CMPE 492

DeepGameExtractor: Harnessing Game Engines for AI Training Data Generation

Ömer Faruk Çelik

Cahid Enes Keleş

Advisor:

Mehmet Turan

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Broad Impact

In today's world of artificial intelligence (AI) and machine learning (ML), having access to a lot of good and varied data is crucial. Being able to use relevant and large amounts of data is key to making strong AI models. This project, focusing on the automated aggregation of game screenshots for AI model training, addresses a critical need within the AI development community, particularly in the domains of game development, computer vision, and interactive media.

DeepGameExtractor does more than just gather images. It marks a smart step forward in how developers and researchers train AI models. By automating the process of data collection, this project significantly reduces the time and effort required to gather diverse datasets. This efficiency allows for a more rapid iteration of AI models, which promotes innovation and improvement in AI applications within gaming and other visual-based domains.

Moreover, the inclusion of a user interface (UI) that guides users through the necessary pre-configuration steps adds a layer of accessibility and user-friendliness to the tool. This also makes training AI more accessible to more people by simplifying the process. Plus, it makes sure that the collected data is reliable and of good quality. Being able to choose specific settings and configurations makes the tool even more useful, as it lets people create custom datasets that fit exactly what different AI projects need.

The broad impact of this project is diverse:

(i) **Enhancement of AI Model Training**: It offers a streamlined method to collect game screenshots, improving AI training, especially for visual data.
(ii) **Support for Research and Development**: It provides an efficient way for

researchers and developers in game development, computer vision, and similar fields to gather specific datasets, speeding up innovation.

(iii) **Accessibility and Democratization**: Making data collection process accessible to a wider audience promotes inclusivity and democratization in the field of AI development, empowering more individuals and organizations to participate in AI research and application development.

(iv) **Quality and Consistency in Data Collection**: The project's emphasis on pre-configuration and user guidance ensures that the datasets generated are not only of high quality but also consistent, which is crucial for the training of effective AI models.

In conclusion, creating a tool that automatically gathers game screenshots is a big advancement in artificial intelligence. This tool is likely to benefit many areas, making AI models better, helping research move forward, and leading to a more open and efficient way of developing AI.

1.2. Ethical Considerations

The development of automated data aggregation tools, especially in the context of collecting game screenshots for AI model training, raise several ethical considerations that must be addressed to ensure responsible use and minimize potential harm. This section outlines key ethical considerations associated with this project:

- **Privacy and Consent**: It is important to consider the privacy implications of aggregating screenshots from games, particularly when these images may include player-generated content or potentially sensitive information. Ensuring that data is collected in a manner that respects user privacy and includes appropriate consent mechanisms is crucial.
- **Intellectual Property Rights**: The collection and use of game screenshots for training AI models must carefully consider the intellectual property rights. This includes obtaining necessary permissions from game developers and publishers,

especially when screenshots involve copyrighted material like game assets.

- **Bias and Fairness**: Given that the AI models trained with these screenshots will reflect the data they are trained on, it is important to consider the diversity and representativeness of the collected data.

Addressing these ethical considerations requires a proactive approach, including the development of guidelines for ethical data collection and the incorporation of ethical principles into the design and implementation of the tool. By prioritizing ethical considerations, this project can contribute positively to the field of AI and ensure that the benefits of automated data aggregation are achieved in a responsible and fair manner.

2. PROJECT DEFINITION AND PLANNING

2.1. Project Definition

DeepGameExtractor is an innovative tool designed to automate the collection of game screenshots, aiming to enrich the pool of training data for artificial intelligence (AI) and machine learning (ML) models. By leveraging the dynamic and visually diverse environments found in video games, this project addresses the critical need for high-quality, varied datasets in AI research, particularly in the field of computer vision. The purpose of DeepGameExtractor is to streamline the data collection process, making it faster and more efficient. DeepGameExtractor focuses on simplifying the process for users to configure and collect the data they need. DeepGameExtractor also aims to facilitate the development of more accurate and efficient AI applications across a range of fields. In simple terms, DeepGameExtractor aims to revolutionize the creation of training data, marking a significant advancement in AI training.

2.2. Project Planning

2.2.1. Project Time and Resource Estimation

The DeepGameExtractor project, scheduled to finish in the spring term, is a collaborative effort by Cahid and Omer. The project schedule is arranged to include different stages of development, such as starting research, planning, building, and checking. Considering the limited time, the project is expected to take about four months from start to finish. Resources are primarily allocated towards software development tools, access to game libraries for screenshot collection, and computational resources for testing. Each team member will dedicate approximately 5 hours per week, balancing project tasks with academic responsibilities. Collaboration tools and version control systems will be utilized to maximize efficiency and facilitate seamless teamwork.

2.2.2. Success Criteria

Success for the DeepGameExtractor project is defined by several key criteria:

- Development of a fully functional tool capable of automatically capturing game screenshots based on specified user settings.
- Creation of a user-friendly interface that simplifies the configuration process for data collection.
- Demonstration of the tool's ability to collect a diverse set of high-quality images that can be directly used for AI model training.
- Positive feedback from a test group of users on the tool's ease of use and effectiveness in generating relevant training data.
- Completion of the project within the allocated timeframe and resource budget.

2.2.3. Risk Analysis

Several risks have been identified that could potentially impact the DeepGameExtractor project:

- **Technical Challenges:** The integration of the tool with various games may present unforeseen technical difficulties, requiring additional time to troubleshoot and resolve. Also, the aim that the tool should work on different computers and operating systems pose serious challenges and may result in failure.
- **Resource Constraints:** Limited access to certain games or computational resources could hinder the project's progress and affect data collection efforts.
- **Time Management:** Balancing project development with academic responsibilities, holding a job, may lead to time constraints that impact the achievement of project milestones.
- **Data Quality:** Ensuring the collected screenshots are of high quality and relevant to AI training needs is crucial. Inadequate data quality could compromise the effectiveness of the tool.

- **User Acceptance:** The tool must meet user expectations in terms of functionality and ease of use. Failure to align with user needs could limit its adoption and utility.

To counter these risks, we will implement mitigation strategies such as adaptable scheduling, consistent progress evaluations, and collaboration with teachers and assistants for feedback and testing.

2.2.4. Team Work

The DeepGameExtractor project is conducted by Cahid and Omer, each bringing own skills and perspectives to the table. Our teamwork strategy is built on clear communication, mutual respect, and a shared dedication to the project's success, strengthened by support from our assistants and advisor.

