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 Külahçıoğlu

Asst. Prof. Dr. Zeynep Behrin Güven Aydın

CONTENTS

ÖZET

İLERİ VİDEO 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 içgörüleri, 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

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 roaster, 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.

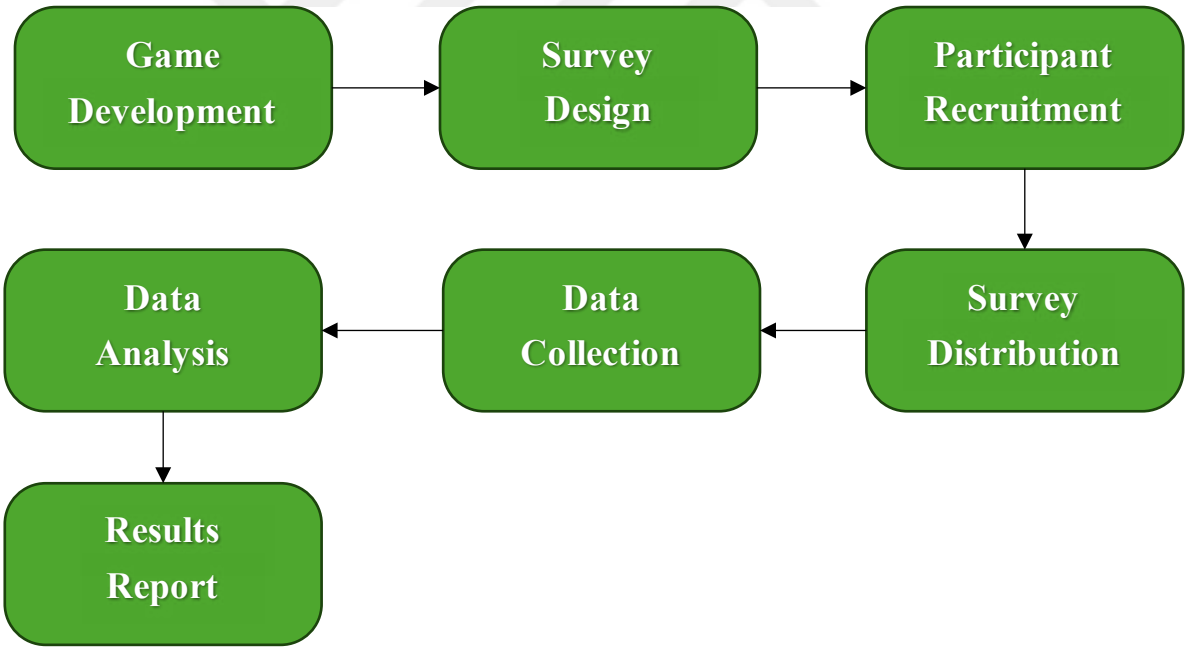


Figure 3.1: Research methodology steps process

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 refluxes 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

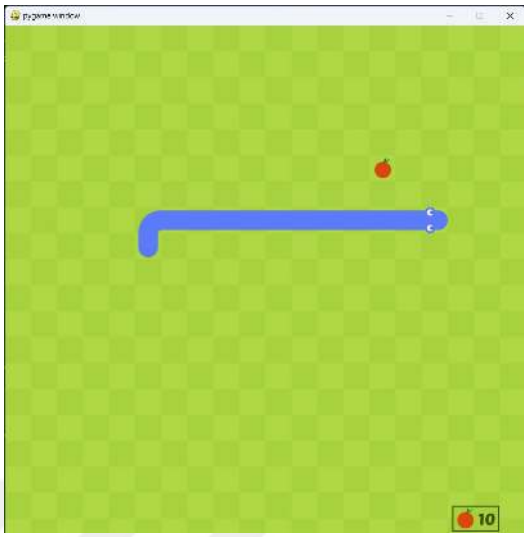


Figure 3.3: Interface of Snake game 2.0

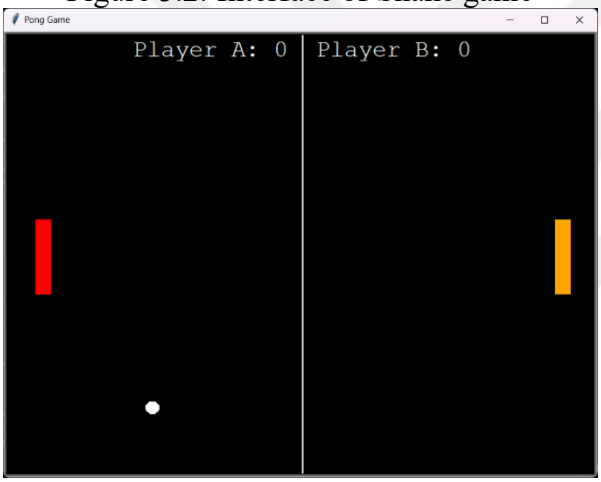


Figure 3.4: Interface of Pong game

Figure 3.5: Interface of Pong Shooter game

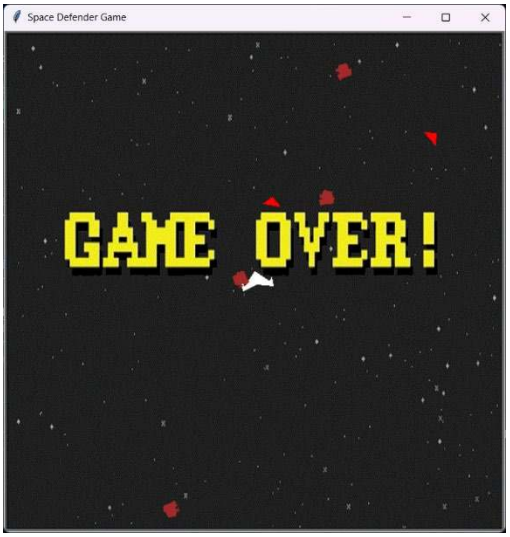


Figure 3.6: Interface of Space Defender game

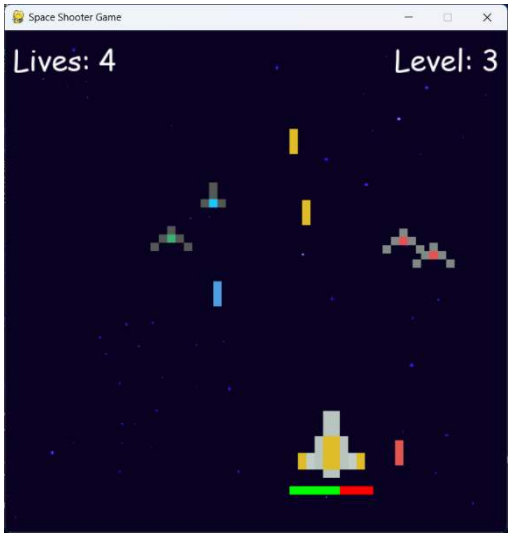


Figure 3.7: Interface of Space Shooter game

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 collects 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 DISTRUBTION:

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.

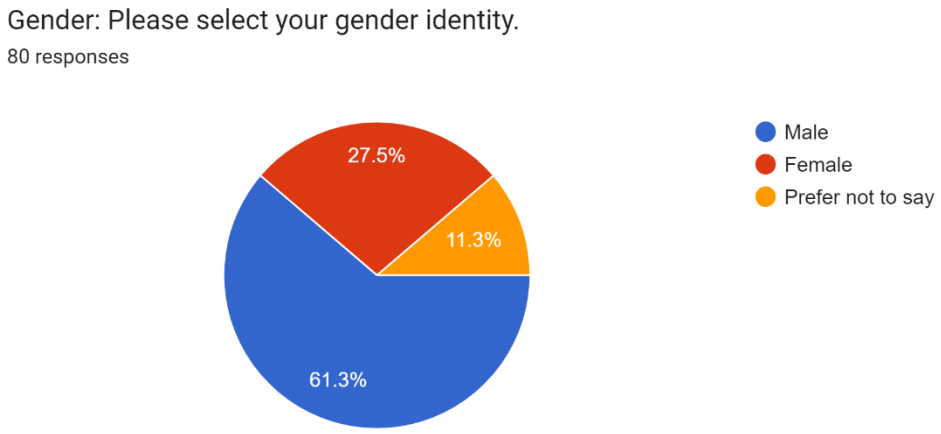


Figure 4.1: The percentage of participation regarding gender

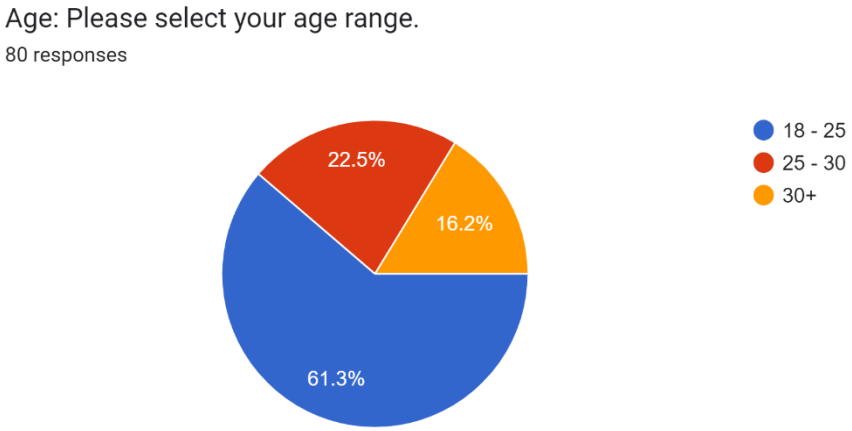


Figure 4.2: The percentage of participation regarding age range

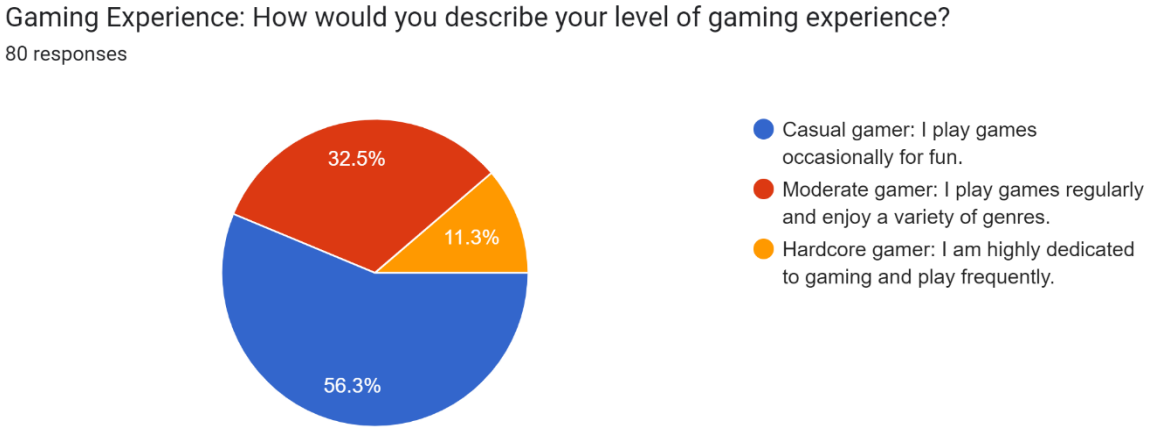


Figure 4.3: The percentage of participation regarding overall game experience

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.

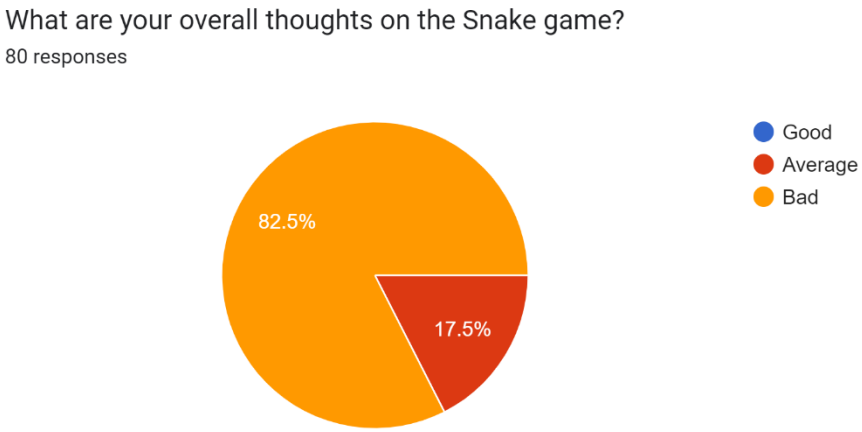


Figure 4.4: The percentage of overall thought on Snake game

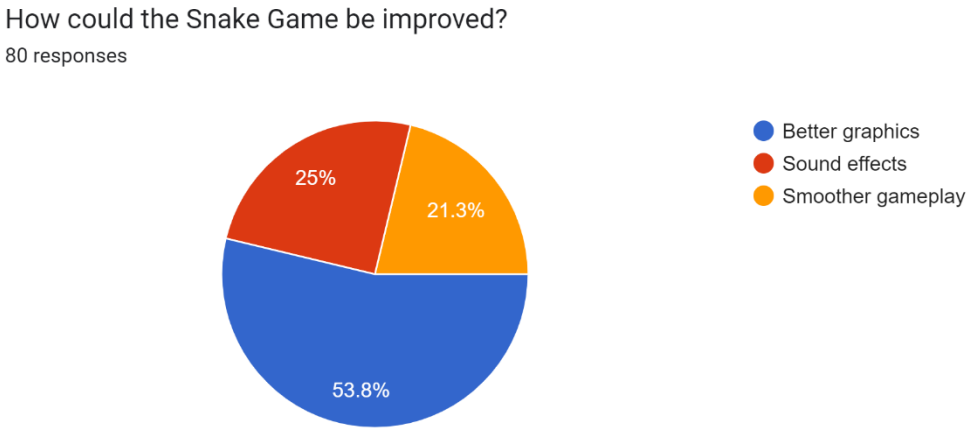


Figure 4.5: The percentage of improvement suggestions for Snake game

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.

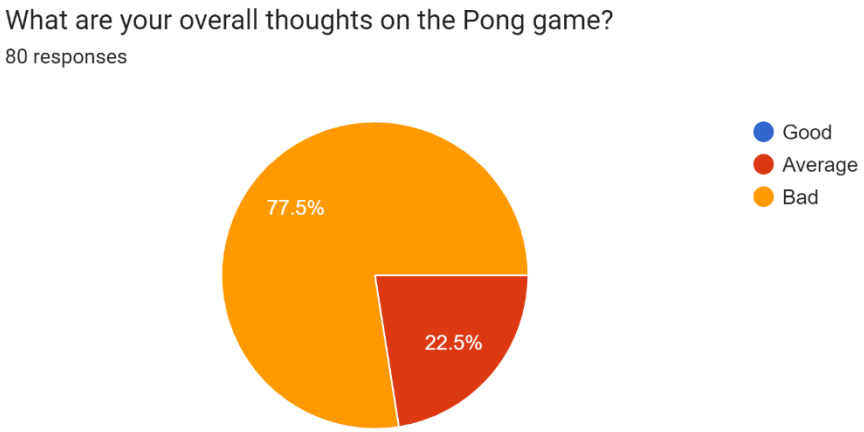


Figure 4.6: The percentage of overall thought on Pong game

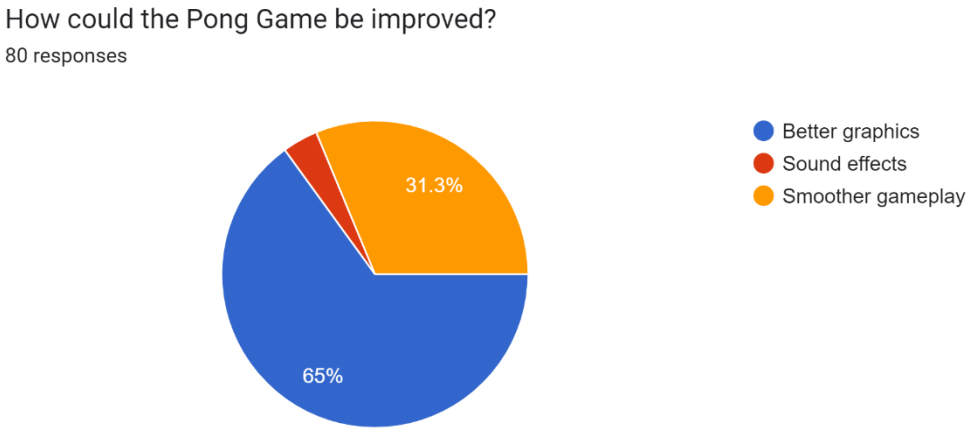


Figure 4.7: The percentage of improvement suggestions for Pong game

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.

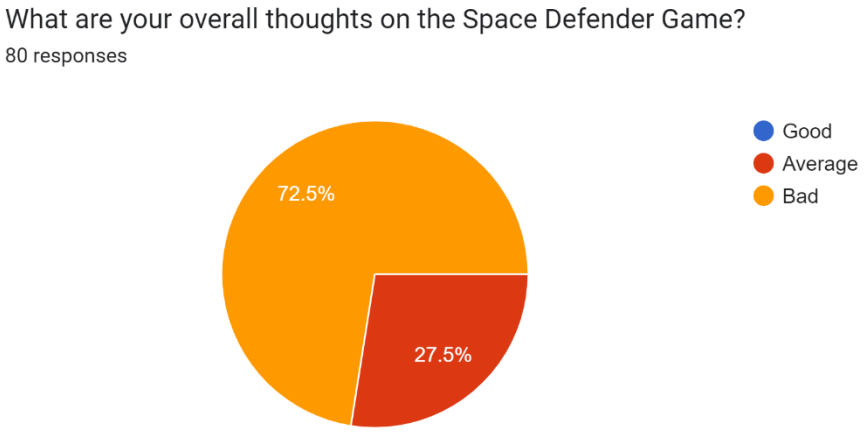


Figure 4.8: The percentage of overall thought on Space Defender game

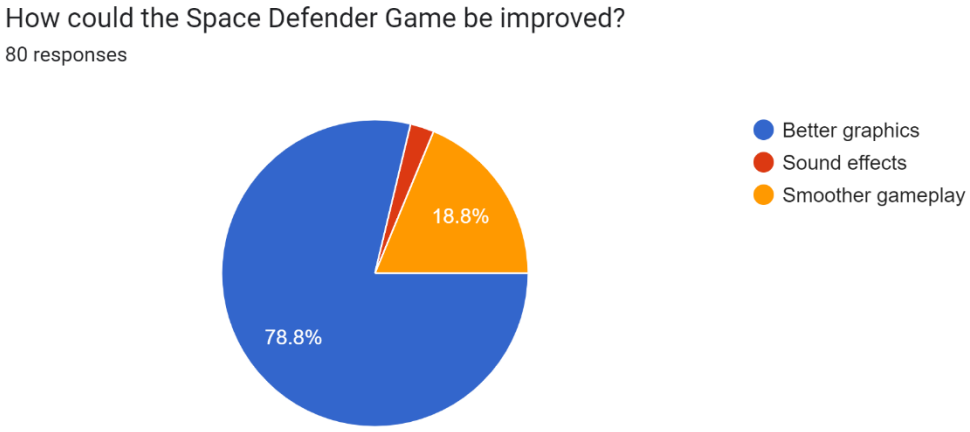


Figure 4.9: The percentage of improvement suggestions for Space Defender game

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.

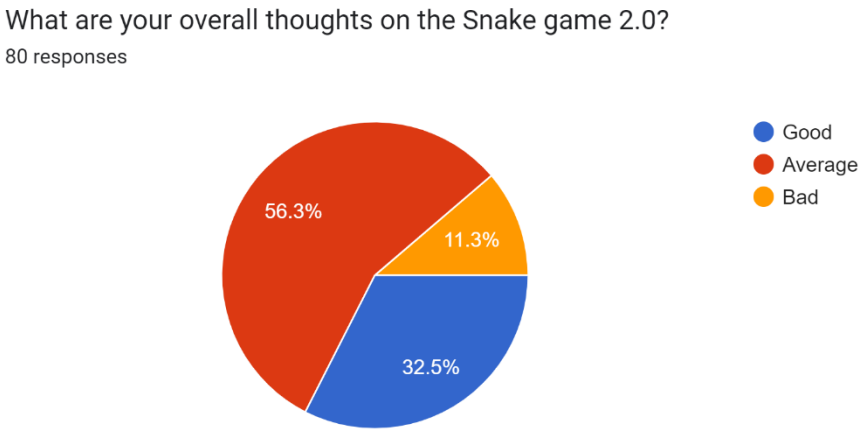


Figure 4.10: The percentage of overall thought on Snake game 2.0

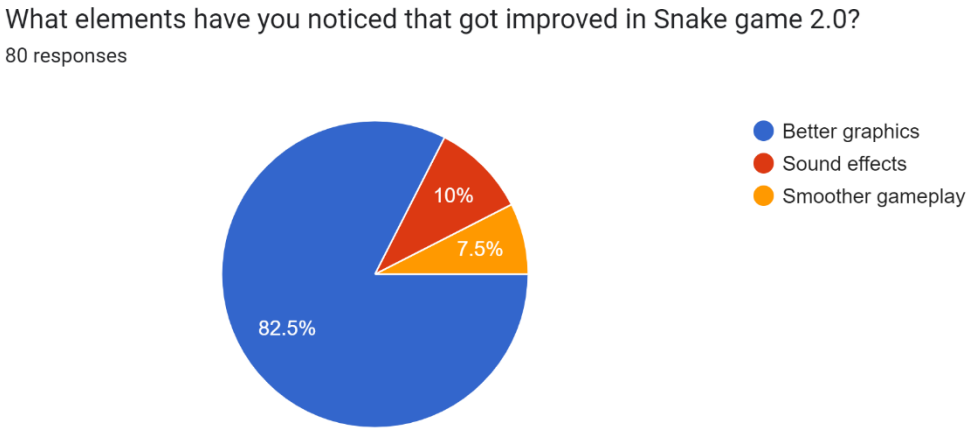


Figure 4.11: The percentage of improvement noticed in Snake game 2.0

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.

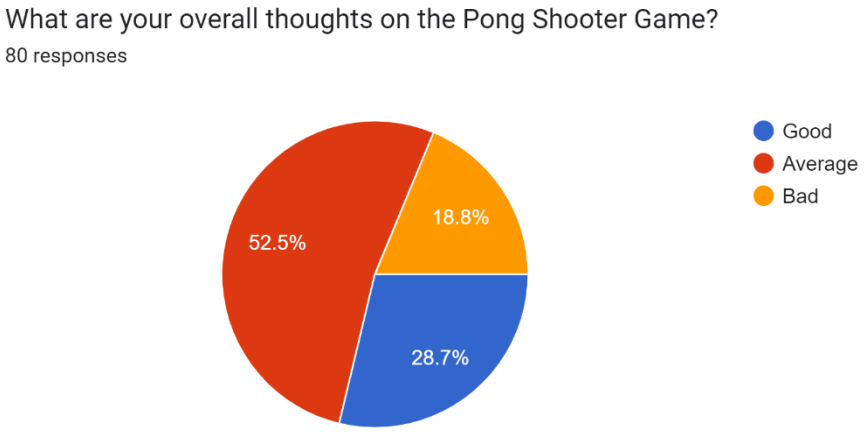


Figure 4.12: The percentage of overall thought on Pong Shooter game

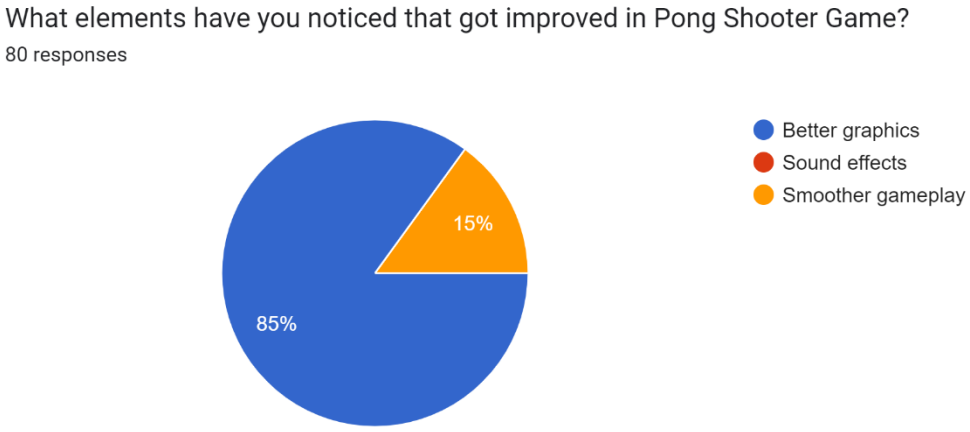


Figure 4.13: The percentage of improvement noticed in Pong Shooter game

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.

