Blackjack Assessment

Andrew Bennett - Year 12 SDD Assessment 1

Problem Definition

In this world, Blackjack, a game beloved by millions, has encountered a dark and unprosperous time, full of unoriginal and repetitive recreations of the game. The repetitive nature of these blackjack projects has diminished the world's engagement in the game as a whole. The Blackjack genre desperately requires a fresh perspective and iteration on the beautifully crafted masterpiece. Technology has allowed Blackjack and other entertainment products to take on a new form, to reinvent themselves and introduce more efficient methods to improve the games playability. In order to fix the lack of quality blackjack games, I have decided to take on the task of developing my own version of the classic casino game. Through my game, I plan to reinvent the aesthetics of the game, to create a unique experience, whilst still ensuring the rules of the game remain the same. In order to fully maximise player engagement, I have decided to make the game multiplayer. Despite players not actually interacting with each other, viewing other players in-game creates a more enjoyable experience for everyone.

Needs and Objectives

Overview

The Blackjack game must contain the expected features of the game, including betting, splitting and doubling. The game should be visually appealing, and have an interactive UI so the game is satisfying to play. The game should be rewarding for the player upon winning a round, to maximise engagement. Lastly, the game will include multiplayer functionality, as the game is significantly more enjoyable to play with others than just with bots.

Modularity

The application must have very modular code, to improve performance and convenience for modifying the game at a later stage, such as behaviour for all bots, and adding additional users to the game. Since the game will be multiplayer, the code must be modular so clients can easily interact with the server.

Lazy Loading

In order to reduce time taken to initialise the program, all images and other assets will be loaded as they are needed. This spreads out the processing required of the application, reducing the initial loading time and battery usage of the game. This method also reduces the RAM usage of the game, as not all sprites will have been loaded until some time has passed.

AI

The AI for the game must be proficient in the rules and strategies of the game. The dealer should follow a strict set of rules when playing, and the bot players should have variation in their playstyle depending on factors such as their balance and unique playstyle.

Multiplayer

Since the game is going to have multiplayer capabilities, many systems of the game must be specifically designed with this in mind, or modules will have to be changed in the future to accommodate the unique demands of a multiplayer game. The game will also contain a single player mode, so differentiating between the game modes is important and they should not interfere with each other.

Boundaries:

- The game must abide by the original rules of blackjack by default, and should be consistent throughout the game. The player may modify specific rules to their preference before the game however, but the game should still fundamentally behave as expected.
- The game will not include custom sprites.
- The game will not allow the ability to establish a dedicated host. The host must also be a player.
- The game will not include a betting system or fantastic aesthetics due to time constraints.
- The game will not have an account system or save game progress or upon leaving the session.
- The game will not contain special actions such as splitting hands.

Feasibility Report

Economic

Economic feasibility is deciding whether the money invested in the system is well used, and if the system will be a financial success. This includes costs such as development, maintenance, hosting, training of staff and infrastructure required (varies per the project). Since the blackjack game is being developed by me alone and does not cost anything, and I am not being paid for my work, economic feasibility does not apply. If the project were to be released to the public, it would be economically successful because any money I receive for payments of use or donations would cover the costs of development. The time used to develop the game is the only loss, as I could have made money by creating a different project.

Technical

The technical feasibility of a project is if the actual technology required to create and run the system exists and is available. The blackjack game I am creating does not require any special hardware or software for the project to be made or run. Any modern computer can be used to run the program, which is widely available. This means my game is technically feasible, as most people have access to a computer.

