACK = (0, 0, 0)
12 RED = (255, 0, 0)
13 YELLOW = (255, 255, 0)
14
15 BORDER = pygame.Rect(WIDTH/2 - 5, 0, 10, HEIGHT)
16
17 #BULLET_HIT_SOUND = pygame.mixer.Sound('Assets/Grenade1.mp3')
18 #BULLET_FIRE_SOUND = pygame.mixer.Sound('Assets/Gunfire1.mp3')
19
20 HEALTH_FONT = pygame.font.Sysfont('comicsans', 40)
21 WINNER_FONT = pygame.font.Sysfont('comicsans', 100)
22
23 FPS = 60
24 VEL = 5
25 BULLET_VEL = 7
26 MAX_BULLETS = 3
27 SPACESHIP_WIDTH, SPACESHIP_HEIGHT = 55, 40
28
29 YELLOW_HIT = pygame.USEREVENT + 1
30 RED_HIT = pygame.USEREVENT + 2
31
32 YELLOW_SPACESHIP_IMAGE = pygame.image.load(
33     os.path.join('Assets', 'spaceship_yellow.png'))
34 YELLOW_SPACESHIP = pygame.transform.rotate(pygame.transform.scale(
35     YELLOW_SPACESHIP_IMAGE, (SPACESHIP_WIDTH, SPACESHIP_HEIGHT)), 90)
36
37 RED_SPACESHIP_IMAGE = pygame.image.load(

```

Figure 7.27: Code of Pong Shooter game in Python

