GAZI UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

Burak ÇETİN – 22118080032 – brakcetin660@gmail.com

İsmail Görkem ULAŞ – 21118080007 – gorkem17ulas@gmail.com

MEDIEVAL CASTLE

PROJECT REPORT

CENG376 – COMPUTER GRAPHICS

MAY 2025

TABLE OF CONTENTS

PROJECT REPORT: MEDIEVAL CASTLE SCENE

ABSTRACT

The Medieval Castle Scene project is an interactive 3D simulation developed as part of the CENG376 Computer Graphics course using WebGL and the Three.js library. The goal of the project is to design a fully navigable virtual environment set in a medieval theme while meeting all the technical criteria of the course, including camera movement in three axes, object interaction, lighting, and textured models. Players can explore the scene from a first-person perspective, interact with gameplay elements such as collecting stones and launching them via a catapult, and switch between day and night modes, which affect the lighting and ambiance of the scene. The system features real-time controls via GUI and HTML elements, dynamic lighting updates, a power bar-based launch mechanic, and immersive sound effects. In addition to meeting all minimum project requirements, the implementation includes extended functionalities such as an inventory panel, scoring system, help menu, and performance optimizations. The project showcases a strong blend of technical implementation, user experience, and creative design.

GitHub: https://github.com/brakcetin/medieval_castle_scene.git

Video: Google Drive Video Link

Keywords: WebGL, Three.js, 3D Graphics, Interactive Simulation, Day/Night Cycle, Catapult Mechanic, Real-time Lighting, Game UI, Medieval Scene, First-Person Navigation, GLTF Models, dat.GUI, Sound Effects

1. INTRODUCTION

Computer graphics plays a fundamental role in many modern-day applications ranging from video games and simulations to scientific visualization and virtual reality. As part of the CENG376 Computer Graphics course, students are expected to demonstrate their understanding of core 3D rendering principles, user interaction, object modeling, lighting, and scene composition through a comprehensive project. This hands-on approach allows students to apply theoretical knowledge in a practical context and experience the challenges of building real-time interactive environments.

The "Medieval Castle Scene" project was conceived to reflect these objectives in a visually appealing and functionally rich 3D setting. Drawing inspiration from medieval architecture and gameplay mechanics, the project combines historical aesthetics with dynamic interactivity. The main purpose of the project is to create an immersive, explorable environment where players can move freely, interact with various objects, and experience a cycle of changing light conditions that affect visibility and gameplay.

The project leverages WebGL through the Three.js framework to achieve cross-platform compatibility and real-time performance within a web browser. Users are placed in a medieval world featuring elements like a castle, a catapult, torches, and scattered stones. Gameplay involves collecting stones, loading them into a catapult, and launching them with precision and force

control. Additionally, environmental features such as a day/night cycle and atmospheric lighting effects are incorporated to enhance realism and challenge.

In the sections that follow, the report details the design choices, technical implementations, and performance considerations that contributed to the development of this project. It also outlines how the system meets and surpasses the course's minimum requirements through creative design and technical enhancements.

2. PROJECT OVERVIEW

This project presents an interactive and visually rich 3D medieval castle scene designed and developed using WebGL and Three.js. The environment consists of multiple textured and animated models such as a castle, a working catapult, realistic torches, and dynamic stones. Users can explore the environment from a first-person perspective with full 3D camera control, interact with gameplay elements like picking and launching stones, toggle between day and night modes, and receive visual feedback via a scoring and inventory system. The aim of the project is to both meet and exceed the course's requirements while delivering a technically robust, immersive, and enjoyable user experience.

3. TECHNICAL IMPLEMENTATION

This section outlines the technical framework, tools, and components employed to develop the "Medieval Castle Scene" project. It provides a structured breakdown of the technologies, interaction systems, model integration, and other key architectural aspects that contribute to the overall function and presentation of the interactive 3D environment.

3.1 Technologies and Tools

- **Three.js**: The primary rendering engine used to build and display the 3D world within the browser. It provides essential functionality for camera control, object rendering, lighting, shadows, and real-time updates.
- **GLTFLoader**: A part of the Three.js extensions, GLTFLoader is responsible for importing and rendering .glb model files efficiently. It supports complex geometry, animation, and material configurations.
- **dat.GUI**: Used to create an interactive developer and user panel for modifying environmental parameters such as light intensity, time of day, and camera sensitivity.
- **HTML/CSS/JavaScript**: HTML structures the UI, CSS handles styling for various gameplay elements such as the score panel and ESC menu, and JavaScript manages logic, event handling, and DOM integration.
- **Audio System (HTML5 Audio)**: Provides immersive sound effects synchronized with user actions (e.g., collecting a stone, launching the catapult).

3.2 Scene Composition and Objects

The environment consists of multiple game-ready 3D models:

- **Castle**: A detailed static mesh that forms the central architectural focus of the scene. Loaded via GLTFLoader and scaled appropriately.

Figure 3.2.1: Castle.glb

Figure 3.2.2: Castle.glb

- **Catapult**: A functional object with separate parts (base, arm, bucket) for animation. Capable of stone launching and sound interaction.