Operational

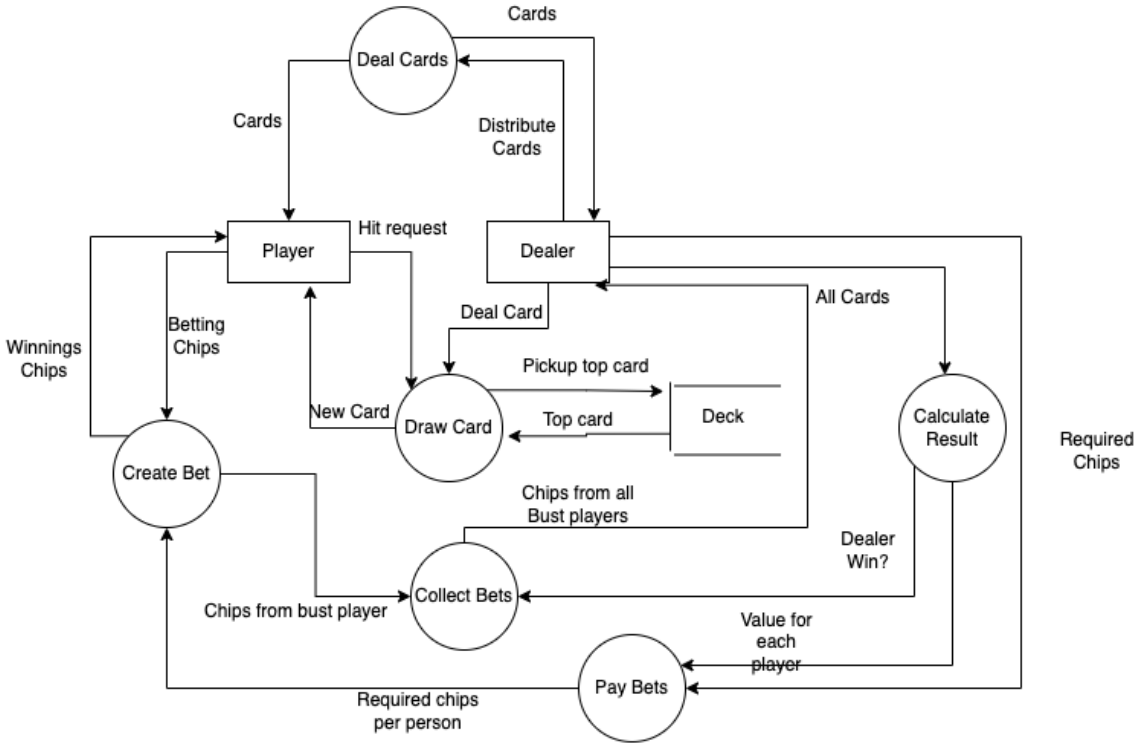
Operational feasibility is whether the system will work in practice and if the project actually solves the problem it was designed to solve, and whether users have the technical skills required to use the program. My blackjack game has some flaws which make it operationally viable, as it must be run directly from an IDE, the program has only been tested on a Linux OS, and the multiplayer functionality may not work on some types of network configurations. This makes my blackjack game not entirely operationally feasible, and severely limits the amount of people who can use my game. My game did satisfy my requirements however, which makes the project successful.

