Simulation Design

In the design of our simulation, we carefully considered the various methods of simulation studied in class. For this particular project, we chose to use Agent-Based Modelling as our basic flow for the simulator. This was because using the ball as an agent would represent an actual football match closest. Throughout the project, we came up with 2 iterations of the simulator. The first version was revised after testing and experimentation.

SoccerSim version 1.0

With the initial iteration of the simulator, we wanted to go for a fluid 2-dimensional representation of the game. Designing around an agent-based simulation, we took the agent as the ball, and its states as **Team A** or **Team B**. Then, we added methods to each state such that teams would try to move the ball forward and score goals when the opportunity arose.

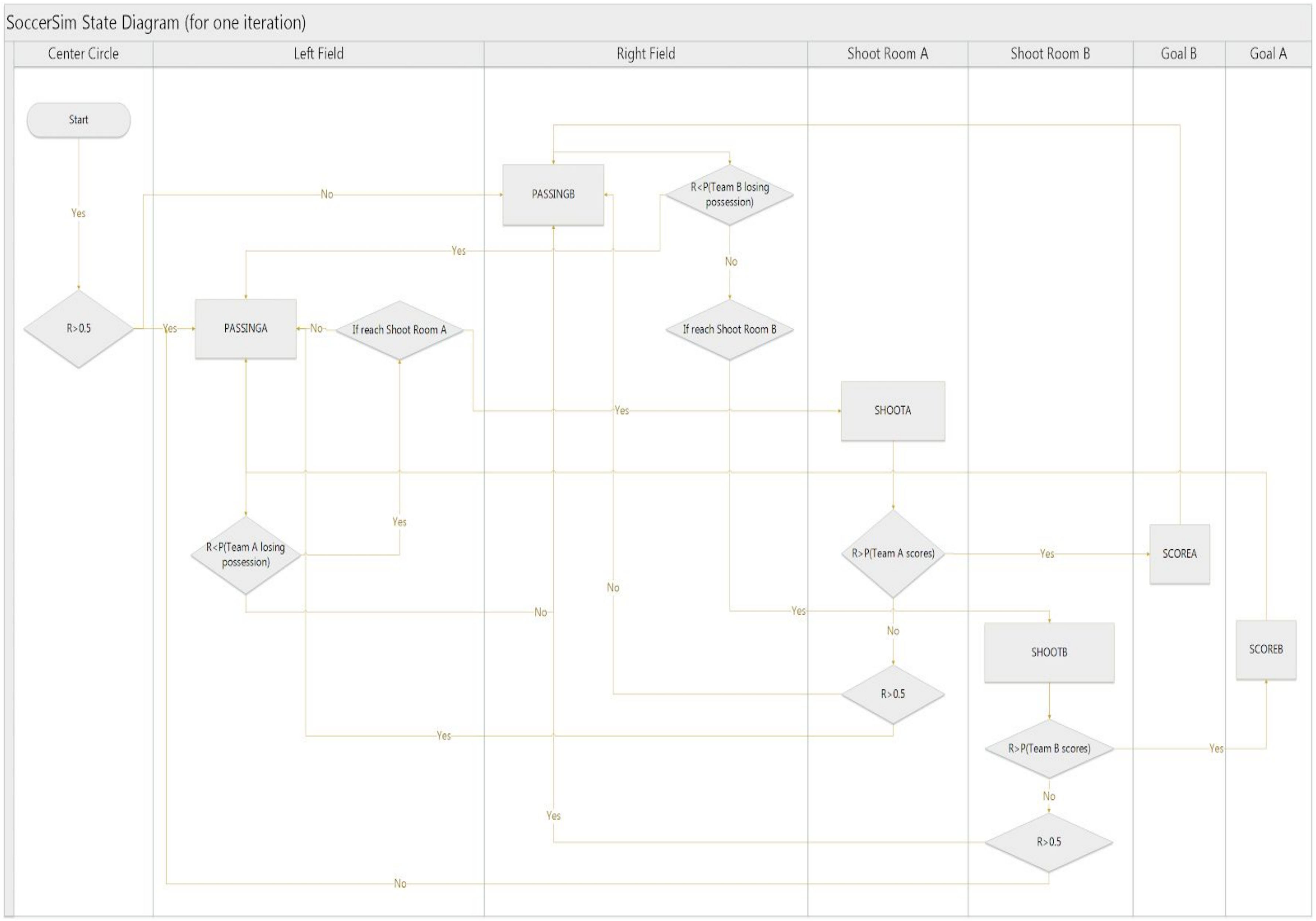
After coding out parts of the javascript and testing it, we came to realize that the initial design was inherently flawed. Not only did the ball seem to move unnaturally, there did not seem to be intent from either team to win the ball back or score. We realized that this stemmed from the treatment of the entire canvas as one area, and by our erroneous definition of states. Thus, we made the risky decision to redesign the simulation from scratch, taking into account interpretability and realism.