Figure 3.2.3: Catapult.glb

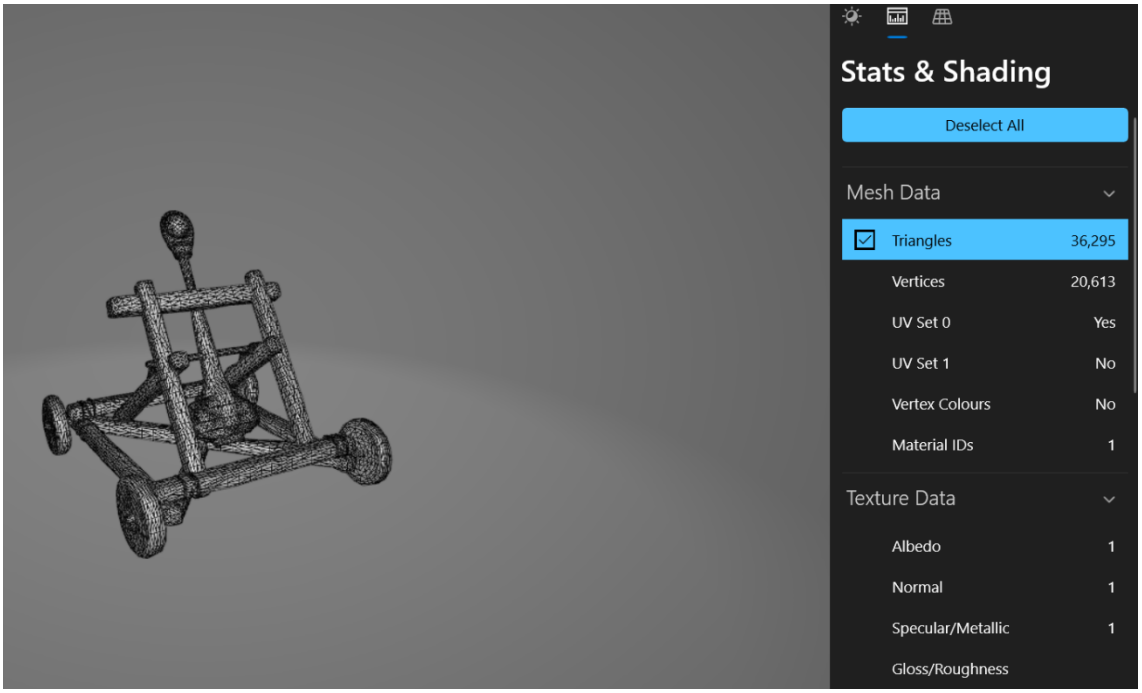

Figure 3.2.4: Catapult.glb

- **Stones**: Dynamic and interactive objects that can be collected and placed into the catapult. Their state changes based on user interactions.

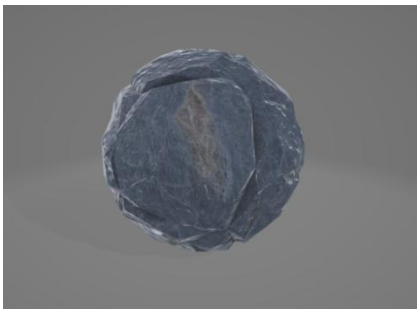

Figure 3.2.5: Stone.glb

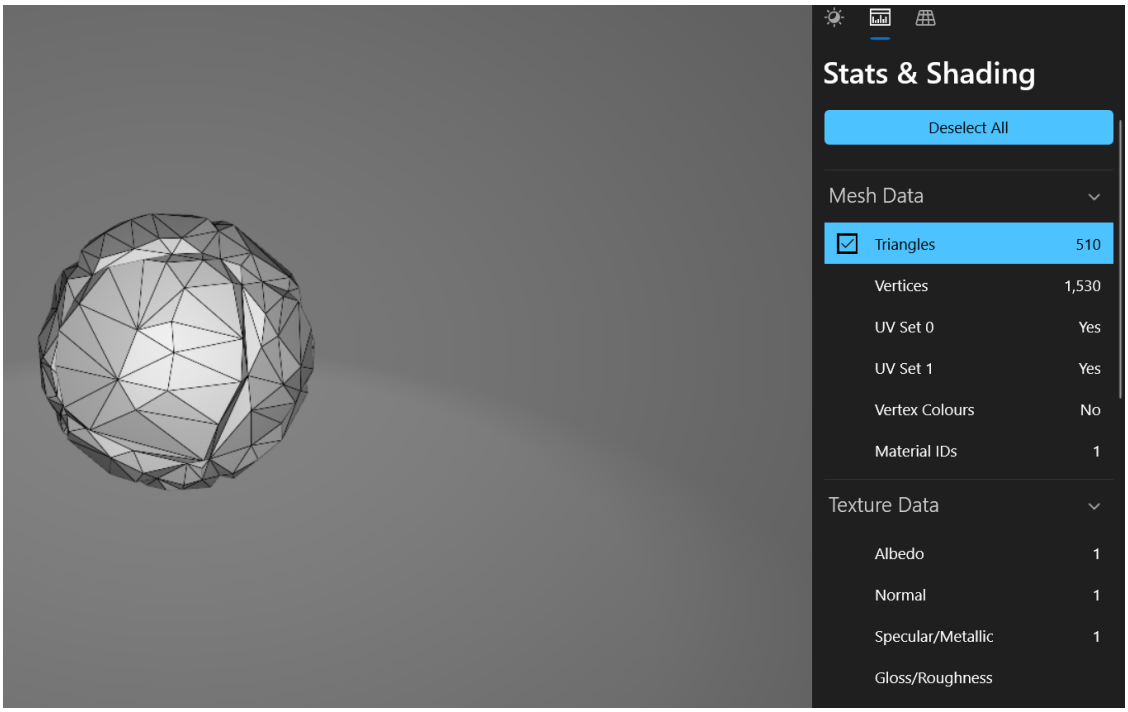

Figure 3.2.6: Stone.glb

- **Torches**: Light-emitting props that are either static (on the castle) or held by the player. Intensity and visibility are adjustable.

