System Design and Architecture

Let us look at the various components that have been used here:

I. **Modules**:

- ***Pygame***: Using this library we can easily develop graphical user interface for user in a game environment.
- ***Random***: Using this I have generated the random coordinates of the players in game.
- ***Keyboard***: This module helps us to interact with the game created and here using space key the game is launched.
- ***Time***: This is used to generate time delays so that the game can work smoothly.
- ***Math***: This is used to do various mathematical operations like calculating square root, and finding $\tan^{-1}$.

II. **Functions**:

- ***Euclidean_Distance***: This calculates the distance between two coordinates which in terms is a positive value in all the cases.
- ***Is_collison***: Here the slope of each line is calculated. Using this we again calculate the $\tan^{-1}$ in order to calculate the angle from the positive x-axis. Then we subtract the smaller one from the greater one to get the relative angel between them. If the angle is more than 5 degree then there is very less chance of collision and the player is available to take the shot. It takes six variables, where 1^{st} two variables represents the x and y coordinate of the nearby red player, next two variables represents the x and y coordinate of blue player who is going to perform the kick, and last two variables represents the x and y coordinate of the position where the ball is being kicked. It returns true if the player is available and false if not available.

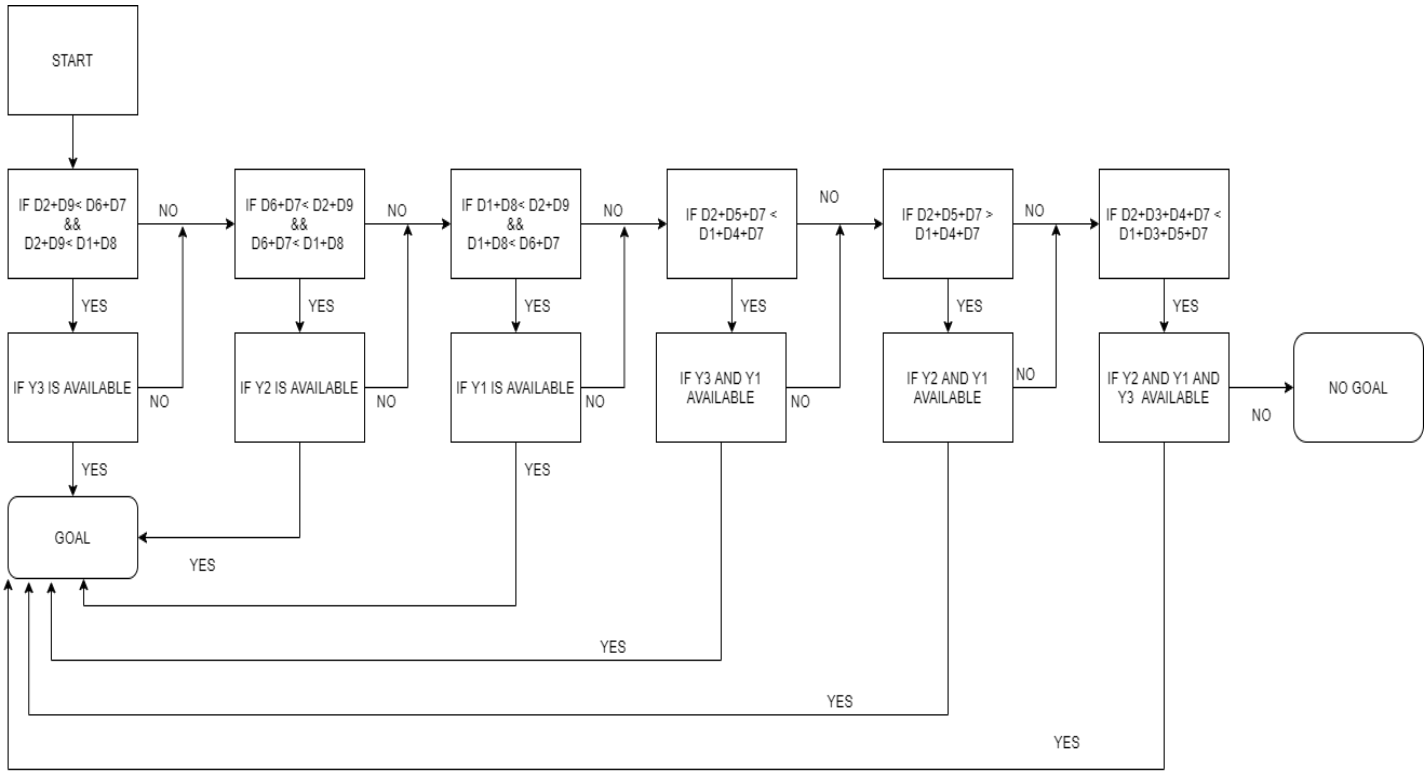
III. **Process Flow**: The following steps are:

a) ***Initialization***: First the pygame is used to initialize the display surface of height 570 pixels, width 400 pixels and color white using the pygame.display.set_mode(). Also the caption of the screen is set using pygame.display.set_caption(). After this all the images are loaded from the relative path i.e. from the photos folder. While loop is created for continuous looping which uses try and except to get rid of any exceptions. Now coordinates are generated for the loaded images randomly using randrange() function. Now all the loaded images are displayed as per the coordinates on the display surface using blit() function.

b) ***Euclidean distance calculation***: Here the distance between blue and blue, blue and red players are calculated.

c) ***Collision checking***: Using the calculated distance between blue and red players it is checked if there would be any collision between them or not.

d) ***Algorithm***: The steps are-



Here the algorithm first checks if there can be goal in 2 passes depending upon the availability of the players checked using is_collision. It takes care of all the 2 pass conditions. If after checking all the 2 pass conditions there is no way out it checks all the 3 pass conditions. If no 3 pass condition is available then finally it checks 4 pass conditions. If none of these is true then we cannot reach goal.

e) ***Quit***: When the close button is pressed the game quit.

Implementation of Environment and Agents

1. ***Environment*:** First I have loaded the images using relative path and then created object containing each individual image. By doing this we could use it multiple times as per our requirement. Using blit() function the images are being shown in the display surface. I have used update and flip functions the images have been rendered. Then using random function the position of each player is randomized according to the rules defined.

2. ***Agent*:** Here the agent takes decision based on our set conditions. It decides the shortest distance in each iteration of goal performed and passes the ball accordingly avoiding as many obstacles as possible.

From the video attached, it is evident that there are some outliers to this approach, as in some cases, the obstacles are not correctly detected. This is because the agent makes decisions based on our rules at any instance. As in every iteration, players' positions change randomly; it becomes quite challenging to get hold. This error can be corrected if the agent had learning capability, i.e. it would have learned from its past experiences and used that past data to predict the path accurately.

Q3 Making simple decisions

Cost of single ticket = 2000/-
Cost of combined ticket = 3000/-

Value of going to movie = 2000/-
Value of going to concert = 2000/-

Probability of finding time for movie (M) = 0.4
Probability of finding time for concert (C) = 0.4

Let us look at all the options available to us

Options	M, C (P = 0.16)	M, ¬C (P = 0.24)	¬M, C (P = 0.24)	¬M, ¬C (P = 0.36)
Combined	cost = 3000 Value = 4000 Total = 1000	cost = 3000 Value = 2000 Total = -1000	cost = 3000 Value = 2000 Total = -1000	cost = 3000 Value = 0 Total = -3000
Single	cost = 4000 value = 4000 Total = 0	cost = 2000 value = 2000 Total = 0	cost = 2000 Value = 2000 Total = 0	cost = 0 value = 0 Total = 0

Here total = Value - Cost

The expected value of buying a combined ticket is

$= (0.16 \times 1000) + (0.24 \times -1000) + (0.24 \times -1000) + (0.36 \times -3000)$

