**Esports Management Simulator Proposal**

Grant Miller

**Abstract**

Over the last 30 years, games that simulate traditional sports have become very popular, as users have wished to take control of a team and make changes that will affect the outcome of a simulated season. While many games simulate these sports by allowing the user to take control of individual players, other popular games allow users to manage the team behind the scenes, watching the results of their changes rather than being a direct part of the “gameday experience”. The most notable game of this genre is Out of the Park Baseball, which allows users to make extensive and precise changes to an accurate simulation of the American baseball system, from Major League Baseball to the lowest of the minor leagues. On the other side of the gaming world, esports have dramatically grown in popularity and viewership over the last 10-15 years, with new games providing avenues for competition every year. This project focuses on an intersection of those two rising industries, as a simulation game will be created based around esports. Over the course of the semester, the game will be created, first with basic features in order to create a solid foundation, then with more features to make a realistic and enjoyable simulation game. In order to create the game, a relational database will be created containing information and statistics about players, teams, and the overall league. This database will provide parameters for simulation code, which will run to “decide” individual matches in the esport simulation. Users will be able to make changes regarding their own team and players, and matches will be simulated over the course of a season. The project will pull on ideas introduced in other simulation games, such as the aforementioned Out of the Park Baseball, in order to create a playable and enjoyable game. In order to both make a realistic game and make reasonable commitments given a four month timeframe, the project will be created using an iterative approach. The initial version of the game will simulate two teams with default players against each other in a basic match. In particular, the esport genre simulated will be a MOBA (massively online battle arena), and there will be no “score” kept. Instead, certain events will trigger at various points in each match, and winning or losing an event will affect the probability of winning or losing future events for each team, until one team is able to win the “win the game” event. Other features will be added on top of the initial framework. In this way, the project will always have the ability to be scaled up or down, depending on progress made throughout the semester. Ultimately, the game will be a realistic and technologically effective simulation of esports for fans of esports or simulation games to enjoy.

**Background/Introduction**

Sports management simulation games, while not widely popular, have existed for decades, and their followers are extremely dedicated. For example, Out of the Park Developments, creators of the aforementioned OOTP, have 1100 members in their Discord, and the 2022 version of OOTP averages 1200 concurrent players. These games involve taking over an existing roster of players and making whatever changes are necessary to make a winning team. However, the user does not control their team’s actions on the field/playing arena, meaning their personnel choices are the only user-influenced factors in their matches. For users, this offers a unique and interesting opportunity to control teams in the same way a general manager might in real life. Additionally, because the matches in these games tend to be shorter than real matches, more games can be simulated, and users can simulate many seasons in a relatively short time span. These games tend to be less popular than other major releases because they require more thought and effort on the side of the user, but for those users who enjoy this style of game, it can be incredibly satisfying and enjoyable.

In a separate genre, esports have grown significantly in the last decade. Since the early days of Starcraft, a number of games have gained significant popularity in the esports scene. For example, North American League of Legends averages 125,000 viewers, and Valorant international events average 389,000 viewers (Esports Charts, 2021). Tried and true esports, such as Defense of the Ancients 2 and Counterstrike: Go, have continued to grow from a solid base, while newer games, such as SMITE and Valorant, have attempted to make themselves the newest popular esport. As games improve and younger people continue to play more video games, esports project to continue to grow and become more relevant in popular and sports culture. An esports management game adds to the genre by incorporating esports, a type of sport that has virtually never been represented in this genre before. Given that esports viewership has grown significantly in the last 10 years, a simulation game fits well into the genre as an option for fans of esports.

The game will simulate matches in the MOBA genre. Existing esports in this genre include League of Legends, Defense of the Ancients 2, and SMITE. In particular, the game will closely resemble the overall mechanics of League of Legends. In League of Legends, each team fields five players. Each team starts in their own base, which includes a Nexus guarded by two towers. By destroying all of the towers in one of the three lanes a player can travel down, as well as an inhibitor, players can attack the two towers, and then the Nexus. The ultimate win condition of the game is to destroy the enemy Nexus.

Because there is no “score” in these kinds of matches, simulation will be significantly different than in other sports. No matter the match state, a team is never “ahead” or “behind” by score. A team can be considered ahead based on their gold lead, control over neutral objectives, and map advantages (ex. A team might be considered “ahead” if they have destroyed more towers than their opponent, even if the gold count is even, because towers give players safety). Because of this, a match cannot be simulated as a collection of scores that lead to one team being ahead at the end of the game; instead, matches will be simulated as a collection of events, where each team has a chance to succeed or fail in the event based on player ratings and past events. As a team succeeds in more and more events, they will have a higher chance to keep succeeding in events, and will eventually have the opportunity to succeed in the “Destroy the Nexus” event, at which point they will win the match. However, teams will virtually never have a 100% chance of winning an event, accounting for the fact that, in this kind of match, a team always has a chance to come back, even from massive gold and neutral objective deficits.