Figure 3.2.7: Torch.glb

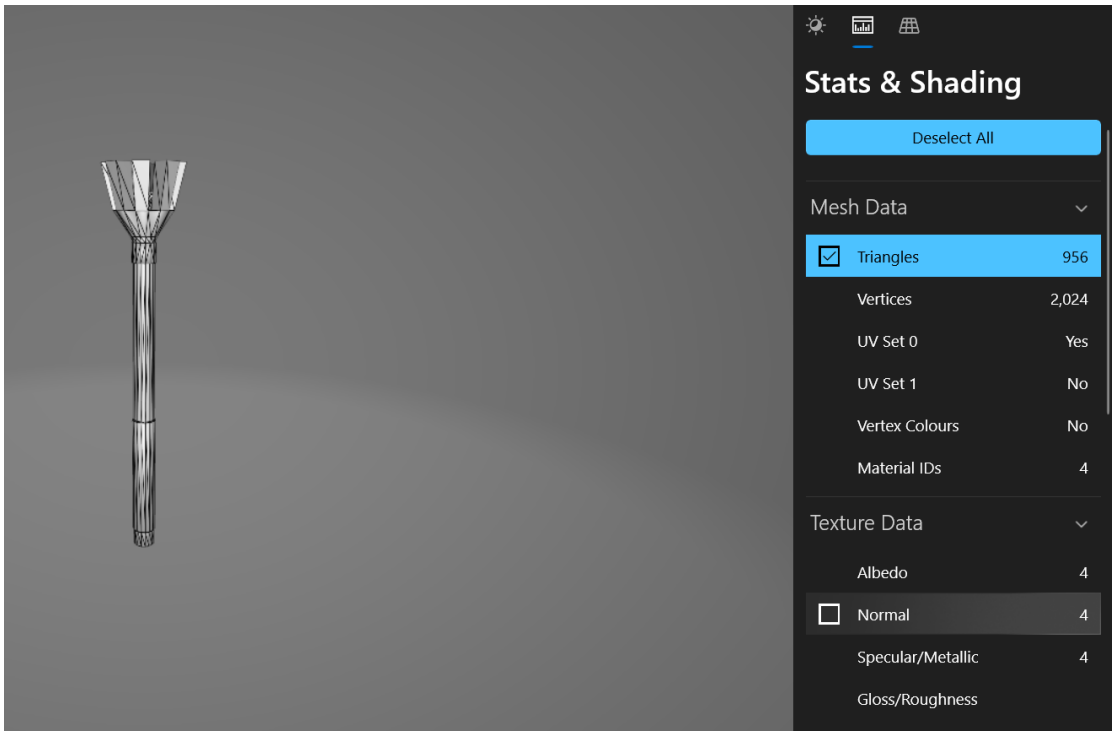

Figure 3.2.8: Torch.glb

3.3 User Interaction and Controls

- **First-Person Navigation**: The camera responds to WASD keys for translation and mouse input for rotation. PointerLock API enables immersive control.
- **Object Interaction**: Implemented using raycasting and bounding sphere detection to allow stone collection and object selection.

- **Power Bar Mechanic**: A visual UI element appears during stone launch preparation. Timing the bar precisely affects launch power and score.
- **Inventory and Scoring System**: Real-time UI updates inform players about their stone inventory status and cumulative score.
- **ESC Menu**: A full overlay panel that pauses the game and provides control instructions, time manipulation options, and visual explanations.

3.4 Lighting and Time System

- **Day/Night Cycle**: A dynamic light system simulates sun movement and transitions between day and night. Ambient and directional lighting are updated accordingly.
- **Torch Lighting**: Torches switch from decorative to primary light sources during nighttime. Light intensity can be adjusted via GUI or game settings.
- **Fog and Background**: Fog density and sky color change with time of day to enhance realism and visual feedback.

3.5 Sound Integration

- **catapult_fire_sound.mp3**: Synchronized with the stone launching event, volume scales based on power level.
- **stone_sound.mp3**: Played when a stone is collected from the ground, providing immediate auditory feedback.
- **Audio Manager Class**: Centralizes sound playback logic and allows easy management of volume levels and resource reuse.

4. COMPLIANCE WITH PROJECT REQUIREMENTS

This section provides a detailed overview of how the project aligns with the core requirements specified in the CENG376 Computer Graphics course guidelines. Each criterion is addressed not only in terms of technical fulfillment but also with regard to its integration into the overall design and user experience. The development team ensured that all fundamental expectations were met and often surpassed them by embedding features in a contextually meaningful and visually coherent way.

REQUIREMENT	STATUS
WEBGL-BASED PROJECT	Yes
3D SCENE (NON-2D CAMERA VIEW)	Yes
3-AXIS CAMERA MOVEMENT AND ROTATION	Yes
AT LEAST 3 OBJECT TYPES WITH DIFFERENT MORPHOLOGIES	Yes
AT LEAST 3 OBJECT TYPES WITH DIFFERENT TEXTURES	Yes
3+ CONTROLLABLE OBJECTS VIA USER INTERACTION	Yes
AT LEAST ONE VISIBLE LIGHT SOURCE	Yes
MOVABLE LIGHT SOURCE WITH ADJUSTABLE BRIGHTNESS	Yes

Table 4.1: Requirement Table

WebGL-based Project: The entire simulation was implemented using WebGL through the Three.js library, providing cross-platform browser support without requiring additional installations or plugins.

3D Scene and Perspective: The project features a fully immersive three-dimensional environment. All objects are rendered with perspective depth, and the camera supports a first-person viewpoint, giving players the feeling of walking within a medieval world.

Three-Axis Camera Movement: The user can navigate the scene using WASD keys and mouse input, enabling movement along the X, Y, and Z axes, with smooth camera rotation in all directions thanks to the PointerLockControls from Three.js.

Object Morphology and Texture Variety: The models used include a castle, torches, a catapult, and stones—all designed with distinct morphologies and textures. For instance, the castle features stone bricks, the torches have a wooden texture, and the stones have rough natural surfaces, which collectively provide visual diversity.

Interactive Objects: Players can interact with more than three different object types: they can collect stones, operate a catapult, and toggle lighting with torches. Each object responds to user actions through mouse events and real-time updates.

Lighting Requirements: A combination of ambient, directional, and point lights is used to illuminate the scene. Torches and sunlight serve as dynamic light sources that change based on the time of day and user inputs.

Adjustable Light Source: The player can toggle and adjust torch brightness and switch between day and night modes. These changes are reflected dynamically, affecting shadow casting and object visibility.

In summary, all minimum project requirements have been not only met but thoughtfully incorporated into gameplay mechanics and aesthetic decisions. This ensures that technical checkmarks translate into an engaging, interactive, and polished user experience.

5. ADDITIONAL FEATURES BEYOND REQUIREMENTS

In addition to satisfying all fundamental requirements specified by the course, the "Medieval Castle Scene" project incorporates a wide range of advanced features designed to enhance both technical quality and user engagement. These additions were implemented not as standalone modules but in a way that seamlessly integrates with the core gameplay and narrative structure of the scene.

5.1 Day/Night Cycle

The scene includes a dynamic day and night system that simulates the passage of time. The transition is smoothly animated, with interpolated changes to ambient light, directional light intensity, and sky background color. During the day, sunlight illuminates the scene, while at night, torches and player-held lights become the primary sources of visibility.

5.2 Time Slider

A time control slider was developed and integrated into the HTML interface. This slider allows players to manually adjust the time of day in the environment. The effects of these adjustments are visualized in real-time, including lighting changes, fog density, and sky tone. It serves both as a gameplay enhancement and as a technical demonstration of real-time environmental control.

5.3 Hand Torch Mechanism

The player is equipped with a controllable hand torch that can be toggled with the 'F' key. This feature enhances playability during night cycles or dark interior areas of the castle. The torch includes a dynamic point light and glow effect, both of which are updated in real-time based on the player’s movement and camera orientation.

5.4 Power Bar Interface

A power bar UI is displayed when preparing to launch a stone with the catapult. This gamified mechanic adds an element of timing and skill, as the effectiveness of the launch depends on when the user clicks during the power bar’s animation. The visual interface is implemented using CSS transitions and canvas drawing logic.

5.5 Inventory and Score Panel

Two separate UI components allow the user to keep track of their collected stones and successful hits. The inventory panel displays the number of stones available, while the score panel updates with each successful launch. Both panels are designed to update dynamically without reloading the interface.

5.6 Settings and Help Menus

An in-game menu accessible via the ESC key provides settings for toggling the day/night cycle, restarting the scene, and viewing help instructions. The menu overlays the canvas and temporarily disables pointer lock to facilitate interaction. It improves accessibility and helps users understand controls more effectively.