Our collaboration is organized to maximize efficiency by assigning tasks according to individual strengths and maintaining the flexibility to help each other as needed. We hold frequent meetings to review progress, tackle challenges, and plan future tasks. These meetings also provide a space for brainstorming, solving problems, and aligning with our project objectives.

2.3. Related Work

The idea of using video games to gather data for artificial intelligence and machine learning research isn't entirely new, but it's still not widely explored, even though it has a lot of potential. Lately, the diversity and complexity of video game environments have been recognized for their utility in training more sophisticated AI models. This section examines various literature and projects that align with the goals of DeepGameExtractor, providing valuable insights to enhance our understanding of the project.

2.3.1. Automating Content Analysis of Video Games

In our endeavor to develop an automated screenshot collection tool for video games, the research conducted by Bullen, Katchabaw, and Dyer-Witheford on automating content analysis of video games serves as a relevant reference point [1]. While their focus lies on automating the content analysis process through software instrumentation, their insights into the challenges of manual data collection and analysis within the gaming industry resonate with our objectives. By leveraging automation techniques, their work demonstrates the potential to streamline labor-intensive tasks and enhance the efficiency of data collection and analysis procedures. Although our project specifically targets the automated collection of screenshots rather than content analysis, the foundational principles of automation and efficiency remain consistent. By building upon the methodologies and approaches outlined in their research, we aim to develop a robust and user-friendly tool that automates the process of capturing screenshots during gameplay, thereby contributing to the advancement of data collection techniques in the field of video game research.

2.3.2. Automating Rocket League Data Collection

Jonathan Collins introduces an innovative project, "Automating Rocket League Data Collection," aiming to streamline data gathering for gameplay analysis. Leveraging Python and Power BI, Collins automates the process, integrating a RL plugin with Ballchasing.com to export game replays into usable CSV files. Despite a manual step in downloading files, Collins anticipates future automation [2]. Python scripts handle data transformation efficiently, with safety measures to ensure data integrity. The resulting Power BI dashboard showcases key performance indicators (KPIs), facilitating data-driven insights despite sharing limitations. Collins highlights the importance of nuanced interpretation alongside quantitative analysis for informed decision-making. This project underscores the value of automation and visualization in leveraging data for competitive gaming analysis.

3. Methodology

The methodology for developing the DeepGameExtractor tool was guided by a series of user and system requirements aimed at creating a robust and versatile application for the automated collection of game screenshots for AI model training.

3.1. Requirement Analysis

The initial phase involved a thorough analysis of user and system requirements. The functional requirements specified that the tool must allow users to add configurations for different games, automatically obtain screenshots from these configured games, and interact with the system through a Graphical User Interface (GUI). Non-functional requirements emphasized the need for compatibility across various machines and operating systems.

3.2. Design

The design process was structured into several distinct parts, represented through a series of UML diagrams:

- A **Use Case Diagram** provided a high-level overview of the system's functionality from the user's perspective.
- An **Information Structure Diagram** detailed the system's data architecture and the relationship between configurations, settings, and screenshots.
- **Activity and Sequence Diagrams** outlined the operational workflow and interaction between the user, program, and data storage.
- A **Class Diagram** showcased the object-oriented structure, detailing the classes and methods that make up the system.
- The **Module Diagram** depicted the system's modular architecture, crucial for maintaining a clean separation of concerns and ensuring system extensibility.

- Finally, the **User Interface Design** was drafted to visualize the interaction points for users, focusing on simplicity and usability.

3.2.1. Implementation

The implementation phase will be focused on:

(i) Building the core modules as laid out in the design phase.
(ii) Implementing a GUI using a suitable framework that supports cross-platform functionality.
(iii) Integrating the GUI with the backend logic to manage game configurations, screenshot extraction, and data management.
(iv) Testing the tool across different games and settings to ensure that all user requirements were met.

3.3. Testing

Rigorous testing will be conducted to ensure that the DeepGameExtractor tool meets all functional and non-functional requirements. The primary methods of testing include:

- Manual testing throughout the system to ensure overall workflow correctness.
- Compatibility testing across different operating systems.
- Testing on a variety of games and scenarios to verify the tool's adaptability and robustness.

The testing phase is crucial for identifying and resolving issues early on, thus enhancing the reliability and usability of the DeepGameExtractor tool.

3.4. Tools and Technologies

The following tools and technologies were employed in the development of DeepGame-Extractor:

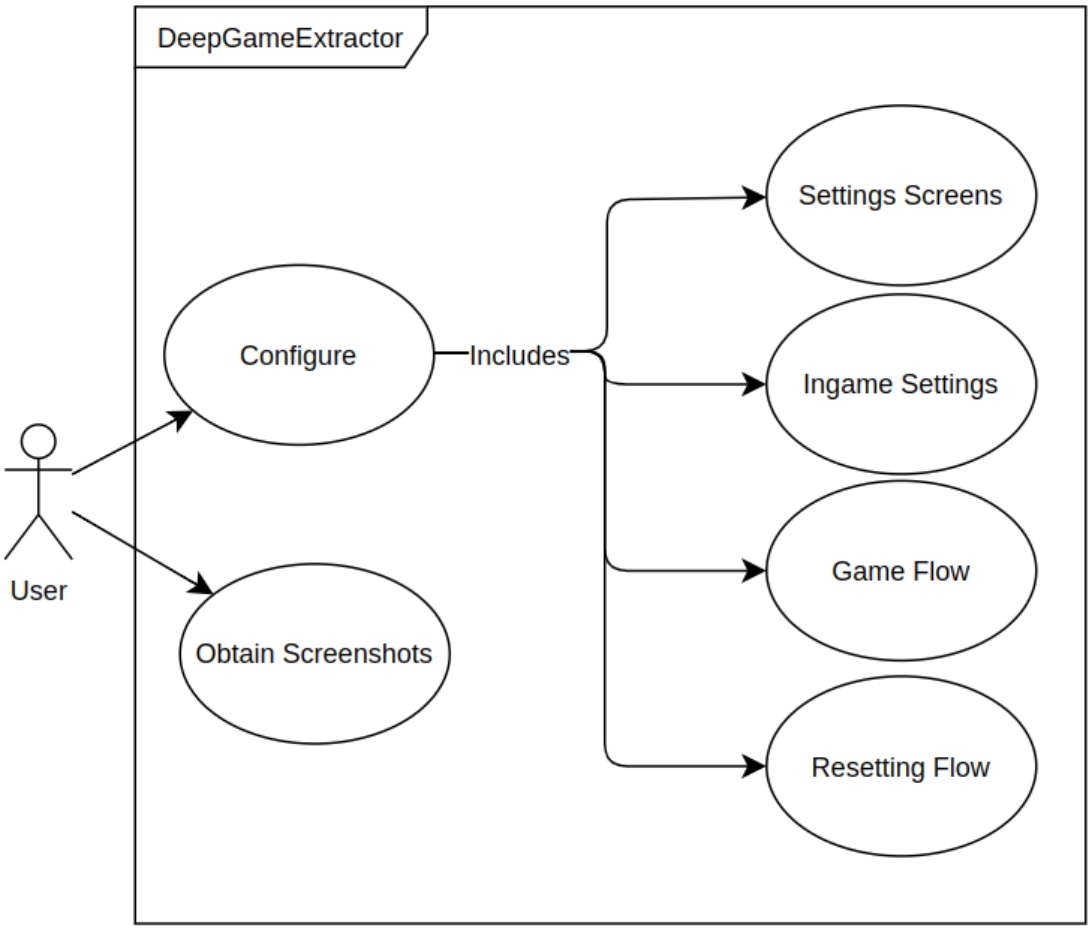
- UML tools for diagramming and design documentation.
- A programming language well-suited for cross-platform development, mainly Python.
- PyGame library for GUI development.
- pynput library for interacting with games and obtaining screenshots
- Version control system Git for source code management.