```

38     os.path.join('Assets', 'spaceship_red.png'))
39 RED_SPACESHIP = pygame.transform.rotate(pygame.transform.scale(
40     RED_SPACESHIP_IMAGE, (SPACESHIP_WIDTH, SPACESHIP_HEIGHT)), 270)
41
42 SPACE = pygame.transform.scale(pygame.image.load(
43     os.path.join('Assets', 'space.png')), (WIDTH, HEIGHT))
44
45
46 def draw_window(red, yellow, red_bullets, yellow_bullets, red_health, yellow_health):
47     WIN.blit(SPACE, (0, 0))
48     pygame.draw.rect(WIN, BLACK, BORDER)
49
50     red_health_text = HEALTH_FONT.render(
51         "Health: " + str(red_health), 1, WHITE)
52     yellow_health_text = HEALTH_FONT.render(
53         "Health: " + str(yellow_health), 1, WHITE)
54     WIN.blit(red_health_text, (WIDTH - red_health_text.get_width() - 10, 10))
55     WIN.blit(yellow_health_text, (10, 10))
56
57     WIN.blit(YELLOW_SPACESHIP, (yellow.x, yellow.y))
58     WIN.blit(RED_SPACESHIP, (red.x, red.y))
59
60     for bullet in red_bullets:
61         pygame.draw.rect(WIN, RED, bullet)
62
63     for bullet in yellow_bullets:
64         pygame.draw.rect(WIN, YELLOW, bullet)
65
66     pygame.display.update()
67
68
69 def yellow_handle_movement(keys_pressed, yellow):
70     if keys_pressed[pygame.K_a] and yellow.x - VEL > 0: # LEFT
71         yellow.x -= VEL
72     if keys_pressed[pygame.K_d] and yellow.x + VEL + yellow.width < BORDER.x: # RIGHT
73         yellow.x += VEL
74     if keys_pressed[pygame.K_w] and yellow.y - VEL > 0: # UP

```

Figure 7.28: Code of Pong Shooter game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop>Pong Shooter Game>Pong Shooter Game.py>...
60 def yellow_handle_movement(keys_pressed, yellow):
61     yellow.x += VEL
62     if keys_pressed[pygame.K_W] and yellow.y - VEL > 0: # UP
63         yellow.y -= VEL
64     if keys_pressed[pygame.K_S] and yellow.y + VEL + yellow.height + HEIGHT - 15: # DOWN
65         yellow.y += VEL
66
67 def red_handle_movement(keys_pressed, red):
68     if keys_pressed[pygame.K_LEFT] and red.x - VEL > BORDER.x + BORDER.width: # LEFT
69         red.x -= VEL
70     if keys_pressed[pygame.K_RIGHT] and red.x + VEL + red.width < WIDTH: # RIGHT
71         red.x += VEL
72     if keys_pressed[pygame.K_UP] and red.y - VEL > 0: # UP
73         red.y -= VEL
74     if keys_pressed[pygame.K_DOWN] and red.y + VEL + red.height + HEIGHT - 15: # DOWN
75         red.y += VEL
76
77 def handle_bullets(yellow_bullets, red_bullets, yellow, red):
78     for bullet in yellow_bullets:
79         bullet.x += BULLET_VEL
80         if red.collidect(bullet):
81             pygame.event.post(pygame.event.Event(RED_HIT))
82             yellow_bullets.remove(bullet)
83         elif bullet.x > WIDTH:
84             yellow_bullets.remove(bullet)
85
86     for bullet in red_bullets:
87         bullet.x += BULLET_VEL
88         if yellow.collidect(bullet):
89             pygame.event.post(pygame.event.Event(YELLOW_HIT))
90             red_bullets.remove(bullet)
91         elif bullet.x < 0:
92             red_bullets.remove(bullet)
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2
```

```

117 def main():
118     pygame.init()
119     screen = pygame.display.set_mode((800, 600))
120     pygame.display.set_caption("Pong Shooter Game")
121     screen.fill((0, 0, 0))
122     # Load images
123     bullet_image = pygame.image.load('bullet.png')
124     red_image = pygame.image.load('red.png')
125     yellow_image = pygame.image.load('yellow.png')
126     bullet_sound = pygame.mixer.Sound('bullet_sound.wav')
127     hit_sound = pygame.mixer.Sound('hit_sound.wav')
128
129     # Create bullet
130     bullet = pygame.sprite.Sprite()
131     bullet.image = bullet_image
132     bullet.rect = bullet_image.get_rect()
133     bullet.rect.x = 50
134     bullet.rect.y = 50
135
136     # Create red and yellow
137     red = pygame.sprite.Sprite()
138     red.image = red_image
139     red.rect = red_image.get_rect()
140     red.rect.x = 400
141     red.rect.y = 300
142
143     yellow = pygame.sprite.Sprite()
144     yellow.image = yellow_image
145     yellow.rect = yellow_image.get_rect()
146     yellow.rect.x = 600
147     yellow.rect.y = 300
148
149     # Create bullet group
150     bullet_group = pygame.sprite.Group()
151     bullet_group.add(bullet)
152
153     # Create red and yellow groups
154     red_group = pygame.sprite.Group()
155     red_group.add(red)
156
157     yellow_group = pygame.sprite.Group()
158     yellow_group.add(yellow)
159
160     # Create bullet fire sound
161     bullet_fire_sound = pygame.mixer.Sound('bullet_fire_sound.wav')
162
163     # Create hit sound
164     hit_sound = pygame.mixer.Sound('hit_sound.wav')
165
166     # Create winner text
167     winner_text = ""
168
169     # Create red health
170     red_health = 100
171
172     # Create yellow health
173     yellow_health = 100
174
175     # Create bullet fire
176     bullet_fire = False
177
178     # Create hit
179     hit = False
180
181     # Create winner
182     winner = False
183
184     # Create game loop
185     while True:
186         # Get events
187         for event in pygame.event.get():
188             if event.type == pygame.QUIT:
189                 pygame.quit()
190                 sys.exit()
191
192             if event.type == pygame.KEYDOWN:
193                 # Fire bullet
194                 if event.key == pygame.K_SPACE and not bullet_fire:
195                     bullet_fire = True
196                     bullet_group.add(bullet)
197                     bullet_fire_sound.play()
198
199                 # Move red
200                 if event.key == pygame.K_LEFT:
201                     red.rect.x -= 5
202
203                 if event.key == pygame.K_RIGHT:
204                     red.rect.x += 5
205
206                 # Move yellow
207                 if event.key == pygame.K_UP:
208                     yellow.rect.y -= 5
209
210                 if event.key == pygame.K_DOWN:
211                     yellow.rect.y += 5
212
213             if event.type == pygame.KEYUP:
214                 # Stop fire
215                 if event.key == pygame.K_SPACE:
216                     bullet_fire = False
217
218         # Update
219         bullet_group.update()
220         red_group.update()
221         yellow_group.update()
222
223         # Check for collisions
224         if bullet_group.spritecollideany(red_group):
225             hit_sound.play()
226             red_health -= 10
227
228         if bullet_group.spritecollideany(yellow_group):
229             hit_sound.play()
230             yellow_health -= 10
231
232         # Draw
233         screen.fill((0, 0, 0))
234         bullet_group.draw(screen)
235         red_group.draw(screen)
236         yellow_group.draw(screen)
237
238         # Display health
239         font = pygame.font.Font('freesansbold.ttf', 16)
240         red_health_text = font.render(str(red_health), True, (255, 0, 0))
241         yellow_health_text = font.render(str(yellow_health), True, (0, 255, 0))
242
243         screen.blit(red_health_text, (100, 100))
244         screen.blit(yellow_health_text, (500, 100))
245
246         # Display winner
247         if winner:
248             font = pygame.font.Font('freesansbold.ttf', 24)
249             winner_text = font.render(winner_text, True, (255, 255, 255))
250             screen.blit(winner_text, (300, 300))
251
252         # Update display
253         pygame.display.flip()
254
255         # Delay
256         pygame.time.wait(100)
257
258     # End game
259     pygame.quit()
260     sys.exit()
261
262 if __name__ == "__main__":
263     main()

```

Figure 7.31: Code of Pong Shooter game in Python

```

1 import turtle
2 import math
3 import random
4
5 # Create screen
6 wn = turtle.Screen()
7 wn.setup(width=600, height=600)
8 wn.title("Space Defender Game")
9 wn.bgcolor("black")
10 wn.bgpic("background.gif")
11 # Stop screen updates
12 wn.tracer(0)
13
14 # Register Shapes
15 player_vertices = ((0,15),(-15,0),(-18,5),(-18, 5),(0,0),(18, 5),(18, 5),(15, 0))
16 wn.register_shape("player", player_vertices)
17
18 asteroid_vertices = ((0, 10), (5, 7), (3,3), (10,0), (7, 4), (8, -6), (0, -10), (-5, -5), (-7, -7), (-10, 0), (-5, 4), (-1, 8))
19 wn.register_shape("asteroid", asteroid_vertices)
20
21 class Sprite(turtle.Turtle):
22     def __init__(self):
23         turtle.Turtle.__init__(self)
24         # Maximum animation speed
25         self.speed(0)
26         self.penup()
27
28     def get_heading_to(t1, t2):
29         x1 = t1.xcor()
30         y1 = t1.ycor()
31
32         x2 = t2.xcor()
33         y2 = t2.ycor()
34
35         heading = math.atan2(y2 - y1, x2 - x1)
36         heading = heading + 180.0 / 3.14159