5.7 Sound System

Auditory feedback plays a crucial role in enhancing user immersion. Specific sound effects are triggered for key interactions, such as picking up a stone and launching it from the catapult. The audio system supports playback control, looping, and conditional volume adjustments. Sounds are preloaded and managed through a centralized sound manager script.

5.8 Performance Optimization

To ensure the application runs smoothly across devices, several performance-focused strategies were implemented:

- **Frustum Culling**: Objects outside the camera's view are ignored to save GPU processing time.
- **Scale Normalization**: All imported models were adjusted to consistent scale units to prevent rendering artifacts.
- **Texture Filtering**: Mipmapping and filtering settings were applied to textures to reduce aliasing and memory usage.
- **Efficient Scene Graph Management**: Objects are hierarchically grouped and updated using minimal change detection to avoid unnecessary rendering cycles.

These additional features collectively contribute to a more engaging, responsive, and technically robust experience that exceeds the base expectations of a course project.

6. CHALLENGES AND SOLUTIONS

During the development of the "Medieval Castle Scene" project, several technical and usability challenges were encountered. This section elaborates on the nature of those challenges and provides insight into the strategies and solutions adopted to overcome them in a structured and performance-conscious manner.

6.1 Dynamic Lighting Performance

One of the most significant technical challenges was managing the computational overhead caused by real-time lighting and shadow rendering. Given the use of dynamic light sources—such as torches, ambient light, and a moving directional light to simulate sun movement—maintaining frame stability was a major concern, particularly during transitions between day and night.

- **Solution**: To mitigate this, lighting updates were regulated using timed intervals and smooth interpolation algorithms. Additionally, shadow map resolution was adjusted dynamically based on the camera's proximity to light sources, reducing unnecessary GPU workload.

6.2 Pointer Lock and ESC Menu Conflict

Another usability issue emerged from the conflict between the game's immersive first-person pointer lock system and the ESC-based menu interactions. Players were initially confused when attempting to pause the game or access settings, as the pointer lock state prevented normal mouse interaction.

- **Solution**: This was resolved by implementing a clean toggle system that unlocks the pointer upon ESC key press and re-locks it upon returning to gameplay. In-game hints and help panels were also added to guide users through the interaction flow.

6.3 Performance Impact of High-Poly Models

The inclusion of detailed .glb models such as the castle, catapult, and torches initially led to significant loading times and runtime lag. These high-poly assets, while visually impressive, strained the rendering pipeline on less powerful machines.

- **Solution**: Model complexity was optimized by reducing polygon counts using mesh decimation tools. Unused texture maps were removed, and all model assets were re-exported with simplified materials. In addition, texture sizes were standardized to power-of-two dimensions for optimal GPU compatibility.

6.4 Synchronization of Power Bar and Catapult Mechanics

The power bar system required precise synchronization with the catapult's animation sequence to deliver responsive and intuitive gameplay. Desynchronization led to inconsistencies in launch strength and gameplay logic.

- **Solution**: A dedicated asynchronous state handler was implemented to manage the entire firing sequence. Callback functions were employed to monitor animation states, ensuring

that stone launch events were only triggered upon animation completion. This resolved all timing issues and enhanced gameplay accuracy.

In summary, each of the challenges encountered during development served as an opportunity to improve the system's stability, responsiveness, and user experience. The applied solutions not only resolved technical limitations but also contributed to a more professional and polished final product.

7. CONCLUSION

The "Medieval Castle Scene" project effectively demonstrates the comprehensive application of WebGL technologies in a fully interactive 3D environment. All minimum project requirements were met and extended with significant enhancements. The technical integration of lighting, sound, gameplay mechanics, and GUI control results in a polished, engaging simulation. The teamwork, planning, and implementation strategies involved reflect both technical skill and user-centric design. Overall, this project serves as a successful capstone in the study of interactive computer graphics.

REFERENCES

[1] Khronos Group. (n.d.). *glTF Overview*. GitHub. https://github.com/KhronosGroup/glTF

[2] dataarts. (n.d.). *dat.GUI*. GitHub. https://github.com/dataarts/dat.gui

[3] Mozilla Contributors. (n.d.). *HTMLAudioElement*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/HTMLAudioElement

[4] Three.js Developers. (n.d.). *Three.js Documentation*. https://threejs.org/docs/

