e-Yantra Robotics Competition - 2020-21
Nirikshak Bot
Task 4C - Theme and Implementation Analysis

NB_887

Team Leader Name	Kallol Saha
College	Manipal Institute of Technology, Manipal
Team Leader Email	kallolsaha1752001@gmail.com
Date	14th January, 2021

Scope

Q1. State the scope of the theme assigned to you. (5)

< Teams should briefly explain in their own words the theme assigned. What in your opinion is the purpose of such an application? You may use figures/diagrams to support your answer (Make neat and labelled diagrams).
Answer format: Text - limit: 100 words. >

The Nirikshak Bot can be used in the production line of a company that manufactures products of different types in the same factory or production line. Normally, if many products need to be sorted into different boxes, this would manually be done by factory workers before the products are transported to the retailer. The Nirikshak Bot can do this automatically by using the ball balancing platforms to drop products into their designated boxes, right after they are manufactured, thus reducing manufacturing cost and time.

Testing your knowledge (Theme and Rulebook analysis)

Q2. Consider the following dictionary written in ball_details.json file:

```
{
   "red"      : ["T3_CB1"],
   "green"    : ["T2_CB2", "T1_CB1"],
   "blue"     : ["T1_CB3", "T3_CB3"]
}
```

Based on the dictionary given above, write the correct Collection Box for the following sequence of balls dispensed by BD: (5)

< This question is to check if you have understood how to interpret the ball_details.json file correctly. Hence fill in the answers carefully in the table below>

Sequence	Color	Collection Box Name
4th	Green	T1_CB1
5th	Blue	T3_CB3
2nd	Blue	T1_CB3
3rd	Red	T3_CB1
1st	Green	T2_CB2

Q3. Consider the JSON configuration given in Q2.

a) **What are the ENTRY and EXIT cell coordinates used by the *first green ball* for all the tables it is passing through? :**
T4: Entry coordinates= (0,5) Exit coordinates= (9,4)
T2: Entry coordinates= (0,4) Exit coordinates= (9,5) **(2)**

b) **What are the ENTRY and EXIT cell coordinates used by the *second blue ball* for all the tables it is passing through?:**
T4: Entry coordinates= (0,5) Exit coordinates= (4,0)
T3: Entry coordinates= (4,9) Exit coordinates= (0,4)
(2)

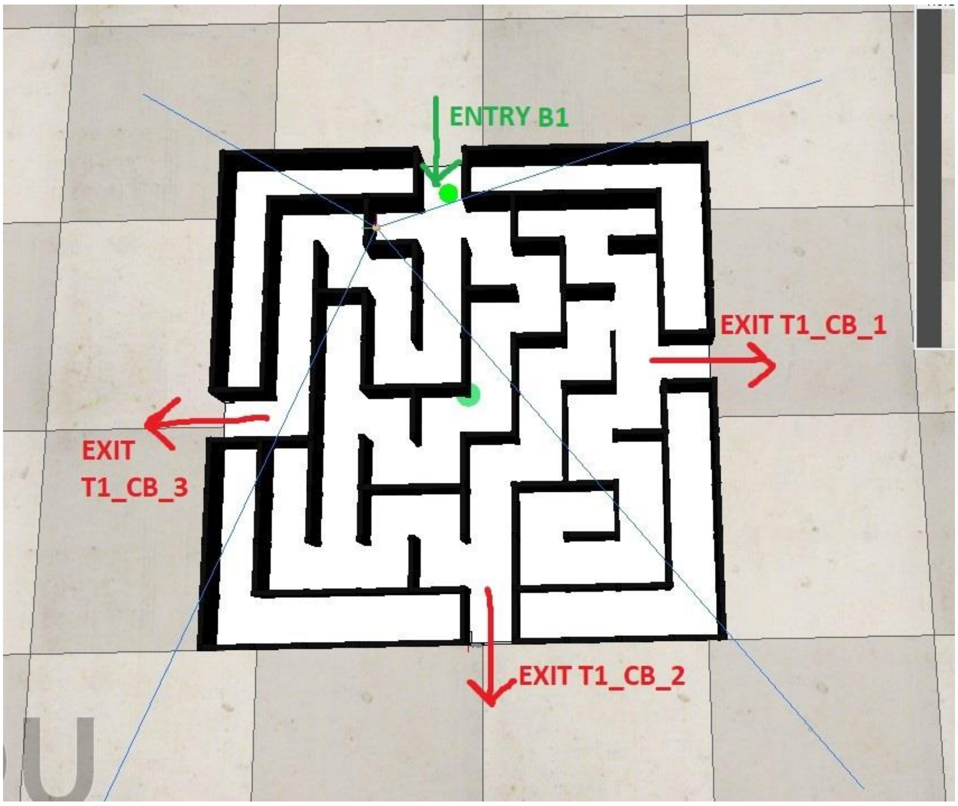
c) **What are the ENTRY and EXIT cell coordinates used by the *first red ball* for all the tables it is passing through?**
T4: Entry coordinates= (0,5) Exit coordinates= (4,0)
T3: Entry coordinates= (4,9) Exit coordinates= (9,5)
(2)