```

Figure 7.32: Code of Space Defender game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Defender Game > space_station.py > ...
29 def get_heading_to(x1, y1, x2, y2):
30     heading = math.atan2(y2 - y1, x2 - x1)
31     heading = heading * 180.0 / 3.14159
32     return heading
33
34 player = Sprite()
35 player.color("white")
36 player.shape("player")
37 player.score = 0
38
39 missiles = []
40 for _ in range(3):
41     missile = Sprite()
42     missile.color("red")
43     missile.shape("arrow")
44     missile.speed = 1
45     missile.state = "ready"
46     missile.hideturtle()
47     missiles.append(missile)
48
49 pen = Sprite()
50 pen.color("white")
51 pen.hideturtle()
52 pen.goto(0, 250)
53 pen.write("Score: 0", False, align="center", font=("Arial", 24, "normal"))
54
55 asteroids = []
56 for _ in range(5):
57     asteroid = Sprite()
58     asteroid.color("brown")
59     asteroid.shape("asteroid")
60     asteroid.speed = random.randint(2, 3)/50
61     asteroid.goto(0, 0)
62     heading = random.randint(0, 360)
63     distance = random.randint(300, 400)

```

Figure 7.33: Code of Space Defender game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Defender Game > space_station.py > ...
72 asteroid.setheading(heading)
73 asteroid.fd(distance)
74 asteroid.setheading(get_heading_to(player, asteroid))
75 asteroids.append(asteroid)
76
77 def rotate_left():
78     player.lt(20)
79
80 def rotate_right():
81     player.rt(20)
82
83 def fire_missile():
84     for missile in missiles:
85         if missile.state == "ready":
86             missile.goto(0, 0)
87             missile.showturtle()
88             missile.setheading(player.heading())
89             missile.state = "fire"
90             break
91
92 # keyboard binding
93 wn.listen()
94 wn.onkey(rotate_left, "Left")
95 wn.onkey(rotate_right, "Right")
96 wn.onkey(fire_missile, "space")
97
98 # Start main game loop
99 Game_Over = False
100 while True:
101     # Do screen update
102     wn.update()
103     player.goto(0, 0)
104
105     # Move the Missile
106     for missile in missiles:
107         if missile.state == "fire":
108             missile.fd(missile.speed)

```

Figure 7.34: Code of Space Defender game in Python

```

109
110 # Border checking
111 if missile.xcor() > 300 or missile.xcor() < -300 or missile.ycor() > 300 or missile.ycor() < -300:
112     missile.hideturtle()
113     missile.state = "ready"
114
115 # Iterate through asteroids
116 for asteroid in asteroids:
117     # Move the asteroid
118     asteroid.fd(asteroid.speed)
119
120 # Check for collisions
121 # Asteroid and missile
122 for missile in missiles:
123     if asteroid.distance(missile) < 20:
124         # Reset Asteroid
125         heading = random.randint(0, 260)
126         distance = random.randint(600, 800)
127         asteroid.setheading(heading)
128         asteroid.fd(distance)
129         asteroid.setheading(get_heading_to(player, asteroid))
130         asteroid.speed += 0.01
131
132         # Reset Missile
133         missile.goto(600, 600)
134         missile.hideturtle()
135         missile.state = "ready"
136
137         # Increase score
138         player.score += 10
139         pen.clear()
140         pen.write("Score: {}".format(player.score), False, align = "center", font = ("Arial", 24, "normal"))
141
142 # Asteroid and Player
143 if asteroid.distance(player) < 20:
144     # Reset Asteroid
145     heading = random.randint(0, 260)

```

Figure 7.35: Code of Space Defender game in Python

```

130     asteroid.speed += 0.01
131
132     # Reset Missile
133     missile.goto(600, 600)
134     missile.hideturtle()
135     missile.state = "ready"
136
137     # Increase score
138     player.score += 10
139     pen.clear()
140     pen.write("Score: {}".format(player.score), False, align = "center", font = ("Arial", 24, "normal"))
141
142 # Asteroid and Player
143 if asteroid.distance(player) < 20:
144     # Reset Asteroid
145     heading = random.randint(0, 260)
146     distance = random.randint(600, 800)
147     asteroid.setheading(heading)
148     asteroid.fd(distance)
149     asteroid.setheading(get_heading_to(player, asteroid))
150     asteroid.speed += 0.005
151     Game_Over = True
152     player.score -= 30
153     pen.clear()
154     pen.write("Score: {}".format(player.score), False, align = "center", font = ("Arial", 24, "normal"))
155
156 if Game_Over == True:
157     player.hideturtle()
158     missile.hideturtle()
159     for a in asteroids:
160         a.hideturtle()
161     pen.clear()
162     wn.bgs pic("end.gif")
163     break
164
165 wn.mainloop()