APPENDIX

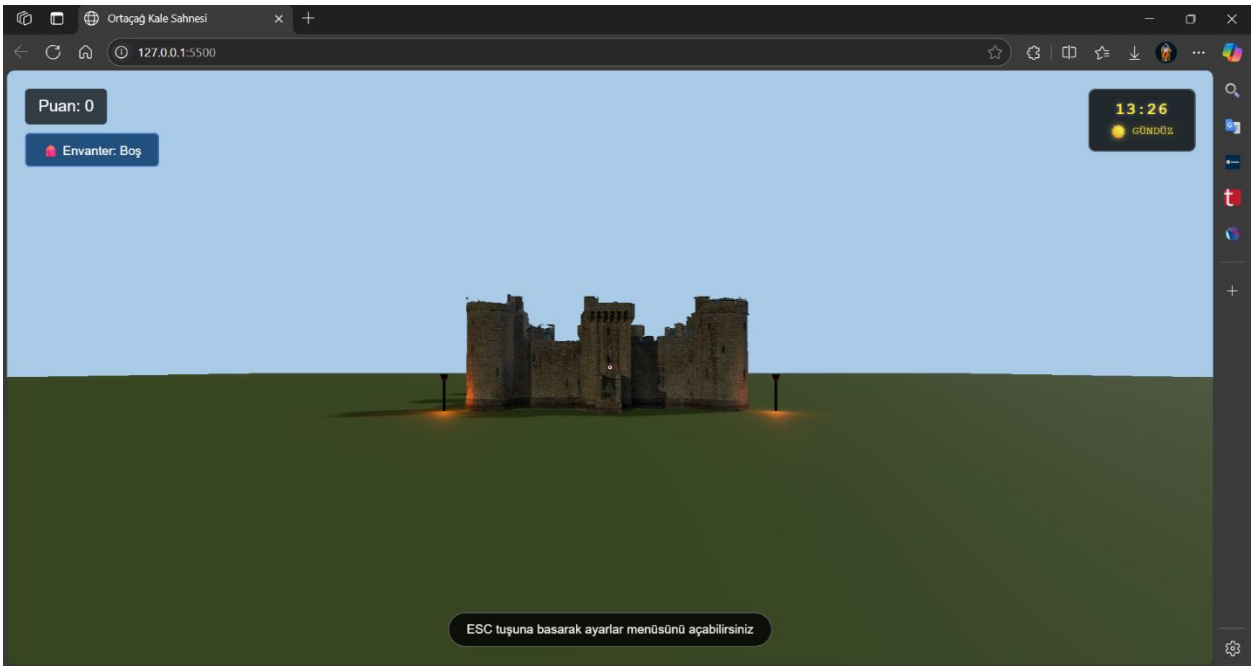

Figure A.1: General Overview

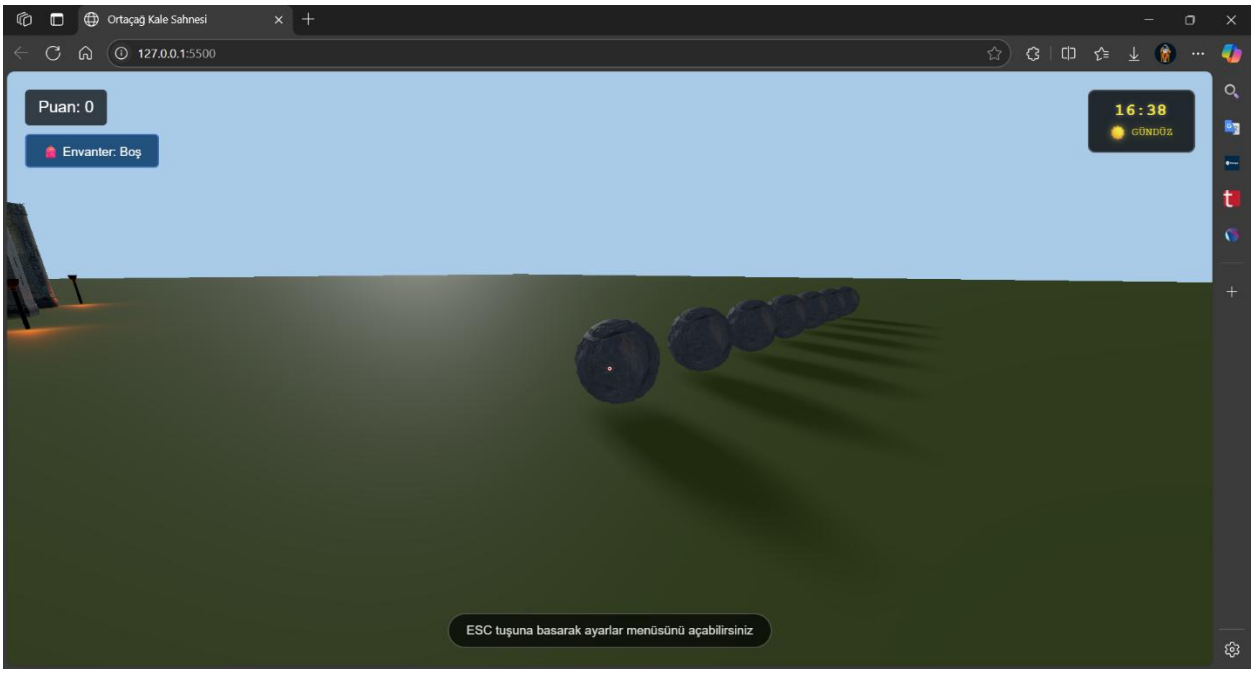

Figure A.2: General Overview – Stones and Shadows

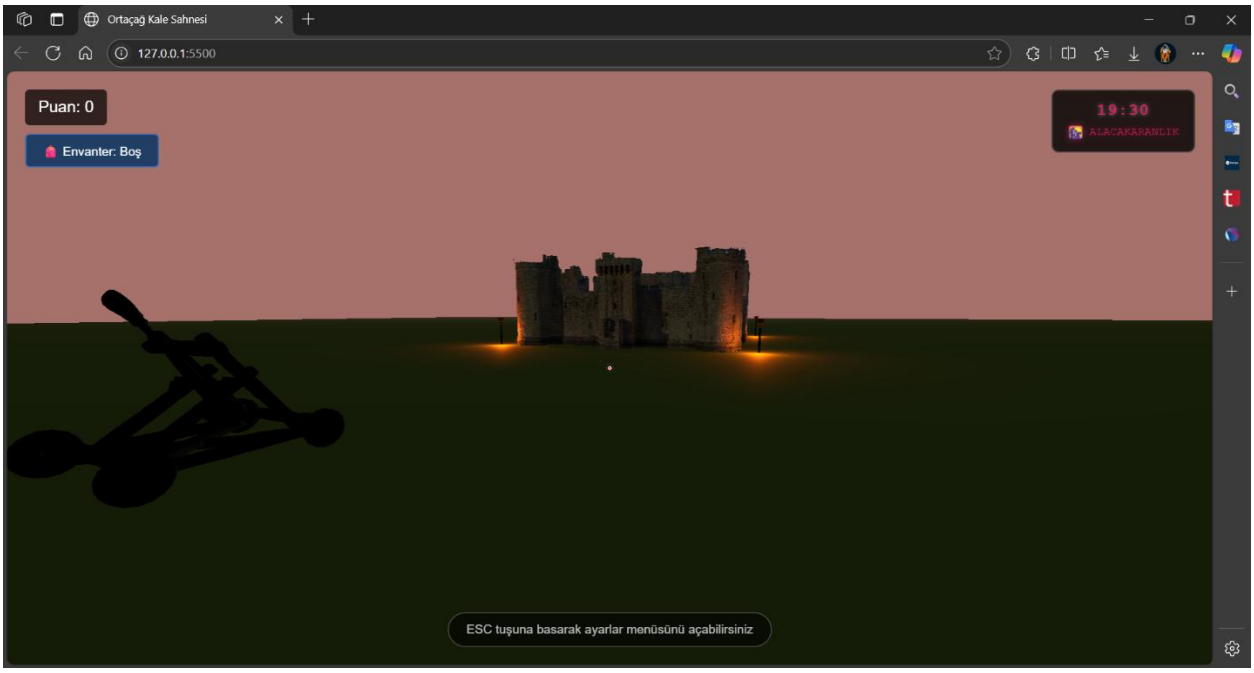

Figure A.3: General Overview – Night