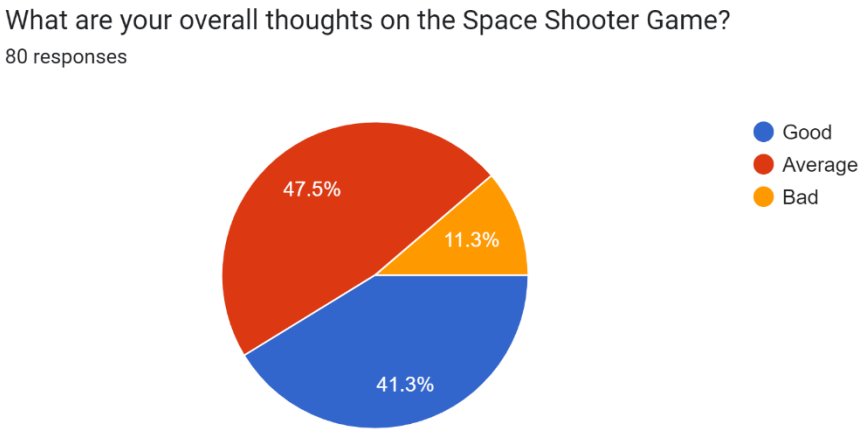


Figure 4.14: The percentage of overall thought on Space Shooter game

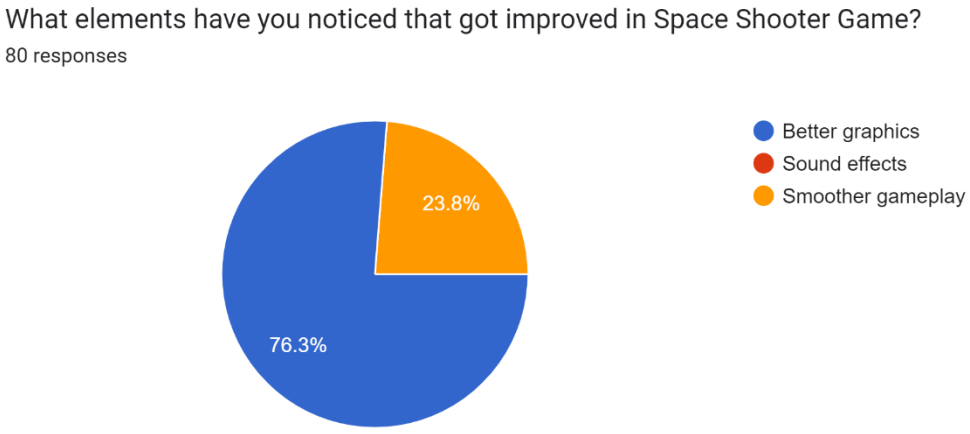


Figure 4.15: The percentage of improvement noticed in Space Shooter game

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).

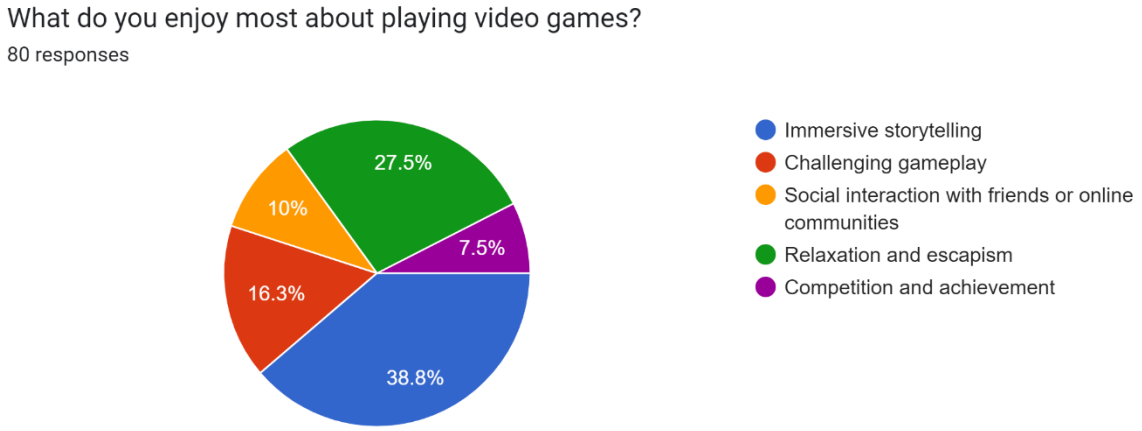


Figure 4.16: The percentage of participants answers regarding overall game enjoyment factors

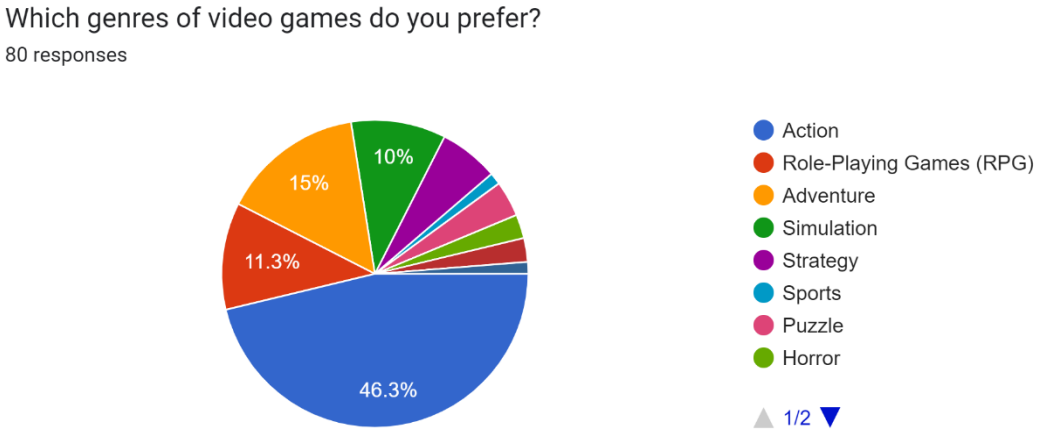


Figure 4.17: The percentage of participants answers regarding preferred game genres

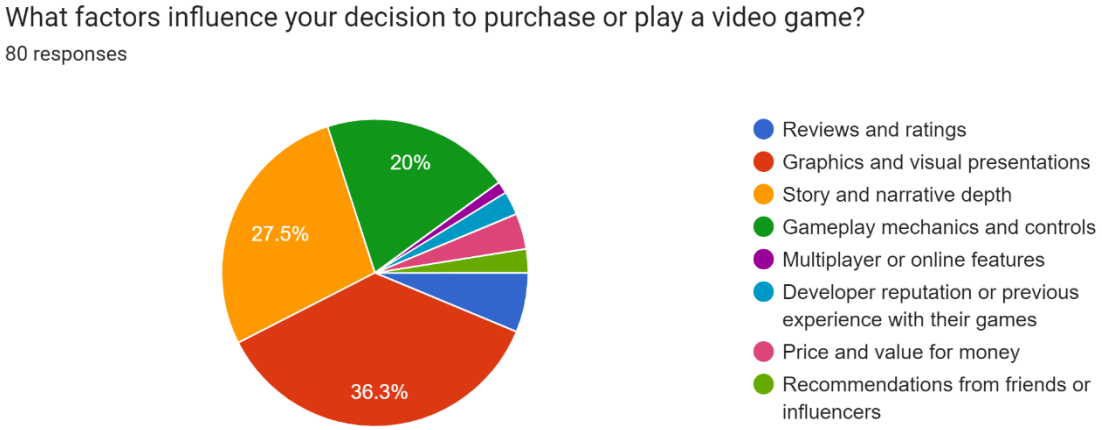


Figure 4.18: The percentage of participants answers on the influence for playing/buying a game

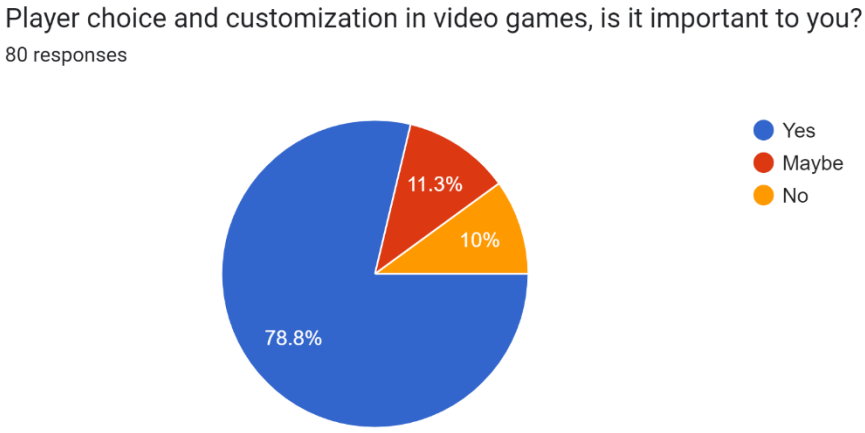


Figure 4.19: The percentage of participants answers regarding the importance of customization in video games

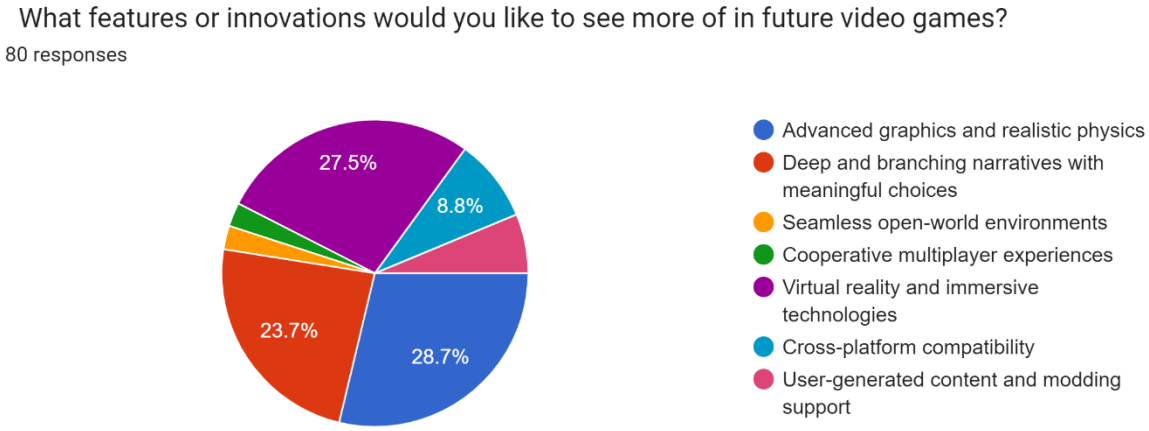


Figure 4.20: The percentage of participants answers regarding preferred innovations in future games

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 & Norbisrath, 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 strategic plannings(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 cand 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

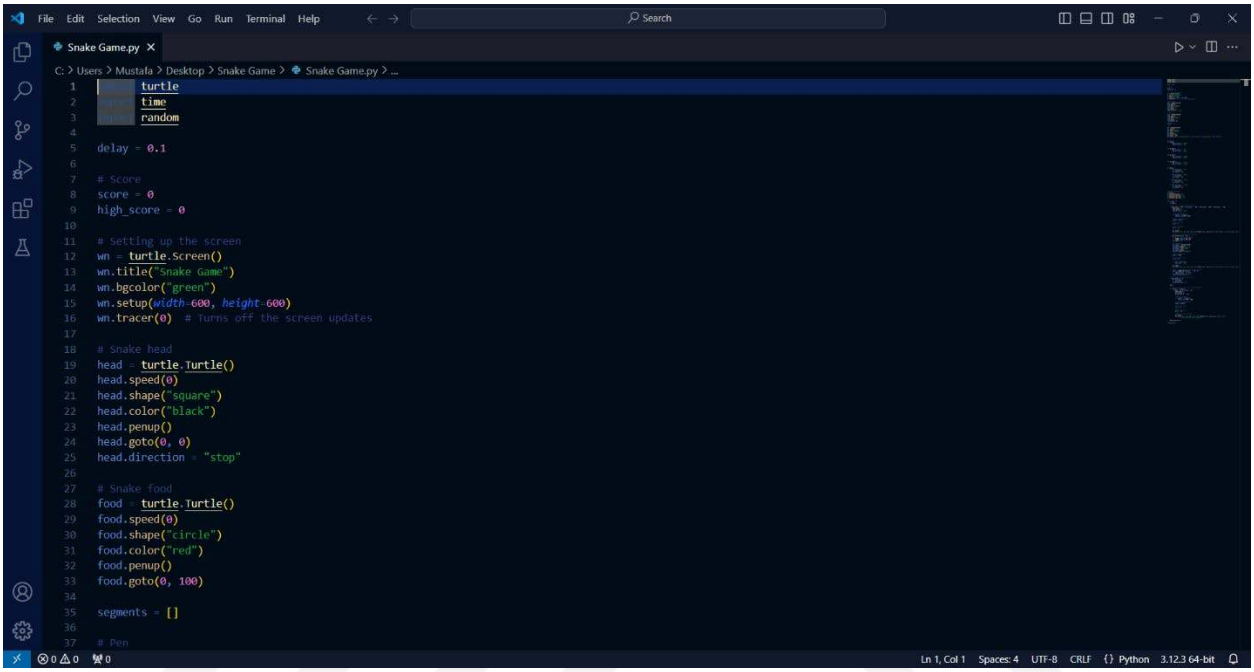
In the end of this study, it is certain that advanced video game programming is a rich tapestry of interwoven elements that creates excellent depth in impact on players. The overwhelming number of themes emerging across the landscape of this far-reaching analysis into industry trends, technological advancement, 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.

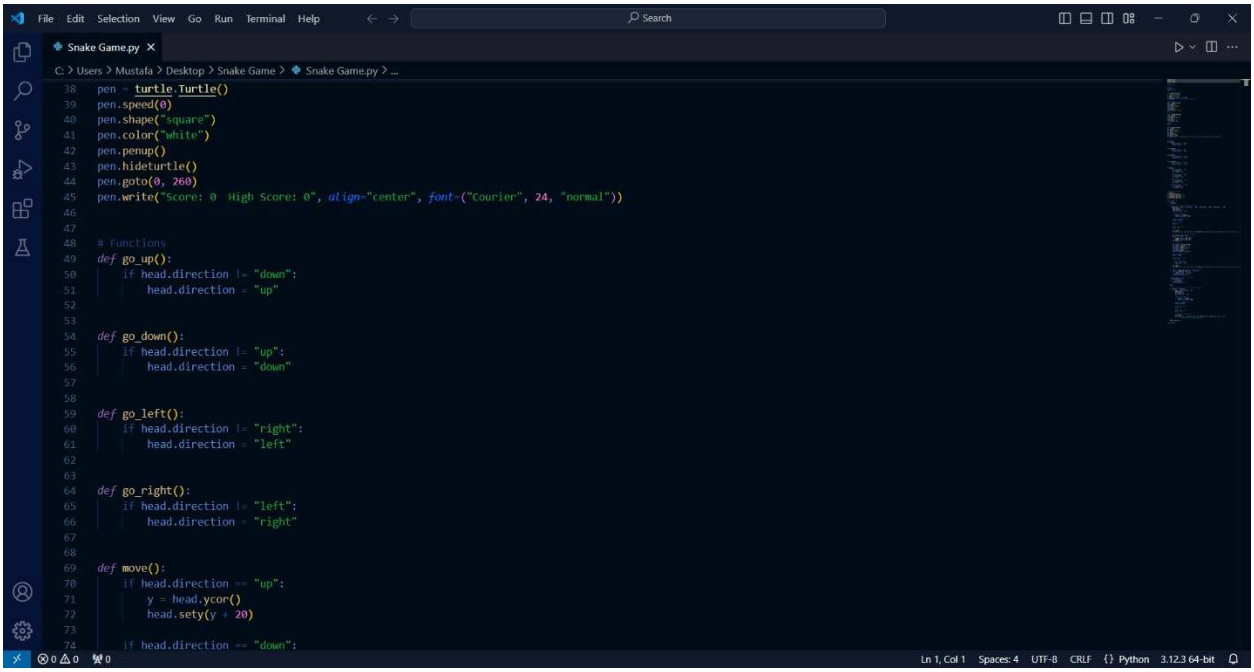
But amidst all the enthusiasm for this technological advancement, the developers should not overlook the ethical factors that should be considered during game development. The developers need to be responsible so that their player base receives their share of enjoyment, and nothing at any end deprives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

7. APPENDIX



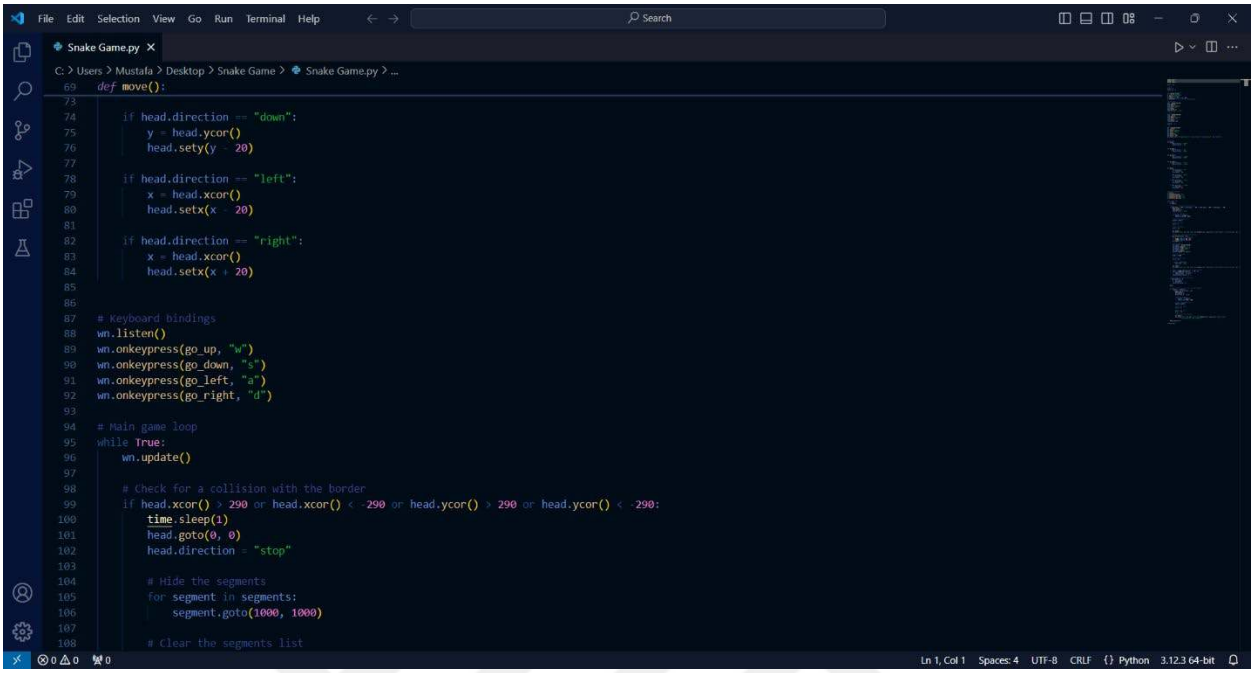
```
import turtle
import time
import random

delay = 0.1

# Score
score = 0
high_score = 0

# Setting up the screen
wn = turtle.Screen()
wn.title("Snake Game")
wn.bgcolor("green")
wn.setup(width=600, height=600)
wn.tracer(0)  # Turns off the screen updates

# Snake head
head = turtle.Turtle()
head.speed(0)
head.shape("square")
head.color("black")
head.penup()
head.goto(0, 0)
head.direction = "stop"

# Snake food
food = turtle.Turtle()
food.speed(0)
food.shape("circle")
food.color("red")
food.penup()
food.goto(0, 100)

segments = []

# Pen
```

Figure 7.1: Code of Snake game in Python



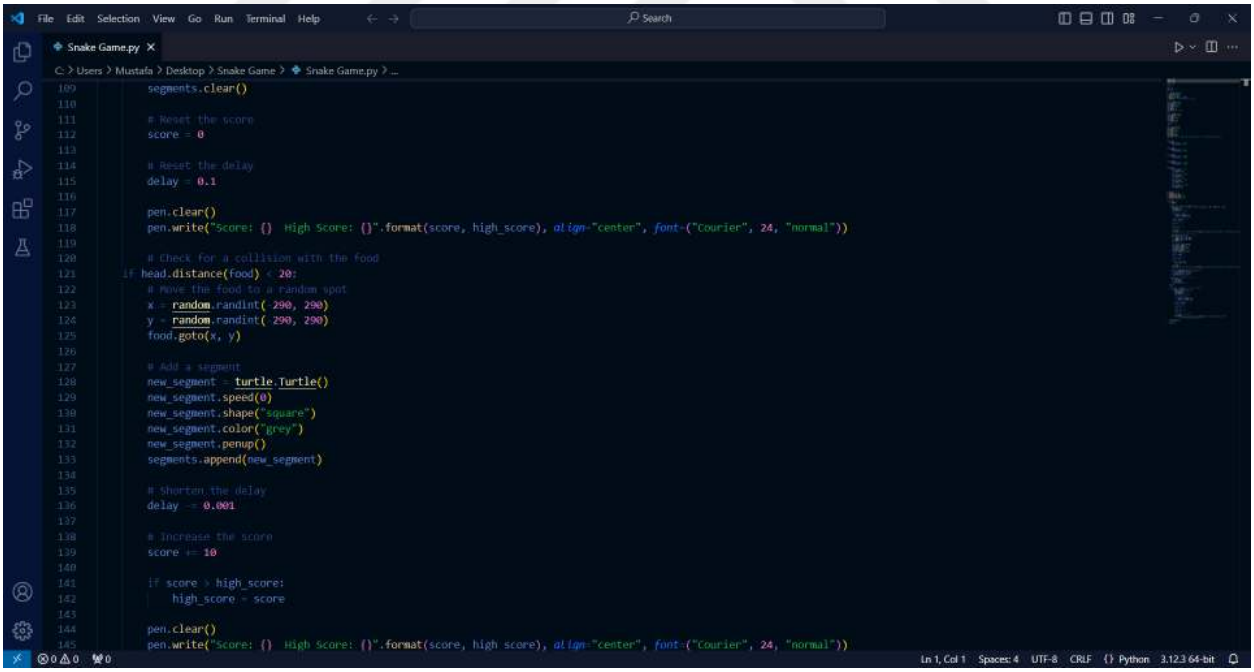
```
pen = turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 260)
pen.write("Score: 0  High Score: 0", align="center", font=("Courier", 24, "normal"))


# Functions
def go_up():
    if head.direction != "down":
        head.direction = "up"


def go_down():
    if head.direction != "up":
        head.direction = "down"


def go_left():
    if head.direction != "right":
        head.direction = "left"


def go_right():
    if head.direction != "left":
        head.direction = "right"


def move():
    if head.direction == "up":
        y = head.ycor()
        head.sety(y + 20)

    if head.direction == "down":
```

