# 15-112 Term Project: Volleyball Simulator

#### RICHARD AHN

#### Project Description:

In this project, I will be designing a volleyball game that runs smoothly and functions like other notorious sports games out there such as the FIFA, NBA 2k, and Madden series. The name will be called Spiking through Competition.

#### Competitive Analysis

Currently, due to the absence of a large league for national men’s volleyball, there aren’t many sports games notorious for volleyball except Spike Volleyball, which does a decent job of presenting the game. Even with Spike Volleyball, there has been numerous complaints about the bugs present within this that does not complement the true joys of playing volleyball and the enjoyment in watching men’s volleyball. Some bugs include the unnatural movement of characters or how they set up to the spiker. Others even involve how the ball moves off a block, and for that game specifically, the block would have the ball rebound and tumble down the net weirdly. Thus, I wanted to present the game that tries to achieve similar or yet better aspects in the gameplay for the player to have fun and develop a more profound appreciation for volleyball.

By the end of the term project, this game would be different from most volleyball games due to the fact that it is implementing a stronger 3-D oriented gameplay, and the moves and ball motion would be more diverse than what current games have to offer. This variety sets apart what makes real volleyball, especially competitive volleyball, more fun to play and watch than the same constant move sets that never change with the same outcomes from the moves.

#### Time Plan with Algorithmic Plan

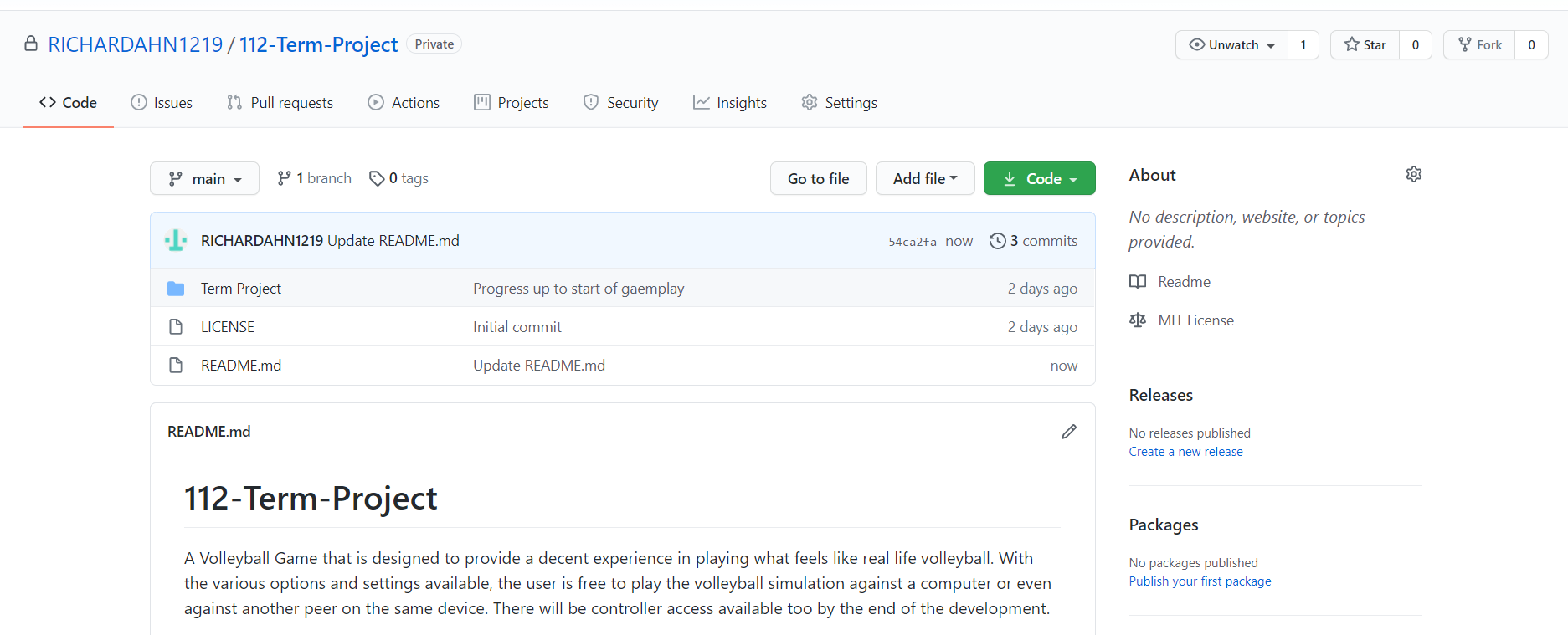
* Foundation
  + Different screens: Minimum will be completed by TP1, more will follow in TP2 and afterwards as it is not the biggest focus
  + Graphic features:
    - 3D gameplay: After MVP
    - Moving backgrounds: After MVP, will have set up before then
  + Movement:
    - Ball movement: Basics done by TP1, gravitational impacts will be implemented during TP2 and completed after MVP
    - Player movement: 2D by Tp1, 3d after TP2 with xbox controller option like jumping
* Calculations
  + Ball gravity (this will present a parabolic curve in the ball motion in 3d gameplay): After MVP
  + Ball curve (This will be implemented inside floaters, not good receives, and cross shots especially): TP2
  + Hitting calculation: basic hitting will be done by TP1 to work in 2d, will be improved upon and modified for different moves by TP2 and 3d purposes after MVP
  + Match Analyses: Generating graph and organizing the calculation to determine the different trends and patterns will be function by TP2 and completed after MVP
  + Gap calculation: Calculating the gap between each player on the opponent side to determine where to hit
* Player interaction
  + Setting: Will begin TP1, will finish moves in TP2 and then implement 3d movement after MVP
  + Spiking: Will get done TP2, will start TP1 as merely increasing ball speed
  + Dig: Done by TP1 as a collision, will be changed after MVP with 3D graphics
  + Player speed: Will start and finish TP2
  + Player jump: After MVP, this requires the player to be seen in 3d format
  + Serving: Will have collisions until MVP serve as serves
* Game AI’s:
  + Individual AI’s (setter, libero) will be given roles and different methods for certain purposes starting TP2
  + General player roles: Minimum to play game in 2d will be implemented starting TP2, format is already organized and ready to be written after TP1
  + Formations will come starting in TP2 and completed after MVP