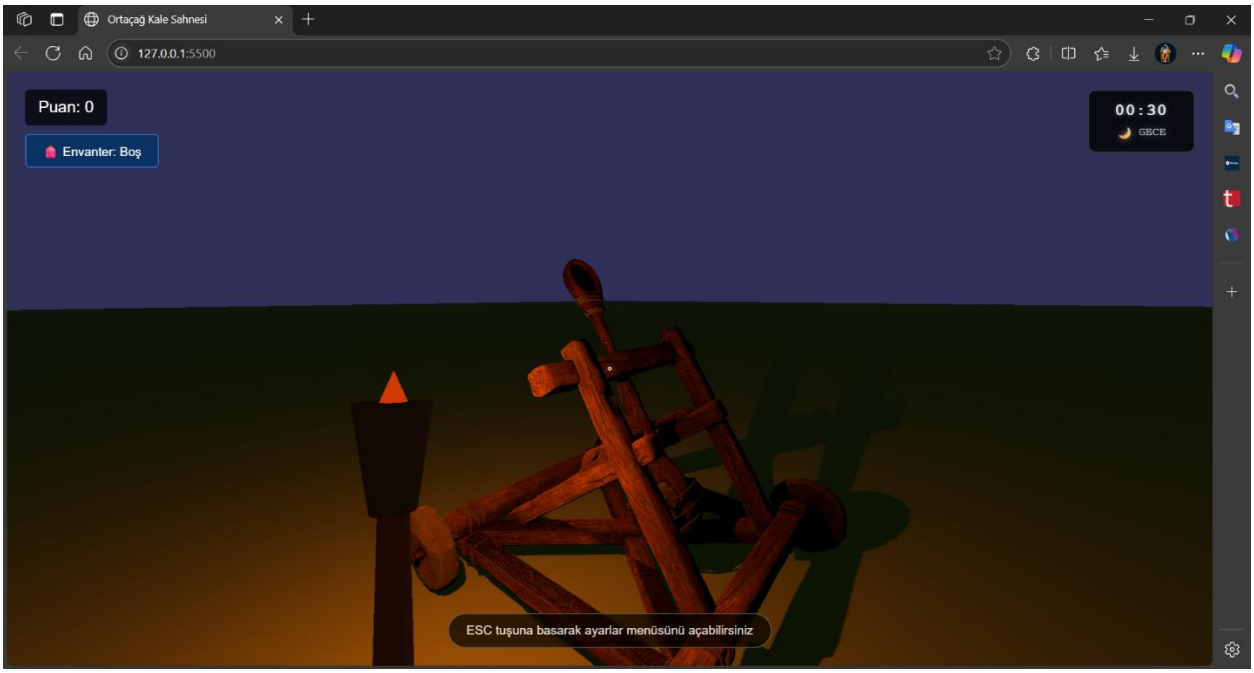

Figure A.4: Torch and Shadow Physics

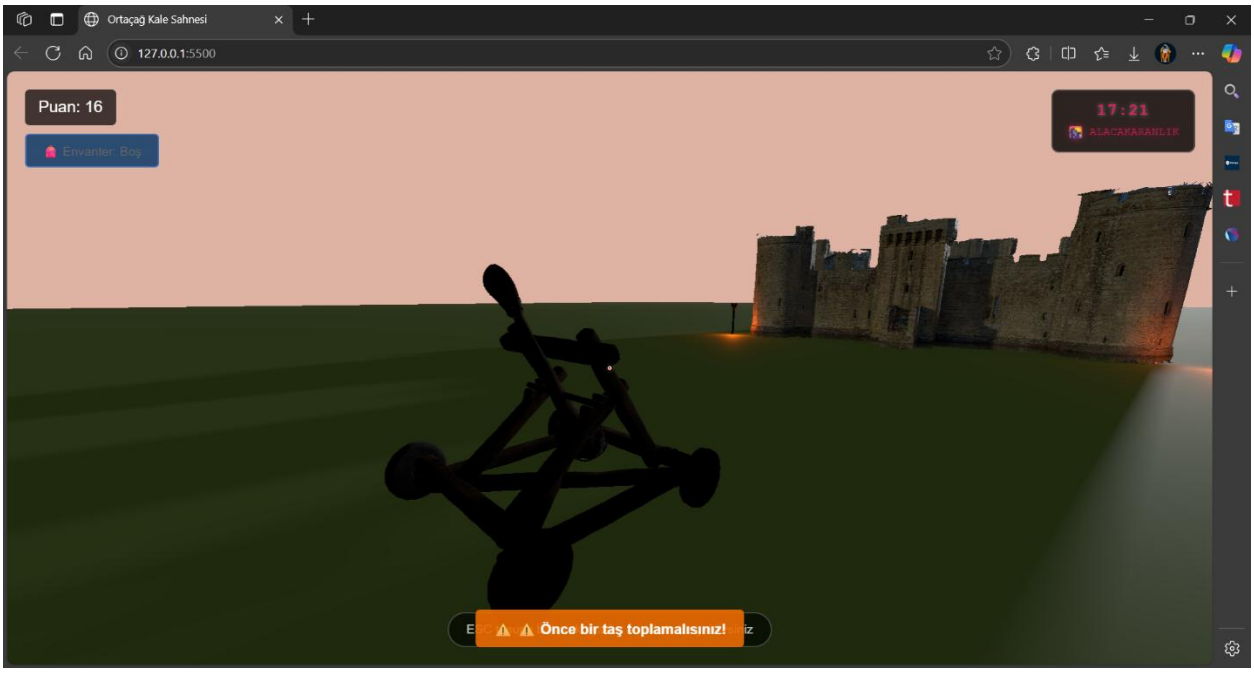

Figure A.5: Warning if no stone

Figure A.6: Information when the stone is collected

Figure A.7: Inventory warning: collect gems when inventory is full