Figure 7.2: Code of Snake game in Python



```
def move():

    if head.direction == "down":
        y = head.ycor()
        head.sety(y - 20)

    if head.direction == "left":
        x = head.xcor()
        head.setx(x - 20)

    if head.direction == "right":
        x = head.xcor()
        head.setx(x + 20)


# Keyboard bindings
wn.listen()
wn.onkeypress(go_up, "w")
wn.onkeypress(go_down, "s")
wn.onkeypress(go_left, "a")
wn.onkeypress(go_right, "d")

# Main game loop
while True:
    wn.update()

    # Check for a collision with the border
    if head.xcor() > 290 or head.xcor() < -290 or head.ycor() > 290 or head.ycor() < -290:
        time.sleep(1)
        head.goto(0, 0)
        head.direction = "stop"

        # Hide the segments
        for segment in segments:
            segment.goto(1000, 1000)

        # Clear the segments list
```

Figure 7.3: Code of Snake game in Python



```
        segments.clear()

        # Reset the score
        score = 0

        # Reset the delay
        delay = 0.1

        pen.clear()
        pen.write("Score: {}  High Score: {}".format(score, high_score), align="center", font=("Courier", 24, "normal"))

    # Check for a collision with the food
    if head.distance(food) < 20:
        # Move the food to a random spot
        x = random.randint(-290, 290)
        y = random.randint(-290, 290)
        food.goto(x, y)

        # Add a segment
        new_segment = turtle.Turtle()
        new_segment.speed(0)
        new_segment.shape("square")
        new_segment.color("grey")
        new_segment.penup()
        segments.append(new_segment)

        # Shorten the delay
        delay -= 0.001

        # Increase the score
        score += 10

        if score > high_score:
            high_score = score

        pen.clear()
        pen.write("Score: {}  High Score: {}".format(score, high_score), align="center", font=("Courier", 24, "normal"))
```

Figure 7.4: Code of Snake game in Python

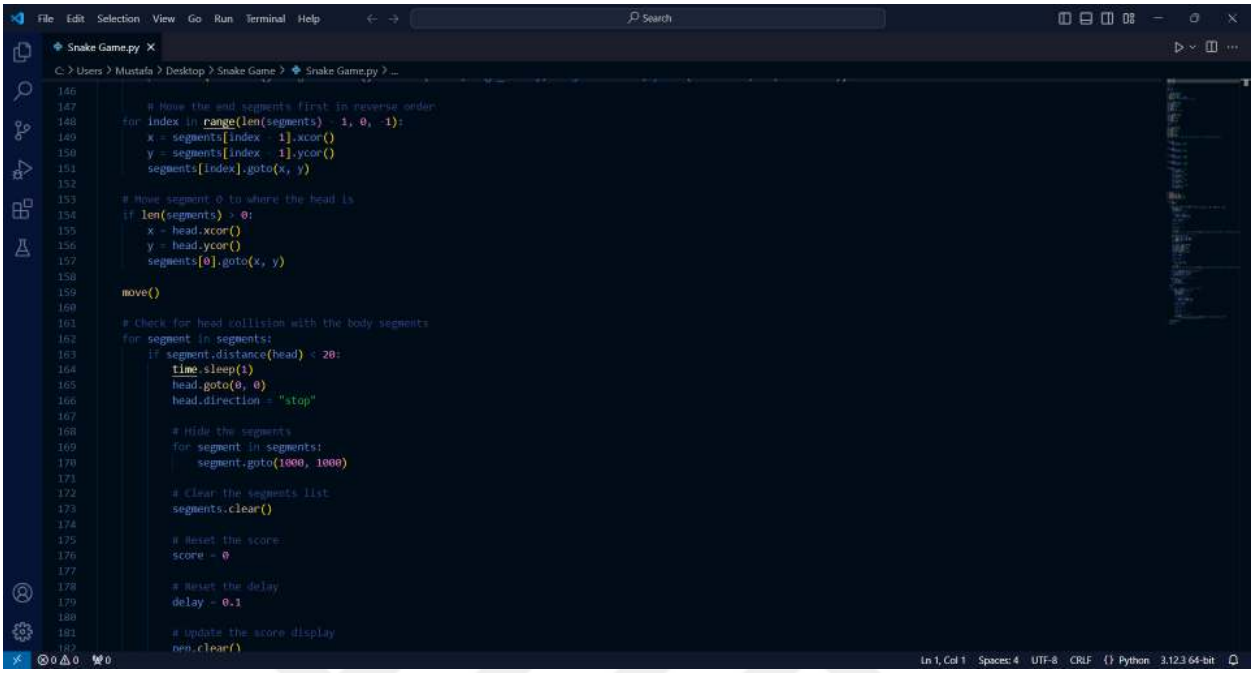


Figure 7.5: Code of Snake game in Python

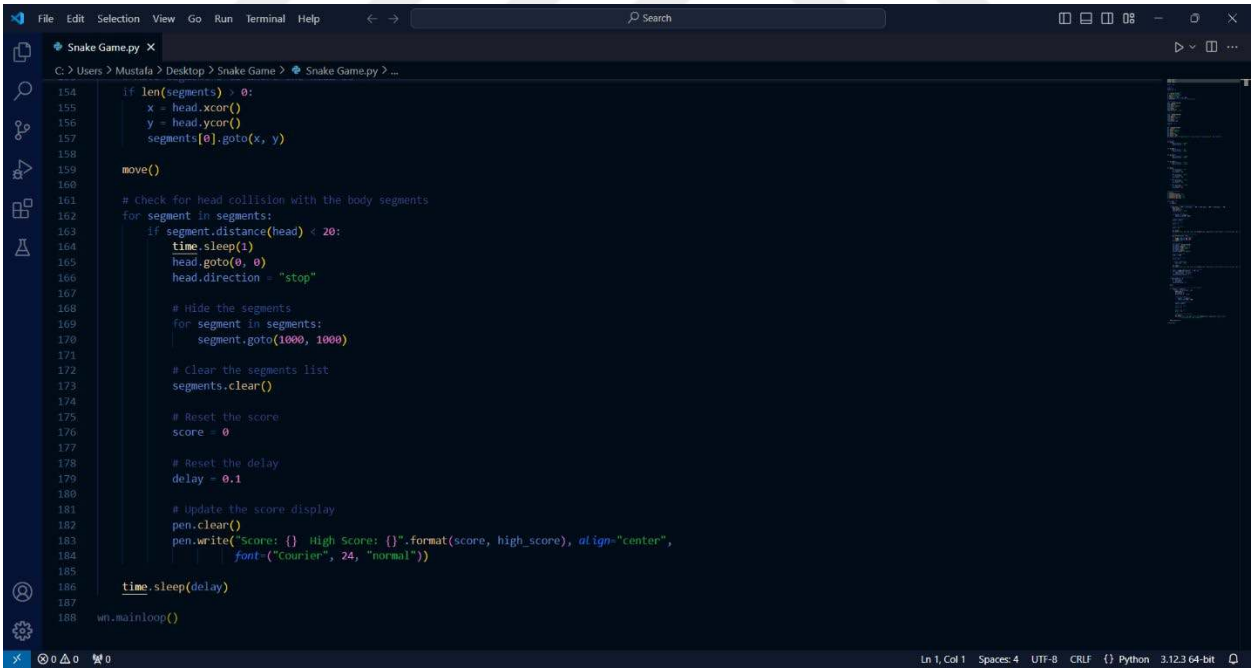
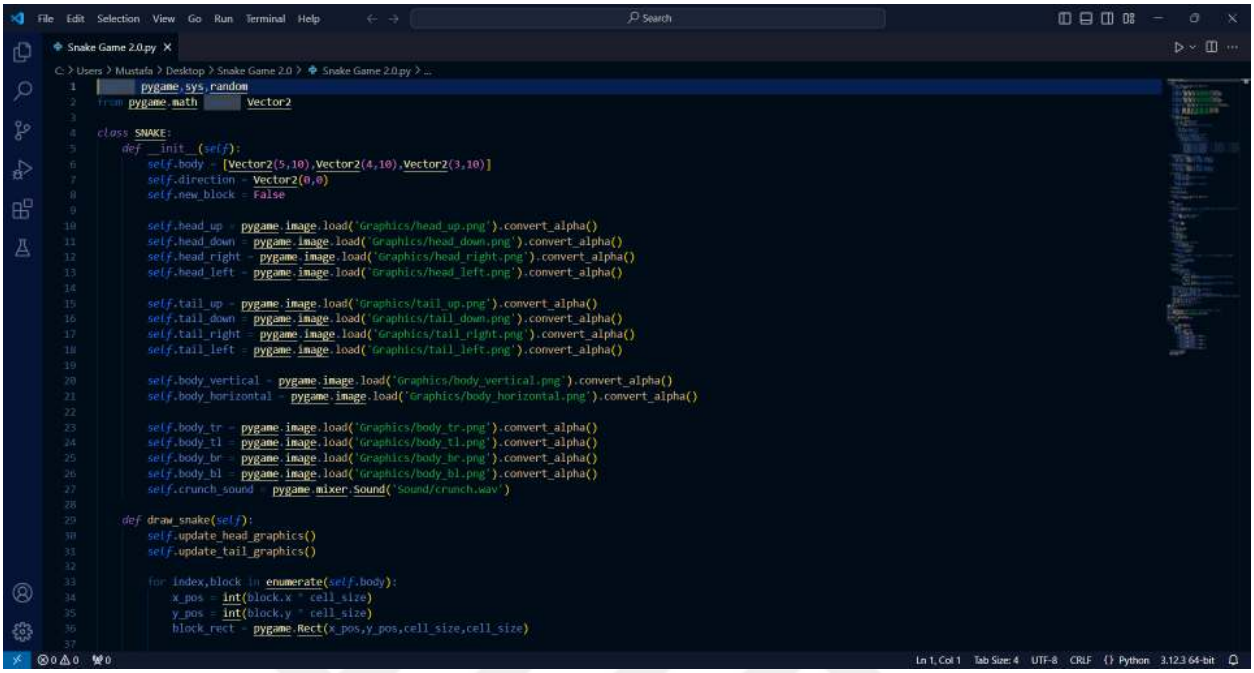
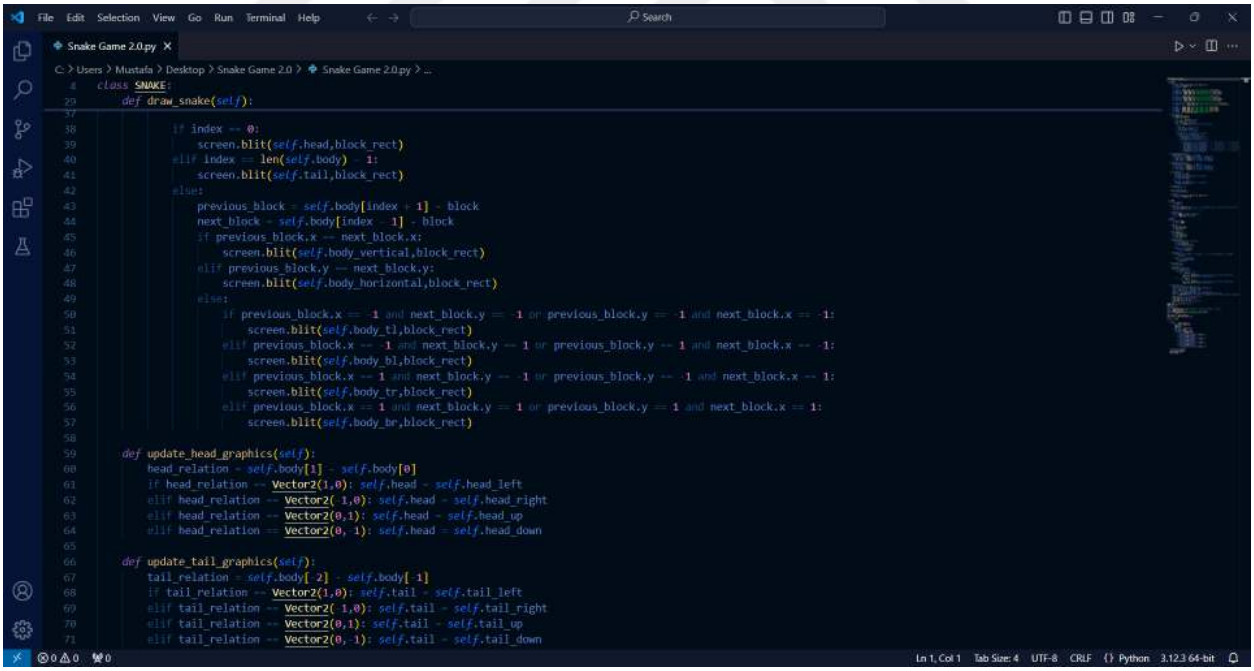


Figure 7.6: Code of Snake game in Python



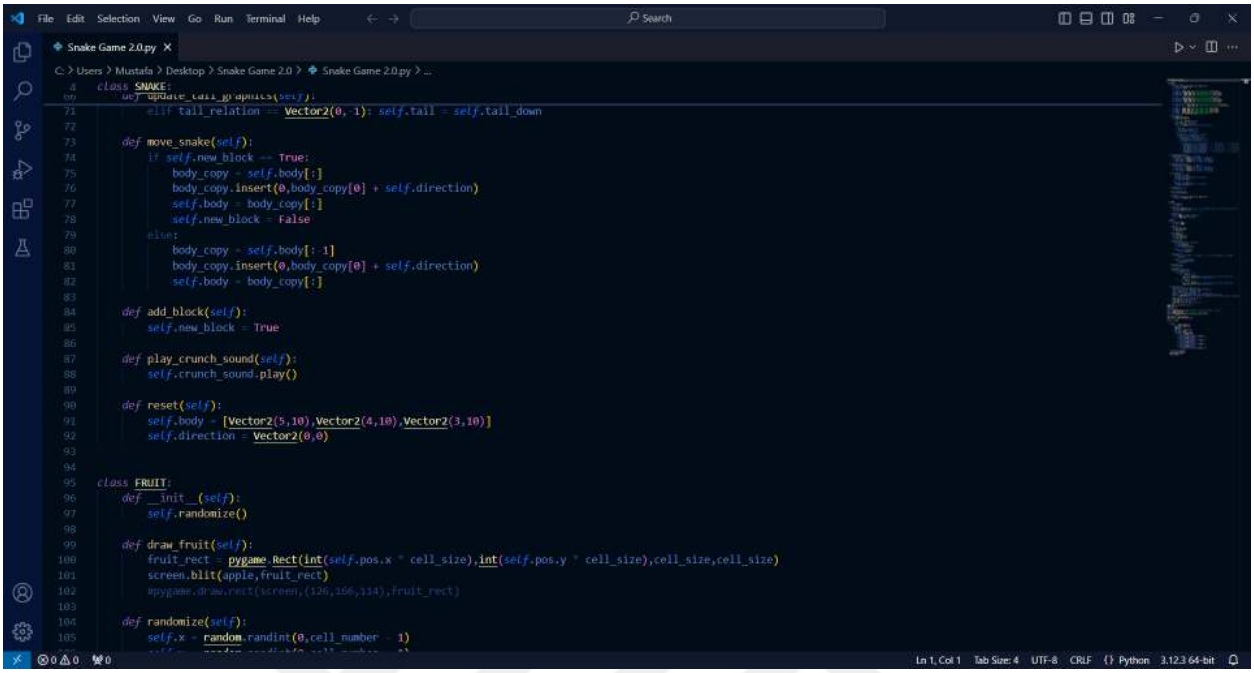
```
import pygame,sys,random
from pygame.math import Vector2

class SNAKE:
    def __init__(self):
        self.body = [Vector2(5,10),Vector2(4,10),Vector2(3,10)]
        self.direction = Vector2(0,0)
        self.new_block = False

        self.head_up = pygame.image.load('Graphics/head_up.png').convert_alpha()
        self.head_down = pygame.image.load('Graphics/head_down.png').convert_alpha()
        self.head_right = pygame.image.load('Graphics/head_right.png').convert_alpha()
        self.head_left = pygame.image.load('Graphics/head_left.png').convert_alpha()

        self.tail_up = pygame.image.load('Graphics/tail_up.png').convert_alpha()
        self.tail_down = pygame.image.load('Graphics/tail_down.png').convert_alpha()
        self.tail_right = pygame.image.load('Graphics/tail_right.png').convert_alpha()
        self.tail_left = pygame.image.load('Graphics/tail_left.png').convert_alpha()

        self.body_vertical = pygame.image.load('Graphics/body_vertical.png').convert_alpha()
        self.body_horizontal = pygame.image.load('Graphics/body_horizontal.png').convert_alpha()

        self.body_tr = pygame.image.load('Graphics/body_tr.png').convert_alpha()
        self.body_tl = pygame.image.load('Graphics/body_tl.png').convert_alpha()
        self.body_br = pygame.image.load('Graphics/body_br.png').convert_alpha()
        self.body_bl = pygame.image.load('Graphics/body_bl.png').convert_alpha()
        self.crunch_sound = pygame.mixer.Sound('Sound/crunch.wav')

    def draw_snake(self):
        self.update_head_graphics()
        self.update_tail_graphics()

        for index,block in enumerate(self.body):
            x_pos = int(block.x * cell_size)
            y_pos = int(block.y * cell_size)
            block_rect = pygame.Rect(x_pos,y_pos,cell_size,cell_size)
```

Figure 7.7: Code of Snake game 2.0 in Python



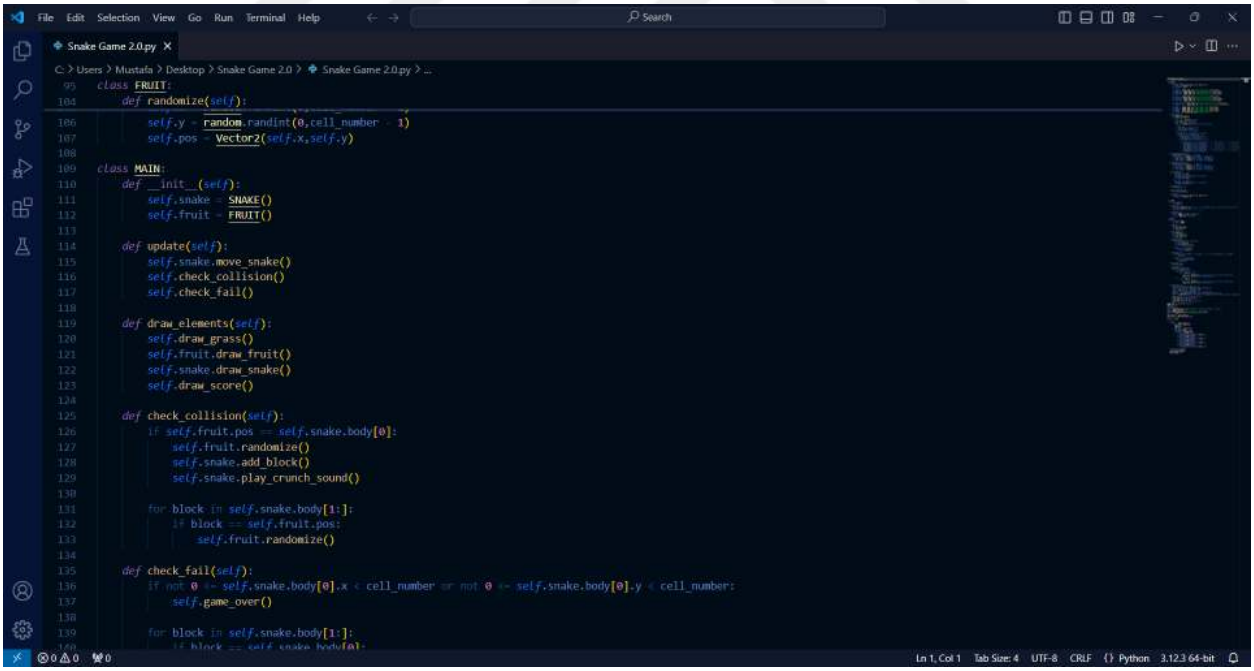
```
class SNAKE:
    def draw_snake(self):

            if index == 0:
                screen.blit(self.head,block_rect)
            elif index == len(self.body) - 1:
                screen.blit(self.tail,block_rect)
            else:
                previous_block = self.body[index + 1] - block
                next_block = self.body[index - 1] - block
                if previous_block.x == next_block.x:
                    screen.blit(self.body_vertical,block_rect)
                elif previous_block.y == next_block.y:
                    screen.blit(self.body_horizontal,block_rect)
                else:
                    if previous_block.x == -1 and next_block.y == -1 or previous_block.y == -1 and next_block.x == -1:
                        screen.blit(self.body_tl,block_rect)
                    elif previous_block.x == -1 and next_block.y == 1 or previous_block.y == 1 and next_block.x == -1:
                        screen.blit(self.body_bl,block_rect)
                    elif previous_block.x == 1 and next_block.y == -1 or previous_block.y == -1 and next_block.x == 1:
                        screen.blit(self.body_tr,block_rect)
                    elif previous_block.x == 1 and next_block.y == 1 or previous_block.y == 1 and next_block.x == 1:
                        screen.blit(self.body_br,block_rect)

    def update_head_graphics(self):
        head_relation = self.body[1] - self.body[0]
        if head_relation == Vector2(1,0): self.head = self.head_left
        elif head_relation == Vector2(-1,0): self.head = self.head_right
        elif head_relation == Vector2(0,1): self.head = self.head_up
        elif head_relation == Vector2(0,-1): self.head = self.head_down

    def update_tail_graphics(self):
        tail_relation = self.body[-2] - self.body[-1]
        if tail_relation == Vector2(1,0): self.tail = self.tail_left
        elif tail_relation == Vector2(-1,0): self.tail = self.tail_right
        elif tail_relation == Vector2(0,1): self.tail = self.tail_up
        elif tail_relation == Vector2(0,-1): self.tail = self.tail_down
```

Figure 7.8: Code of Snake game 2.0 in Python



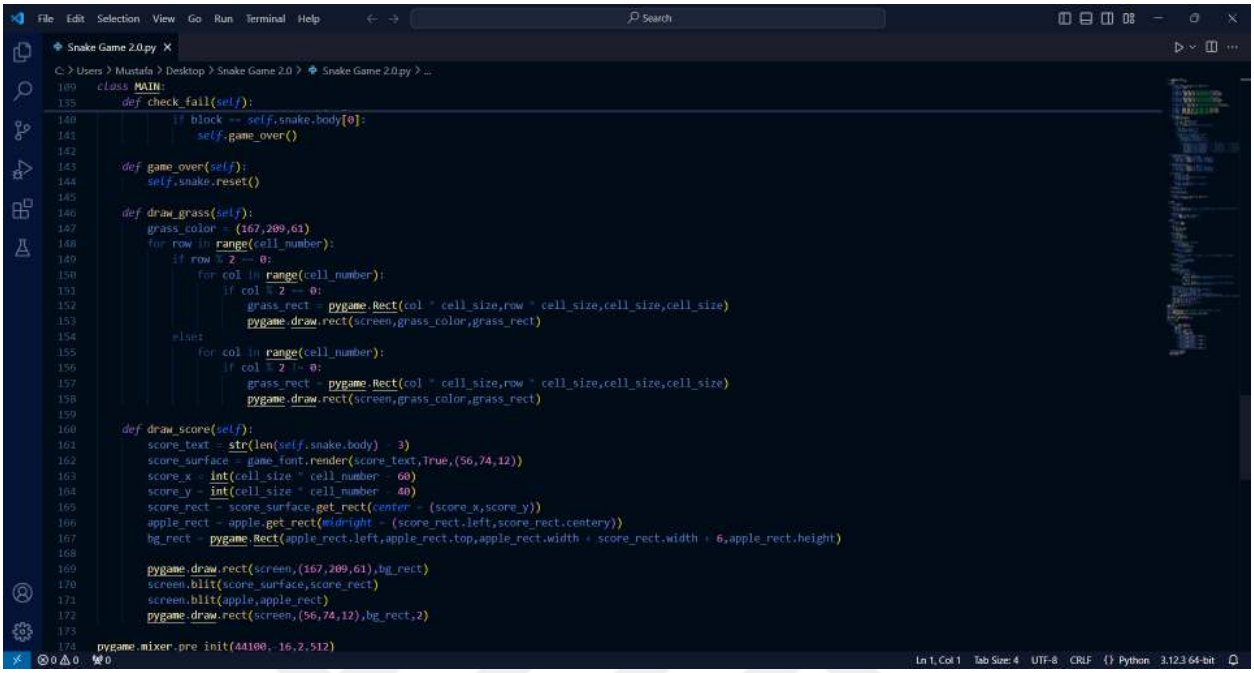
```
class SNAKE:
    def update_tail_graphics(self):
        elif tail_relation == Vector2(0,-1): self.tail = self.tail_down

    def move_snake(self):
        if self.new_block == True:
            body_copy = self.body[:]
            body_copy.insert(0,body_copy[0] + self.direction)
            self.body = body_copy[:]
            self.new_block = False
        else:
            body_copy = self.body[:-1]
            body_copy.insert(0,body_copy[0] + self.direction)
            self.body = body_copy[:]

    def add_block(self):
        self.new_block = True

    def play_crunch_sound(self):
        self.crunch_sound.play()

    def reset(self):
        self.body = [Vector2(5,10),Vector2(4,10),Vector2(3,10)]
        self.direction = Vector2(0,0)


class FRUIT:
    def __init__(self):
        self.randomize()

    def draw_fruit(self):
        fruit_rect = pygame.Rect(int(self.pos.x * cell_size),int(self.pos.y * cell_size),cell_size,cell_size)
        screen.blit(apple,fruit_rect)
        #pygame.draw.rect(screen,(126,166,114),fruit_rect)

    def randomize(self):
        self.x = random.randint(0,cell_number - 1)
```