```

Figure 7.36: Code of Space Defender game in Python

```

1  import pygame
2  import os
3  import time
4  import random
5  pygame.font.init()
6
7
8  # Global vars & consts
9  WIDTH, HEIGHT = 600, 600
10 WIN = pygame.display.set_mode((WIDTH, HEIGHT))
11 pygame.display.set_caption("Space Shooter Game")
12 main_font = pygame.font.SysFont("comicsans", 35)
13 lost_font = pygame.font.SysFont("comicsans", 45)
14
15
16 # Global Functions
17 # detect collision between two objects
18 def collide(obj1, obj2):
19     offset_x = obj2.x - obj1.x
20     offset_y = obj2.y - obj1.y
21     return obj1.mask.overlap(obj2.mask, (offset_x, offset_y)) != None
22
23
24 # Draw screen UI
25 def draw_lives(lives):
26     lives_label = main_font.render(f"Lives: {lives}", 1, (255, 255, 255))
27     WIN.blit(lives_label, (10, 10))
28
29
30 def draw_level(level):
31     level_label = main_font.render(f"Level: {level}", 1, (255, 255, 255))
32     WIN.blit(level_label, (WIDTH - level_label.get_width() - 10, 10))
33
34
35 def draw_lost():
36     lost_label = lost_font.render(f"You Lost!", 1, (255, 255, 255))
37     WIN.blit(lost_label, (WIDTH/2 - lost_label.get_width()/2, 350))

```

Figure 7.37: Code of Space Shooter game in Python

```

38
39
40 # Loading Assets
41 RED_SPACE_SHIP = pygame.image.load(
42     os.path.join("assets", "pixel_ship_red_small.png"))
43 GREEN_SPACE_SHIP = pygame.image.load(
44     os.path.join("assets", "pixel_ship_green_small.png"))
45 BLUE_SPACE_SHIP = pygame.image.load(
46     os.path.join("assets", "pixel_ship_blue_small.png"))
47
48 # Player ship
49 YELLOW_SPACE_SHIP = pygame.image.load(
50     os.path.join("assets", "pixel_ship_yellow.png"))
51
52 # Lasers
53 RED_LASER = pygame.image.load(os.path.join("assets", "pixel_laser_red.png"))
54 GREEN_LASER = pygame.image.load(
55     os.path.join("assets", "pixel_laser_green.png"))
56 BLUE_LASER = pygame.image.load(os.path.join("assets", "pixel_laser_blue.png"))
57 YELLOW_LASER = pygame.image.load(
58     os.path.join("assets", "pixel_laser_yellow.png"))
59
60 # Background
61 BG = pygame.transform.scale(pygame.image.load(
62     os.path.join("assets", "background-black.png")), (WIDTH, HEIGHT))
63
64
65 # Ship Classes
66 class Ship:
67     FRAMES_BETWEEN_SHOTS = 30 # .5 second
68
69     def __init__(self, x, y, health=100):
70         self.x = x
71         self.y = y
72         self.health = health
73         self.ship_img = None
74         self.laser_img = None

```

Figure 7.38: Code of Space Shooter game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Shooter Game > Space Shooter Game.py > ...
66 class Ship:
67     def __init__(self, x, y, health=100):
74         self.laser_img = None
75         self.lasers = []
76         self.frames_counter = 0
77
78     # Ship Update
79     # move the ship lasers and check collision with the player and if off_screen
80     def move_lasers(self, vel, obj): # obj is the player
81         self.frames_counter += 1
82         for laser in self.lasers:
83             laser.move(vel)
84             if laser.off_screen(HEIGHT):
85                 self.lasers.remove(laser)
86             elif laser.collison(obj):
87                 obj.health -= 10
88                 self.lasers.remove(laser)
89
90     # handle shooting lasers
91     def shoot(self):
92         if self.frames_counter <= 0:
93             laser = Laser(self.x, self.y, self.laser_img)
94             self.lasers.append(laser)
95             self.frames_counter = self.FRAMES_BETWEEN_SHOTS
96         else:
97             self.frames_counter -= 1
98
99     # get width and height of the ship
100     def get_width(self):
101         return self.ship_img.get_width()
102
103     def get_height(self):
104         return self.ship_img.get_height()
105
106     # Ship Draw
107     # draw ship and its lasers
108     def draw(self, window):

```

Figure 7.39: Code of Space Shooter game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Shooter Game > Space Shooter Game.py > ...
107     # draw ship and its lasers
108     def draw(self, window):
109         window.blit(self.ship_img, (self.x, self.y))
110         for laser in self.lasers:
111             laser.draw(window)
112
113 class Laser:
114     def __init__(self, x, y, img):
115         self.x = x
116         self.y = y
117         self.img = img
118         self.mask = pygame.mask.from_surface(self.img)
119
120     # Laser Update
121     def move(self, vel):
122         self.y += vel
123
124     # check if laser is out of the screen
125     def off_screen(self, height):
126         return self.y > height or self.y < 0
127
128     def collision(self, obj):
129         return collide(self, obj)
130
131     # Laser Draw
132     def draw(self, window):
133         window.blit(self.img, (self.x, self.y))
134
135 # Player & Enemy Ships
136 class Player(Ship):
137     def __init__(self, x, y, health=100):
138         super().__init__(x, y, health)
139         self.ship_img = YELLOW_SPACE_SHIP