Scheduling

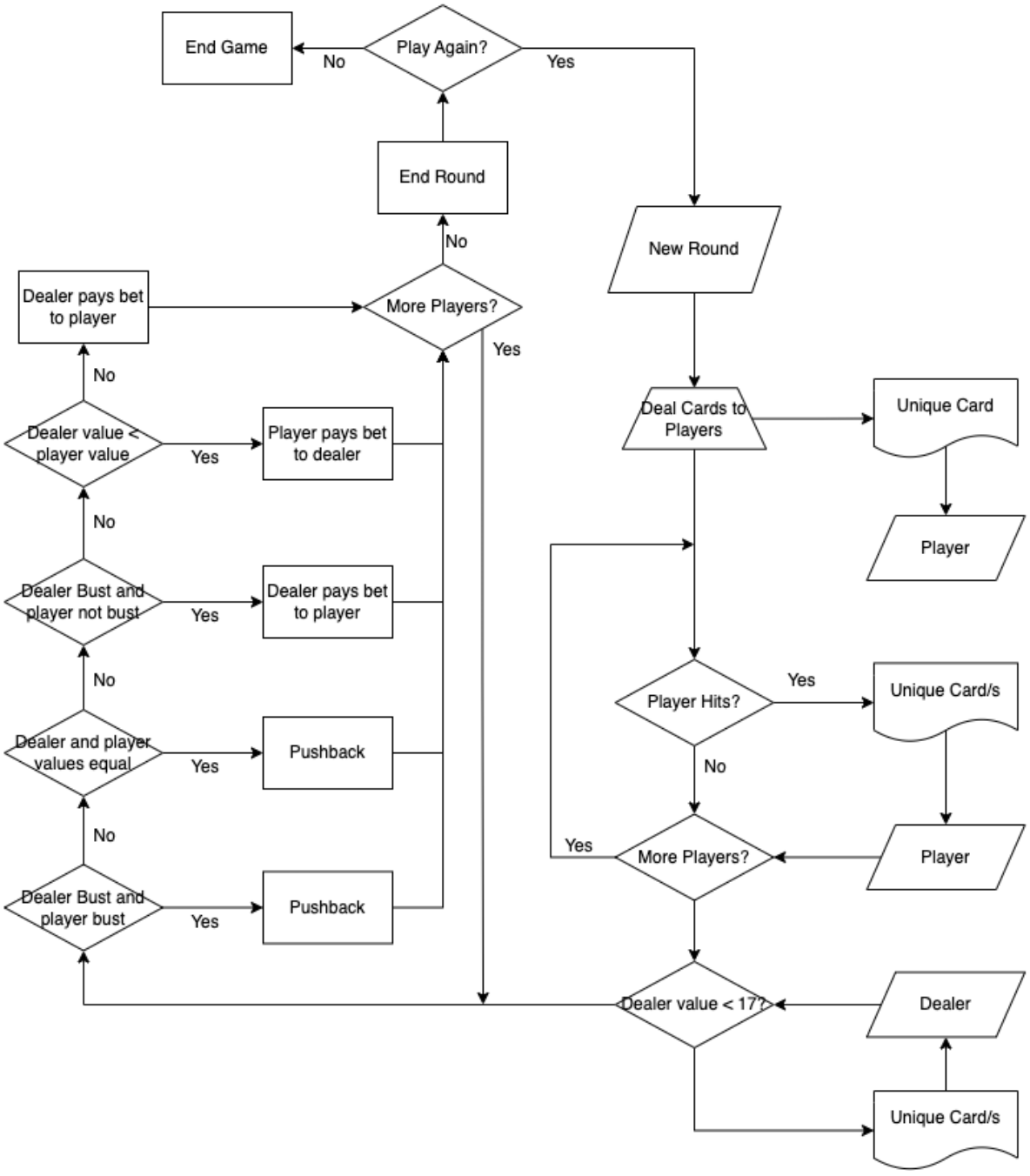
Scheduling feasibility is whether a project can be finished in a certain amount of time. This includes how long it takes to acquire technology required, how long programming takes, and bug fixes. Scheduling feasibility also determines consequences for unfinished solutions, and if an unfinished solution can be implemented if the deadline has been reached. For my blackjack project, it is mostly feasible in terms of scheduling. The restrictive deadline has enforced a strict project plan, which ensures critical features are created first, followed by other additions. This allows the solution to be finished whenever necessary, despite not all planned features being completed. For example, betting was left till last as it was not necessary for the games completion, multiplayer was considered a higher priority.

Diagrams

Data Flow Diagram



System Flowchart



Build Log

Build log present on the Forum:
https://scotsict.com/t/sdd-2022-blackjack-task-andrew-bennett/1093?u=andrew_bennett
Github shows push history and changes in the project:
https://github.com/andrewbennett1/Blackjack-Project

	W1	W2	W3	W4	W5	W6	W7
Theory							
Game logic and code structure							
User, Bot, Dealer Logic							
Graphics and visual elements							
Interactivity with visual elements							
Single Player fully working							
Multiplayer							
Polish							