Outside of individual matches, a user will have the opportunity to make changes to their roster, improve their players, and view results of other teams around the league. A database will supply users the ability to constantly evaluate and change their roster, and simulation code will supply accurate representations of individual matches. In short, similar to Out of the Park Baseball, the game will focus on the mechanical aspects of MOBA games, giving users an accurate representation of esports management.

**Project Description**

(Note: To avoid confusion, when discussing the project itself, I will use the term “game”, whereas when I reference an individual simulated “game” or “match”, I will use the term “match”. I will also discuss in-match entities as Non Playable Characters, or NPCs, while I will refer to the user as the user or player.)

To start, we will discuss how individual matches inside the game will run, then discuss how the user will interact with the game on a higher level.

Individual matches will be run using two key systems: a database using SQLite, and simulation code written in Python. The database contains a table for default players with predetermined attributes, along with a table with teams for each of the players. It will also contain empty tables for Player Statistics, Team Statistics, and League Statistics. The list of player attributes is: Team, Position, Age, Overall, Aggressiveness, Mechanical Skill, Shot-Calling, Laning, and Teamfighting (this list is subject to change as the project continues). Most of the player attributes will be between 50-99 as part of a rating system. The Overall rating will be a combination of all of the other ratings. Not all attributes will be ratings; for example, Age will function as expected.

The simulation code will pull specific attributes of players from the players database and use them as parameters in a larger simulation. A match will start with each team under “equal” circumstances, as neither team will have a gold lead or any control over neutral objectives. In any given match, a number of events will occur, including (but not limited to): Lane Fight, Attempted Tower Takedown, Jungle Fight, Dragon Fight, Baron Fight, and Base Fight. These events will occur based on game state (ex. If a team has won enough lane fights to reach their opponent’s base, the option for a Base Fight will open up, and a team’s combined aggressiveness will decide whether a Base Fight will occur), and a team’s chances to succeed in a given event will be based on player attributes as well as current game state. While a certain event may be possible, a team’s combined Aggressiveness will raise or lower the chance that the team decides to engage in that event, or wait for a different opportunity. The game’s state will change after any event, with each team’s gold count, neutral objective control, and tower count changing the chances of future events. Some events will not include every NPC on a team, and some events will not use all attributes from the NPCs involved (for example, the result of a Lane Fight does not rely on the NPC’s shotcalling, as it usually starts spontaneously, but it will rely on the NPC’s Mechanics to decide whether the NPC will win the fight). Certain events will only be available after previous events have occurred (ex. A team will not be able to initiate a Base Fight in the opposing team’s base unless they have succeeded in enough Attempted Tower Takedowns to reach the enemy team’s base). Eventually, the events will lead to a team being able to attempt the Destroy the Enemy Nexus event, and if that event succeeds, that team will win the match.

Because the game is a management game, once a match starts, the user will not have control over their NPCs. They will be able to see which events have occurred, as well as gold count, neutral objectives controlled by either side, and towers taken by either side. Esports do not contain substitutions other than in between matches, so the user will not be able to make NPC substitutions. During the match, information will be stored about the players and teams, and after a match is completed, that information will be sent to the database to be stored in the Player Statistics and Team Statistics. This information will include individual player contributions (kills, assists, deaths, etc.) for Player Statistics, and broader contributions (towers taken, neutral objectives taken, etc.) for Team Statistics. These statistics will also be updated after matches from other teams have been simulated.

When the game (not an individual match) starts, a user will choose the team they wish to control, and be shown the players on that team, along with their attributes. Then, they will see buttons for Play Match, Simulate Match, Roster Status and Changes, and Stats. Play Match will show the user the simulation of their team’s next match. This will be an in-depth look at the match, with the results of individual events being listed over time leading to an eventual winner being declared. This is the closest the user will get to watching a “real” match. Simulate Match, on the other hand, will simulate the game in a similar fashion, but will not show the user each individual event. Only the result and a handful of statistics will be given, allowing the user a quicker way to simulate through matches. After a match is “played” or “simulated”, the user will be given the option to Simulate to Next Match, which will simulate each other team’s matches, but will only show the user the results of those matches. Whenever a match is “played” or “simulated”, statistics regarding events and player performance will be stored in the Statistics tab, which will be discussed later. After Simulate to Next Match is pressed, the user’s team will move on to the next match on their predetermined schedule, and Play Match or Simulate Match will become available again for the user to press to continue their season.