Figure A.10: Power Bar, Red: -10 points, Yellow: +7 points, Green: +15 points

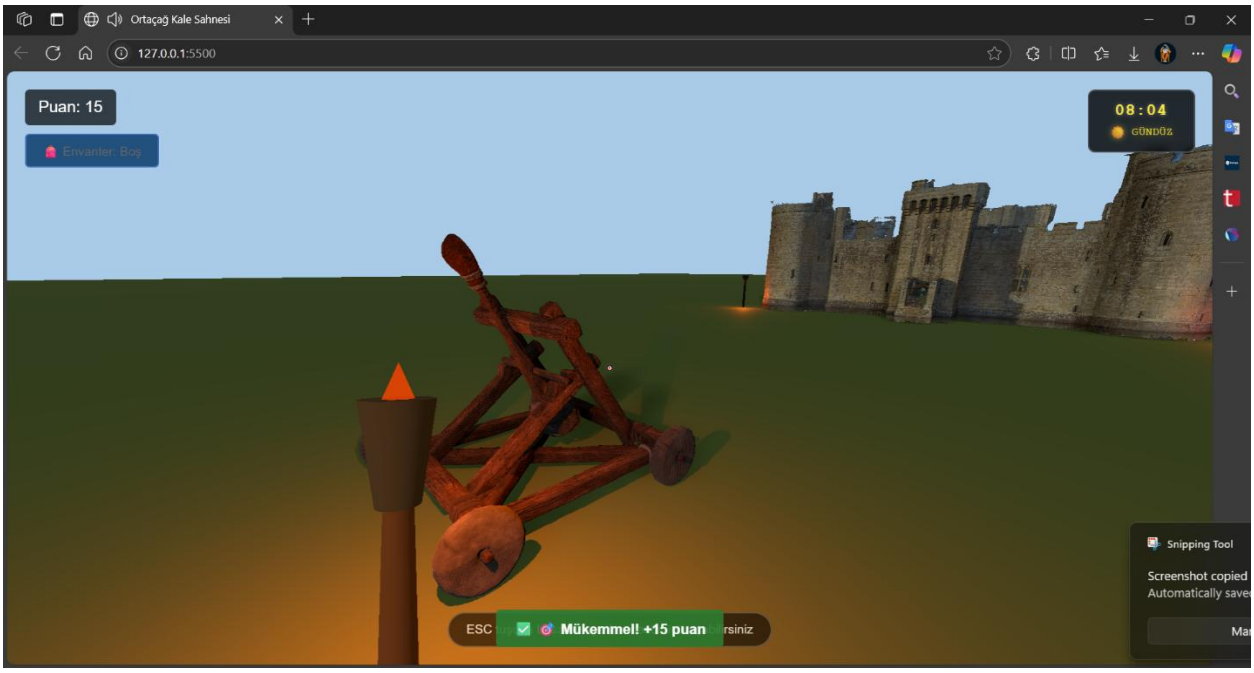

Figure A.11: Catapult shot fired

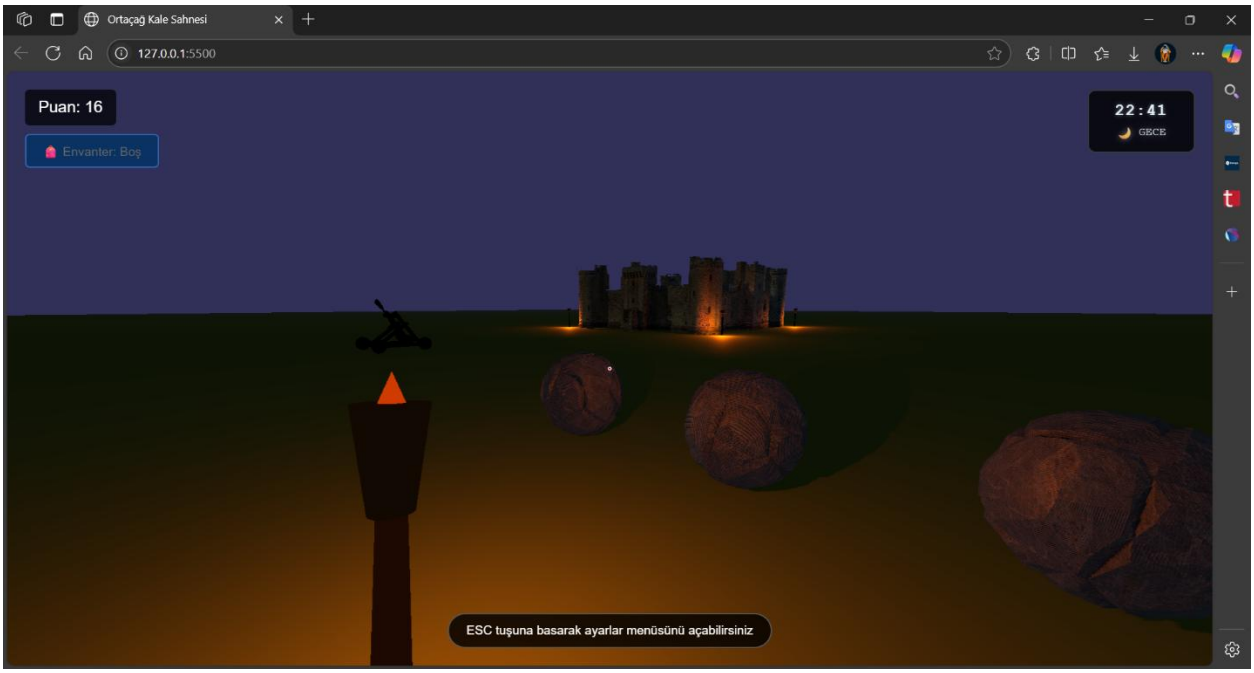

Figure A.12: Night view