Evaluation of Implementation

Structured

The structured approach to software development (also known as structured approach) is a formal set of stages for development which have to be followed in order. The structured approach is best suited for large projects which must be efficient and reliable. This method is often very expensive and time consuming however. The first stage is Defining and Understanding the Problem. This is when the programmers gather all the necessary information regarding the problem which must be solved. It is critical that the information be correct, as changing anything later in the process will become very difficult and expensive. The next stage is Planning and Designing. The actual solution is planned out, and diagrams and pseudocode are used to map out the plan. Implementing is when the code is actually created. All the modules combine to form the complete system. The system is then Tested and Evaluated, which checks that the system works as intended, and it actually solves the problem and is fit for purpose. Lastly, the application then goes under

maintenance, which ensures the system continues to satisfy the problem. Issues are fixed and minor features are introduced in this stage.
The structured approach is not suitable for my blackjack game, as a single person is developing the game with no money to spend. Efficiency and reliability of the system is not necessary, and time constraints prevent detailed planning.

Prototyping

The prototyping approach to software development focuses on the generation of prototype applications which gradually fulfil all the requirements over time. Prototyping allows the developers to present multiple iterations of a feature to the client so one can be chosen. Prototyping allows for cheap mockups to be made, saving money and time on something that may not have met the requirements of the client. When creating a prototype, the developers must define and understand the problem, where constant communication is established between the developers and client. The new prototype should then be planned and designed, where changes from the previous iteration are decided. The prototype may be scrapped at this stage if an alternative is preferred. The new prototype is then implemented using code. The UI gains functionality and the application works as expected. The prototype is then tested and evaluated. Informal testing occurs during and after each iteration of the prototype, while thorough testing occurs at the final prototype. This prototype will then become the basis for future prototypes. The problem must be redefined with the client, so future prototypes can be created. Once the final prototype has been developed and satisfies all the requirements, it is then deployed as the final solution.
This software development approach may be useful for my blackjack project, as many designs can be tested then evaluated before significant work is completed, reducing wasted time. Since the project is largely unstructured, this approach can be used to test new ideas throughout the development process.

End-user development

The end-user development approach is when the user of the system develops and maintains the system. This is quite beneficial for less sophisticated systems, as end-user development is cheap, fast and doesn't require detailed documentation. This approach can be disadvantageous however, as the user must have the relevant skills to develop the system. The quality of the output is also lower, and can introduce more bugs and inefficiencies.
This approach is very suitable for this blackjack project, as the developer is also using the product. No money is being spent on the project, and features are created depending on the priority determined by the user. An unstructured project also means work can begin much faster, allowing the project to meet the deadline.

Agile

The agile approach to software development is an ad hoc system of development, as no detailed plan is created, and the requirements of the project change over time. A small team of developers are generally used which work closely together and constantly communicate with each other and the client. This development approach has the advantages of being fairly cheap and has a quick development time, as well as being well suited to changing specifications. This approach is also disadvantageous, as a lack of proper structure can create inefficiencies and introduce bugs. Outsourcing aspects of the project can also be difficult, as changing requirements may result in expensive work being wasted.

The agile approach is viable for this project, as the development of the game is very unstructured and requirements may change often.

RAD

Rapid Application Development when a project is intended to be created and operational as fast as possible. This often involves utilising existing code modules and applications to reduce the work required of the creator. This significantly reduces the time taken to create the project, but is often more expensive than other development approaches. This approach can introduce vulnerabilities and bugs into the code which can not be fixed, and is only suited to small scale projects.

The RAD approach is not viable for this blackjack project, as no money is being spent and all of the code being used is developed by me. The reduced time needed to create this project could be beneficial however, but the project must not spend money or it will become economically infeasible.

Recommendation

The End-user approach should be used for the development of this blackjack project, as the small scale and free development of the application does not require much structure. The End-user approach means the developer is the primary user of the application, which is reasonable for an assessment task with very few people using the application. This approach is also generally faster than other methods due to the well defined problem which is to be solved and the lack of coordination needed due a single developer present.