The user can also click on Roster Status and Changes, which will take them to a screen with buttons for Roster, League Rosters, and Free Agents. The Roster button will show the team’s current roster of players, including the starting players and the substitution players, along with all of their attributes. Rosters will be made up of 10 players (five starters and five substitution players), and users will be required to have 10 players on their roster at all times. League Rosters will show the first team alphabetically in the league, and the user will use a dropdown menu to choose which team’s roster they want to see. Free Agents will show a list of free agent players and allow the user to add a free agent to their roster. If the user’s roster is full, they will have the option to cut a player, adding that player to the free agent pool, in order to add the free agent to the team and keep 10 players on their roster.

If the user is on the opening screen and selects Statistics, they will see buttons for Player, Team, League, and Legacy statistics. If they click on Player, the user will see statistics for every player in the league, sorted alphabetically. The user will be able to sort players by any of their statistics. If the user instead clicks on Team, they will see statistics for every team in the league, and will again be able to sort by statistics. If the user clicks on League, they will see a list of league-wide statistics, but will not be able to interact with any of these statistics. If the user clicks on Legacy and has completed a season, they will see a list of past years played as buttons. If they click on one of the years, the three Player, Team, and League buttons will be available, with the same statistics as above for that specific year. If the user has not completed a season, Legacy will not be available.

Seasons will consist of 10 matches. Once the user reaches the end of their season, they will see two buttons, one for Start From Scratch or Continue. If the user chooses Start From Scratch, they will be taken to the original start screen, where they will choose a team and start over their game (statistics from that season will still be stored in Legacy). If the user chooses Continue, they will be taken to the screen with Play Match, Simulate Match, Roster Status and Changes, and Statistics buttons. Their roster will be the same as it was at the end of the prior season, as will all of the rosters of the other teams in the league. Any attribute changes that occur yearly will have occurred, changing some player attributes. The user will be able to continue on as if they had restarted, only with a team that has completed a year of play.

**Foundations**

Simulations exist for a vast group of subjects, as being able to represent real events without needing to create those events is incredibly valuable. Some of the best practices of simulations are detailed in Modeling and Simulation Fundamentals, which serves as a more than adequate starting point for any project regarding simulation (Sokolowski, 2009). One key message is that a simulation is made using a model, which is a representation of a system, which is, in this case, esports. However, while the model attempts to represent the system, it cannot represent it perfectly, as there are too many variables to track each variable accurately and effectively. Because of this, a model must only include those variables that have a significant effect on the system. In the project, a major focus will be on what aspects of esports truly affect the outcomes of matches, as those aspects will need to be included in the simulation code and effectively represented for the user. While each simulation will involve some probability, and no match will be a guaranteed win, the user should feel that, if they improve their performance in key aspects, they will significantly improve their chances of winning, and less important aspects should not have as noticeable of an effect.

Another key text in understanding and creating the game is Modelling and Simulation for One Day Cricket, by Swartz, Gill, and Muthukumarana (Swartz, 2019). While their simulation of cricket is not meant as a game, they dive into the probabilities of outcomes based on events in a single cricket match. One directly related idea to the project is the idea of player aggressiveness, meaning how likely a player is to swing on any given pitch in a match. Player aggressiveness can be tracked on a game to game basis, meaning that individual moments in the game can influence the player’s actions later in the game, similar to what happens in a MOBA. While cricket is otherwise not very similar to esports, the paper supplies a strong baseline mathematical model for simulation of events based on past events, as the esports game will be doing. Their approach towards simulating cricket is similar in many ways to the approach proposed for the game, and while the MOBA simulator will not keep score, the mathematical ideas and event-driven progression displayed in the paper will be a useful reference point for the project.

In theory, a MOBA could be simulated in a similar way to more traditional sports, as the gold count and neutral objectives could be converted to a “score”, and whichever team has players with better ratings could be given the higher gold count and neutral objective control and eventually finish with a “higher score”. However, one key inspiration in the event-focused nature of the game comes from Andrew Rollings and Ernest Adams on game design, and specifically their chapter on sports games (Rollings and Adams, 2003). In the chapter, they reference how one type of match event can lead to a number of other match events. For example, in soccer, a corner kick can lead to continuing play, a free kick, a retaken corner kick, or other options depending on the outcome of the corner kick. In this way, sports can be considered as a collection of events that eventually lead to one team winning a match, instead of as an addition of scores together to determine a winner. The former idea is the one that our game will focus on, as the events in a match will lead to other events and, eventually, to one team being named a winner, rather than a score being kept to determine the winner.