Figure 7.9: Code of Snake game 2.0 in Python



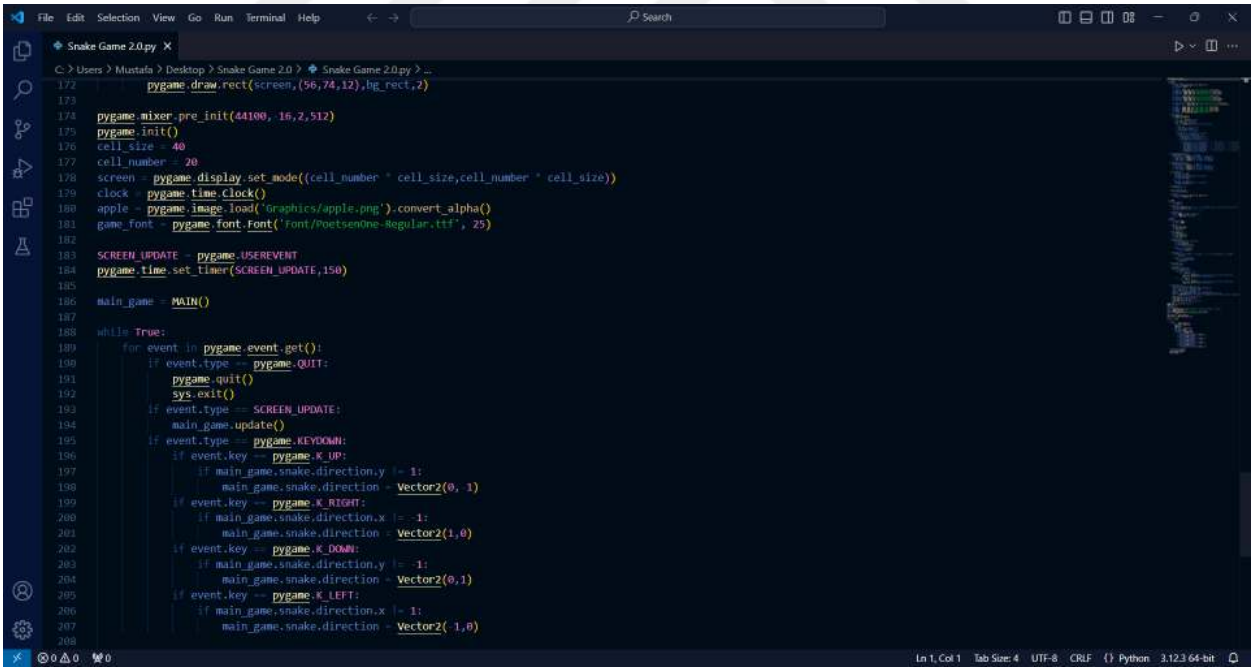
```
class FRUIT:
    def randomize(self):
        self.y = random.randint(0,cell_number - 1)
        self.pos = Vector2(self.x,self.y)

class MAIN:
    def __init__(self):
        self.snake = SNAKE()
        self.fruit = FRUIT()

    def update(self):
        self.snake.move_snake()
        self.check_collision()
        self.check_fail()

    def draw_elements(self):
        self.draw_grass()
        self.fruit.draw_fruit()
        self.snake.draw_snake()
        self.draw_score()

    def check_collision(self):
        if self.fruit.pos == self.snake.body[0]:
            self.fruit.randomize()
            self.snake.add_block()
            self.snake.play_crunch_sound()

        for block in self.snake.body[1:]:
            if block == self.fruit.pos:
                self.fruit.randomize()

    def check_fail(self):
        if not 0 <= self.snake.body[0].x < cell_number or not 0 <= self.snake.body[0].y < cell_number:
            self.game_over()

        for block in self.snake.body[1:]:
```

Figure 7.10: Code of Snake game 2.0 in Python



```
class MAIN:
    def check_fail(self):
            if block == self.snake.body[0]:
                self.game_over()

    def game_over(self):
        self.snake.reset()

    def draw_grass(self):
        grass_color = (167,209,61)
        for row in range(cell_number):
            if row % 2 == 0:
                for col in range(cell_number):
                    if col % 2 == 0:
                        grass_rect = pygame.Rect(col * cell_size,row * cell_size,cell_size,cell_size)
                        pygame.draw.rect(screen,grass_color,grass_rect)
            else:
                for col in range(cell_number):
                    if col % 2 != 0:
                        grass_rect = pygame.Rect(col * cell_size,row * cell_size,cell_size,cell_size)
                        pygame.draw.rect(screen,grass_color,grass_rect)

    def draw_score(self):
        score_text = str(len(self.snake.body) - 3)
        score_surface = game_font.render(score_text,True,(56,74,12))
        score_x = int(cell_size * cell_number - 60)
        score_y = int(cell_size * cell_number - 40)
        score_rect = score_surface.get_rect(center = (score_x,score_y))
        apple_rect = apple.get_rect(midright = (score_rect.left,score_rect.centery))
        bg_rect = pygame.Rect(apple_rect.left,apple_rect.top,apple_rect.width + score_rect.width + 6,apple_rect.height)

        pygame.draw.rect(screen,(167,209,61),bg_rect)
        screen.blit(score_surface,score_rect)
        screen.blit(apple,apple_rect)
        pygame.draw.rect(screen,(56,74,12),bg_rect,2)

pygame.mixer.pre_init(44100,-16,2,512)
```

Figure 7.11: Code of Snake game 2.0 in Python



```
        pygame.draw.rect(screen,(56,74,12),bg_rect,2)

pygame.mixer.pre_init(44100, -16,2,512)
pygame.init()
cell_size = 40
cell_number = 20
screen = pygame.display.set_mode((cell_number * cell_size,cell_number * cell_size))
clock = pygame.time.Clock()
apple = pygame.image.load('Graphics/apple.png').convert_alpha()
game_font = pygame.font.Font('Font/PoetsenOne-Regular.ttf', 25)

SCREEN_UPDATE = pygame.USEREVENT
pygame.time.set_timer(SCREEN_UPDATE,150)

main_game = MAIN()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == SCREEN_UPDATE:
            main_game.update()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                if main_game.snake.direction.y != 1:
                    main_game.snake.direction = Vector2(0, -1)
            if event.key == pygame.K_RIGHT:
                if main_game.snake.direction.x != -1:
                    main_game.snake.direction = Vector2(1,0)
            if event.key == pygame.K_DOWN:
                if main_game.snake.direction.y != -1:
                    main_game.snake.direction = Vector2(0,1)
            if event.key == pygame.K_LEFT:
                if main_game.snake.direction.x != 1:
                    main_game.snake.direction = Vector2(-1,0)
```

Figure 7.12: Code of Snake game 2.0 in Python

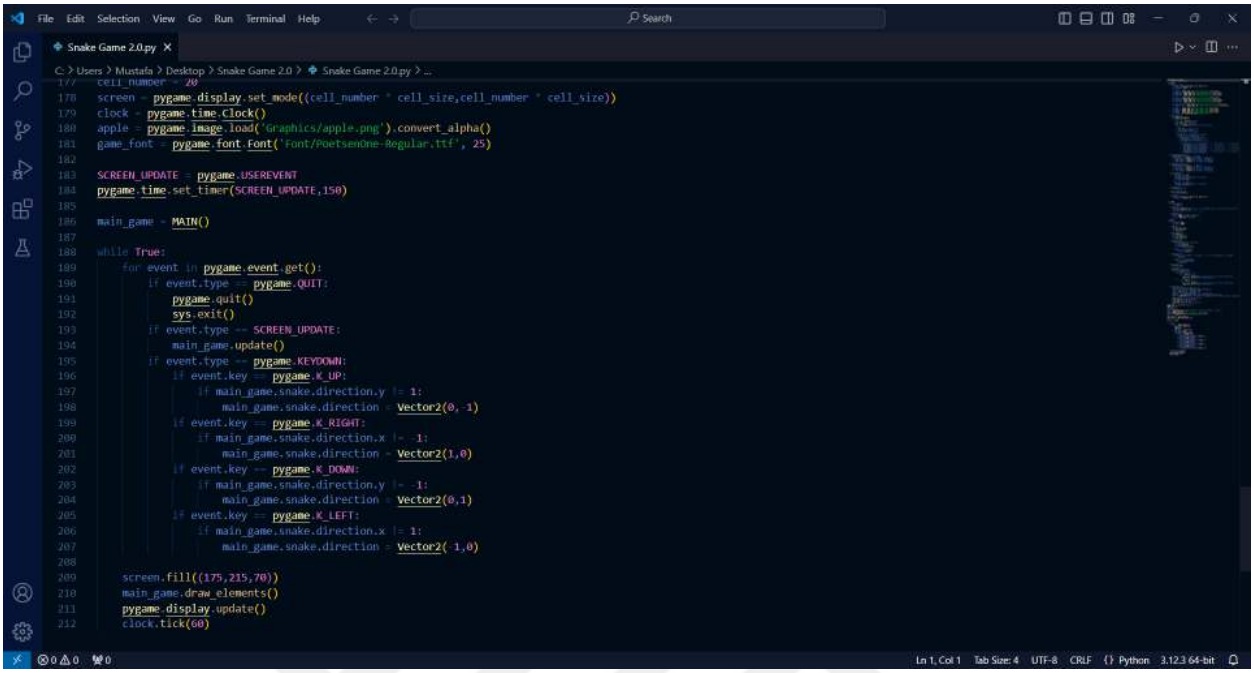


Figure 7.13: Code of Snake game 2.0 in Python

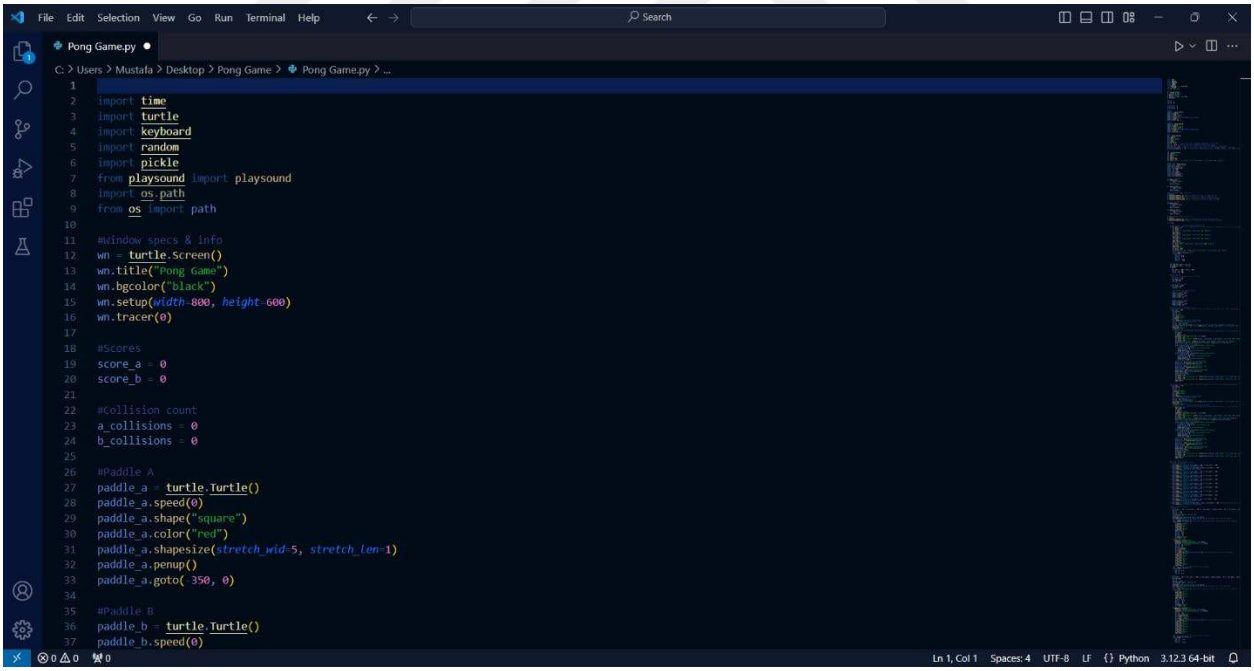
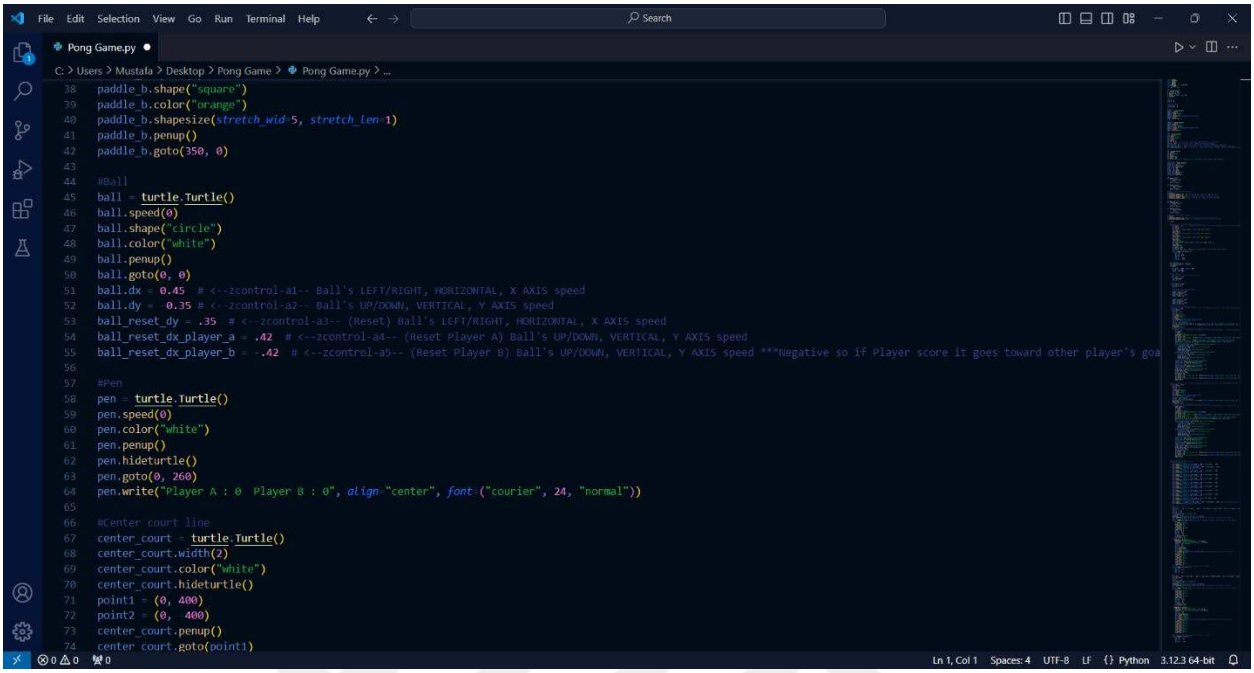
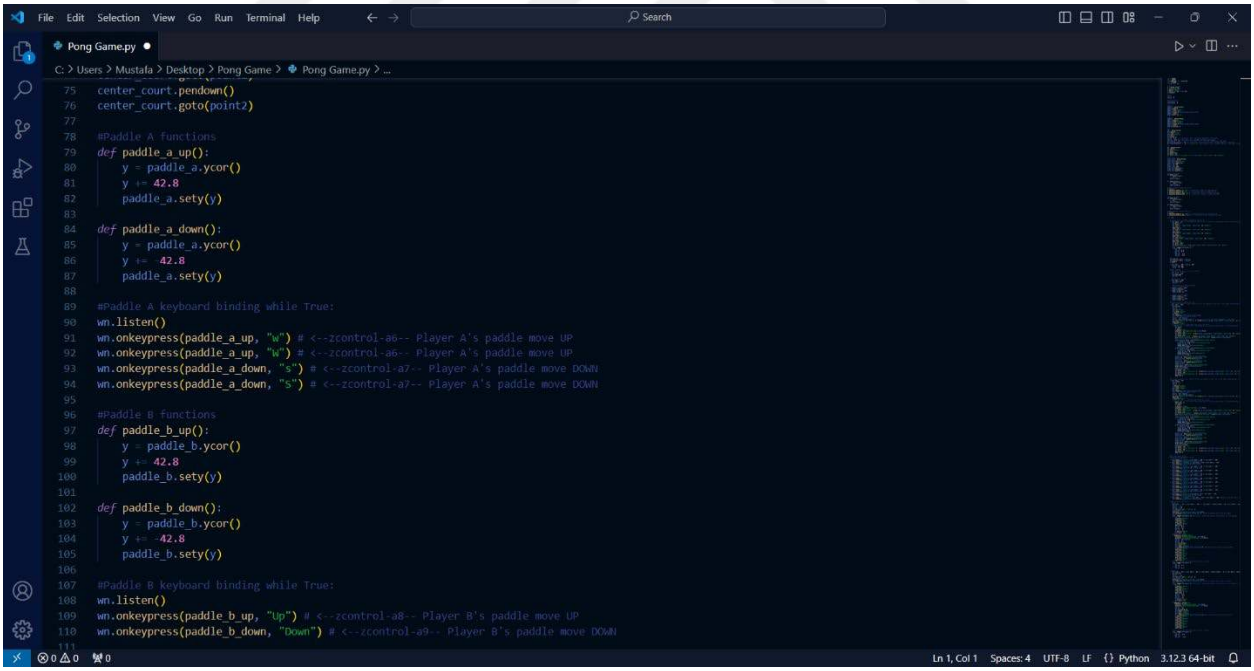


Figure 7.14: Code of Pong game in Python



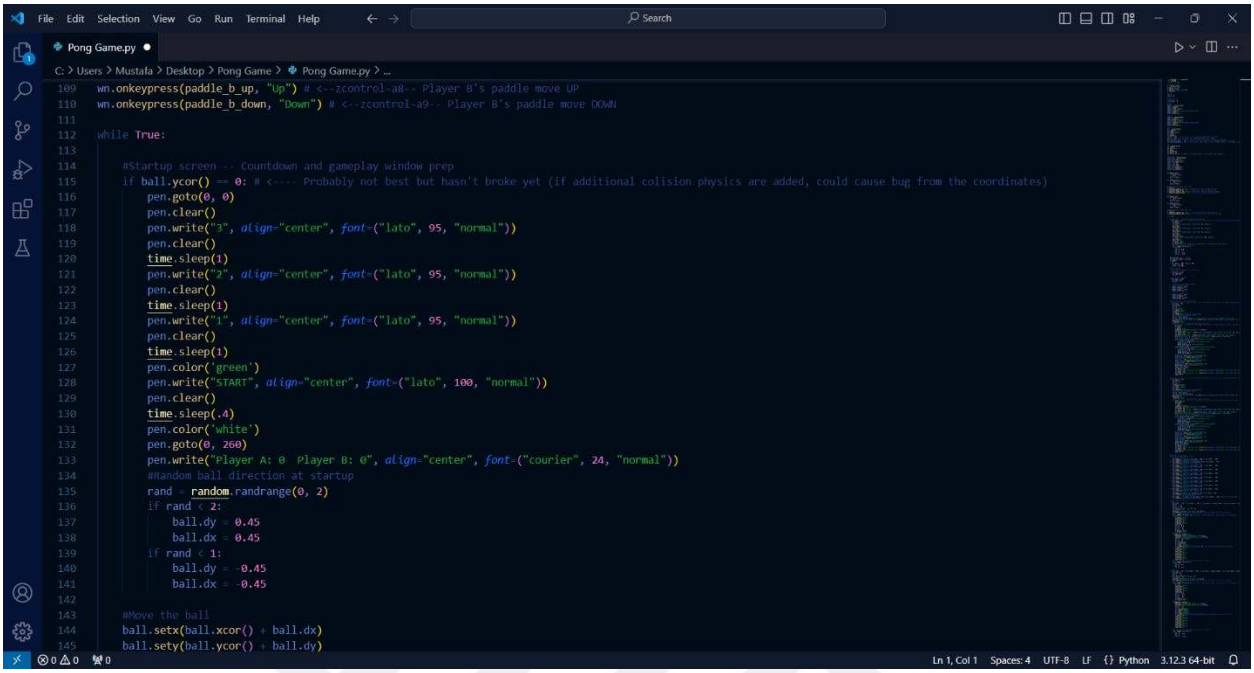
```
paddle_b.shape("square")
paddle_b.color("orange")
paddle_b.shapesize(stretch_wid=5, stretch_len=1)
paddle_b.penup()
paddle_b.goto(350, 0)

#Ball
ball = turtle.Turtle()
ball.speed(0)
ball.shape("circle")
ball.color("white")
ball.penup()
ball.goto(0, 0)
ball.dx = 0.45  # <--zcontrol-a1-- Ball's LEFT/RIGHT, HORIZONTAL, X AXIS speed
ball.dy = -0.35 # <--zcontrol-a2-- Ball's UP/DOWN, VERTICAL, Y AXIS speed
ball_reset_dy = .35  # <--zcontrol-a3-- (Reset) Ball's LEFT/RIGHT, HORIZONTAL, X AXIS speed
ball_reset_dx_player_a = .42  # <--zcontrol-a4-- (Reset Player A) Ball's UP/DOWN, VERTICAL, Y AXIS speed
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 goa

#Pen
pen = turtle.Turtle()
pen.speed(0)
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 260)
pen.write("Player A : 0  Player B : 0", align="center", font=("courier", 24, "normal"))

#Center court line
center_court = turtle.Turtle()
center_court.width(2)
center_court.color("white")
center_court.hideturtle()
point1 = (0, 400)
point2 = (0, -400)
center_court.penup()
center_court.goto(point1)
```