#### Modules:

* PIL
* Pygames
* panda3d (some 3d module)
* Tkinter

#### Version Control Plan

For version control, I will be storing my term project in a GitHub repository and plan to update it frequently, allowing myself to see what changes have been made every time I push the code to the branch.



#### Timesheet

**1st session**

11:00 am - 1:00 pm 11/20

**2nd session**

5- 7 pm 11/20

**3rd session**

11/20, 2-4 AM

**4th session**

11/22, 12-5 pm

**5th session**

8-10 pm, 11/22

**6th session**

11/23: 7 - 10 am, 5 - 8pm

**7th session**

11/25: 11- 5 pm

**8th session**

11/27: 2 - 6 pm

**9th session**

11/29: 9 am – 3:30 pm

#### Algorithmic Plan Continued

I will be using gap calculations and a hit gage to determine how the player will determine a certain set of moves and how they will be running. The most daunting task will be designing the AI around these parameters and how to write them specifically with the calculations working and not working at times. The AI will determine how many touches there have been so far to the ball, so we will keep track a touchCount to measure this, and for a certain touchCount up to 3, we will need to prepare moves that account for each different value of the touchCount (1 would have generally a form of overhand/underhand receiving, 2 would have a toss or dump or spike from the setter, and the 3rd would be ideally an outside hitting it). We also want the hitting gage to have different methods of calculations for the differing difficulty modes (easy has random selection of values from a certain range, medium closes the range to a smaller range, hard is a consistent 90s accuracy). The method of selecting where to hit will be the gaps—the gaps will be selected randomly for easy mode, a new list with some of the larger values for the medium mode, and only the largest gap for the hard mode. The moves the AI players make will also rely on their class type, for if they are a libero, the AI will try to have a configuration where the libero receives and passes it to the setter, who tosses it to the outside. Another would be measuring the accuracy of plays from each spiker, as this would benefit a more diverse gameplay, where the setter tosses both to the front and backrow. The complexity would increase with this variety of moves as the difficulty increases. The ball speed will differ for the different difficulties, and this setting will be hard set in the Match Class. the reaction time would be the time between the ball being a certain distance from the player (like 5 to 10) and the time it takes to receive. This would create different toss paces ultimately for the hard mode AI’s and so this would simulate real volleyball where the spiker may run before or after the toss-up. The reaction time will be faster or more diverse for different AI roles depending on their positions. Liberos would have faster reaction times, which is crucial for faster ball speeds, whereas the spiker would have a varied reaction time, and this would have a more complex player interaction which expands the number of options of move configurations available to the player and AI. The other complex AI aspect is the same-team AI’s. The same-team AI’s will need to move and perform certain functions, and the opponent AI needs to predict moves based off of what the same-team AI does. Thus, it is necessary for the design to be a mix of easy mode and medium mode for the same-team AI to be a supportive teammate that ultimately involves the player’s gameplay, such as actually moving towards the direction of the ball or opponent spiker.

class Player(App):

    def \_\_init\_\_(*self*, *playerNumber*, *country*, *position*, *number*, *courtPosition*):

        self.playerNumber = playerNumber

        self.country = country

        self.position = position

        self.number = number

        self.courtPosition = courtPosition

    def bump(*self*):

*#reduce ball velocity, hit it like to front or across court*

*#send ball to setter*

*pass*

    def toss(*self*):

*#this is default set*

*#set to person to the right or middle*

*pass*

class Setter(Player):

    def C\_Toss(*self*):

*if* self.courtPosition == 3:

*#Set the ball coordinates to the be within range of the coordinates of the player in position 1*

*pass*

*elif* self.courtPosition == 1:

*#Set the ball coordinates to the be within range of the coordinates of the player in position 1*

*pass*

    def Default(*self*):

*pass*

    def Dump(*self*):