```

Figure 7.40: Code of Space Shooter game in Python

```

139 class Player(Ship):
140     def __init__(self, x, y, health=100):
141         self.ship_img = pygame.image.load('assets/ship.png')
142         self.laser_img = YELLOW_LASER
143         self.mask = pygame.mask.from_surface(self.ship_img)
144         self.max_health = health
145
146     # Player update
147     # move the player ship lasers and check collision with the enemies and if off screen
148     def move(self, vel):
149         self.frames_counter -= 1
150         for laser in self.lasers:
151             laser.move(vel)
152             if laser.off_screen(HEIGHT):
153                 self.lasers.remove(laser)
154             else:
155                 for obj in self.objs:
156                     if laser.collide(obj):
157                         obj.remove()
158                         if laser in self.lasers:
159                             self.lasers.remove(laser)
160
161     # Player Draw
162     def draw(self, window):
163         super().draw(window)
164         self.draw_healthbar(window)
165
166     # for drawing healthbar
167     def draw_healthbar(self, window):
168         pygame.draw.rect(window, (255, 0, 0), (self.x, self.y +
169             self.ship_img.get_height() - 10, self.ship_img.get_width(), 10))
170         pygame.draw.rect(window, (0, 255, 0), (self.x, self.y + self.ship_img.get_height(
171             ) - 10, self.ship_img.get_width() * (self.health/self.max_health), 10))
172
173 class Enemy(Ship):
174     COLOR_MAP = {
175         "red": (RED_SPACE_SHIP, RED_LASER),
176         "green": (GREEN_SPACE_SHIP, GREEN_LASER),
177         "blue": (BLUE_SPACE_SHIP, BLUE_LASER),
178     }
179
180     def __init__(self, x, y, color, health=100):
181         super().__init__(x, y, health)
182         self.ship_img, self.laser_img = self.COLOR_MAP[color]
183         self.mask = pygame.mask.from_surface(self.ship_img)
184
185     # Enemy update
186     def move(self, vel):
187         self.y += vel
188
189     def shoot(self):
190         if self.frames_counter <= 0:
191             laser = Laser(self.x - 20, self.y, self.laser_img)
192             self.lasers.append(laser)
193             self.frames_counter = self.FRAMES_BETWEEN_SHOTS
194         else:
195             self.frames_counter -= 1
196
197 # Main Game Loop
198 def main():
199     run = True
200     FPS = 60
201     level = 0
202     lives = 5 # number of enemy pass
203
204     enemies = []
205     wave_length = 5
206     enemy_vel = 1
207
208     player_vel = 5

```

Figure 7.41: Code of Space Shooter game in Python

```

175 class Enemy(Ship):
176     COLOR_MAP = {
177         "red": (RED_SPACE_SHIP, RED_LASER),
178         "green": (GREEN_SPACE_SHIP, GREEN_LASER),
179         "blue": (BLUE_SPACE_SHIP, BLUE_LASER),
180     }
181
182     def __init__(self, x, y, color, health=100):
183         super().__init__(x, y, health)
184         self.ship_img, self.laser_img = self.COLOR_MAP[color]
185         self.mask = pygame.mask.from_surface(self.ship_img)
186
187     # Enemy update
188     def move(self, vel):
189         self.y += vel
190
191     def shoot(self):
192         if self.frames_counter <= 0:
193             laser = Laser(self.x - 20, self.y, self.laser_img)
194             self.lasers.append(laser)
195             self.frames_counter = self.FRAMES_BETWEEN_SHOTS
196         else:
197             self.frames_counter -= 1
198
199 # Main Game Loop
200 def main():
201     run = True
202     FPS = 60
203     level = 0
204     lives = 5 # number of enemy pass
205
206     enemies = []
207     wave_length = 5
208     enemy_vel = 1
209
210     player_vel = 5

```

Figure 7.42: Code of Space Shooter game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Shooter Game > Space Shooter Game.py > ...
201 def main():
210     player_vel = 5
211     laser_vel = 5
212
213     player = Player(250, 450)
214
215     clock = pygame.time.Clock()
216
217     lost = False
218     lost_count = 0
219
220
221     def redraw_window():
222         WIN.blit(BG, (0, 0))
223         # draw text
224         draw_lives(lives)
225         draw_level(level)
226
227         for enemy in enemies:
228             enemy.draw(WIN)
229
230         player.draw(WIN)
231
232         if lost:
233             draw_lost()
234
235         pygame.display.update()
236
237     while run:
238         clock.tick(FPS)
239         redraw_window()
240
241         if lives <= 0 or player.health <= 0:
242             lost = True
243             lost_count += 1
244
245         if lost:

```

Figure 7.43: Code of Space Shooter game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Shooter Game > Space Shooter Game.py > ...
244
245     if lost:
246         if lost_count > FPS * 3:
247             run = False
248         else:
249             continue
250
251     if len(enemies) == 0:
252         level += 1
253         wave_length += 5
254         for i in range(wave_length):
255             enemy = Enemy(random.randrange(
256                 50, WIDTH-100), random.randrange(-100, 100), random.choice(["red", "blue", "green"]))
257             enemies.append(enemy)
258
259     # Player Movement
260     keys = pygame.key.get_pressed()
261     if keys[pygame.K_a] and player.x - player_vel > 0: # left
262         player.x -= player_vel
263     if keys[pygame.K_d] and player.x + player_vel + player.get_width() < WIDTH: # right
264         player.x += player_vel
265     if keys[pygame.K_w] and player.y - player_vel > 0: # up
266         player.y -= player_vel
267     # down (15 for healthbar)
268     if keys[pygame.K_s] and player.y + player_vel + player.get_height() + 15 < HEIGHT:
269         player.y += player_vel
270
271     # for shooting:
272     if keys[pygame.K_SPACE]:
273         player.shoot()
274
275     for enemy in enemies[:]:
276         enemy.move(enemy_vel)
277         enemy.move_lasers(laser_vel, player)
278
279     if random.randrange(0, 2*60) == 1:

```

Figure 7.44: Code of Space Shooter game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Shooter Game > Space Shooter Game.py > ...
278 def main():
279     if random.randrange(0, 2*60) == 1:
280         enemy.shoot()
281
282     if collide(enemy, player):
283         player.health -= 10
284         enemies.remove(enemy)
285     elif enemy.y + enemy.get_height() > HEIGHT:
286         lives -= 1
287         enemies.remove(enemy)
288
289     player.move_lasers( laser_vel, enemies)
290
291     for event in pygame.event.get():
292         if event.type == pygame.QUIT:
293             quit()
294
295 # Draw Main Menu
296 def main_menu():
297     title_font = pygame.font.SysFont("comicsans", 45)
298     run = True
299     while run:
300         WIN.blit(86, (0, 0))
301         title_label = title_font.render(
302             "Press left click to begin...", 1, (255, 255, 255))
303         WIN.blit(title_label, (WIDTH/2 - title_label.get_width()/2, 350))
304         pygame.display.update()
305         for event in pygame.event.get():
306             if event.type == pygame.QUIT:
307                 run = False
308             if event.type == pygame.MOUSEBUTTONDOWN:
309                 main()
310     pygame.quit()
311
312
313
Ln 315, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.3 64-bit

```

Figure 7.45: Code of Space Shooter game in Python