Figure 7.15: Code of Pong game in Python



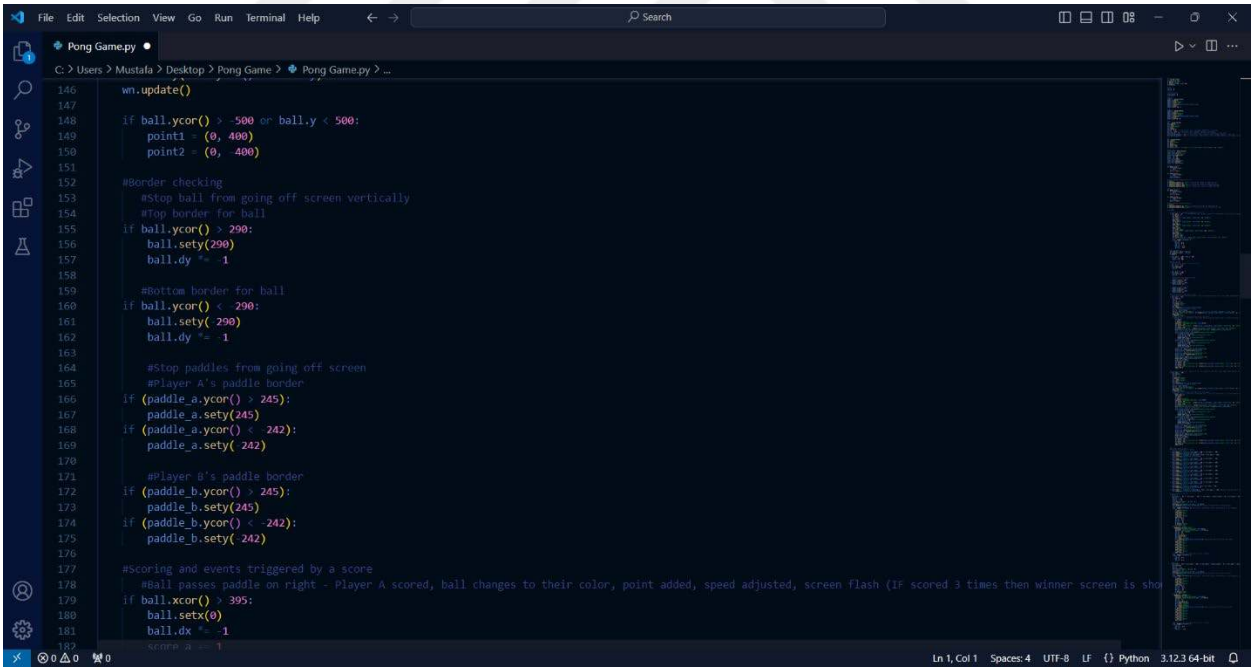
```
center_court.pendown()
center_court.goto(point2)

#Paddle A functions
def paddle_a_up():
    y = paddle_a.ycor()
    y += 42.8
    paddle_a.sety(y)

def paddle_a_down():
    y = paddle_a.ycor()
    y += -42.8
    paddle_a.sety(y)

#Paddle A keyboard binding while True:
wn.listen()
wn.onkeypress(paddle_a_up, "w") # <--zcontrol-a6-- Player A's paddle move UP
wn.onkeypress(paddle_a_up, "W") # <--zcontrol-a6-- Player A's paddle move UP
wn.onkeypress(paddle_a_down, "s") # <--zcontrol-a7-- Player A's paddle move DOWN
wn.onkeypress(paddle_a_down, "S") # <--zcontrol-a7-- Player A's paddle move DOWN

#Paddle B functions
def paddle_b_up():
    y = paddle_b.ycor()
    y += 42.8
    paddle_b.sety(y)

def paddle_b_down():
    y = paddle_b.ycor()
    y += -42.8
    paddle_b.sety(y)

#Paddle B keyboard binding while True:
wn.listen()
wn.onkeypress(paddle_b_up, "Up") # <--zcontrol-a8-- Player B's paddle move UP
wn.onkeypress(paddle_b_down, "Down") # <--zcontrol-a9-- Player B's paddle move DOWN
```

Figure 7.16: Code of Pong game in Python



```
wn.onkeypress(paddle_b_up, "Up") # <--zcontrol-a8-- Player B's paddle move UP
wn.onkeypress(paddle_b_down, "Down") # <--zcontrol-a9-- Player B's paddle move DOWN

while True:

    #Startup screen -- Countdown and gameplay window prep
    if ball.ycor() == 0: # <---- Probably not best but hasn't broke yet (if additional colision physics are added, could cause bug from the coordinates)
        pen.goto(0, 0)
        pen.clear()
        pen.write("3", align="center", font=("lato", 95, "normal"))
        pen.clear()
        time.sleep(1)
        pen.write("2", align="center", font=("lato", 95, "normal"))
        pen.clear()
        time.sleep(1)
        pen.write("1", align="center", font=("lato", 95, "normal"))
        pen.clear()
        time.sleep(1)
        pen.color('green')
        pen.write("START", align="center", font=("lato", 100, "normal"))
        pen.clear()
        time.sleep(.4)
        pen.color('white')
        pen.goto(0, 260)
        pen.write("Player A: 0  Player B: 0", align="center", font=("courier", 24, "normal"))
        #Random ball direction at startup
        rand = random.randrange(0, 2)
        if rand < 2:
            ball.dy = 0.45
            ball.dx = 0.45
        if rand < 1:
            ball.dy = -0.45
            ball.dx = -0.45

    #Move the ball
    ball.setx(ball.xcor() + ball.dx)
    ball.sety(ball.ycor() + ball.dy)
```

Figure 7.17: Code of Pong game in Python



```
    wn.update()

    if ball.ycor() > 500 or ball.y < 500:
        point1 = (0, 400)
        point2 = (0,  400)

    #Border checking
        #Stop ball from going off screen vertically
        #Top border for ball
    if ball.ycor() > 290:
        ball.sety(290)
        ball.dy *= -1

        #Bottom border for ball
    if ball.ycor() < -290:
        ball.sety(-290)
        ball.dy *= -1

        #Stop paddles from going off screen
        #Player A's paddle border
    if (paddle_a.ycor() > 245):
        paddle_a.sety(245)
    if (paddle_a.ycor() < -242):
        paddle_a.sety(-242)

        #Player B's paddle border
    if (paddle_b.ycor() > 245):
        paddle_b.sety(245)
    if (paddle_b.ycor() < -242):
        paddle_b.sety(-242)

    #Scoring and events triggered by a score
        #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 sho
    if ball.xcor() > 395:
        ball.setx(0)
        ball.dx *= -1
        score_a += 1
```