4. REQUIREMENTS SPECIFICATION

4.1. Requirements

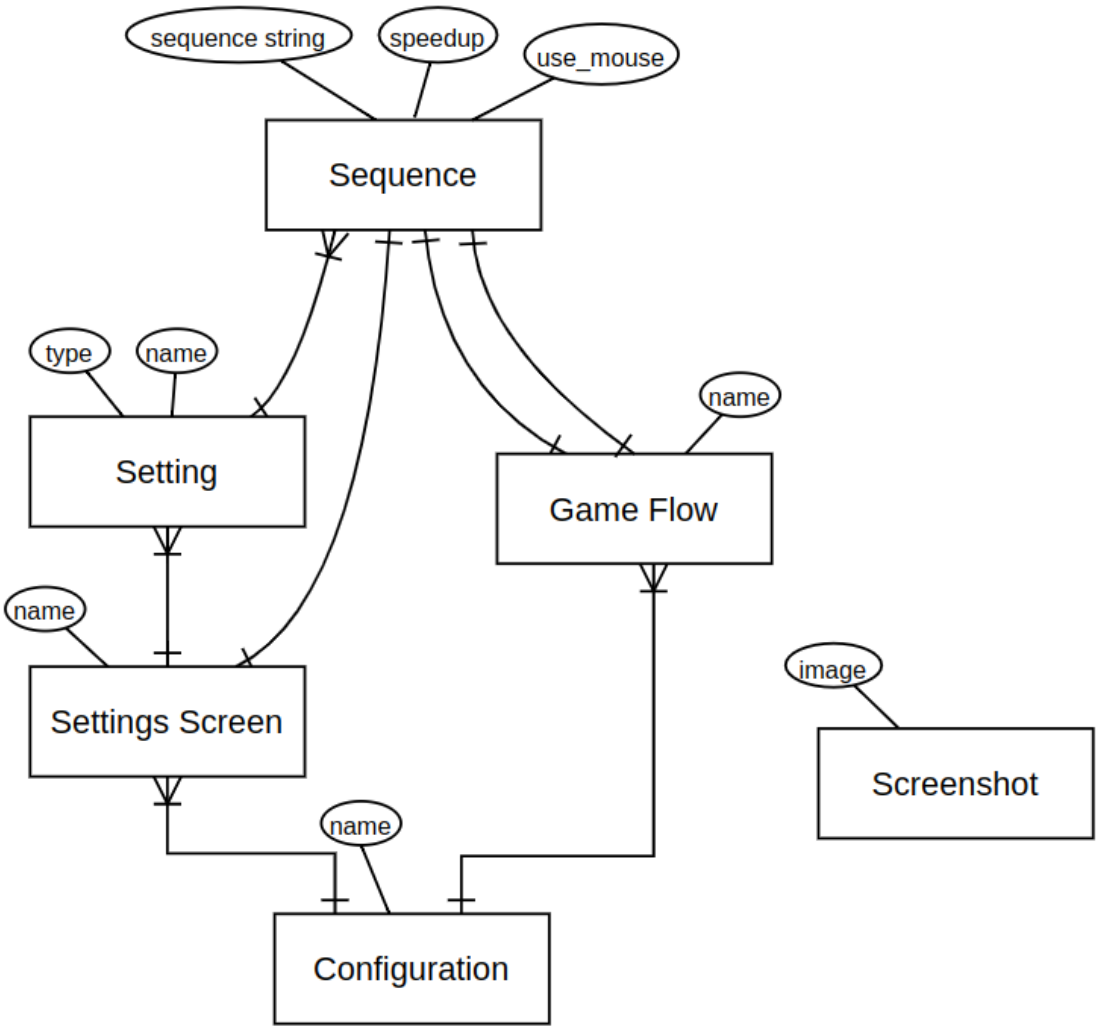
- **1.** Functional Requirements
  - **1.1.** User Requirements
    - **1.1.1.** Users should be able to add configurations to extract images from different games.
    - **1.1.2.** Users should be able to save configurations and recordings.
    - **1.1.3.** Users should be able to obtain screenshots automatically from a configured game.
    - **1.1.4.** Users should be able to interact with the program using a Graphical User Interface.
    - **1.1.5.** Users should be able to switch between light and dark themes.
  - **1.2.** System Requirements
    - **1.2.1.** Configurations
      - **1.2.1.1.** Configurations should include entering different settings screens.
      - **1.2.1.2.** Configurations should include changing any in-game setting.
      - **1.2.1.3.** Configurations should include simulating game flow.
      - **1.2.1.4.** Any recording should be speed-uppable.
      - **1.2.1.5.** Any recording should be able to record with or without mouse movements.
- **2.** Non-Functional Requirements
  - **2.1.** The tool should be working on different machines and different operating systems