```

File Edit Selection View Go Run Terminal Help
C:\Users\Mustafa\Desktop> Space Shooter Game > Space Shooter Game.py > ...
281
282     if collide(enemy, player):
283         player.health -= 10
284         enemies.remove(enemy)
285     elif enemy.y + enemy.get_height() > HEIGHT:
286         lives -= 1
287         enemies.remove(enemy)
288
289     player.move_lasers( laser_vel, enemies)
290
291     for event in pygame.event.get():
292         if event.type == pygame.QUIT:
293             quit()
294
295 # Draw Main Menu
296 def main_menu():
297     title_font = pygame.font.SysFont("comicsans", 45)
298     run = True
299     while run:
300         WIN.blit(86, (0, 0))
301         title_label = title_font.render(
302             "Press left click to begin...", 1, (255, 255, 255))
303         WIN.blit(title_label, (WIDTH/2 - title_label.get_width()/2, 350))
304         pygame.display.update()
305         for event in pygame.event.get():
306             if event.type == pygame.QUIT:
307                 run = False
308             if event.type == pygame.MOUSEBUTTONDOWN:
309                 main()
310     pygame.quit()
311
312
313
314 main_menu()
315
Ln 315, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.3 64-bit

```

Figure 7.46: Code of Space Shooter game in Python

7.1 SURVEY STRUCTURE

Section 1 of 6

Welcome to the Player Engagement and Gaming Experience Survey! This survey aims to gather valuable insights into players' experiences with video games. After playing the games, we'd like to hear your thoughts on your experience with these games, as well as your preferences and feedback on what you want to see in video games in general. Your participation will help us better understand how advanced video game programming impacts player engagement and enjoyment. Thank you for taking the time to share your thoughts and experiences with us!.

Section 2 of 6

Demographics

Gender: Please select your gender identity.

- Male
- Female
- Prefer not to say

Age: Please select your age range.

- 18 – 25
- 25 – 30
- 30+

Gaming Experience: How would you describe your level of gaming experience?

- **Casual gamer:** I play games occasionally for fun.
- **Moderate gamer:** I play games regularly and enjoy a variety of genres.
- **Hardcore gamer:** I am highly dedicated to gaming and play frequently.

Section 3 of 6

First User Feedback: We'd like to hear your thoughts and feedback on the first set of games you played, the Snake game, Pong game and Space Defender Game. Please take a moment to answer the following questions based on your experience with this game. Your feedback will help us understand what aspects of the games worked well and areas for improvement. Thank you for your participation!

-What are your overall thoughts on the Snake game?

Good / Average / Bad

- How could the Snake Game be improved?

Better graphics / Sound effects / Smoother gameplay

-Would you recommend the Snake Game in this state to a friend?

Yes / No

-What are your overall thoughts on the Pong game?

Good / Average / Bad

- How could the Pong Game be improved?

Better graphics / Sound effects / Smoother gameplay

-Would you recommend the Pong Game in this state to a friend?

Yes / No

-What are your overall thoughts on the Space Defender game?

Good / Average / Bad

- How could the Space Defender Game be improved?

Better graphics / Sound effects / Smoother gameplay

-Would you recommend the Space Defender Game in this state to a friend?

Yes / No

Section 4 of 6

Second User Feedback: We value your feedback on the second set of improved games you played, Please take a moment to share your thoughts and opinions on this game. Your feedback will help us understand what aspects of the game were successful and areas for improvement.

-What are your overall thoughts on the Snake game 2.0?

Good / Average / Bad

- What elements have you noticed that got improved in Snake game 2.0?

Better graphics / Sound effects / Smoother gameplay

-Would you recommend the Snake Game 2.0 in this state to a friend?

Yes / No

-What are your overall thoughts on the Pong Shooter Game?

Good / Average / Bad

- What elements have you noticed that got improved in Pong Shooter Game?

Better graphics / Sound effects / Smoother gameplay

-Would you recommend the Pong Shooter Game in this state to a friend?

Yes / No

-What are your overall thoughts on the Space Shooter Game?

Good / Average / Bad

- What elements have you noticed that got improved in Space Shooter Game?

Better graphics / Sound effects / Smoother gameplay

-Would you recommend the Space Shooter Game in this state to a friend?

Yes / No

Section 5 of 6

General Thoughts on Video Games: We're interested in hearing your opinions on video games in general. Please take a moment to share your thoughts on various aspects of gaming. Your feedback will provide valuable insights into the preferences and expectations of players like you. Thank you for your participation!

-What do you enjoy most about playing video games?

- Immersive storytelling
- Challenging gameplay
- Social interaction with friends or online communities
- Relaxation and escapism
- Competition and achievement

-Which genres of video games do you prefer?

- Action
- Role-Playing Games (RPG)
- Adventure
- Simulation
- Strategy
- Sports
- Puzzle
- Horror
- Fighting
- Multiplayer Online Battle Arena (MOBA):

-What factors influence your decision to purchase or play a video game?

- Reviews and ratings
- Graphics and visual presentations
- Story and narrative depth
- Gameplay mechanics and controls
- Multiplayer or online features
- Developer reputation or previous experience with their games
- Price and value for money
- Recommendations from friends or influencers

-Player choice and customization in video games, is it important to you?

Yes / Maybe / No

-What features or innovations would you like to see more of in future video games?

- Advanced graphics and realistic physics
- Deep and branching narratives with meaningful choices
- Seamless open-world environments
- Cooperative multiplayer experiences
- Virtual reality and immersive technologies
- Cross-platform compatibility
- User-generated content and modding support

Section 6 of 6

Additional Comments: If you have any additional comments, suggestions, or feedback you'd like to share with us regarding video games or your experience with this survey, please feel free to do so in the space provided below. Your insights are greatly appreciated!

8. REFERENCES

- Almulla, A. A., Khasawneh, M. A. S., Abdel-Hadi, S. A., & Jarrah, H. Y. (2024). Influence of Non-linear Storytelling in Video Games on Cognitive Development among Students with Learning Disabilities. *International Journal of Learning, Teaching and Educational Research*, 23(1), 84–97. <https://doi.org/10.26803/ijlter.23.1.5>