Figure 7.18: Code of Pong game in Python

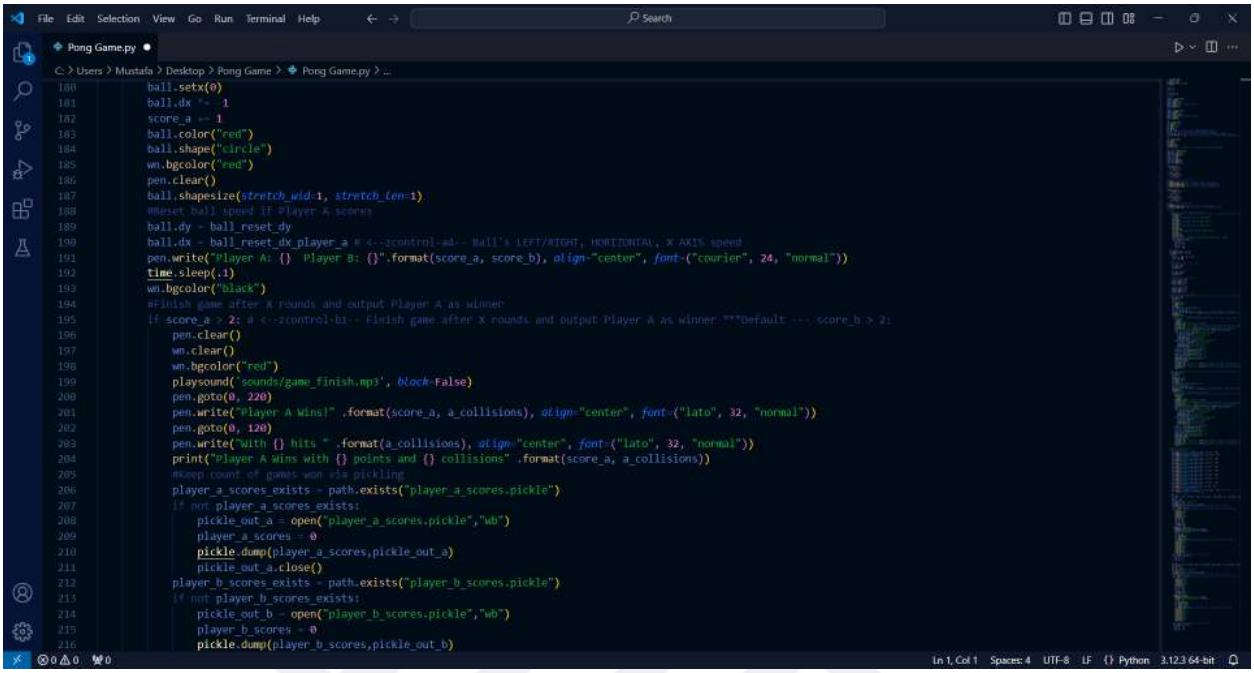


Figure 7.19: Code of Pong game in Python

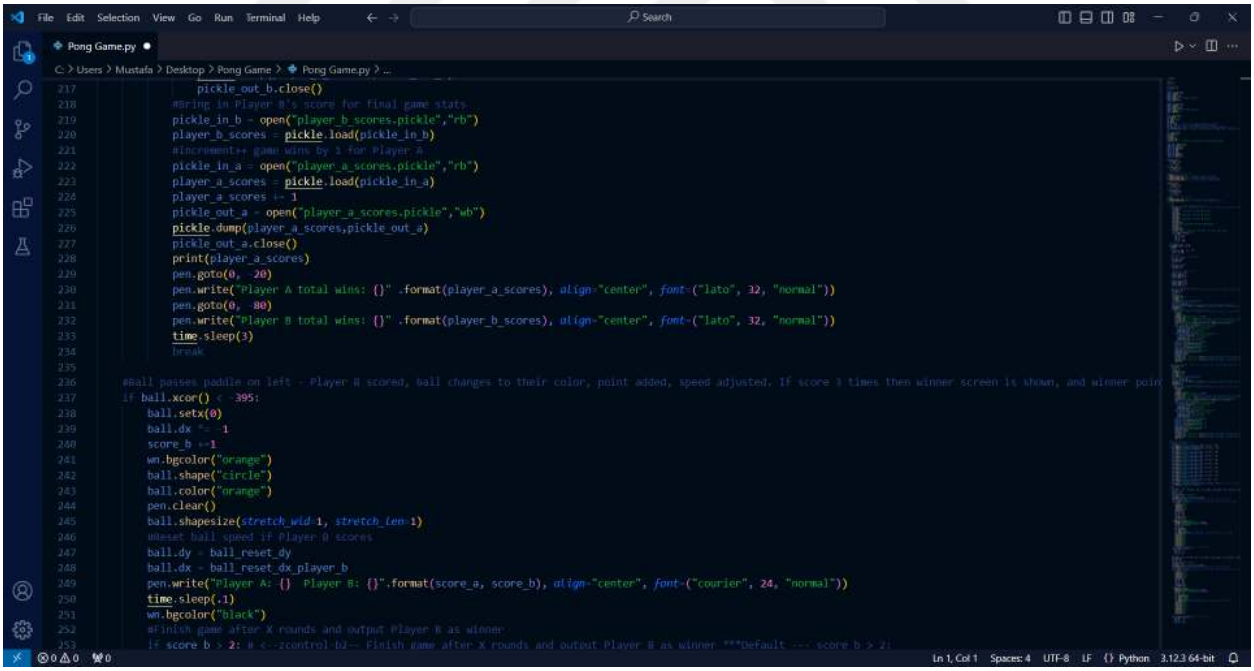


Figure 7.20: Code of Pong game in Python

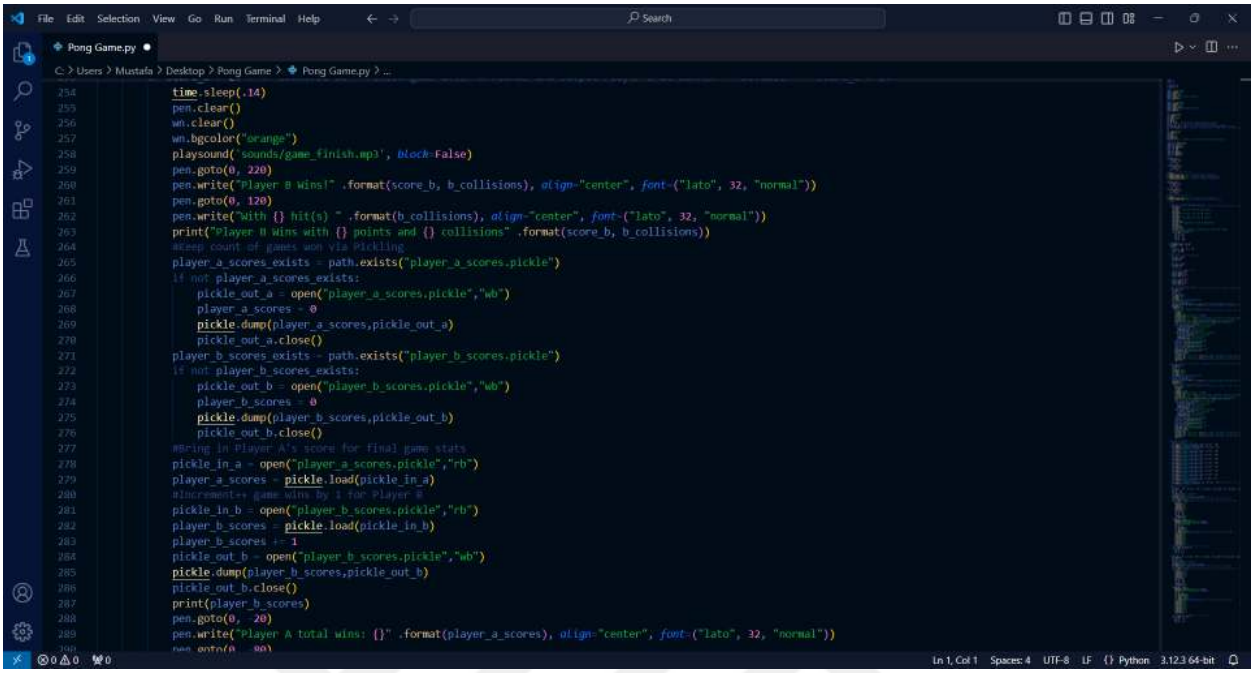


Figure 7.21: Code of Pong game in Python

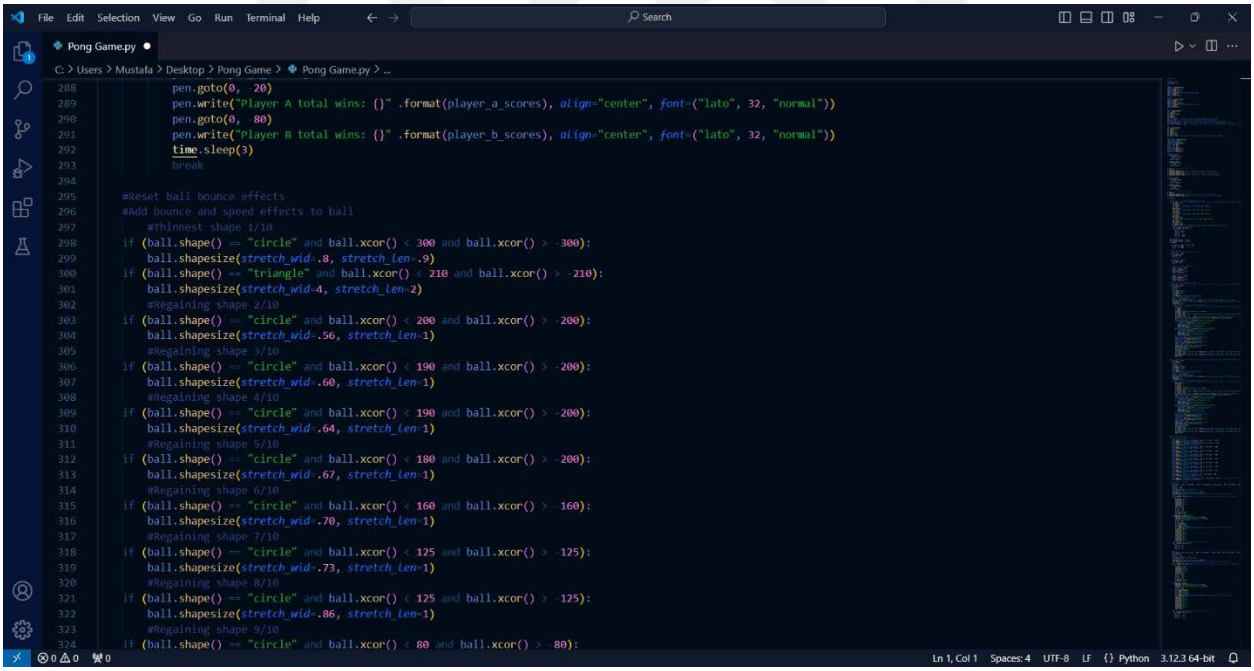


Figure 7.22: Code of Pong game in Python

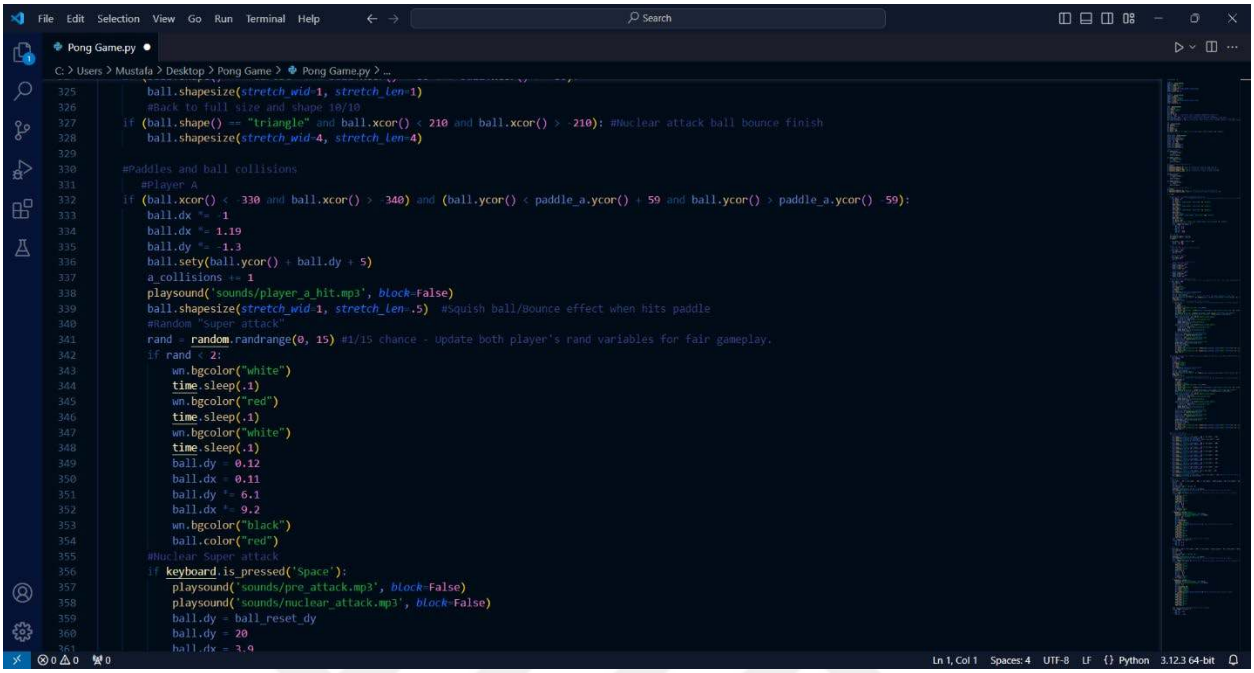


Figure 7.23: Code of Pong game in Python

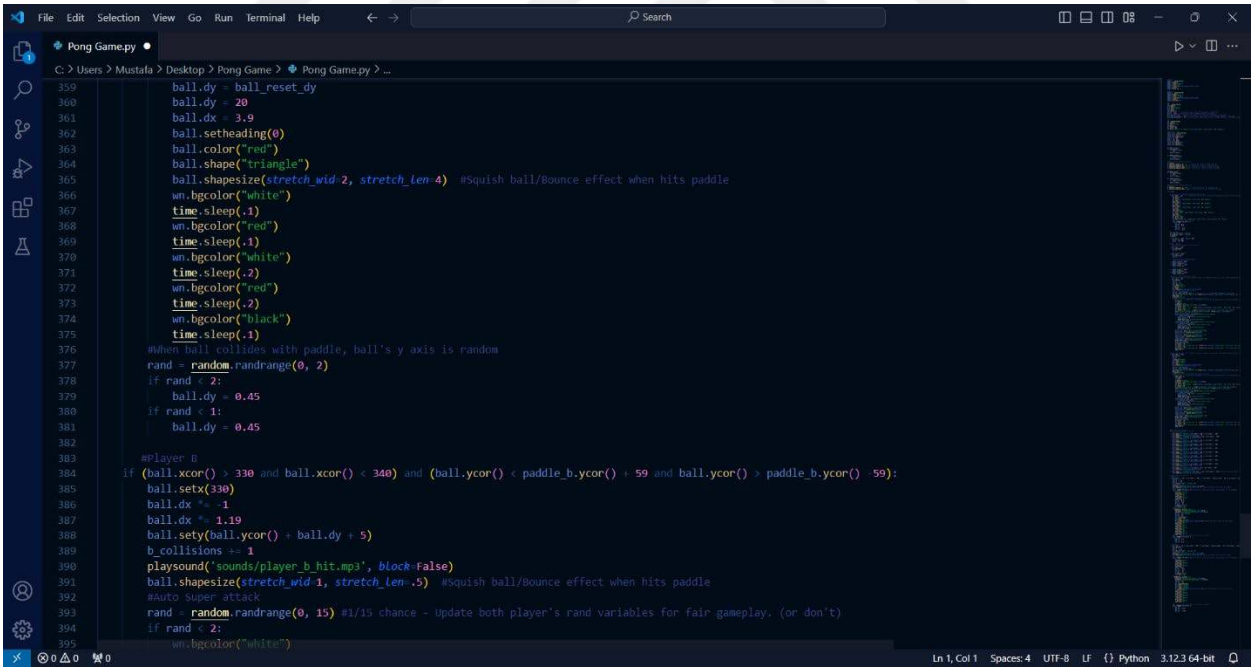


Figure 7.24: Code of Pong game in Python

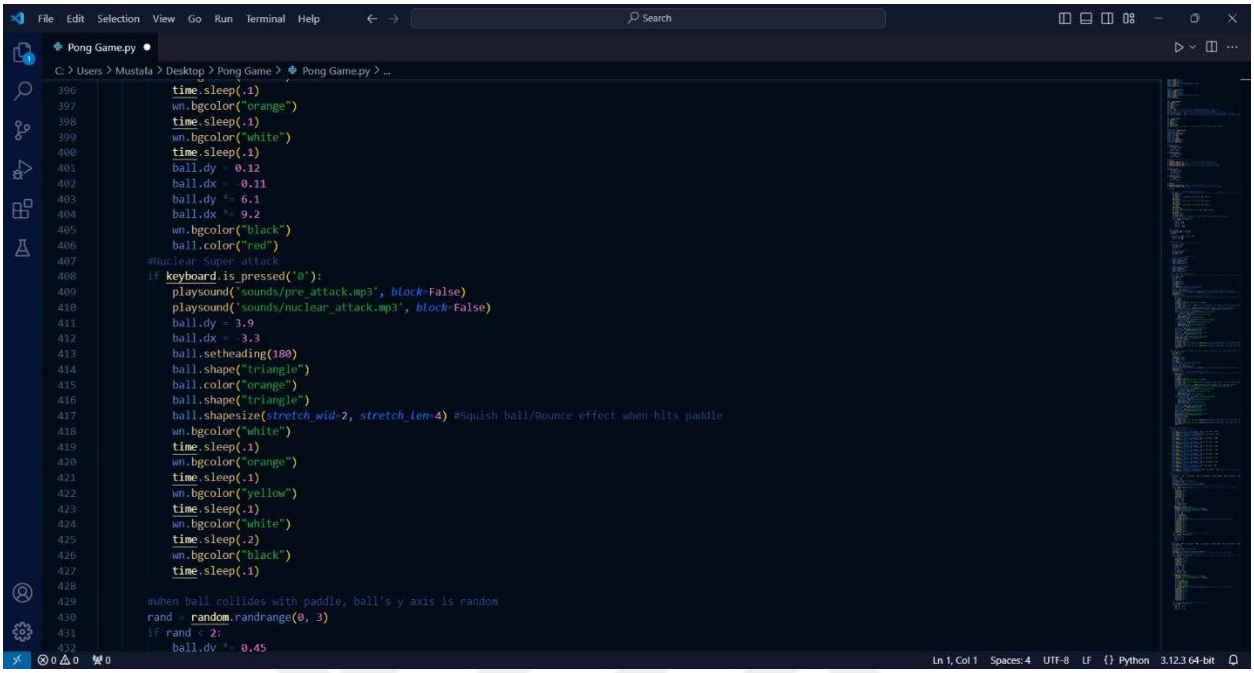


Figure 7.25: Code of Pong game in Python

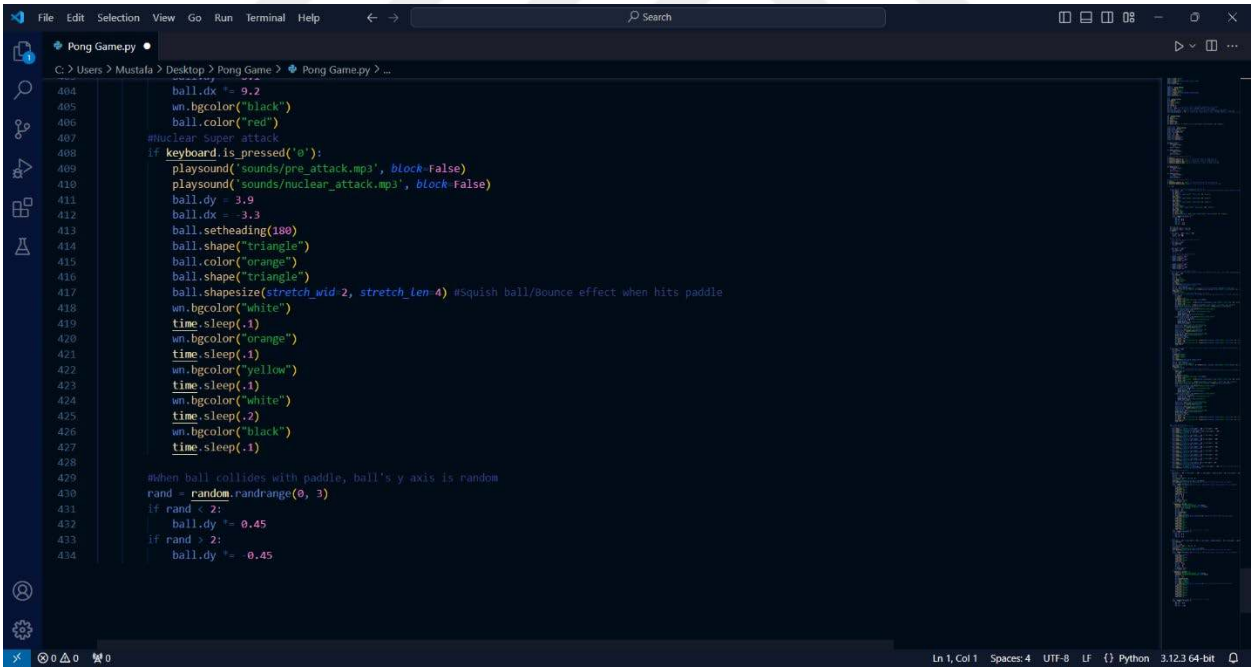
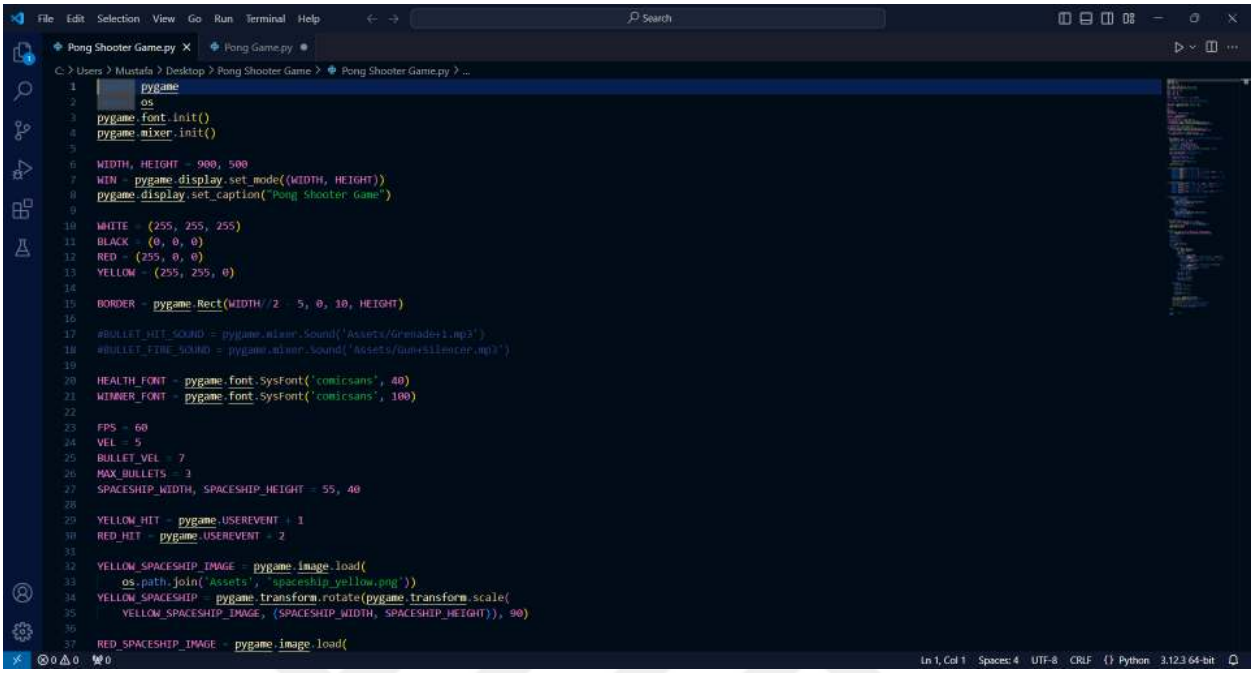
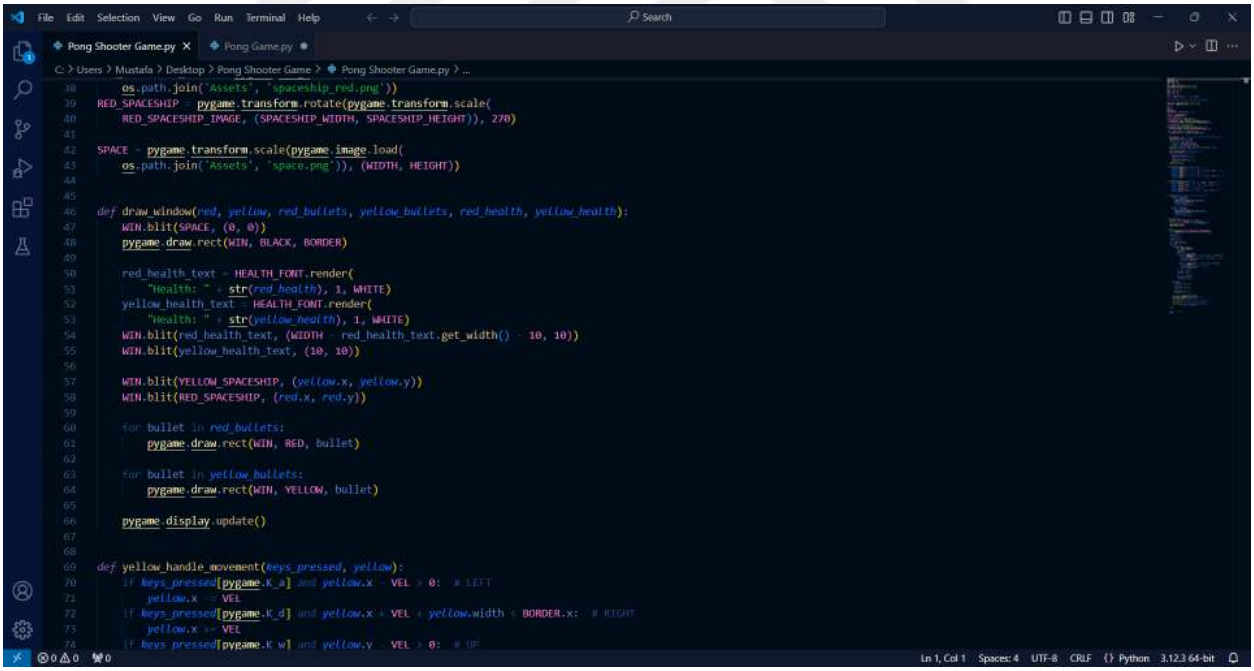


Figure 7.26: Code of Pong game in Python



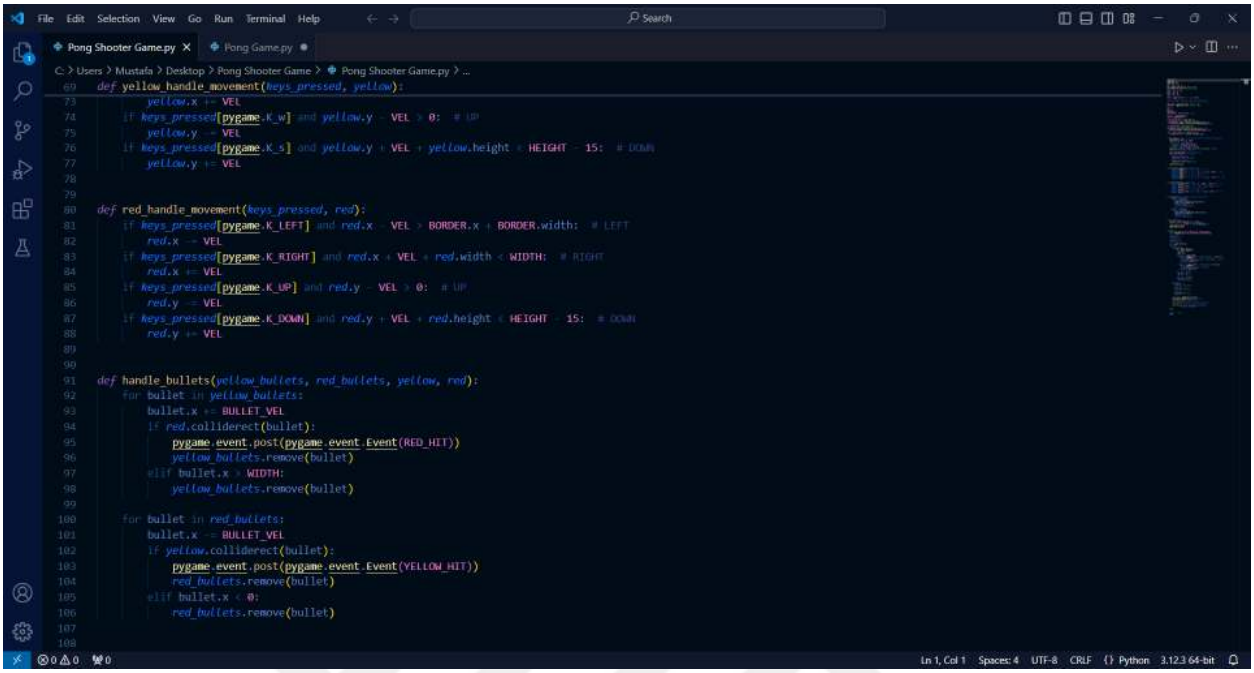
```
import pygame
import os
pygame.font.init()
pygame.mixer.init()

WIDTH, HEIGHT = 900, 500
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Pong Shooter Game")

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
YELLOW = (255, 255, 0)

BORDER = pygame.Rect(WIDTH//2 - 5, 0, 10, HEIGHT)

#BULLET_HIT_SOUND = pygame.mixer.Sound('Assets/Grenade+1.mp3')
#BULLET_FIRE_SOUND = pygame.mixer.Sound('Assets/Gun+Silencer.mp3')

HEALTH_FONT = pygame.font.SysFont('comicsans', 40)
WINNER_FONT = pygame.font.SysFont('comicsans', 100)

FPS = 60
VEL = 5
BULLET_VEL = 7
MAX_BULLETS = 3
SPACESHIP_WIDTH, SPACESHIP_HEIGHT = 55, 40

YELLOW_HIT = pygame.USEREVENT + 1
RED_HIT = pygame.USEREVENT + 2

YELLOW_SPACESHIP_IMAGE = pygame.image.load(
    os.path.join('Assets', 'spaceship_yellow.png'))
YELLOW_SPACESHIP = pygame.transform.rotate(pygame.transform.scale(
    YELLOW_SPACESHIP_IMAGE, (SPACESHIP_WIDTH, SPACESHIP_HEIGHT)), 90)

RED_SPACESHIP_IMAGE = pygame.image.load(
```

Figure 7.27: Code of Pong Shooter game in Python



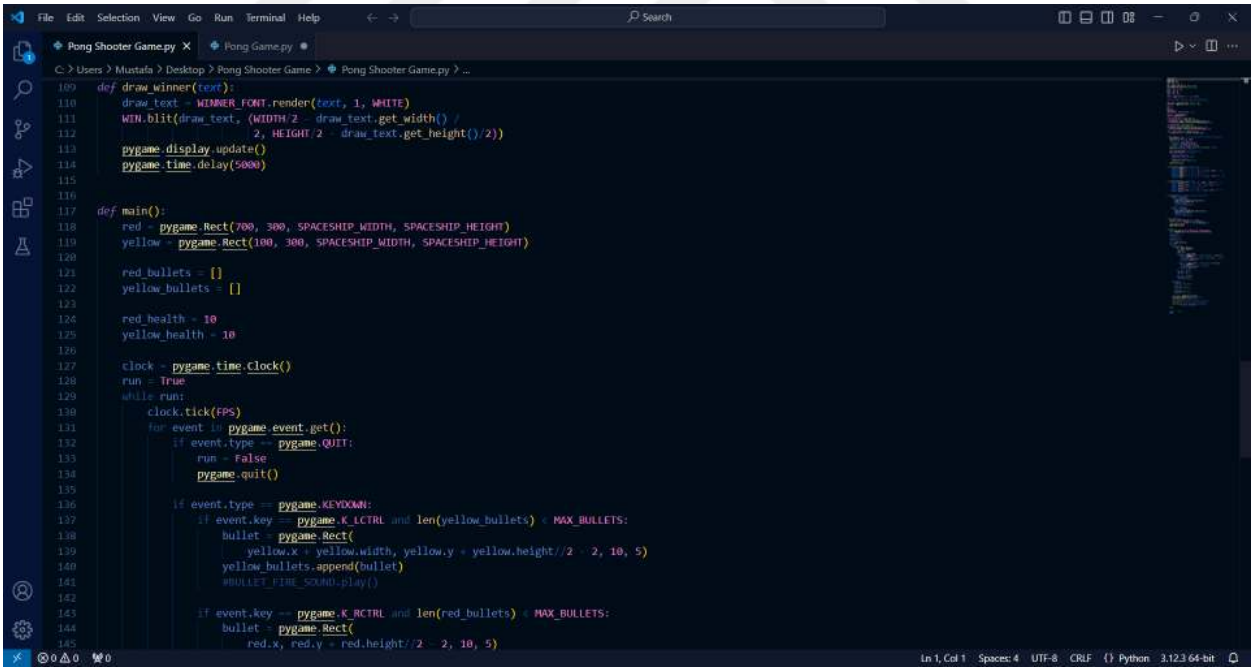
```
    os.path.join('Assets', 'spaceship_red.png'))
RED_SPACESHIP = pygame.transform.rotate(pygame.transform.scale(
    RED_SPACESHIP_IMAGE, (SPACESHIP_WIDTH, SPACESHIP_HEIGHT)), 270)

SPACE = pygame.transform.scale(pygame.image.load(
    os.path.join('Assets', 'space.png')), (WIDTH, HEIGHT))


def draw_window(red, yellow, red_bullets, yellow_bullets, red_health, yellow_health):
    WIN.blit(SPACE, (0, 0))
    pygame.draw.rect(WIN, BLACK, BORDER)

    red_health_text = HEALTH_FONT.render(
        "Health: " + str(red_health), 1, WHITE)
    yellow_health_text = HEALTH_FONT.render(
        "Health: " + str(yellow_health), 1, WHITE)
    WIN.blit(red_health_text, (WIDTH - red_health_text.get_width() - 10, 10))
    WIN.blit(yellow_health_text, (10, 10))

    WIN.blit(YELLOW_SPACESHIP, (yellow.x, yellow.y))
    WIN.blit(RED_SPACESHIP, (red.x, red.y))

    for bullet in red_bullets:
        pygame.draw.rect(WIN, RED, bullet)

    for bullet in yellow_bullets:
        pygame.draw.rect(WIN, YELLOW, bullet)

    pygame.display.update()


def yellow_handle_movement(keys_pressed, yellow):
    if keys_pressed[pygame.K_a] and yellow.x - VEL > 0:  # LEFT
        yellow.x -= VEL
    if keys_pressed[pygame.K_d] and yellow.x + VEL + yellow.width < BORDER.x:  # RIGHT
        yellow.x += VEL
    if keys_pressed[pygame.K_w] and yellow.y - VEL > 0:  # UP
```

Figure 7.28: Code of Pong Shooter game in Python



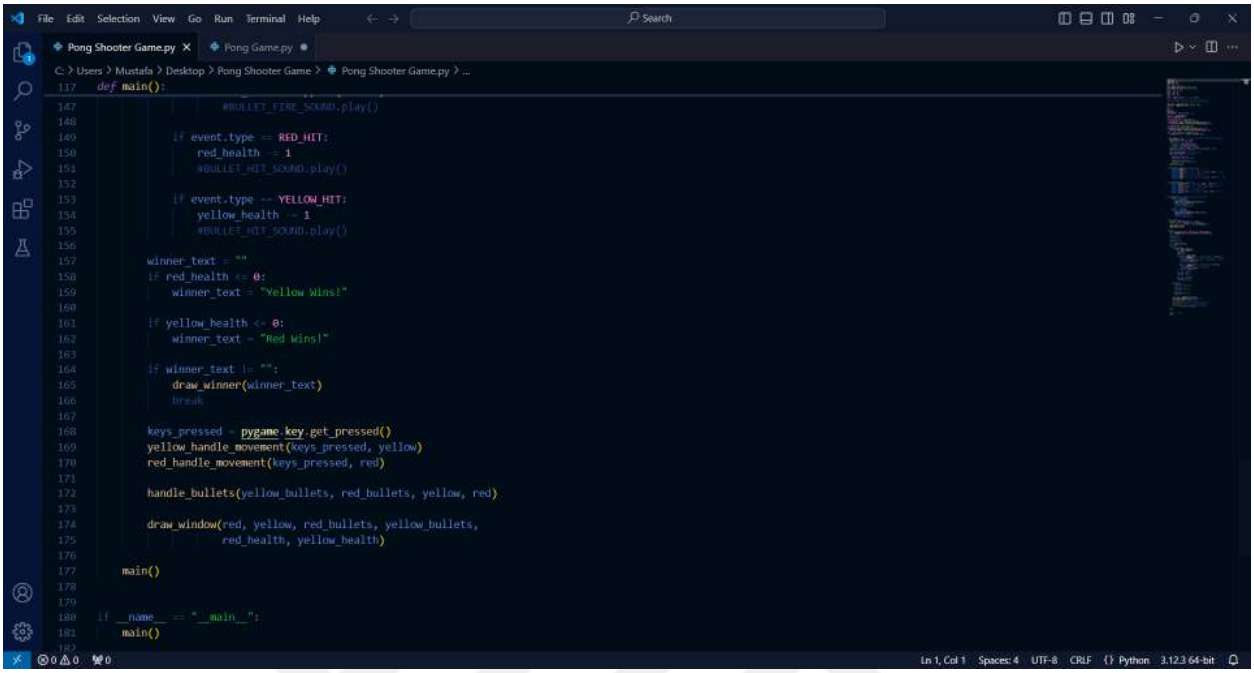
```python
def yellow_handle_movement(keys_pressed, yellow):
        yellow.x += VEL
    if keys_pressed[pygame.K_w] and yellow.y - VEL > 0:  # UP
        yellow.y -= VEL
    if keys_pressed[pygame.K_s] and yellow.y + VEL + yellow.height < HEIGHT - 15:  # DOWN
        yellow.y += VEL


def red_handle_movement(keys_pressed, red):
    if keys_pressed[pygame.K_LEFT] and red.x - VEL > BORDER.x + BORDER.width:  # LEFT
        red.x -= VEL
    if keys_pressed[pygame.K_RIGHT] and red.x + VEL + red.width < WIDTH:  # RIGHT
        red.x += VEL
    if keys_pressed[pygame.K_UP] and red.y - VEL > 0:  # UP
        red.y -= VEL
    if keys_pressed[pygame.K_DOWN] and red.y + VEL + red.height < HEIGHT - 15:  # DOWN
        red.y += VEL


def handle_bullets(yellow_bullets, red_bullets, yellow, red):
    for bullet in yellow_bullets:
        bullet.x += BULLET_VEL
        if red.colliderect(bullet):
            pygame.event.post(pygame.event.Event(RED_HIT))
            yellow_bullets.remove(bullet)
        elif bullet.x > WIDTH:
            yellow_bullets.remove(bullet)

    for bullet in red_bullets:
        bullet.x -= BULLET_VEL
        if yellow.colliderect(bullet):
            pygame.event.post(pygame.event.Event(YELLOW_HIT))
            red_bullets.remove(bullet)
        elif bullet.x < 0:
            red_bullets.remove(bullet)
```

Figure 7.29: Code of Pong Shooter game in Python



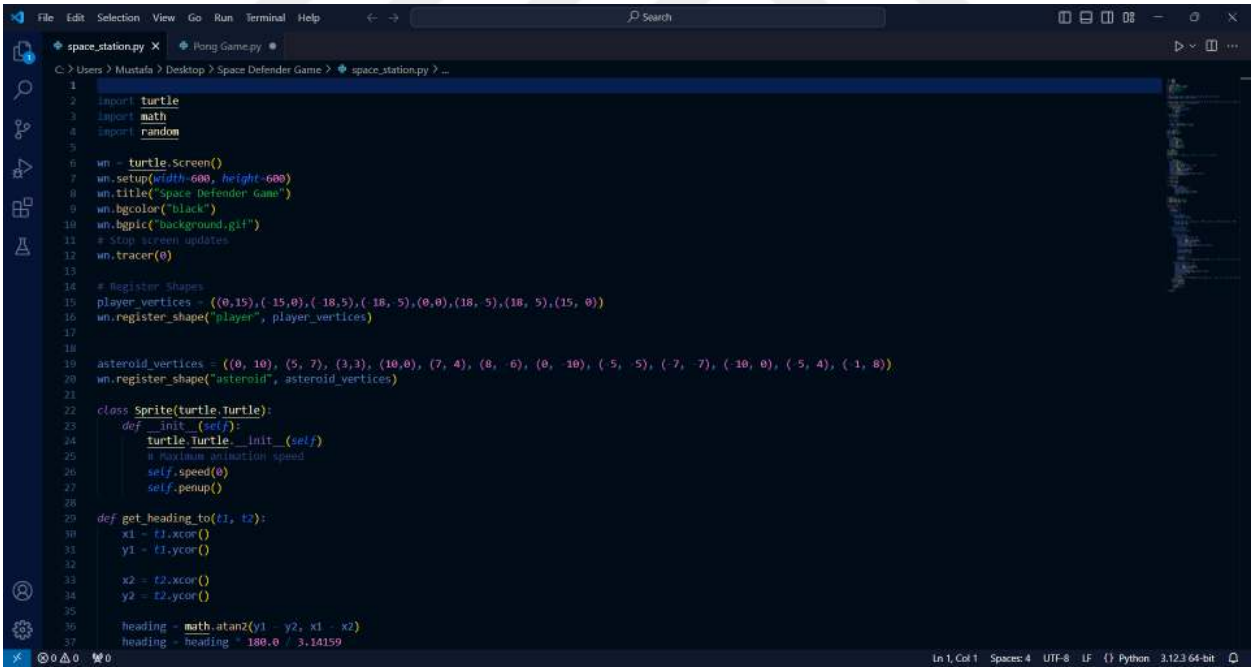
```python
def draw_winner(text):
    draw_text = WINNER_FONT.render(text, 1, WHITE)
    WIN.blit(draw_text, (WIDTH/2 - draw_text.get_width() /
                         2, HEIGHT/2 - draw_text.get_height()/2))
    pygame.display.update()
    pygame.time.delay(5000)


def main():
    red = pygame.Rect(700, 300, SPACESHIP_WIDTH, SPACESHIP_HEIGHT)
    yellow = pygame.Rect(100, 300, SPACESHIP_WIDTH, SPACESHIP_HEIGHT)

    red_bullets = []
    yellow_bullets = []

    red_health = 10
    yellow_health = 10

    clock = pygame.time.Clock()
    run = True
    while run:
        clock.tick(FPS)
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
                pygame.quit()

            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LCTRL and len(yellow_bullets) < MAX_BULLETS:
                    bullet = pygame.Rect(
                        yellow.x + yellow.width, yellow.y + yellow.height//2 - 2, 10, 5)
                    yellow_bullets.append(bullet)
                    #BULLET_FIRE_SOUND.play()

                if event.key == pygame.K_RCTRL and len(red_bullets) < MAX_BULLETS:
                    bullet = pygame.Rect(
                        red.x, red.y + red.height//2 - 2, 10, 5)
```