4.2. Use Case Diagram



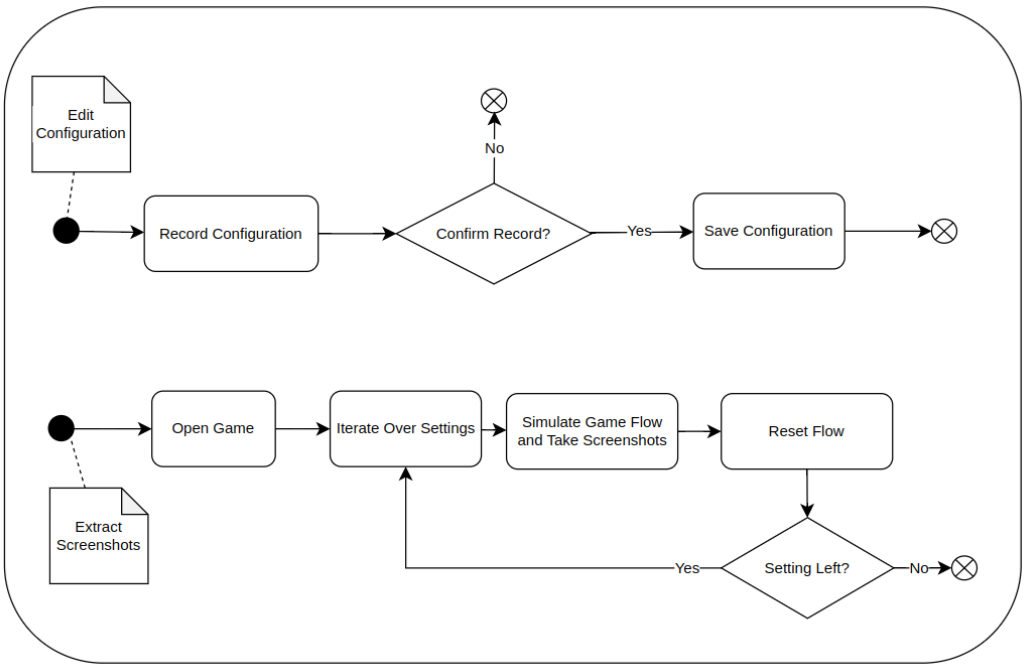
5. DESIGN

5.1. Information Structure

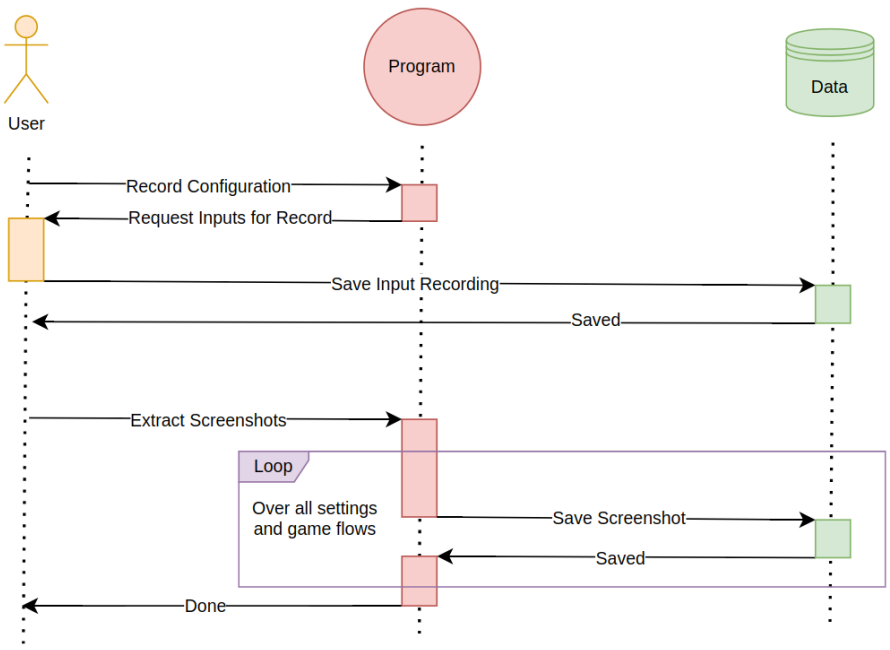


5.2. Information Flow

5.2.1. Activity Diagram

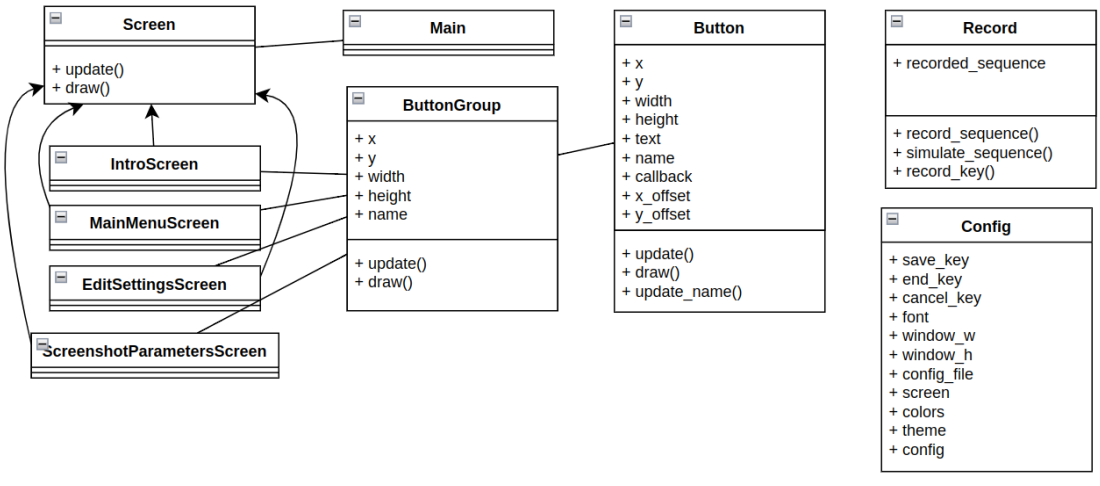


5.2.2. Sequence Diagram

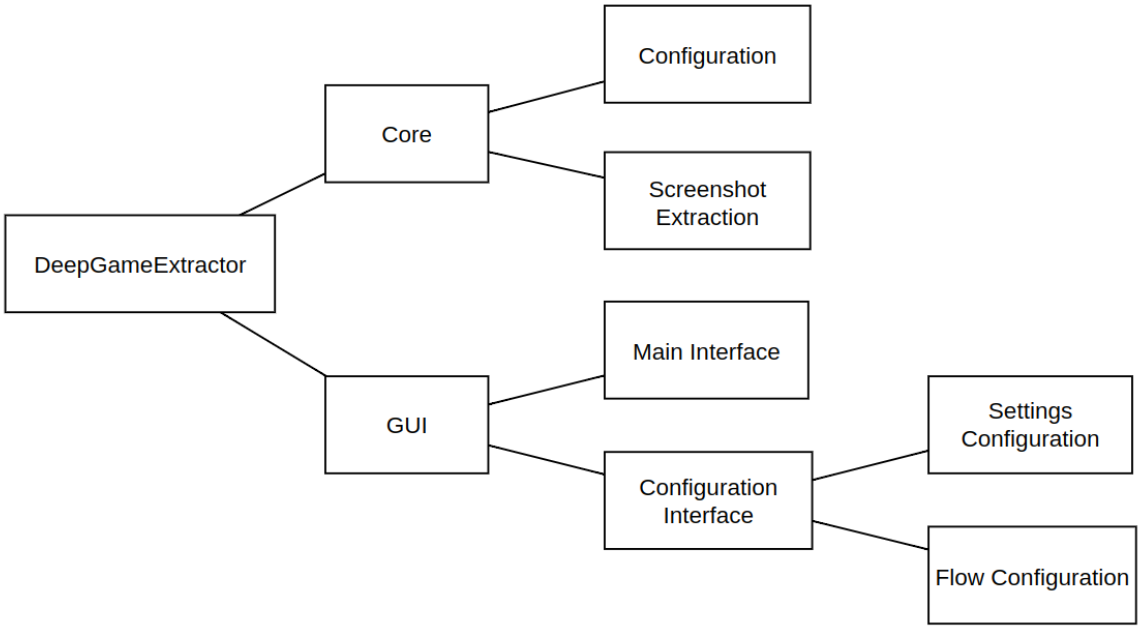


5.3. System Design

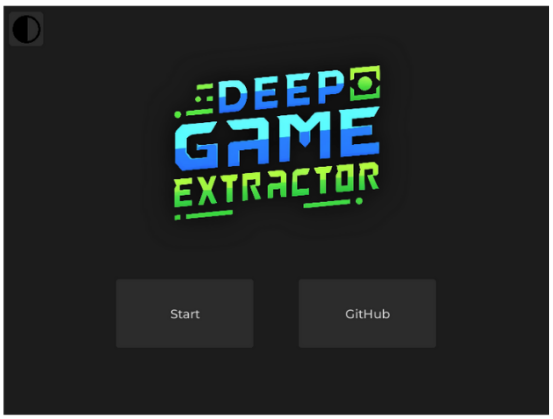
5.3.1. Class Diagram

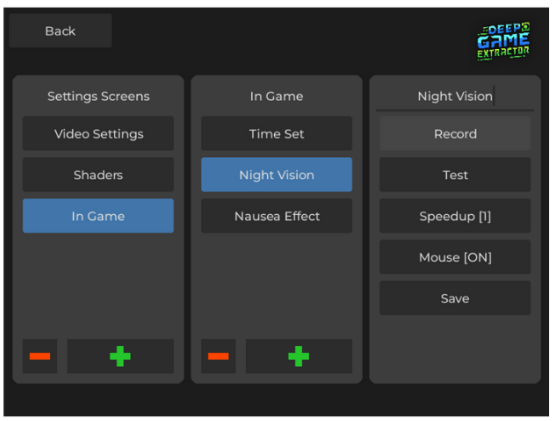


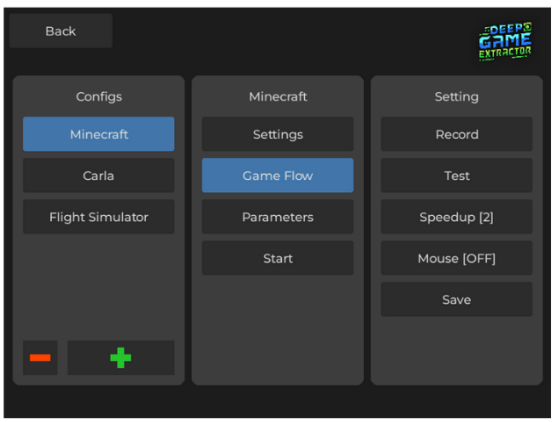
5.3.2. Module Diagram



5.4. User Interface Design







6. IMPLEMENTATION AND TESTING

6.1. Implementation

We aimed our application to be cross-platform, easy to install and easy to use. To achieve these goals, we used Python as it is very flexible, can be used in various operating systems, and offers many libraries that we need. We used PyGame library as GUI library for its easy implementation and flexibility. We used Pynput library to record and simulate mouse and keyboard. We used Pillow to capture screenshots.

Our project consists of 5 files.

6.1.1. main.py

This file contains the basic structure of a pygame implementation like initialization, game loop, drawing, and fps.

6.1.2. config.py

This file contains various configurations for the program. These configurations are not meant to be changed by user, but to make global changes easy. The configurations include

- split, terminate and cancel keys
- fonts
- screen window size
- current screen
- colors for the UI
- theme (light or dark)