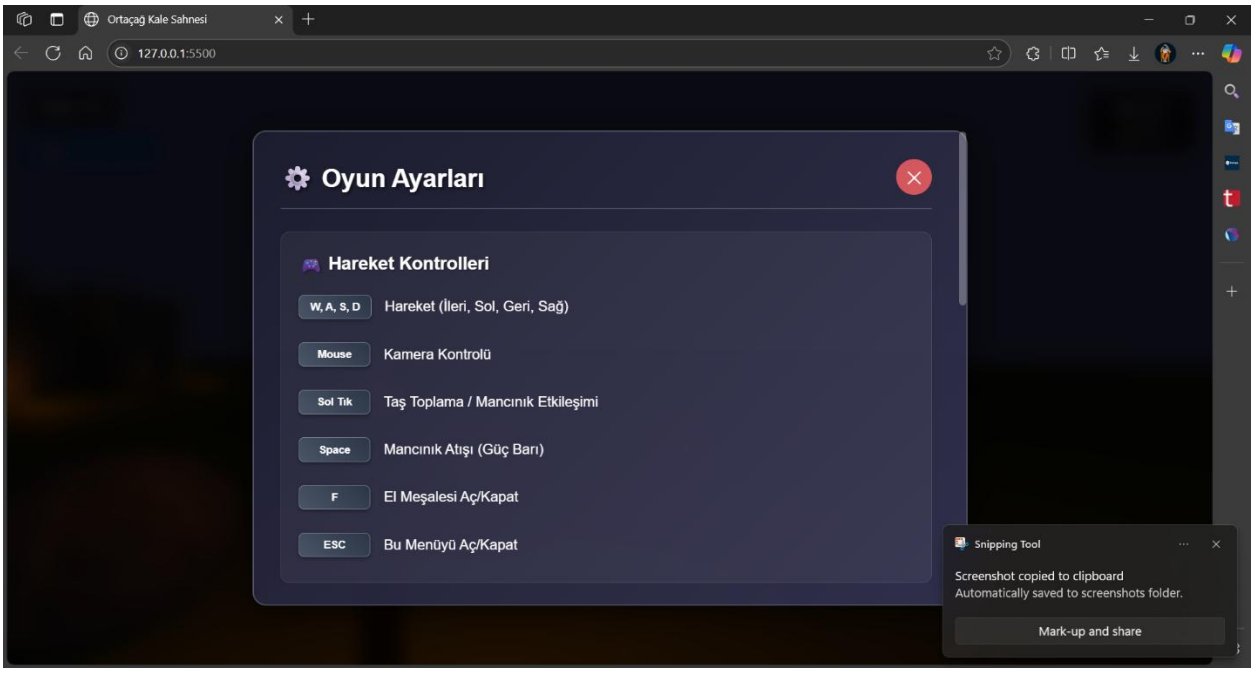

Figure A.13: Menu Screen-1 after pressing ESC

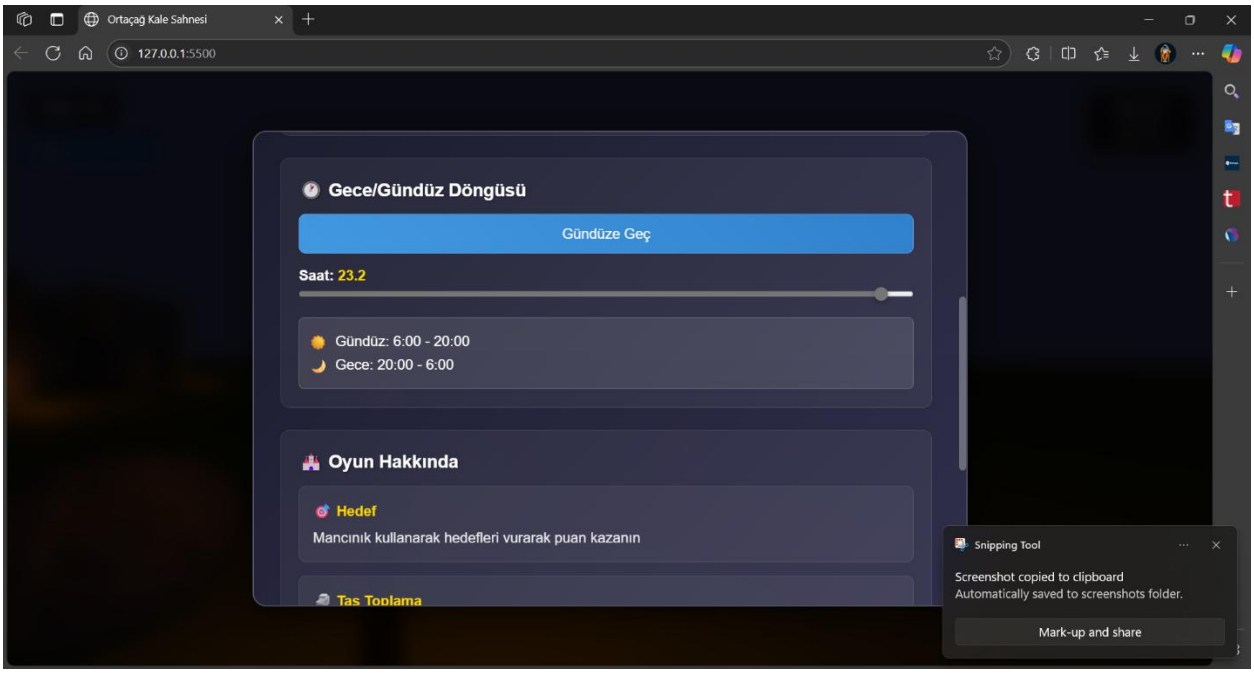

Figure A.14: Menu Screen-2 after pressing ESC

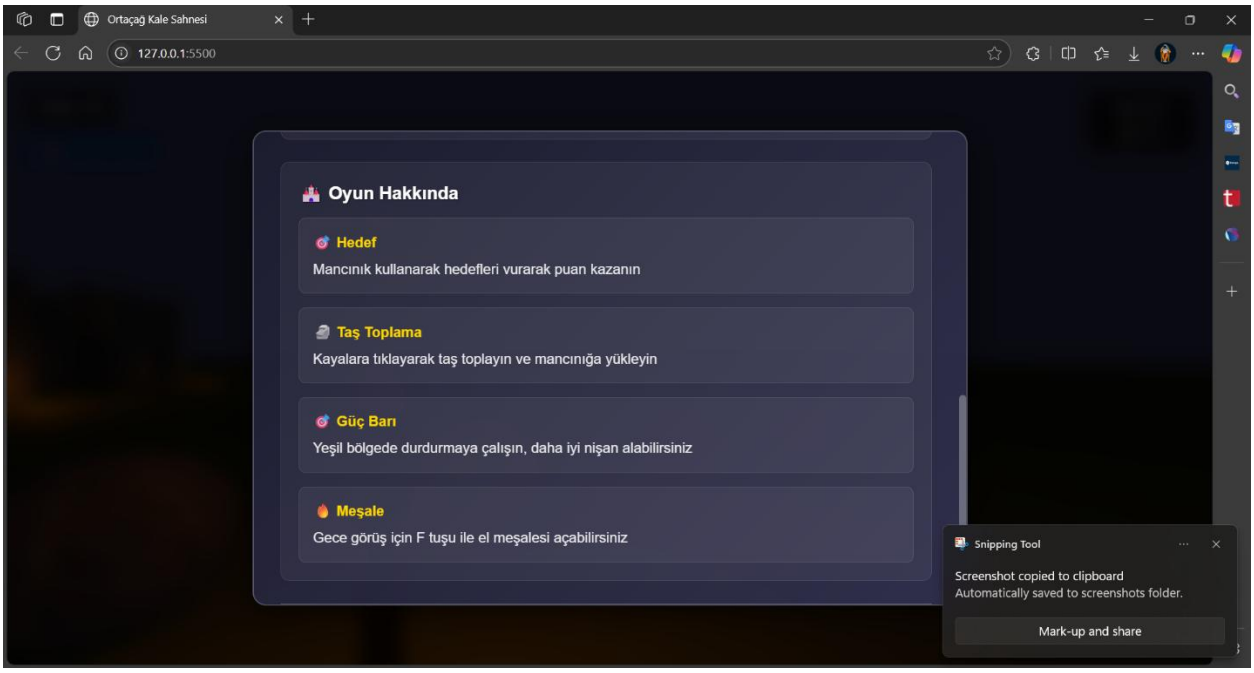

Figure A.15: Menu Screen-3 after pressing ESC