Sports simulation games serve to represent individual sports and the potential outcomes of any given match, and management simulation games cover a wide variety of sports. Some notable examples are Out of the Park Baseball and Football Manager, both of which act as relatively accurate simulations of their respective sports. However, while those games are relatively popular, their documentation is relatively light. Instead, the project will use the work of Jeremy Scheff, also known as ZenGM. He is a programmer who has made a number of free, moderately popular sports simulation games. While Scheff is not an academic in the field, his work will serve as a foundation for the project, as it is open-source and well-documented. Because of this, his projects will be easier to follow and replicate than other sports management simulation games. He has even created a MOBA management game, and while it is older and not updated, it will provide a valuable reference point for the game. Scheff advocates for the three main aspects the project will be using as the three main aspects of any sports management simulation game: a database to contain player and team information, simulation code, and a GUI for visualization (Scheff, 2019). He also advocates for an iterative approach, beginning with a simple creation that does not present a realistic or enjoyable simulation, but that takes a set of parameters (NPCs and teams) and returns a result of a match. This iterative approach is the exact approach that will be taken in the project.

The simulation code will be mostly handmade, with inspiration taken from Scheff’s code, SimPy documentation (see Implementation Plan and Timeline), and other previous simulation projects. Paired with the simulation code will be a functioning database. Relational database management systems (RDBMSs) provide a number of advantages in storing data. These systems allow for faster and easier storage of data, as well as connections between different tables, rather than alternative storage systems without this functionality. In the project, relationships between Player and Team tables will be crucial, meaning the latter aspect of the database will be significant in the completion of the project. Database System Concepts, a book by Silberschatz, Korth, and Sudershan, will be used to make sure the database is being used correctly on a conceptual level, and the SQLite documentation will be used to make sure library-specific best practices are being followed (Silberschatz, 2019). Again, Scheff’s work will be a good reference point, as he has multiple examples of connecting databases with code to make similar games.

Another esports simulation game, called Esports Manager, does exist. In concept, it accomplishes many of the same goals of the project. The game focuses on simulating esports seasons using simulated teams, and provides avenues for player and team improvement. However, in practice, the game does not accomplish the same goals as the project aims to fulfill. The game’s underlying systems are relatively shallow, and there are few options for deep team customization. Additionally, the game has a heavy focus on visualizing the simulation, a focus that will be secondary in the project. By creating more in-depth simulation code, the project will create a more realistic and deep experience, rather than the shallow implementation of Esports Manager.

**Implementation Plan and Timeline**

Below is a preliminary biweekly timeline for the project.

Winter Break:

Learn more about past simulation code projects, especially Scheff’s work in his ZenGM projects. Find best practices about similar projects, and create a skeleton of the project (Github, starting files, starting database and tables, etc.)

Jan 11-Jan 25:

Work on simulation code, with the idea of being able to simulate a single match. Create two sample teams in database for testing purpose.

Jan 25-Feb 8:

Continue to work on simulation code. Either continue to try to simulate a single match, or add events to existing simulation code if that step is already completed. Work on connecting code with database

Feb 8-Feb 22:

Continue to work on simulation code, adding more events. Create new sample teams and continue testing for realism. Continue to work on connecting database with simulation code.

Feb 22-Mar 8:

Continue creating and testing simulation code. Begin working on seasons and legacy data.

Mar 8-Mar 22:

Continue working on seasons and simulation. Begin working on visuals.

Mar 22-Apr 5:

Continue working on simulation, seasons, or any other features if time allows. Continue working on visuals, working towards making a readable and useful end product for hte user.

Apr 5-Apr 19:

Finish work on visuals and simulation. Add any outstanding features, if time allows.

Apr 19-May 3:

Create poster and get ready to present.

The project will be, by design, iterative and flexible in its scope. At its core, it will be a game that simulates matches between two teams of simulated players. All other features, while necessary for realism and enjoyment, are secondary, in no small part because those features are less heavy on technical difficulty. Because the bulk of the technical work will be in creating accurate simulation code and a functioning database, the majority of my time will be spent working on those aspects, and all cuts of features will come in the order listed above.