$= -1400$

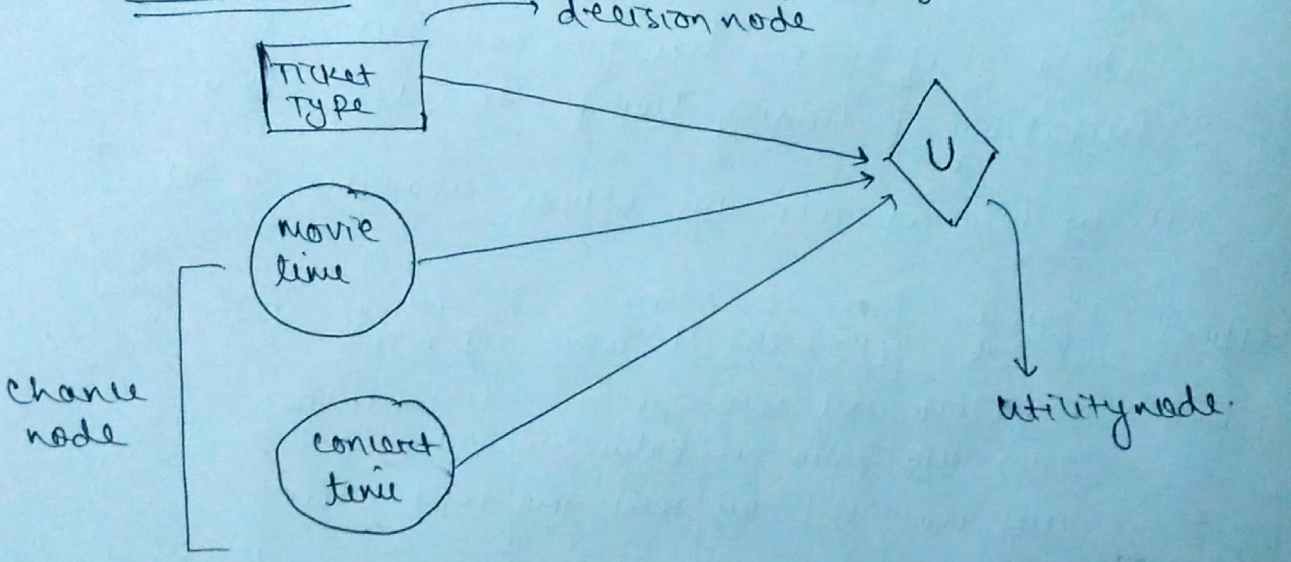
The expected value of ~~finding~~ buying single ticket