- Bowman, N. D., Condis, M., Yoshimura, K., & Bohaty, E. (2023). Vicarious Nostalgia? Playing Retrogames Fosters an Appreciation for Gaming History. *AoIR Selected Papers of Internet Research*. <https://doi.org/10.5210/spir.v2023i0.13400>
- Caroux, L. (2023). Presence in video games: A systematic review and meta-analysis of the effects of game design choices. In *Applied Ergonomics* (Vol. 107). Elsevier Ltd. <https://doi.org/10.1016/j.apergo.2022.103936>
- Castro, H., Pinto, N., Pereira, F., Ferreira, L., Ávila, P., Bastos, J., Putnik, G. D., & Cruz-Cunha, M. (2021). Cyber-Physical Systems using Open Design: An approach towards an Open Science Lab for Manufacturing. *Procedia Computer Science*, 196, 381–388. <https://doi.org/10.1016/j.procs.2021.12.027>
- Dankov, Y. (2023). Digital Presentation and Preservation of Cultural and Scientific Heritage. *Conference Proceedings*, 13.
- Deliyannis, I., Kaimara, P., Maria Poulimenou, S., & Lampoura, S. (2023). Introductory Chapter: Games, Gamification, and Ludification, Can They Be Combined? In *Gamification - Analysis, Design, Development and Ludification*. IntechOpen. <https://doi.org/10.5772/intechopen.109101>
- Fernández-Raga, M., Aleksić, D., İkiz, A. K., Markiewicz, M., & Streit, H. (2023). Development of a Comprehensive Process for Introducing Game-Based Learning in Higher Education for Lecturers. *Sustainability (Switzerland)*, 15(4). <https://doi.org/10.3390/su15043706>
- Filipović, A. (2023). The Role of Artificial Intelligence in Video Game Development. *Kultura Polisa*, 20(3), 50–67. <https://doi.org/10.51738/kpolisa2023.20.3r.50f>
- Gordillo, A., Lopez-Fernandez, D., & Tovar, E. (2022). Comparing the Effectiveness of Video-Based Learning and Game-Based Learning Using Teacher-Authored Video Games for Online Software Engineering Education. *IEEE Transactions on Education*, 65(4), 524–532. <https://doi.org/10.1109/TE.2022.3142688>

- HLADONIK, G., & VÁRADI, K. (2023). Video Games – A New Source of Language Acquisition. *Humanities Science Current Issues*, 1(67), 207–212. <https://doi.org/10.24919/2308-4863/67-1-28>
- Jin, Y. (2024). Application of cyberspace security in video games industry. *Applied and Computational Engineering*, 53(1), 197–204. <https://doi.org/10.54254/2755-2721/53/20241371>
- Koçak, Ö. E., Gorgievski, M., & Bakker, A. B. (2023). Recovery from work by playing video games. *Applied Psychology*. <https://doi.org/10.1111/apps.12519>
- Kuo, H. J., Yeomans, M., Ruiz, D., & Lin, C. C. (2024). Video games and disability—a risk and benefit analysis. In *Frontiers in Rehabilitation Sciences* (Vol. 5). Frontiers Media SA. <https://doi.org/10.3389/fresc.2024.1343057>
- Laine, T. H., & Lindberg, R. S. N. (2020). Designing Engaging Games for Education: A Systematic Literature Review on Game Motivators and Design Principles. *IEEE Transactions on Learning Technologies*, 13(4), 804–821. <https://doi.org/10.1109/TLT.2020.3018503>
- Ma, R., Li, Y., & Kou, Y. (2023, April 19). Transparency, Fairness, and Coping: How Players Experience Moderation in Multiplayer Online Games. *Conference on Human Factors in Computing Systems - Proceedings*. <https://doi.org/10.1145/3544548.3581097>
- Parong, J., & Green, S. (n.d.). *HOW VIDEO GAMES CHANGE THE BRAIN YOUNG REVIEWERS: AKSHARA*.
- Passos, E. B., Medeiros, D. B., Neto, P. A. S., & Clua, E. W. G. (n.d.). *Turning Real-World Software Development into a Game*. <http://www.redcritertracker.com>
- Pozo, J. I., Cabellos, B., & Sánchez, D. L. (2022). Do teachers believe that video games can improve learning? *Heliyon*, 8(6). <https://doi.org/10.1016/j.heliyon.2022.e09798>
- Qosimov, S. (n.d.). *The Role of Video Games in Education*. <https://www.researchgate.net/publication/375005093>
- Rodrigues, L., & Brancher, J. (n.d.). *The Impact of Procedural Level Generation on Players' Experiences and In-game Behavior*. <http://spacemath.rpbtecnologia.com.br>
- Shliakhovchuk, E. (2023). Learning About Refugee Life With Educational Video Games. *Information Technologies and Learning Tools*, 98(6), 94–106. <https://doi.org/10.33407/itlt.v98i6.5168>
- Sinar, T. S., Zein, T. T., Huszka, B., Yusuf, M., Maharani, P., & Sanjaya, D. (2024). Learning English Literacy through Video Games: A Multimodal Perspective. *Journal of Curriculum and Teaching*, 13(1), 102–118. <https://doi.org/10.5430/jct.v13n1p102>

- Spring, D. (2015). Gaming history: Computer and video games as historical scholarship. In *Rethinking History* (Vol. 19, Issue 2, pp. 207–221). Routledge.
<https://doi.org/10.1080/13642529.2014.973714>
- Stevano H. Tobing, D. (2024). Game Design in the Design Thinking Process For a Video Game Development Model to Support Tourism. *KnE Engineering*.
<https://doi.org/10.18502/keg.v6i1.15413>
- The Video Game debate*. (n.d.).
- Tunnel, R.-H., & Norbistrath, U. (2023). Classification of Video Games Bachelor's Curricula. *Journal of Education and Learning*, 12(2), 39. <https://doi.org/10.5539/jel.v12n2p39>
- Vuorre, M., Johannes, N., Magnusson, K., & Przybylski, A. K. (2022). Time spent playing video games is unlikely to impact well-being. *Royal Society Open Science*, 9(7).
<https://doi.org/10.1098/rsos.220411>
- Wagener, G. L., & Melzer, A. (2023). *Seligman's PERMA Model in Video Games – The Positive Influence of Video Games on Well-Being* (pp. 43–58). https://doi.org/10.1007/978-981-99-8248-6_4
- Wulf, T., Bowman, N. D., Rieger, D., & Anthony Velez, J. (2018). *Video Games as Time Machines: Video Game Nostalgia and the Success of Retro Gaming*.
<https://doi.org/10.17645/mac.v6i2.131>
- Zhang, R. Y., Chopin, A., Shibata, K., Lu, Z. L., Jaeggi, S. M., Buschkuehl, M., Green, C. S., & Bavelier, D. (2021). Action video game play facilitates “learning to learn.” *Communications Biology*, 4(1). <https://doi.org/10.1038/s42003-021-02652-7>

9. BIOGRAPHY

Omar ALQAYSI is master's student in the Computer Engineering program at Beykoz University, where he took courses in a variety of computer engineering fields such as Image Processing, Computer Ethics, Information Retrieval, Cryptology, Software Reliability, and Theory of Computation.

Omar earned his bachelor's degree in software engineering from Odesa Polytechnic National University in Ukraine. During his undergraduate studies, ALQAYSI was passionate about exploring how video games are developed, understanding game mechanics, and implementing advanced and diverse programming methods in the video game industry. ALQAYSI is a native Arabic speaker, fluently speaking English, and has a certificate in the Russian and Turkish language.

ALQAYSI plans to continue his research and pursue a career in the gaming industry, where he hopes to influence the development of innovative and engaging games that resonate with players worldwide.

Date: October 2024