In particular, the simulation code will likely be the most difficult and time-consuming aspect of the game to create. While packages exist for simulations and even sports simulations, none exist in a form that fit effectively with the idea of the game. A combination of standard simulation ideas will need to be combined with ideas of Monte Carlo simulation and other sports simulation concepts in order to make the game function effectively and somewhat realistically. The code will be written using Python, and the random package in Python’s standard library will likely be used extensively to add realistic random aspects of the game. More importantly, the SimPy package will be extremely useful, as it has a number of different aspects that help support simulation in Python. Specifically, SimPy gives a framework for event-based simulation, as different events and outcomes will change the state of the simulation and lead into different future events and outcomes. However, SimPy’s support for probability appears to be relatively shallow, so much of that work may need to be done by hand. The mathematical model shown in the Swartz cricket paper will provide a strong starting point for any probabilistic modelling, and more exploration will be done to find packages that can more adequately model these probabilistic events. One other package that will be used is Pandas, which provides dataframes to manage the tables of information that will be sent from the database before a match.

In terms of database management, the project will use SQLite as the database system of choice. Given that some database systems can be difficult to use, SQLite represents a simple and usable system that will integrate well with Python and allow a greater focus on the aforementioned simulation code. Having already worked with SQLite, the major focus of the work in the spring on SQLite will involve integrating it with Python and giving the simulation code the parameters it needs to run. The other significant obstacle will be collecting data during match simulation to send to the database for storage. Some solutions to this include a separate database for in-match statistics, individual Pandas dataframes that are created at the start of a match and hold data during matches, or other solutions yet to be found. Scheff’s solution to these problems in his games will again provide a useful reference point.

Finally, the visuals will be created using PySimpleGUI. Because the focus of the project is on the simulation and database work, the visuals will be relatively simple, and the work done on the visuals should be minimal. PySimpleGUI serves that purpose well, as it is relatively simple to use and will provide whatever visuals are needed for the user to maneuver the game and understand the results of matches.

**Conclusion**

In essence, the goal of the project is to create an accurate simulation of esports management in a usable/playable format for users. This will be done through the combination of a relational database and simulation code, which will interact to present results of simulated esports matches to the user. The bulk of the work will be done in creating a usable and effective database system, as well as creating code that will function both effectively and realistically given the data sent from the database. By following the above plan, the finished product should be a game that represents a realistic simulation of a MOBA esport from a manager’s perspective. The user should be able to simulate matches and seasons, and keep track of player and team statistics without struggling to understand the menus or mechanics of the game. The game should also perform well and integrate databases and simulation coding in an effective and understandable way. The code should be well-organized and neat, with consistent checkups to make sure bloat is avoided whenever possible.

While the above timeline gives a general idea of where the project should head, there is a large amount of flexibility in how complex the final project will be. This flexibility is provided through the iterative nature of the project, as baseline success will include only a usable match simulator using the database and simulation code, and new features will only be added after a certain baseline is met. In this way, new challenges and issues can be examined and solved with as much time as is needed, without needing to worry about future features. A successful project will be one that involves creating a functioning game, as well as learning how to code simulation and combine databases with Python coding. If those goals are accomplished, the project will have succeeded, regardless of added realism or visuals.

**Bibliography**

Esports Charts. (2021, June 4). *LCS Summer 2021 Viewership Stats*. Esports Charts. Retrieved September 15, 2021, from https://escharts.com/tournaments/lol/lcs-2021-summer.

Esports Charts. (2021, June 15). *RLCS Season X North American Championship Viewership Stats*. Esports Charts. Retrieved September 15, 2021, from https://escharts.com/tournaments/rl/rlcs-season-x-north-american-championship.

Esports Charts. (2021, September 10). *VCT 2021: Stage 3 Masters Berlin Viewership Stats*. Esports Charts. Retrieved September 15, 2021, from https://escharts.com/tournaments/valorant/vct-2021-stage-3-masters-berlin.

Rollings, A., & Adams, E. (2003). *Andrew Rollings and Ernest Adams on game design*. New Riders - an imprint of Pearson Education.

Scheff, J. (2019, July 18). *So you want to write a sports simulation game*. So you want to write a sports simulation game " Blog " ZenGM. Retrieved December 9, 2021, from https://zengm.com/blog/2019/07/so-you-want-to-write-a-sports-sim-game/.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database system concepts* (6th ed.). McGraw-Hill Education.

Sokolowski, J. A., & Banks, C. M. (2010). *Modeling and Simulation Fundamentals: Theoretical underpinnings and practical domains*. John Wiley.

Swartz, T. B., Gill, P. S., & Muthukumarana, S. (2009). Modelling and simulation for one-day cricket. *Canadian Journal of Statistics*, *37*(2), 143–160. https://doi.org/10.1002/cjs.10017

Takahashi, D. (2021, March 12). *Newzoo: Game livestreaming should grow 10% TO 728.8m viewers this year*. VentureBeat. Retrieved September 15, 2021, from https://venturebeat.com/2021/03/09/newzoo-game-livestreaming-will-grow-10-to-728-8m-viewers-this-year/.