= 0

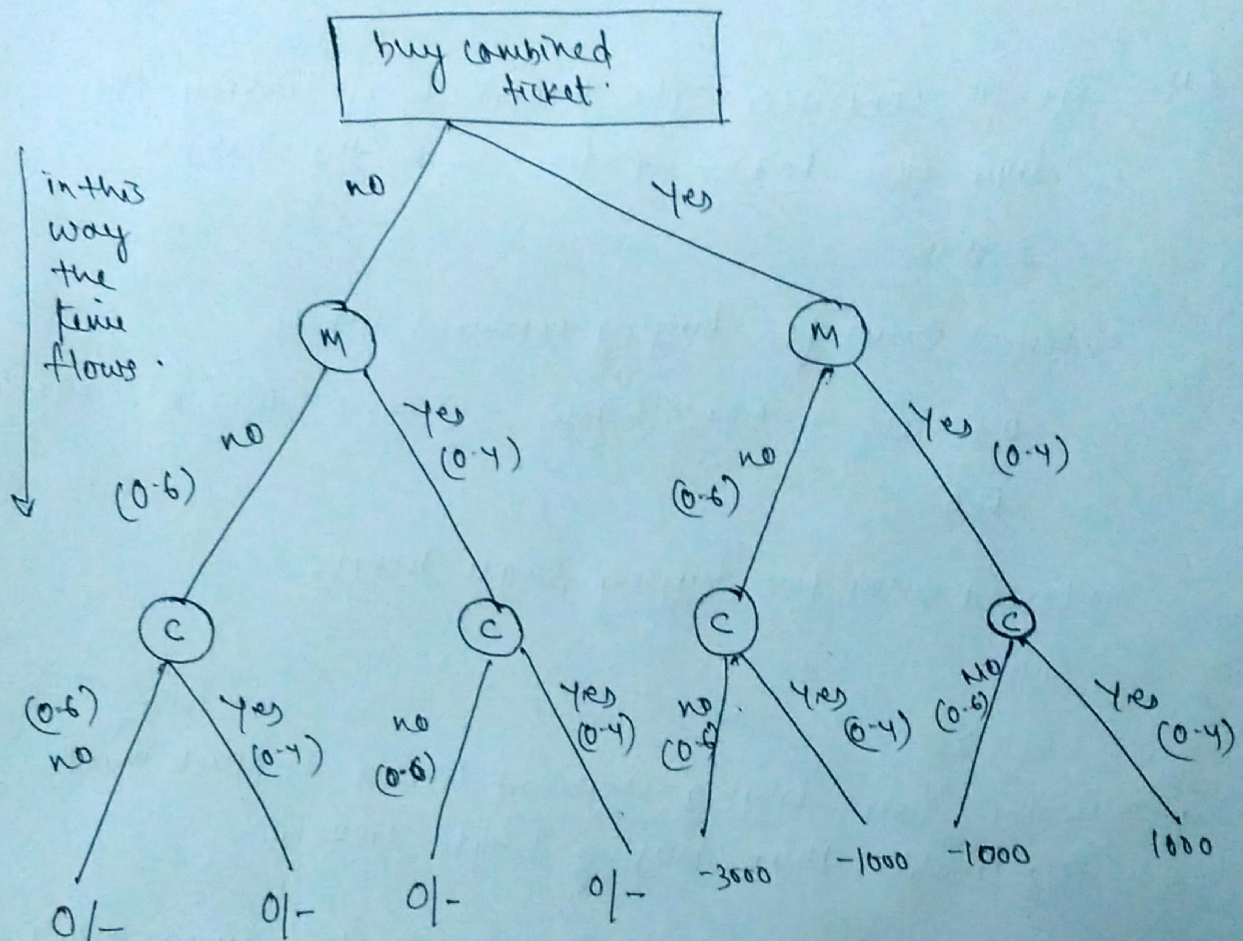
as 0 > -1400 so expected value of single ticket > combined ticket.

∴ The rational choice is to buy 'single tickets'.

The decision network for ~~finding~~ buying tickets.



Decision tree for buying tickets.



Calculate the expected value of buying tickets for 3 different probabilities.

(i) Let us consider the probability of finding the time for both the events

= 0.9

expected value for buying combined tickets

= 0.81 × 1000 + 0.09 × (-1000) + (0.09) × (-1000) + (0.01) × -3000

= 600

expected value for buying single tickets

= 0

as 600 > 0

∴ In this case buying combined tickets is much more Preferable than buying single tickets.

(ii) Let the probability of finding time for movies = 0.7
Let the probability of finding time for concert = 0.4

expected value for buying combined tickets

= 0.28 × 1000 + (0.42) × (-1000) + (0.12) × (-1000) + (0.18) × (-3000)

= 280 + (-420) + (-120) + (-540)

= -800

expected value for buying single tickets

= 0

as 0 > -800 ∴ In this case it is preferable to buy single tickets.

(iii) Let the probability of finding time for movies = 0.8
Let the probability of finding time for concert = 0.6

expected value of buying combined ticket

$= 0.48 \times 1000 + (0.32) \times (-1000) + (0.12) \times (-1000) + (0.08) \times (-3000)$

$= 480 + (-320) + (-120) + (-240)$

$= -200$

expected value of buying single ticket

$= 0$

as $0 > -200$.

∴ In this case it is preferable to buy single tickets.