Figure 7.30: Code of Pong Shooter game in Python



```
def main():
                    #BULLET_FIRE_SOUND.play()

            if event.type == RED_HIT:
                red_health -= 1
                #BULLET_HIT_SOUND.play()

            if event.type == YELLOW_HIT:
                yellow_health -= 1
                #BULLET_HIT_SOUND.play()

        winner_text = ""
        if red_health <= 0:
            winner_text = "Yellow Wins!"

        if yellow_health <= 0:
            winner_text = "Red Wins!"

        if winner_text != "":
            draw_winner(winner_text)
            break

        keys_pressed = pygame.key.get_pressed()
        yellow_handle_movement(keys_pressed, yellow)
        red_handle_movement(keys_pressed, red)

        handle_bullets(yellow_bullets, red_bullets, yellow, red)

        draw_window(red, yellow, red_bullets, yellow_bullets,
                    red_health, yellow_health)

    main()


if __name__ == "__main__":
    main()
```

Figure 7.31: Code of Pong Shooter game in Python



```
import turtle
import math
import random

wn = turtle.Screen()
wn.setup(width=600, height=600)
wn.title("Space Defender Game")
wn.bgcolor("black")
wn.bgpic("background.gif")
# Stop screen updates
wn.tracer(0)

# Register Shapes
player_vertices = ((0,15),(-15,0),(-18,5),(-18,-5),(0,0),(18,-5),(18,5),(15,0))
wn.register_shape("player", player_vertices)


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))
wn.register_shape("asteroid", asteroid_vertices)

class Sprite(turtle.Turtle):
    def __init__(self):
        turtle.Turtle.__init__(self)
        # Maximum animation speed
        self.speed(0)
        self.penup()

def get_heading_to(t1, t2):
    x1 = t1.xcor()
    y1 = t1.ycor()

    x2 = t2.xcor()
    y2 = t2.ycor()

    heading = math.atan2(y1 - y2, x1 - x2)
    heading = heading * 180.0 / 3.14159
```

Figure 7.32: Code of Space Defender game in Python

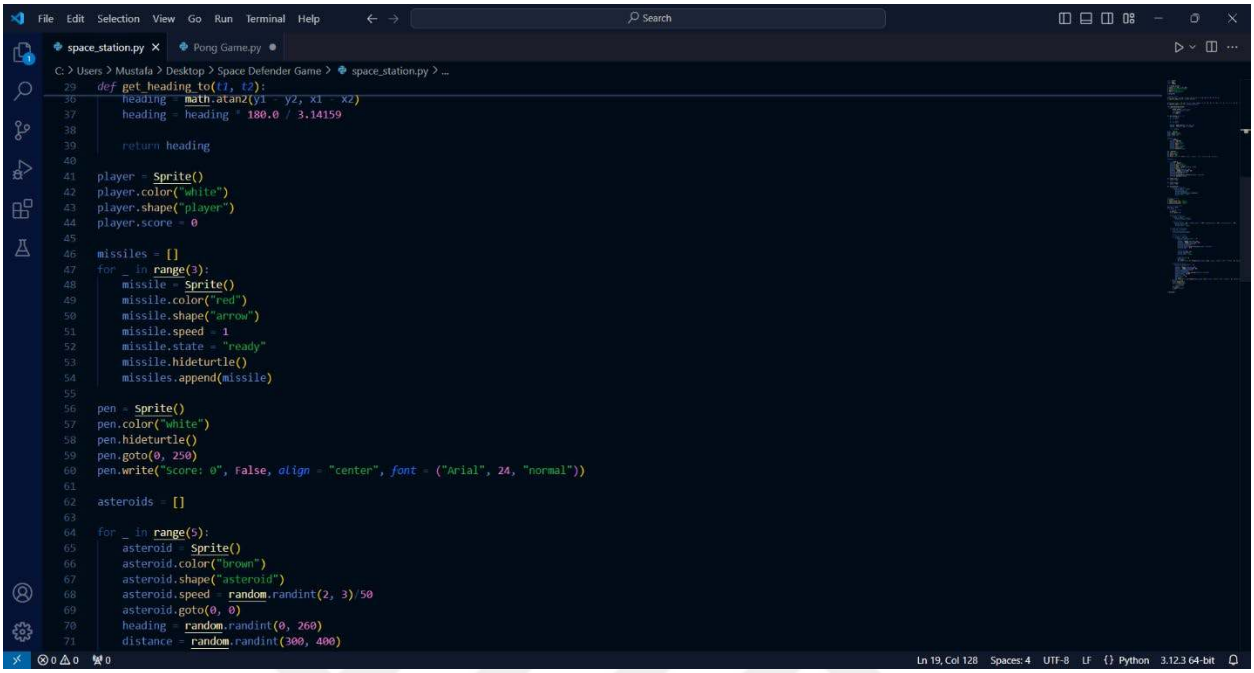


Figure 7.33: Code of Space Defender game in Python

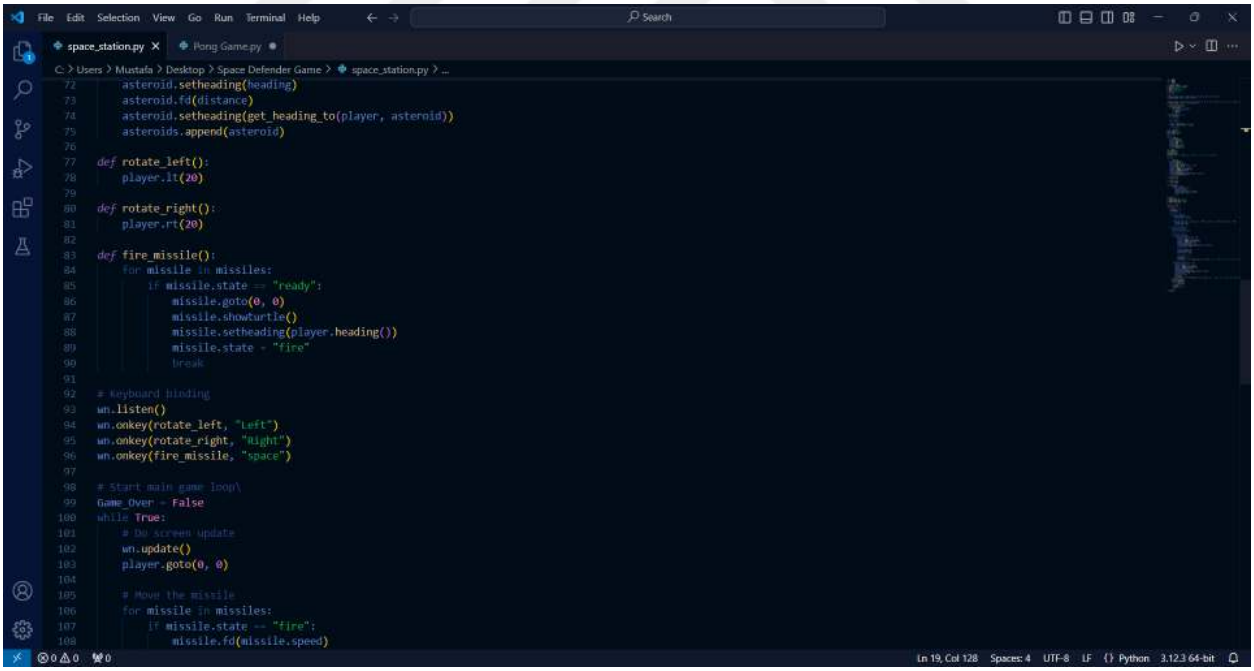
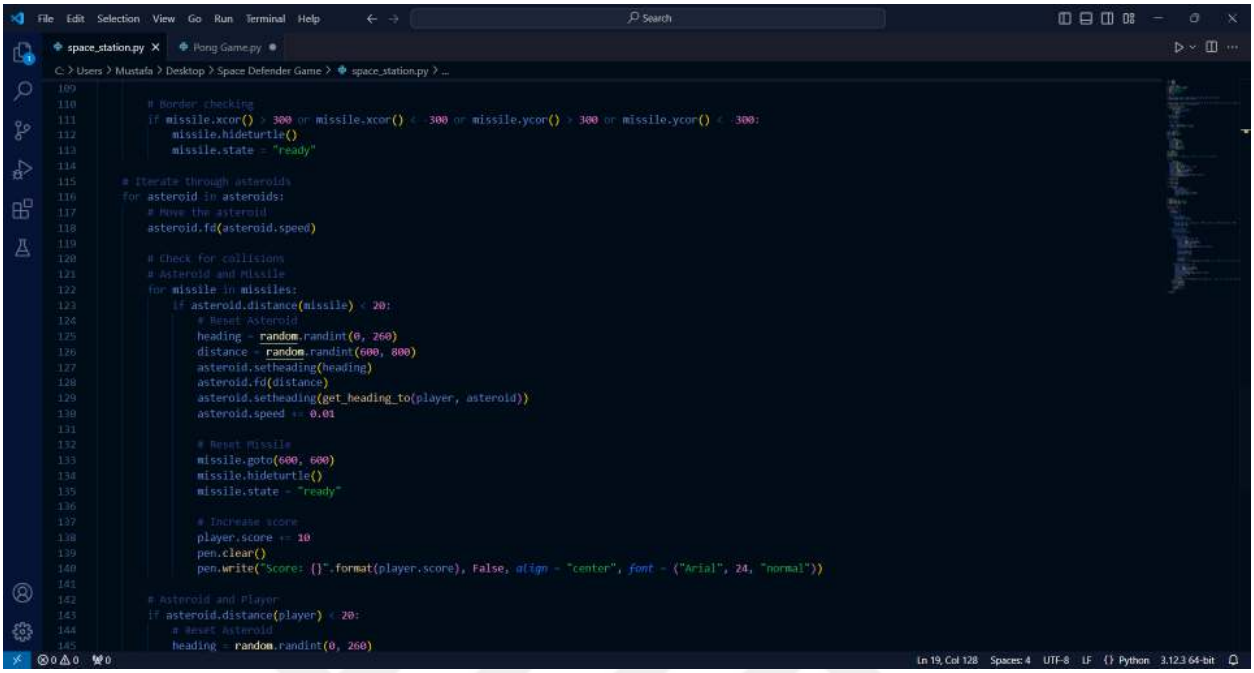
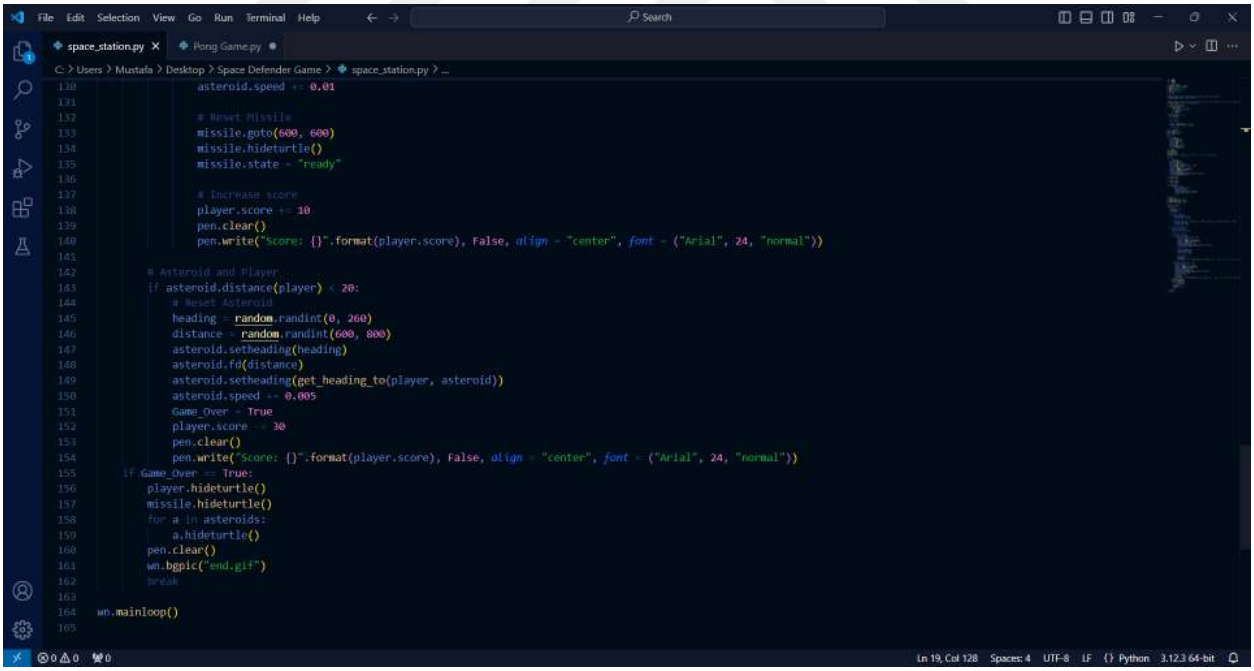


Figure 7.34: Code of Space Defender game in Python



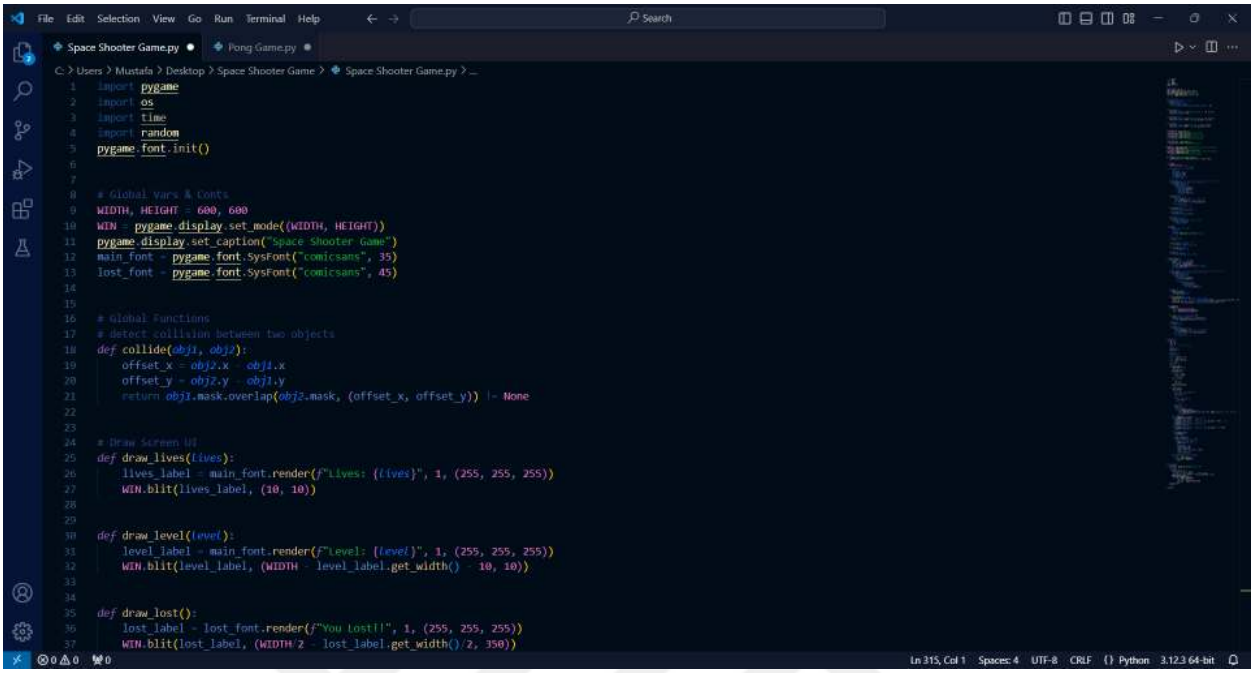
```
        # Border checking
        if missile.xcor() > 300 or missile.xcor() < -300 or missile.ycor() > 300 or missile.ycor() < -300:
            missile.hideturtle()
            missile.state = "ready"

    # Iterate through asteroids
    for asteroid in asteroids:
        # Move the asteroid
        asteroid.fd(asteroid.speed)

        # Check for collisions
        # Asteroid and Missile
        for missile in missiles:
            if asteroid.distance(missile) < 20:
                # Reset Asteroid
                heading = random.randint(0, 260)
                distance = random.randint(600, 800)
                asteroid.setheading(heading)
                asteroid.fd(distance)
                asteroid.setheading(get_heading_to(player, asteroid))
                asteroid.speed += 0.01

                # Reset Missile
                missile.goto(600, 600)
                missile.hideturtle()
                missile.state = "ready"

                # Increase score
                player.score += 10
                pen.clear()
                pen.write("Score: {}".format(player.score), False, align = "center", font = ("Arial", 24, "normal"))

        # Asteroid and Player
        if asteroid.distance(player) < 20:
            # Reset Asteroid
            heading = random.randint(0, 260)
```

Figure 7.35: Code of Space Defender game in Python



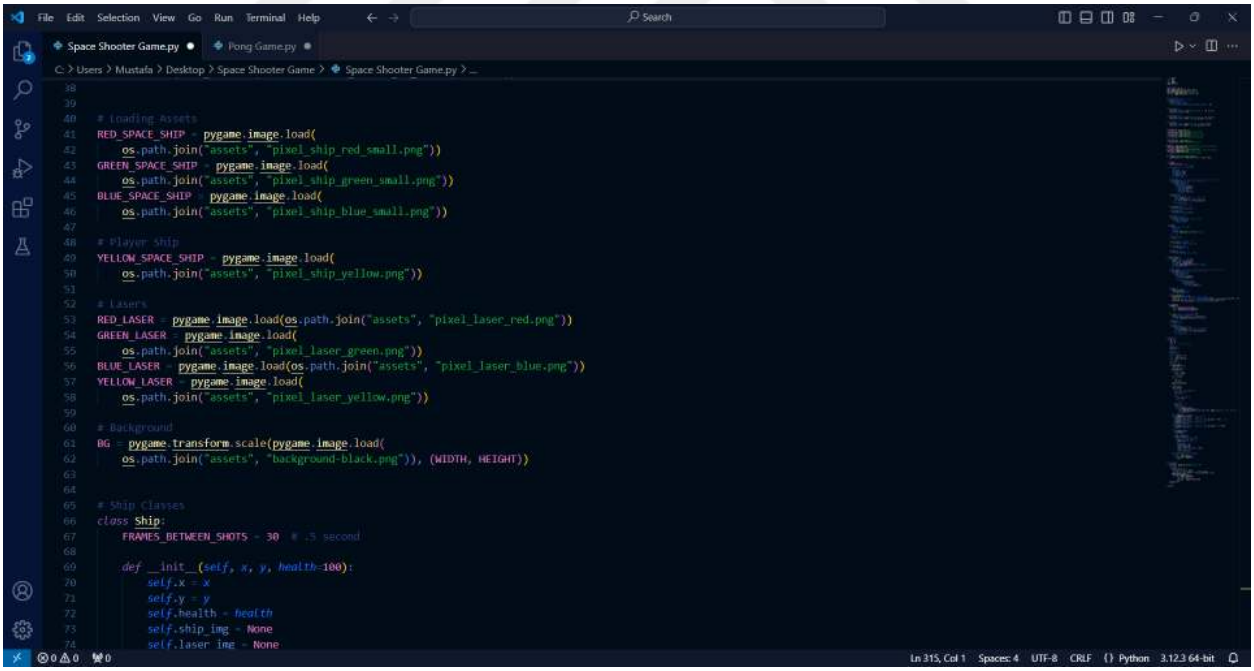
```
                asteroid.speed += 0.01

                # Reset Missile
                missile.goto(600, 600)
                missile.hideturtle()
                missile.state = "ready"

                # Increase score
                player.score += 10
                pen.clear()
                pen.write("Score: {}".format(player.score), False, align = "center", font = ("Arial", 24, "normal"))

        # Asteroid and Player
        if asteroid.distance(player) < 20:
            # Reset Asteroid
            heading = random.randint(0, 260)
            distance = random.randint(600, 800)
            asteroid.setheading(heading)
            asteroid.fd(distance)
            asteroid.setheading(get_heading_to(player, asteroid))
            asteroid.speed += 0.005
            Game_Over = True
            player.score -= 30
            pen.clear()
            pen.write("Score: {}".format(player.score), False, align = "center", font = ("Arial", 24, "normal"))
    if Game_Over == True:
        player.hideturtle()
        missile.hideturtle()
        for a in asteroids:
            a.hideturtle()
        pen.clear()
        wn.bgpic("end.gif")
        break

wn.mainloop()
```

Figure 7.36: Code of Space Defender game in Python



```python
import pygame
import os
import time
import random
pygame.font.init()


# Global Vars & Consts
WIDTH, HEIGHT = 600, 600
WIN = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Space Shooter Game")
main_font = pygame.font.SysFont("comicsans", 35)
lost_font = pygame.font.SysFont("comicsans", 45)


# Global Functions
# detect collision between two objects
def collide(obj1, obj2):
    offset_x = obj2.x - obj1.x
    offset_y = obj2.y - obj1.y
    return obj1.mask.overlap(obj2.mask, (offset_x, offset_y)) != None


# Draw Screen UI
def draw_lives(lives):
    lives_label = main_font.render(f"Lives: {lives}", 1, (255, 255, 255))
    WIN.blit(lives_label, (10, 10))


def draw_level(level):
    level_label = main_font.render(f"Level: {level}", 1, (255, 255, 255))
    WIN.blit(level_label, (WIDTH - level_label.get_width() - 10, 10))


def draw_lost():
    lost_label = lost_font.render(f"You Lost!!", 1, (255, 255, 255))
    WIN.blit(lost_label, (WIDTH/2 - lost_label.get_width()/2, 350))
```

Figure 7.37: Code of Space Shooter game in Python



```python

# Loading Assets
RED_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_red_small.png"))
GREEN_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_green_small.png"))
BLUE_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_blue_small.png"))

# Player Ship
YELLOW_SPACE_SHIP = pygame.image.load(
    os.path.join("assets", "pixel_ship_yellow.png"))

# Lasers
RED_LASER = pygame.image.load(os.path.join("assets", "pixel_laser_red.png"))
GREEN_LASER = pygame.image.load(
    os.path.join("assets", "pixel_laser_green.png"))
BLUE_LASER = pygame.image.load(os.path.join("assets", "pixel_laser_blue.png"))
YELLOW_LASER = pygame.image.load(
    os.path.join("assets", "pixel_laser_yellow.png"))

# Background
BG = pygame.transform.scale(pygame.image.load(
    os.path.join("assets", "background-black.png")), (WIDTH, HEIGHT))


# Ship Classes
class Ship:
    FRAMES_BETWEEN_SHOTS = 30  # .5 second

    def __init__(self, x, y, health=100):
        self.x = x
        self.y = y
        self.health = health
        self.ship_img = None
        self.laser_img = None
```