SoccerSim version 2.0

The new simulator splits the playing field into multiple smaller areas, each with their own coordinates. This allowed for a smoother, more natural looking passing phase. In addition to that, each iteration of the simulation automatically terminates after 90 in-game minutes, based on the animation speed chosen.

We decided to use javascript and D3 to generate the simulation. A key challenge we faced was deciding the position and movements of the ball. Initially, our design was a 2D canvas without predefined areas. In other words, each team always had a running probability of attempting to pass or attempting to shoot. However, we realized that this continuous simulation made ball movements look somewhat unnatural. This is because passes in real life move in a straight line before being received by the target player, whereas a continuous simulation made it possible for circular movements.

Thus, we decided to create grids in the canvas, which significantly improved the movements of the ball, as the equations simply had to pick a coordinate and the ball would travel there in a straight line. Based on these grids and the teams' tactics, they would switch states as shown by the state diagram below:


SoccerSim State Diagram (for one iteration)
Center Circle
Left Field
Right Field
Shoot Room A
Shoot Room B
Goal B
Goal A
Start
Yes
R>0.5
Yes
No
PASSINGA
No
If reach Shoot Room A
PASSINGB
R<P(Team B losing possession)
Yes
No
If reach Shoot Room B
Yes
SHOOTA
R<P(Team A losing possession)
Yes
No
R>P(Team A scores)
Yes
SCOREA
No
Yes
No
SHOOTB
R>0.5
Yes
R>P(Team B scores)
Yes
SCOREB
No
Yes
R>0.5
No

Due to the nature of the soccer match, there would be plenty of interactions between players and the soccer ball itself. However, as using all 22 players inside the model would be too complex for our model, we choose to scale down the model as soccer ball itself being the sole agent inside the model. All the interactions and actions are based on the soccer ball. With the using of D3 and svg, we are able to define the court area into 60 rows and 105 columns. Hence, the ball would position itself into the target grid based on our mathematical model with the user input values.

As we could see from the state diagram above, we are dividing the soccer field into seven areas, center circle, Left Field, Right Field, Shoot Room A, Shoot Room B, Goal A and Goal B. Upon reaching these various areas, the state of the ball would change accordingly. For each decision making, we would use JavaScript to generate an uniformly distributed number, R, in between 0 and 1 to compare to the probability of each event which may happen. The formulas using for us to generate the probability would be found inside the Mathematical Modelling document. Hence, the major states would be explained below.

1. KICKOFF state: the ball will be at the kickoff state upon the start of the game and its position will be in the center circle. Each team would be having an equal chance of getting the ball in possession. This would lead the state change into either PASSINGA or PASSINGB. With the random number generated being larger than 0.5, the state of the ball would change to PASSINGA, otherwise will be in PASSINGB state.
2. PASSINGA state: this is the state that Team A possesses the ball. Upon entering the PASSINGA state, the ball would be positioning itself inside the Left Field as Team A is holding the ball. Before the ball trying to reach its next target position, it would under a probability of Team A losing possession of it. If the R generated is smaller than P (Team A losing possession), the state of the ball would change into PASSINGB, which means Team B gets the ball from Team B. Otherwise, we could proceed with the position checking for the ball. If the ball reaches Shoot Room A, the state of the ball would change to SHOOTA state as it's in a nearer position to the Goal B. Otherwise, the ball would remain in the PASSINGA state for its next positioning.
3. SHOOTA state: this state would be triggered when the ball is in Shoot Room A. For the ball in its SHOOTA state if the R generated is larger than P(Team A scores), the ball will enter the SCOREA state. Otherwise, there would be another R generated for the model to determine either the ball is entering the PASSINGA or PASSINGB state.
4. SCOREA state: as the ball in the state of SCOREA, Team A will get one point, and it would be reflected on the page. After reaching the goal, the next position of the ball would be reset at the center point of the court, and it will change to the state of PASSINGB as according to the soccer rule that Team B will have the ball after Team A scores.
5. The same simulation process will be applied to Team B under the state of PASSINGB, SHOOTB as well as SCOREB

This state diagram only explains how one match is simulated under our simulator. SoccerSim follows the actual soccer match timing, which is 90 minutes for one game. After simulation time reaches 90 minutes, the ball will be reset to the center circle and another match will be simulated. The number of matches the SoccerSim simulates is based on the user input for Simulation Runs.