UNO Flip Remix - Development Plan

Team 24
Zain-Alabedeen Garada
Kevin Ishak
Mingyang Xu
Jianhao Wei
Zheng Bang Liang

November 26, 2024, 2024

Development Plan

Our team will hold weekly team meetings to track progress and align on project tasks. Each team member is expected to attend these meetings. Additionally, we will organize informal work sessions throughout the week to collaborate on specific tasks and document our progress. The link to our GitHub: simon-0215/UNO-Flip-3D

Team Meeting Plan

Scrum Meetings: Weekly Scrum meetings will be conducted with all team members attending. Each meeting will last between 15 and 30 minutes and will occur on Microsoft Teams. Members will discuss progress and any blockers encountered. At least one additional work session per week will be held where coding and design tasks will be conducted collaboratively.

Team Communication Plan

Primary communication will occur on Microsoft Teams, supplemented by cell phone coordination when necessary. The team will also utilize GitHub for version control and code reviews.

Team Member Roles

- **Kevin Ishak**: Full-Stack Developer, Will do research and implement the different types of cards that are possible to add in the game to make the game challenging whilst keeping the core essence of the card game intact.

- **Zain-Alabedeen Garada**: Developer/Project Manager, Will do research and implement the server-side development and real-time game synchronization.
- **Mingyang Xu**: Full Stack Developer, Oversees back-end implementation.
- **Jianhao Wei**: Full Stack Developer, Will do research and implement 3D visualizations and user interface of the game.
- **Zheng Bang Liang**: Full Stack Developer, Oversees front-end design and implement 3D visualizations.

Workflow Plan

The project will follow a Git-flow branching model to streamline development and ensure effective collaboration among team members. In this model, each new feature or bug fix will be developed in its own dedicated branch. This approach allows for focused work on individual components without disrupting the main codebase.

Once a feature is completed, team members will submit a pull request to merge their changes into the master branch. Each pull request will undergo a review process where other team members can provide feedback, suggest improvements, and identify potential issues before integration. This collaborative review process enhances code quality and promotes knowledge sharing within the team.

In addition to code reviews, we will implement unit testing to ensure the reliability and functionality of our code. Each module or feature will include a set of unit tests that validate its behavior and performance against specified requirements. These tests will be written alongside the code and run regularly to catch any regressions early in the development process. By emphasizing unit testing, we aim to maintain high standards of code quality and reduce the likelihood of bugs in the final product.

Proof of Concept (POC) Demonstration Plan

In this demo, we will present the client and server architecture of the multiplayer game we have developed, highlighting the seamless synchronization of game state and player interactions in real-time. The demonstration will emphasize key aspects such as core functionality, real-time synchronization, and interactive features.

1.We will begin by explaining the foundational architecture of the game, focusing on how the client and server communicate to ensure a consistent game state across all connected players. This includes showcasing the role of the server as the authoritative source of truth and the clients as interfaces for player actions.

2.The demo will illustrate the real-time exchange of data between the client and server. We will demonstrate how player actions, such as movement, object interactions, or other in-game events, are captured by the client and synchronized with the server. The server then processes these events and updates all connected clients to maintain a unified game experience.

Key features will include:

3.State Consistency: We'll demonstrate how the server handles multiple clients to maintain consistency in the game state, even under conditions like simultaneous actions or conflicting inputs.

4.Latency Handling: A brief insight into how our game minimizes latency effects, ensuring a smooth user experience.

5.Scalability: We'll explain how the architecture supports scalability, enabling multiple clients to join and participate in the game without compromising performance.

6.For the card display part, we will be demonstrating the card functionality from the viewpoint of each individual person. This include how cards are being withdrawn from the unknown card pool, how the cards are received, stored and played by the user, the algorithm that the program will use to determine the validity of each card play (by only allowing the cards with the same colour or larger number to be outputted), as well as the clearance of cards on the deck. This part of the game will later be modified into 3D and integrated into the multiplayer's scene.

To conclude, we will run a live demonstration of the game, showing two or more clients interacting in real-time. Attendees will see how actions performed by one player are instantly reflected in the game environment of other players, illustrating the robustness and reliability of our synchronization mechanism.

Technology Stack

1.Unity (C#): Ideal for 3D game development with robust tools for rendering, animations, and physics.
2.C#: If backend integrates tightly with Unity's networking libraries.

3.Version Control:Git and GitHub: For code collaboration and version tracking.

4.Netcode:creating networked games or applications in Unity by providing tools to handle synchronization, remote procedure calls (RPCs), and other networking needs.

5.TensorFlow: For implementing reinforcement learning algorithms for AI behavior.

6.Socket.IO: To handle real-time communication for multiplayer games.

7.GitHub Actions: For CI/CD integration to streamline testing and deployment.

Risks and Mitigation

AI Complexity: The complexity of AI learning may exceed expectations. To mitigate this, a simpler rule-based AI will be developed first.
Network Latency: Real-time synchronization may suffer from latency. Rollback networking techniques will be explored to manage this risk.

Coding Standards

- The code will follow JavaScript ES6 standards for front-end and back-end development.
- Pylint will be used to ensure Python code (for AI) adheres to PEP8.
- Testing will be implemented using Mocha (for JavaScript) and Pytest (for Python-based AI modules).

Project Scheduling

A Gantt chart has been created. Here is the link: Gantt Chart