Figure 7.38: Code of Space Shooter game in Python

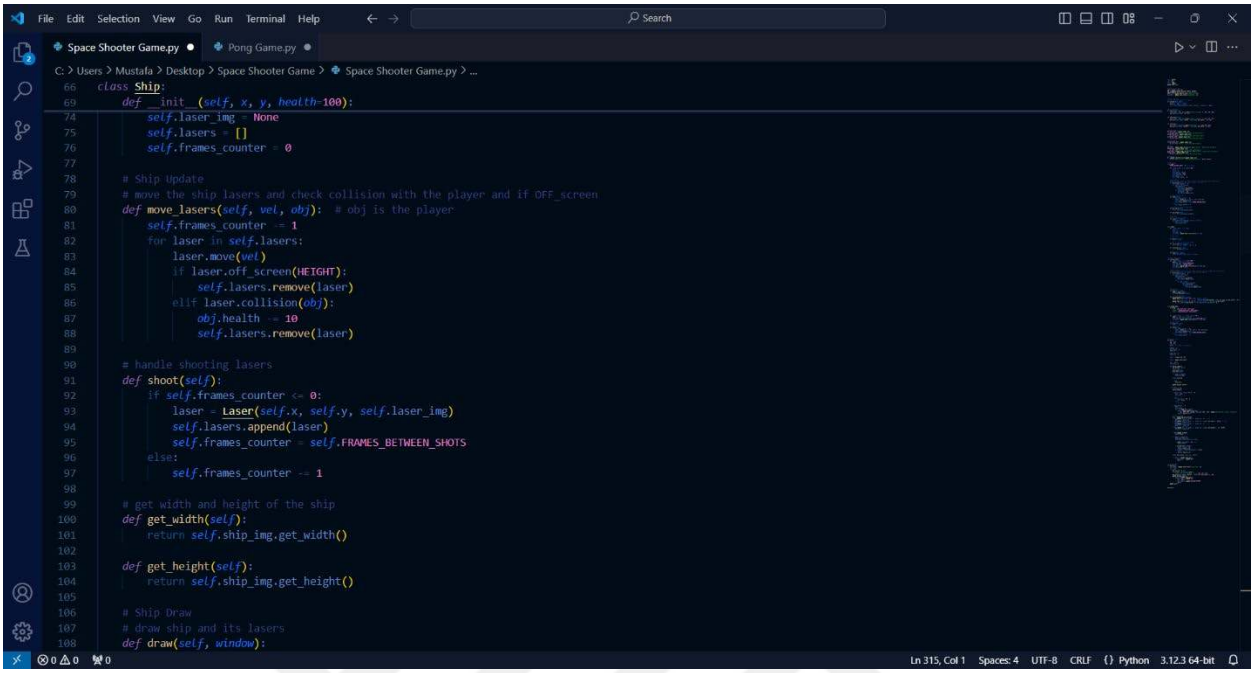


Figure 7.39: Code of Space Shooter game in Python

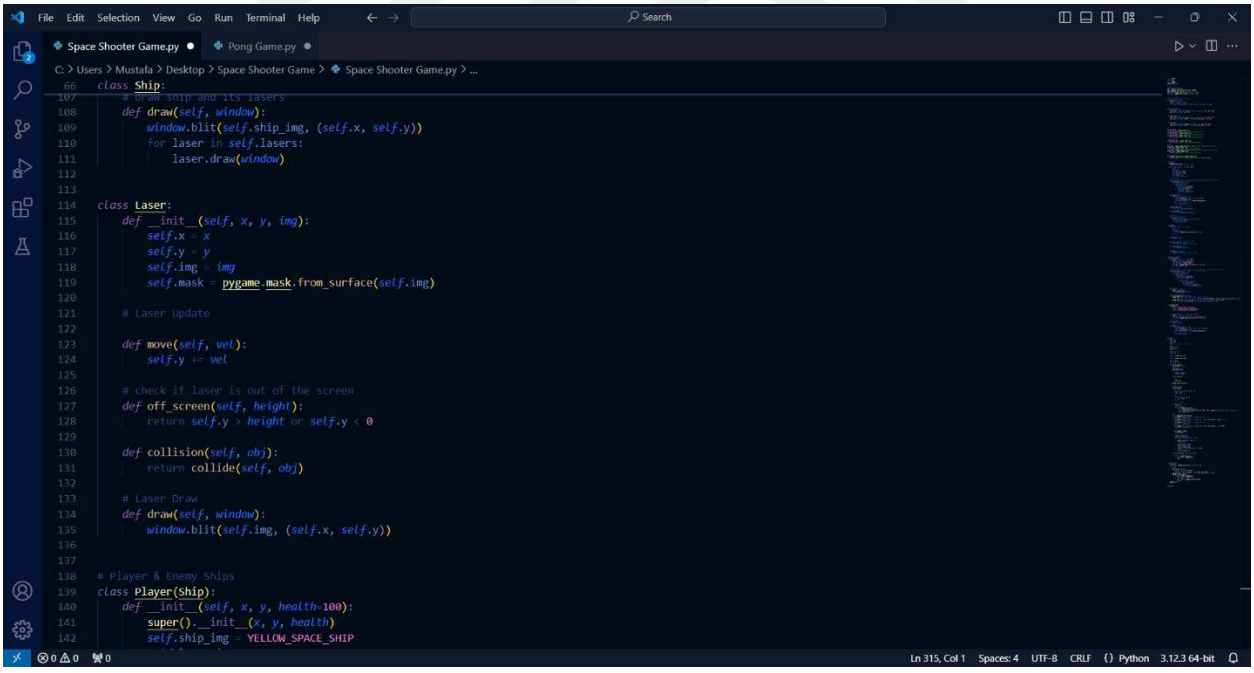


Figure 7.40: Code of Space Shooter game in Python

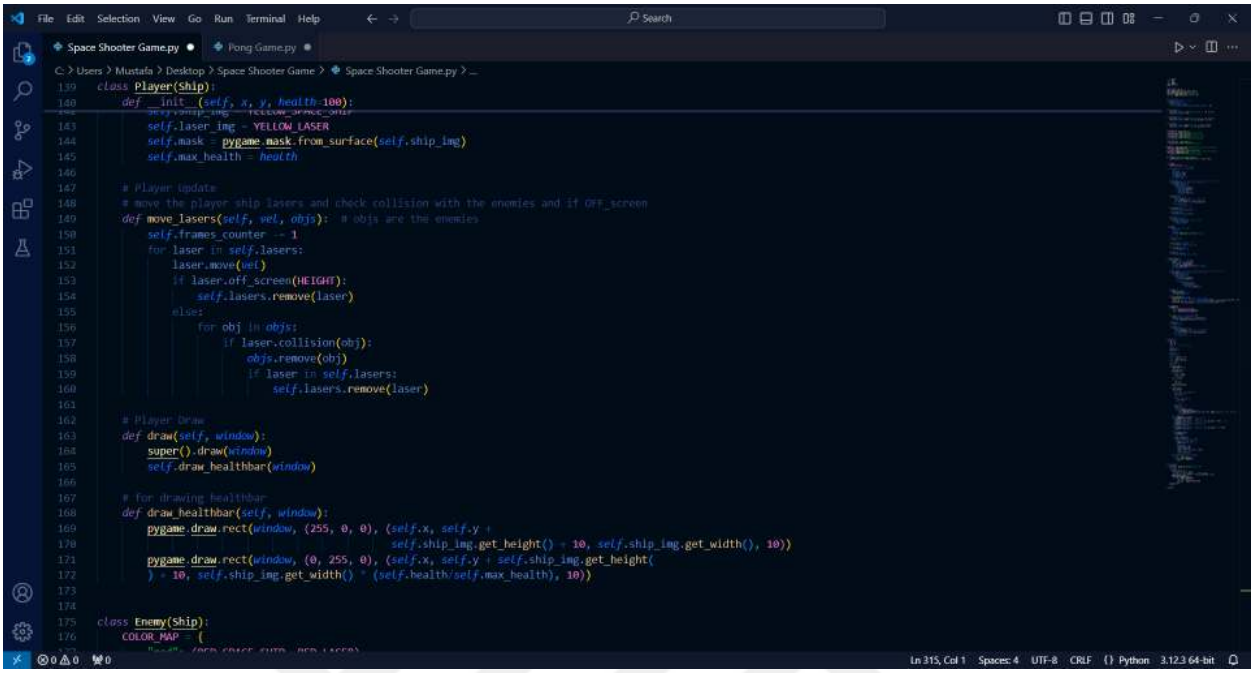


Figure 7.41: Code of Space Shooter game in Python

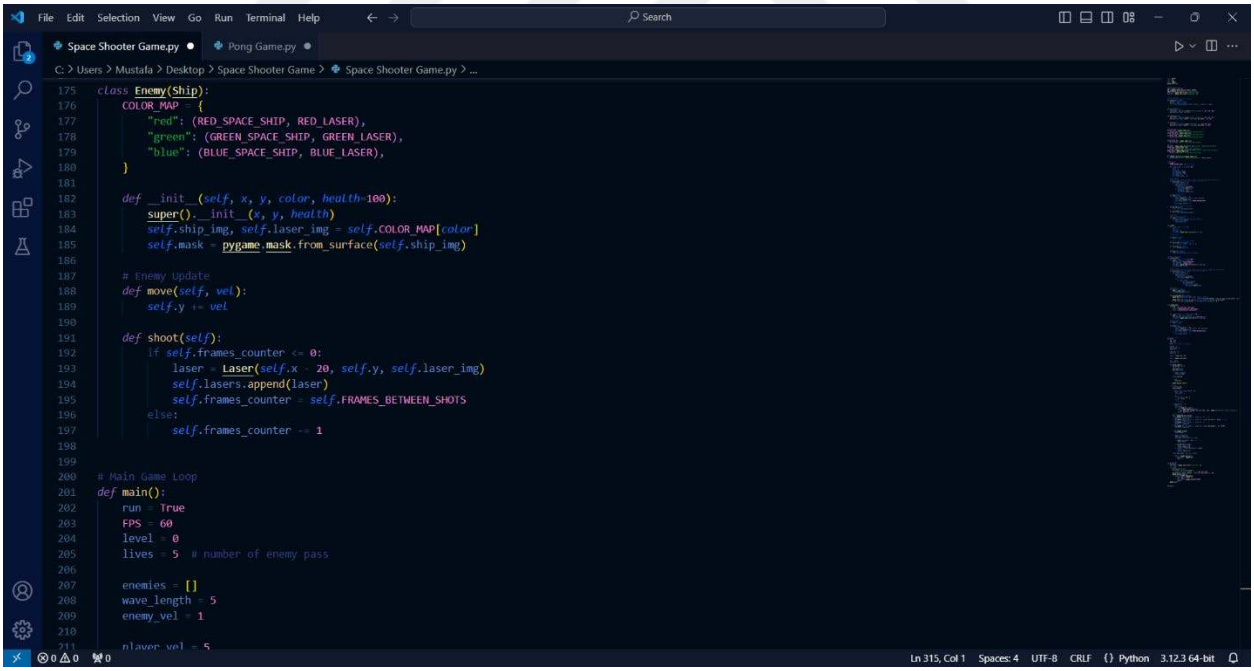
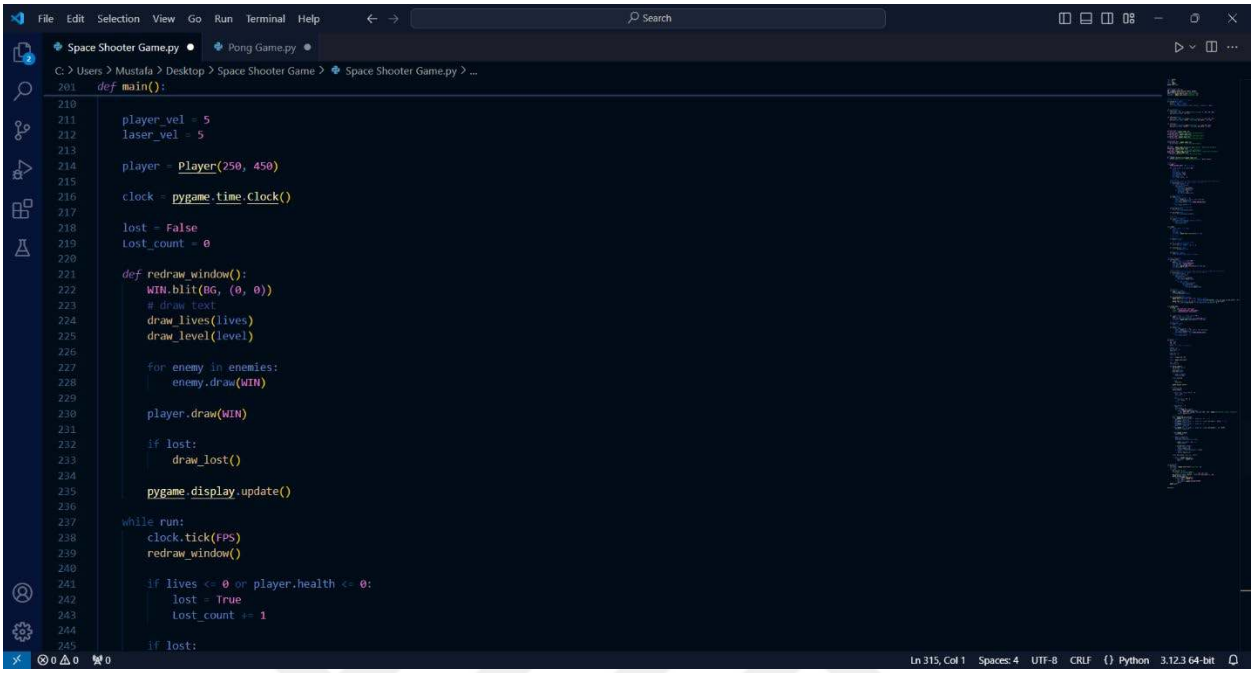
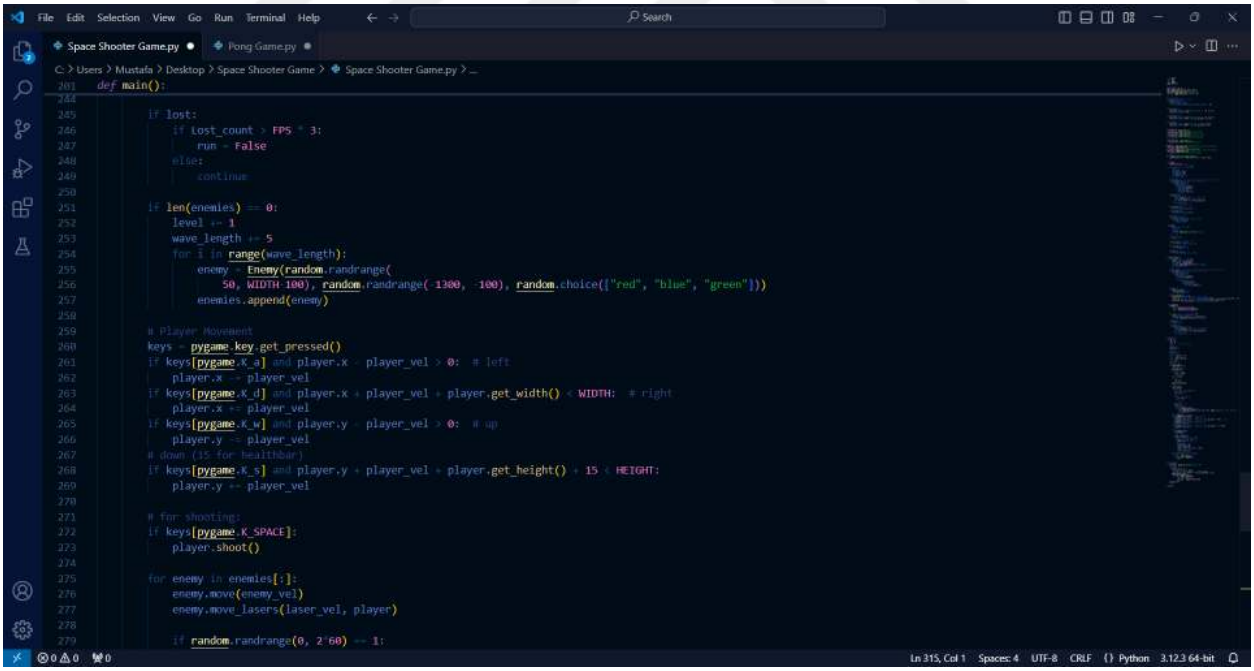


Figure 7.42: Code of Space Shooter game in Python



```
def main():

    player_vel = 5
    laser_vel = 5

    player = Player(250, 450)

    clock = pygame.time.Clock()

    lost = False
    lost_count = 0

    def redraw_window():
        WIN.blit(BG, (0, 0))
        # draw text
        draw_lives(lives)
        draw_level(level)

        for enemy in enemies:
            enemy.draw(WIN)

        player.draw(WIN)

        if lost:
            draw_lost()

        pygame.display.update()

    while run:
        clock.tick(FPS)
        redraw_window()

        if lives <= 0 or player.health <= 0:
            lost = True
            lost_count += 1

        if lost:
```

Figure 7.43: Code of Space Shooter game in Python



```
def main():

        if lost:
            if lost_count > FPS * 3:
                run = False
            else:
                continue

        if len(enemies) == 0:
            level += 1
            wave_length += 5
            for i in range(wave_length):
                enemy = Enemy(random.randrange(
                    50, WIDTH-100), random.randrange(-1300, -100), random.choice(["red", "blue", "green"]))
                enemies.append(enemy)

        # Player Movement
        keys = pygame.key.get_pressed()
        if keys[pygame.K_a] and player.x - player_vel > 0:  # left
            player.x -= player_vel
        if keys[pygame.K_d] and player.x + player_vel + player.get_width() < WIDTH:  # right
            player.x += player_vel
        if keys[pygame.K_w] and player.y - player_vel > 0:  # up
            player.y -= player_vel
        # down (15 for healthbar)
        if keys[pygame.K_s] and player.y + player_vel + player.get_height() + 15 < HEIGHT:
            player.y += player_vel

        # for shooting:
        if keys[pygame.K_SPACE]:
            player.shoot()

        for enemy in enemies[:]:
            enemy.move(enemy_vel)
            enemy.move_lasers(laser_vel, player)

            if random.randrange(0, 2*60) == 1:
```

Figure 7.44: Code of Space Shooter game in Python

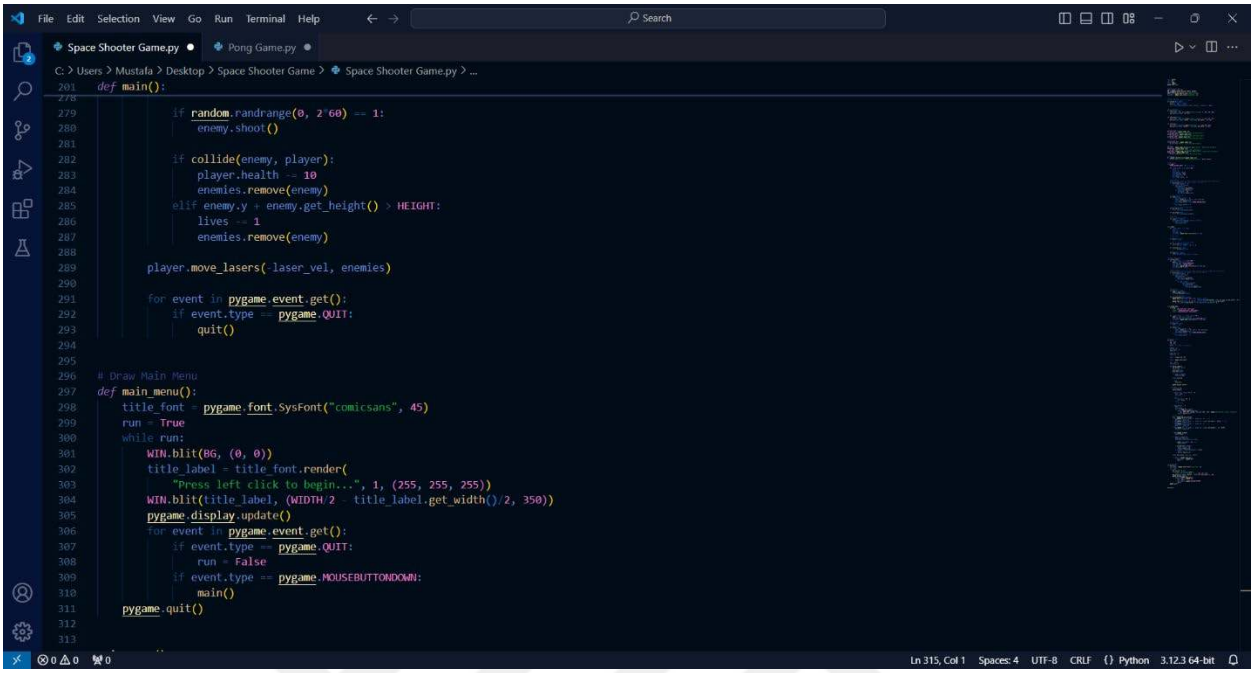


Figure 7.45: Code of Space Shooter game in Python

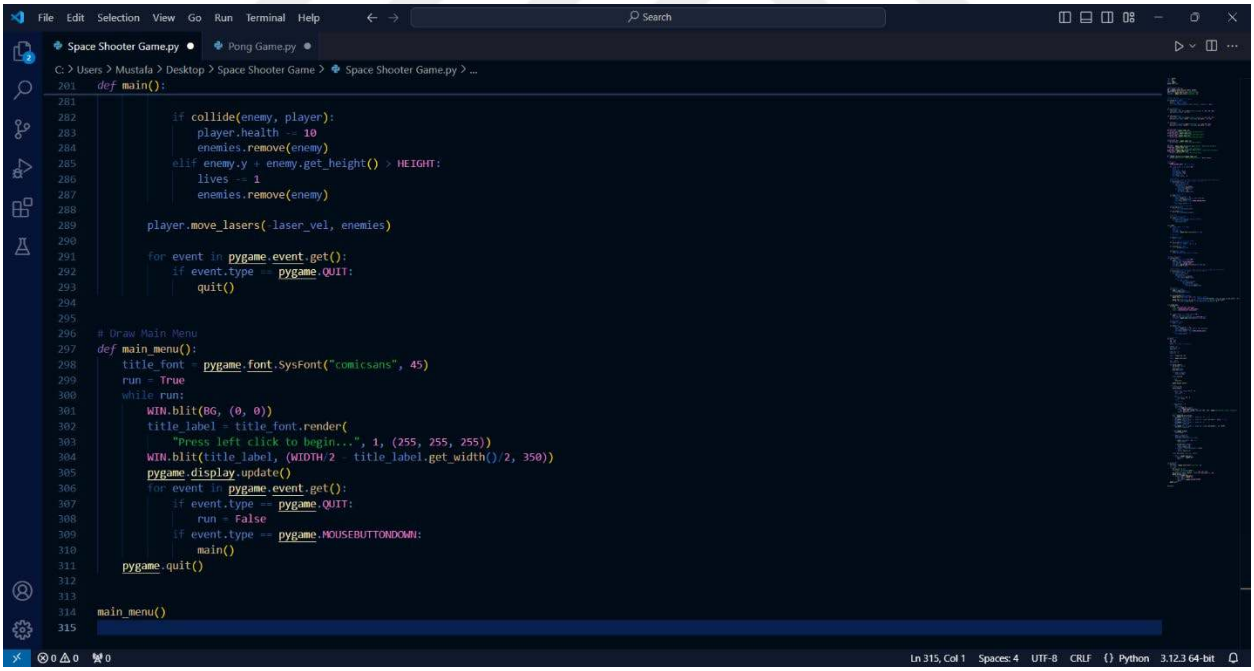


Figure 7.46: Code of Space Shooter game in Python

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.redcrittertracker.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., & Norbisrath, 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