*pass*

    def A\_Toss(*self*):

*pass*

    def diagonal(*self*):

*pass*

    def B\_Toss(*self*):

*pass*

    def backRow(*self*):

*pass*

class Spiker(Player):

    def lineShot(*self*):

*pass*

    def crossShot(*self*):

*pass*

class Libero(Player):

    def underhandReceive(*self*):

*pass*

    def overhandReceive(*self*):

*pass*

class MiddleBlocker(Blocker):

*#Main blockers*

    def block(*self*):

*#Jump up and try to stop block*

*#RESULTS*

*#Kill block*

*#Touch*

*#Miss*

*#Jumping will be estimated through button press*

*pass*

class MiddleBlocker(Blocker):

*#Main blockers*

    def readBlock(*self*):

*#follow one of the front row opponents*

*pass*

    def guessBlock(*self*):

*#guess a direction the ball will go and block in that direction*

*pass*

    def assistBlock(*self*):

*#stick next to another blocker*

*pass*

class Receiver(Libero):

*#Be able to receive, then spike for all number of players modes*

    def lineShot(*self*):

*pass*

    def crossShot(*self*:

*pass*)

#### Structural Plan

STRUCTURE

CLASS HOME

- Construct the properties of the homescreen

- Implement the buttons to press which are

- Play button

- Tutorial Button

- Controls Button

If the tutorial button is pressed, run the tutorials page

CLASS TUTORIAL:

- Include the roles of each player

- Include the basic rules the game follows

- Have a list of tactics and moves that are used, and button combinations for those

CLASS CONTROLS:

- Includes a list of keybinds and bidns for the controller,

which present what button or joystick performs what action

- In a table format when presented on the screen

If play is pressed, then run the match options page:

CLASS MATCHOPTIONS:

- The options will be available to select through buttons or with the joystick and button selection

- First option is number of players. This will change how the AI's move across the courts and perform moves

- Second option is difficulty. The difficulty will alter how the CPU plays the game on the opposing side

- Have this page have a button to continue to SIDES page

CLASS SIDES:

- Let the page have two randomly selected or purposely chosen countries from a dropdown list

- Let the player1, and if there is a player2, select what side they want to be on

- Continue to the MATCH page

CLASS MATCH

- Generate the court

- Generate the ball

- Generate the players with respect to what team and number of players there are

- Have there be an indicator for what player is being selected then

- Implement a hit gauge and a visual to indicate the timing to press/hold the button for the different moves

- Standard method of digging will be having player in path of ball and pressing a certian button to dig it

- Have there be different ways of swapping between players for different numbers of players on the court

- Let the code run on timerFired for the moving velocity of the ball,

and ensure the ball moves within the bounds of the screen

- Have a timer in 2D to estimate when the ball is hitting the ground

- Implement gravitational force when in 3D

- In keyPressed, map moving, swapping between characters, hitting, and the pause menu interaction keys

- Implement more for the controller afterwards

- Have a ballIntersects function to determine that the ball met with a certain player,

and depending on the player's role, to have a certain list of moves available to be performed

- Have the gap calculator between each player, and have all designated gap areas to have

their values input into a list that can constantly switch the values in the list

- Have the game AI select from this gap calculator and choose where to hit the ball in a certain area

or where to prevent some other players like where the blocker would be

- Each rally begins with the serve

- When serving, provide a little option on the side of the player

- Have a counter in the corner for 5 seconds until button pressed for serve

- Have a little indication (like a shooting bar in fifa or 2k) where it estimates the quality of the type of serve being served

- Types of serve to include in the little option side: Float, jump, topspin

- For different difficulties, have different ways of choosing a move. For easy, have random selection, but have hard AI's process

the gaps and character positions. This way the hard AI would choose the ideal spot with a faster ball speed

- Have game end when the points in the third set reach 25 or if there is a deuce at 24 - 24, until there is a 2-point difference

- Run the MATCHOVER page

CLASS MATCHOVER

- Print message showing which player won, including when the player quits in the middle of the game

- Have a background in here, as well as buttons that allow for a rematch (which reruns Match) Match Analysis, or Home

CLASS MATCHANALYSIS

- Provides an option on which team to analyze

- Have a dropdown list on what part of the game to look at such as serving count,

spiking accuracy, average ball speed on spikes

- Have different diagrams available to view this data on in a dropdown list,

and have calculations process in different classes for each individual graph (TBD)

CLASS PLAYER

- sets properties for each player in general such as position, country, number, difficulty

- Have the player inherit basic movesets like bumping, spiking, tossing. This makes it easier

for testing, debugging, and a more general character for the 1v1 mode

- Gap calculation and hit gauge: Random selection of a specific range from 1 to 100, 30 – 80, and 80- - 99 for accuracy in hitting

- Have a set motion to follow ball in some way for idle AI’s on the same team

CLASS SPIKER, MIDDLE BLOCKER, SETTER, LIBERO, etc.

- Have the individual roles for each of the different positions and restrictions

- Have motion be different for each of these different characters

- Have movement restrictions for each of these different roles, like the libero not moving forward