· configuration for recordings

6.1.3. screen.py

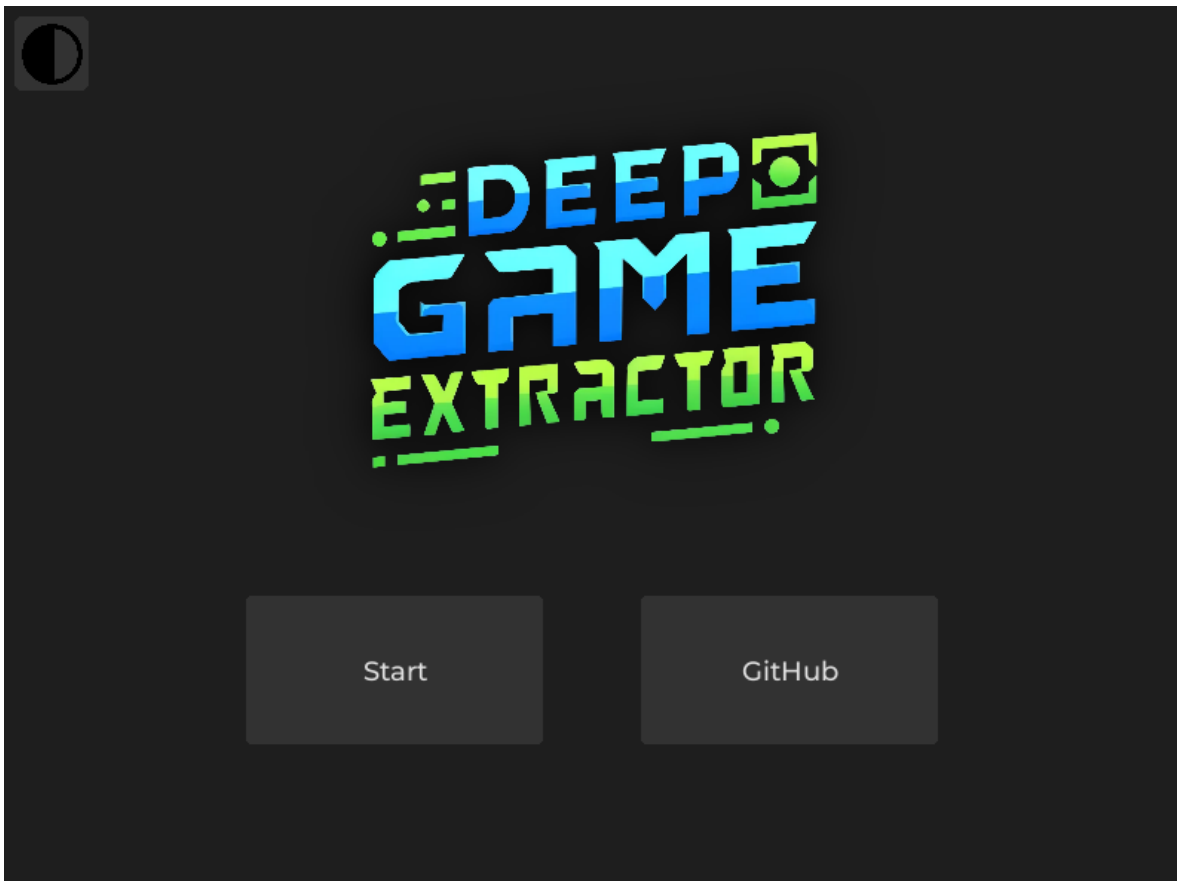
This file contains all the screens that our program contains. The screen classes are

· IntroScreen
· MainMenuScreen
· EditSettingsScreen
· ScreenshotParametersScreen

Instances of these classes are stored in the config.py file and rendered in main.py file.

These classes make use of buttons and recordings. The main logic of the program is implemented in these classes.

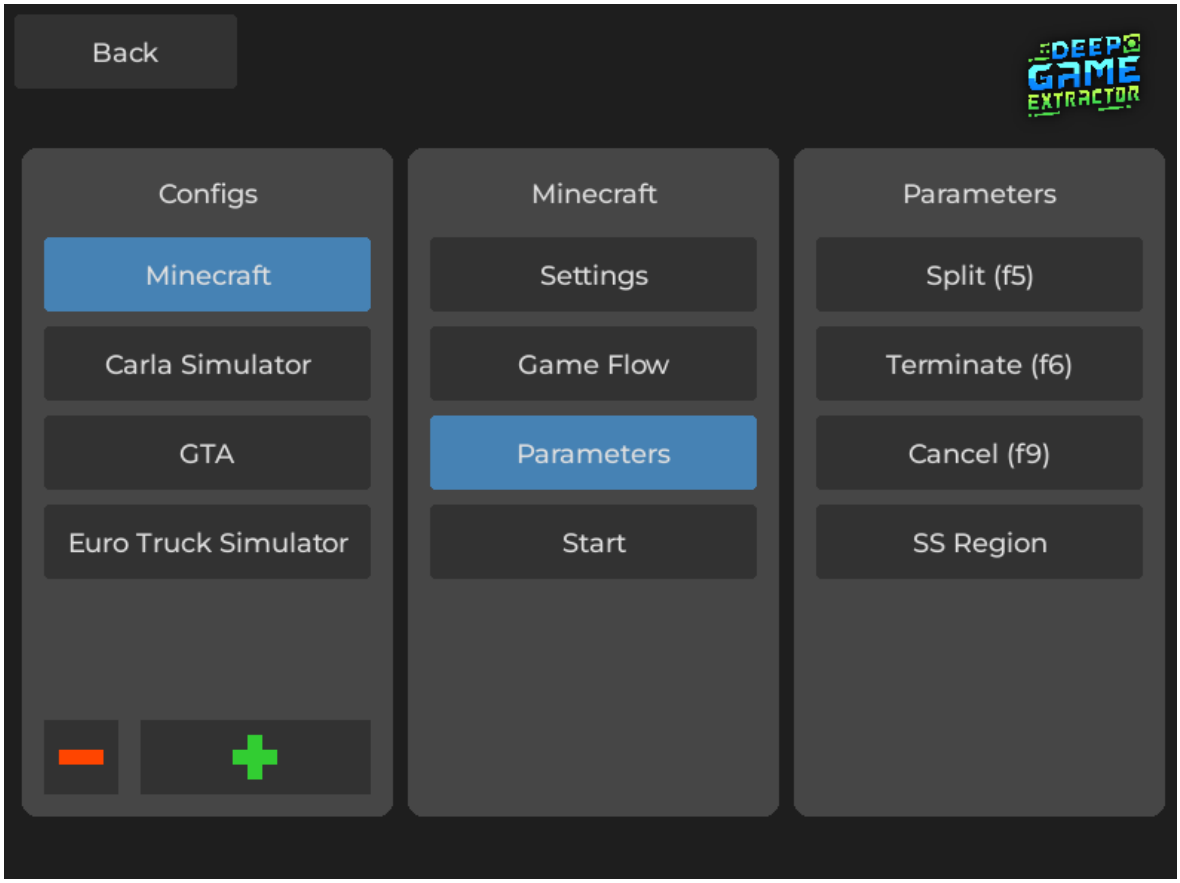
6.1.3.1. IntroScreen. This screen serves an introduction purpose. It contains the logo of the program, a button that opens the github page of the project, a button that changes the app theme, and a start button that leads to main menu.



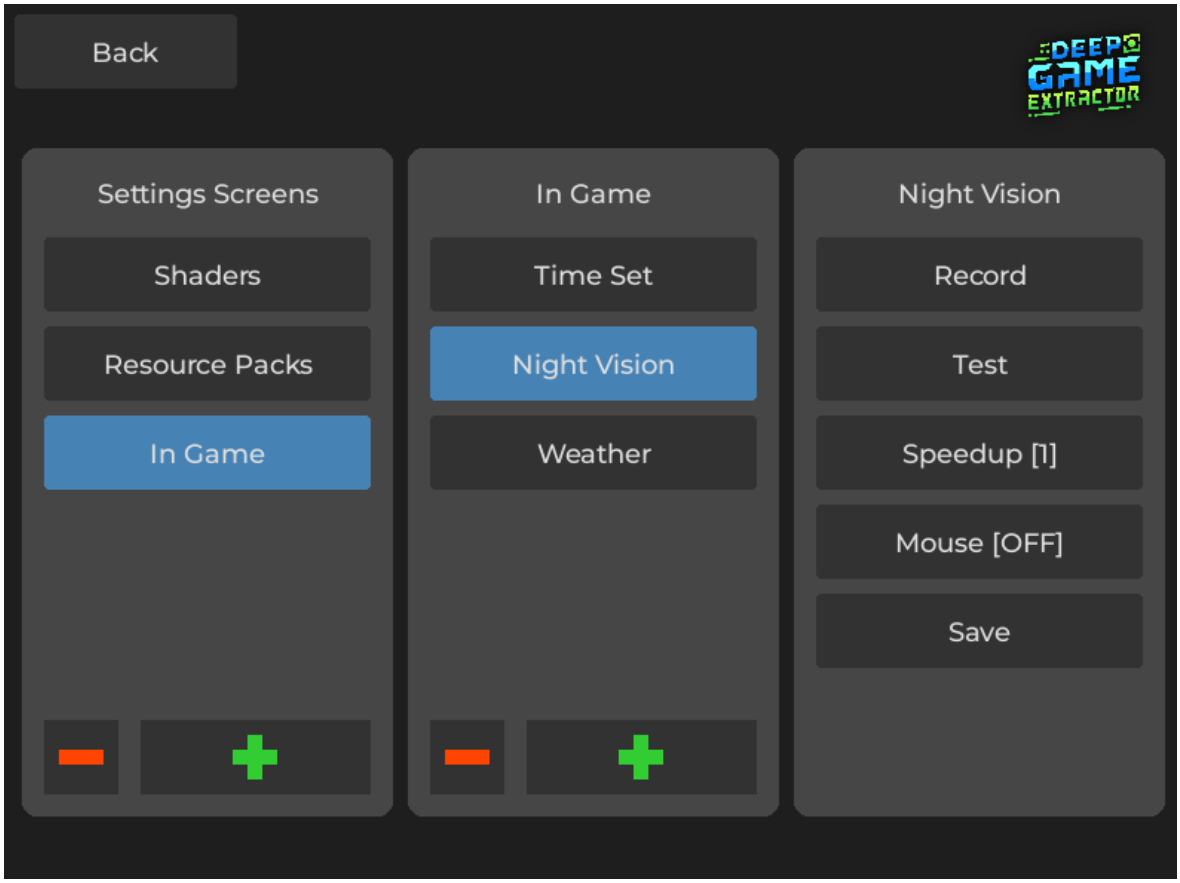
6.1.3.2. MainMenuScreen. This screen contains the top level of the configurations. In the leftmost button group, there is a list of games. When a game is selected, a middle button group appears. This middle button group contains four buttons:

- Settings: opens Edit Settings Screen
- Game Loop: opens a third button group to record and test game loop.
- Parameters: opens a third button group to set the following parameters for the game: split, terminate, cancel keys, and screenshot area.
- Start: starts the screenshot collection process.

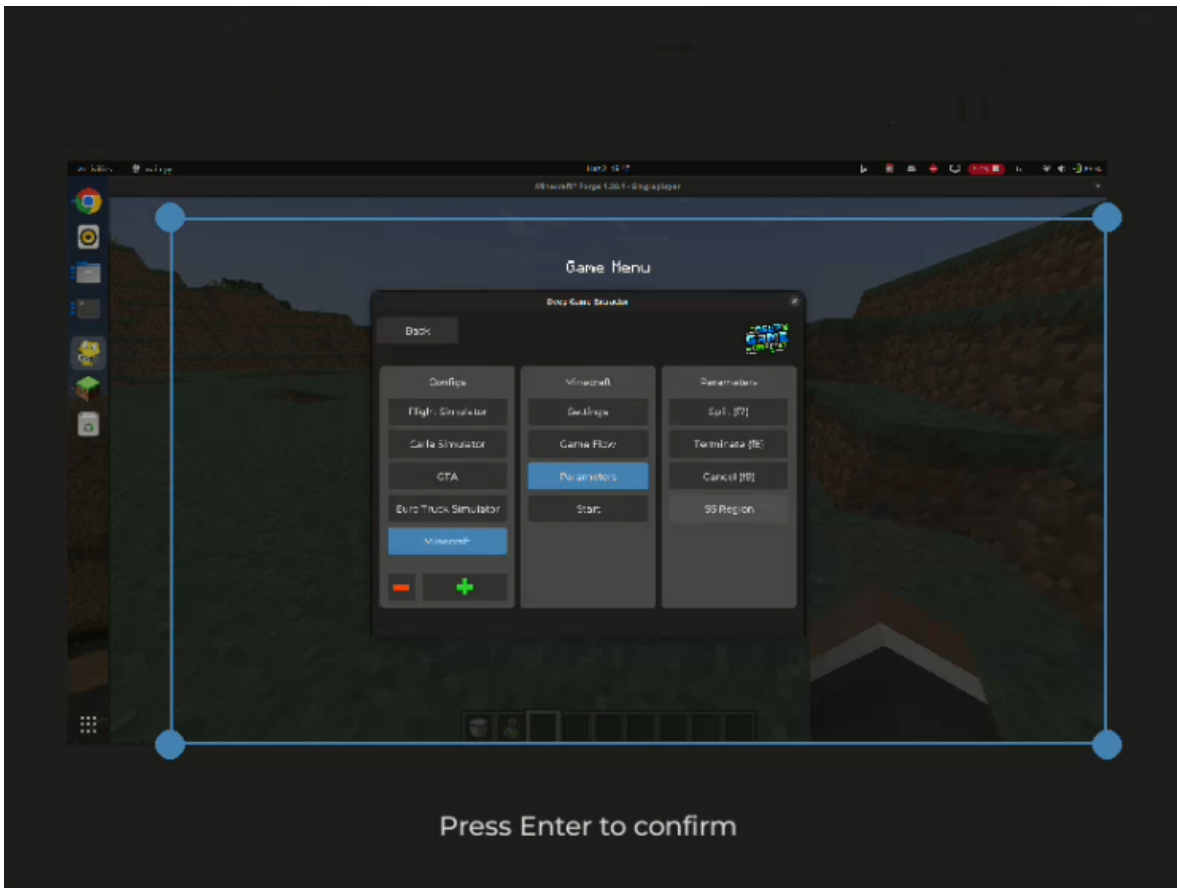
All the configured settings for the game are saved to a JSON file.



6.1.3.3. EditSettingsScreen. This screen contains the configurations for the in-game settings. In the left button group, there is a list of setting screens of the game. When a setting screen is selected, a middle button group with saved settings appears. When a specific setting is selected, a third button group with recording and testing options appears.



6.1.3.4. ScreenshotParametersScreen. This screen contains an area selection tool for screenshots. When a screenshot is captured, it is cropped according to this area to remove unwanted regions to appear in the outputs.



6.1.4. button.py

This file contains a few button classes that are used in screens. These classes make button creation easy and centered. There are a few different button implementations. These include

- Button: Simple button implementation
- IconButton: An extension of Button, contains icon instead of text.
- AddButton: An extension of Button, contains plus icon.
- DeleteButton: An extension of Button, contains minus icon.
- ButtonGroup: A group of buttons. New buttons can be added, deleted, and listed properly.
- EditButtonGroup: An extension of ButtonGroup, the name can be edited.

6.1.5. record.py

This file contains necessary for recording and simulating the mouse and keyboard events. The events are recorded to a string and stored as a string. This string is later can be simulated. The string contains time and cursor movement information. The time can be sped up when simulating, and the cursor movement information can be ignored. This file contains 3 main methods:

- record
- record_key
- simulate

There are also two other helper methods:

- merge_recordings: this method merges two recordings by adjusting the time information.
- split_recordings: this method splits a recording by a delimiter by adjusting the time information.

The source for the project can be found in the following repository: https://github.com/cahid game-extractor

6.2. Testing

Manual testing was conducted throughout the development phase. This involved executing the application, observing its behavior, and verifying that it met the requirements. After initial development, the application was shown to a select group of users to gather feedback.

6.3. Deployment

All the project files are stored in a github repository, along with the required libraries. To install our program, python should be installed on the system. Then the requirements can be installed using the following line:

pip install -r requirements.txt

And the program can be run using the following line:

python3 main.py

7. Results

DeepGameExtractor has achieved successful results, showing that it's versatile and effective in improving AI model training. Here's an in-depth overview of the project's achievements:

7.1. Platform Performance

DeepGameExtractor has been successfully performed on different games, including **Minecraft**, **CS2** and **Track Mania**. In each case, DeepGameExtractor effectively captured relevant game scenarios, providing rich datasets for AI training purposes.

7.2. Operating System Compatibility

DeepGameExtractor has shown decent compatibility and performance across multiple operating systems, functioning seamlessly on **Windows** and **Linux** platforms. This cross-platform capability ensures that it can be easily integrated into various development environments without compatibility issues.

7.3. Efficiency in Data Collection

One of the most significant impacts of DeepGameExtractor is its ability to drastically reduce the time and effort required for data aggregation. By automating the screenshot capture process and utilizing speed settings to optimize capture intervals, the tool has reduced data collection effort significantly.

7.4. Automation and Speed Settings

The automation features, coupled with adjustable speed settings, have proven to be critical in achieving efficient and effective data collection. These features allow users

to precisely control the frequency and conditions under which screenshots are captured, tailoring the process to the specific needs of the project and thereby enhancing the relevance and quality of the collected data.

These results underscore the practical utility and robust performance of the DeepGameExtractor.

8. Conclusion

This project, the DeepGameExtractor, aimed to address the crucial need for high-quality, varied datasets in artificial intelligence (AI) and machine learning (ML) training through the automated collection of game screenshots.

8.1. Achievements

The DeepGameExtractor has successfully met its primary objectives:

- **Automated Data Collection:** DeepGameExtractor automates the screenshot capture process, allowing for efficient and systematic collection of images.
- **Cross-Platform Compatibility:** Demonstrated seamless functionality across multiple operating systems, ensuring broad usability and integration into different development workflows.
- **User-Friendly Interface:** The graphical user interface (GUI) developed is intuitive and accessible, making it easy for users to configure and use the tool.
- **Enhanced Efficiency:** DeepGameExtractor has significantly reduced the energy required for dataset creation.

8.2. Impact and Future Directions

DeepGameExtractor promotes the development of more sophisticated AI applications by providing high-quality training data that is diverse and relevant. As AI continues to evolve, the demand for such specialized tools will increase, highlighting the significance of our project in the broader context of AI and ML research.

Looking forward, there are several areas where further development could enhance the functionality and utility of the DeepGameExtractor:

(i) **Augmentation of Collected Screenshots:** Developing features to augment the collected screenshots can significantly enhance the value of the data. Techniques such as scaling, cropping, rotating, and color modification can create more diverse datasets from a single source image, thereby improving the robustness of AI models. Implementing automated augmentation tools within the DeepGameExtractor would enable real-time generation of varied training data, tailored to specific AI training scenarios.

(ii) **Custom Labels for Data Annotation:** Introducing functionality for users to add custom labels to screenshots during collection can facilitate more precise data categorization and usage, reducing the time needed for manual data labeling post-collection.

(iii) **GUI Enhancements:** Implementing a few key GUI improvements such as confirmations for delete actions to prevent accidental data loss, and an indicator displaying the remaining time for data collection processes to enhance user experience and management of data collection sessions.

(iv) **Optimization of Settings for Duration:** Prioritizing the development of long-lasting settings that reduce the need for frequent adjustments can shorten the overall time required for data collection, making the DeepGameExtractor more efficient.

REFERENCES

1. Bullen, T., M. Katchabaw and N. Dyer-Witheford, "Automating Content Analysis of Video Games", *The University of Western Ontario*, 2023, https://journals.sfu.ca/loading/index.php/loading/article/download/5/2.

2. Collins, J., "Automating Rocket League Data Collection", *Python in Plain English*, 2023, https://python.plainenglish.io/automating-rocket-league-data-collection-7bb20c26ae18.