< This question is to check if you have understood Arena section of the Rulebook. Write your answers point wise for (a), (b) and (c)>

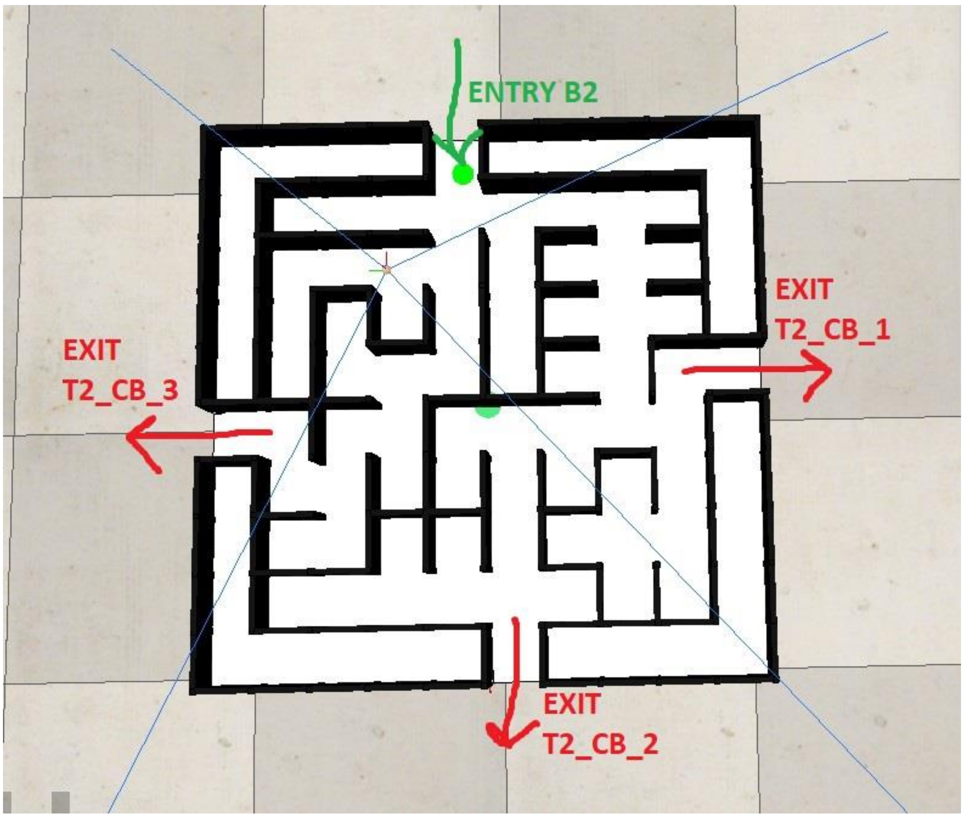
Q4. Download the *task_4c_maze_images.zip* file from this link (from Task 4C page). The images have been named maze_t1.jpg, maze_t2.jpg and so on (according to the Theme Run Requirements part under Theme Description section of the Rulebook). Generate these mazes on the single Platform Table one by one according to the resultant maze images shown in Figure 10 and 12 of Arena section in CoppeliaSim and capture a top-view screenshot for all of them. **(4)**

< Make sure to carve the respective EXIT points for all the mazes on Platform Table. Paste all the screenshots in this document. All the screenshot images should be properly labelled with ENTRY and EXIT clearly marked>

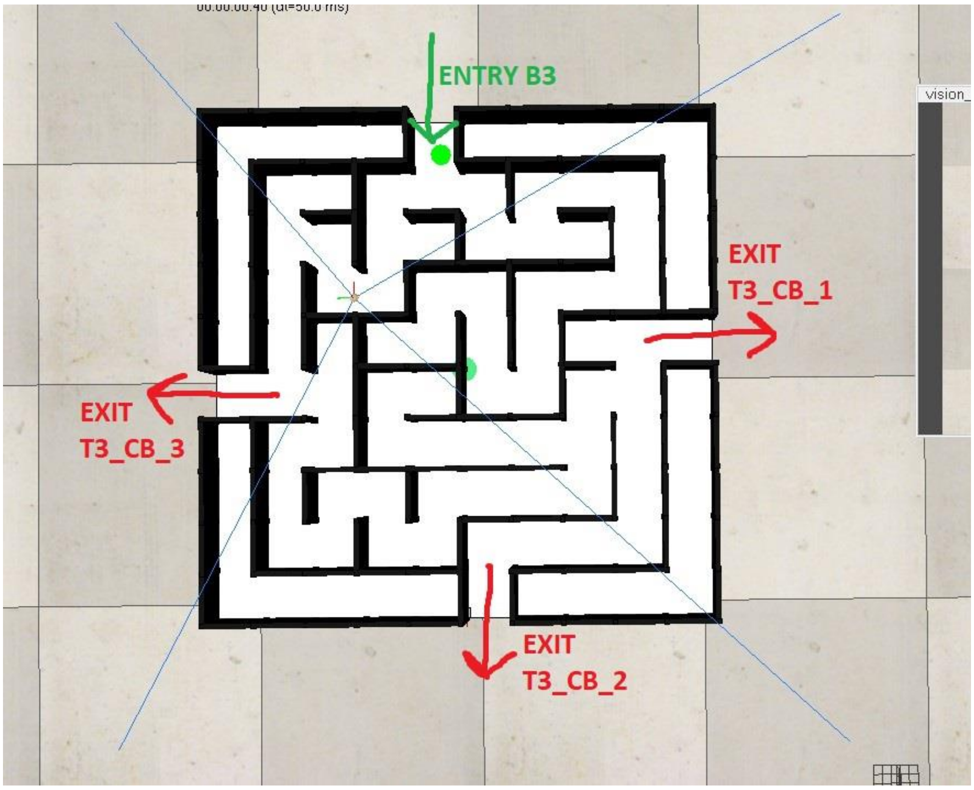
MAZE T1:



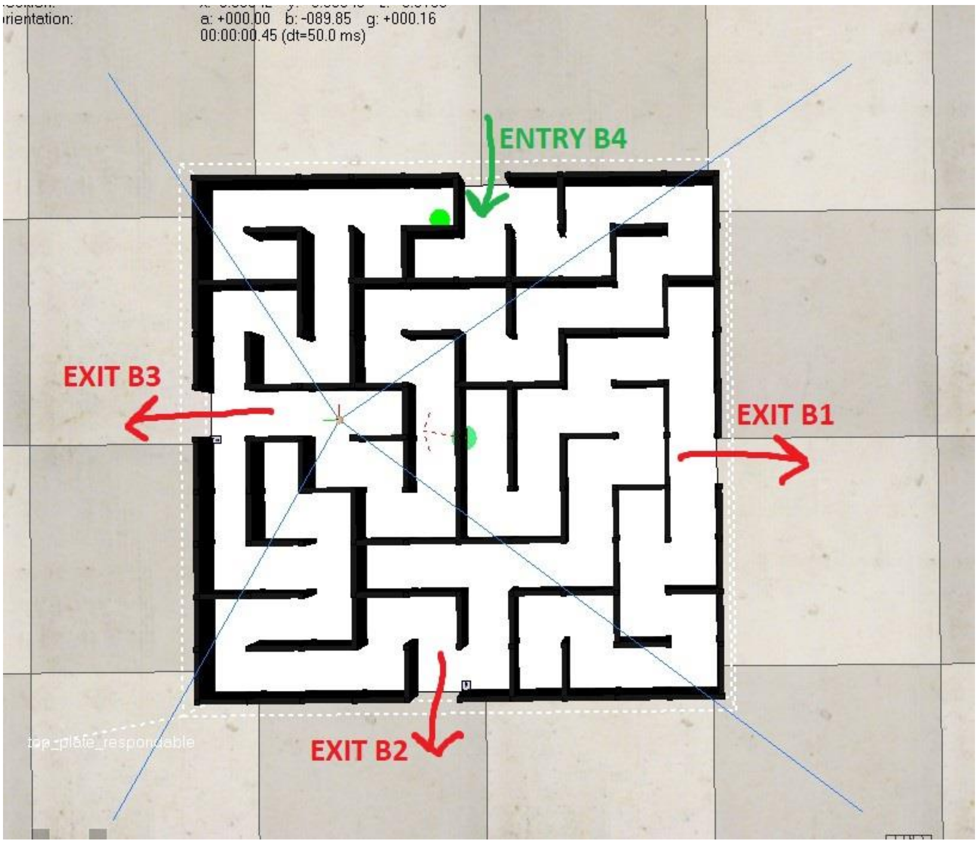
MAZE T2:



MAZE T3:



MAZE T4:



Q5. Consider the following table showing the scenario for each ball and calculate the final score:
Final Score = 2550
(5)

	CI	CP	CD	CT_4	CT_x	TB_4	TB_x	HP
1st	0	1	1	1	1	1	1	1
2nd	1	1	1	1	1	5	10	8
3rd	0	1	0	1	1	3	13	4
4th	1	0	0	1	1	10	20	15
5th	1	1	0	1	1	2	17	2

CM1	CM2	CM3	CM4
1	1	0	1

< Show your calculations in detail below>

$$Score = \sum_{1}^{n} [(CI*10)+(CP*100)+(CD*50)+(CM_4*(T_4*100)$$
$$(CP*CM_2*(T_2*100)+(CI*CP*CM_4*TB_4*10)$$
$$+(CI*CD*CM_4*TB_2*10)-(HP*10)]$$
$$+[sum(CM)*50]$$

$$\Rightarrow [\{0+100+50+100+100+0+0-10\}+\{10+100+50+500$$
$$+100+50+100-80\}+\{0+100+0+300+0+0+0-40\}+$$
$$\{10+0+0+1000+0+0+0-150\}+\{10+100+0+\cancel{1000}+$$
$$200+0+20+0-20\}]+[(1+1+0+1)*50)$$

$$=[340+830+360+860+310]+[3*50]$$

$$= 2700-150$$

$$=\boxed{2550}$$

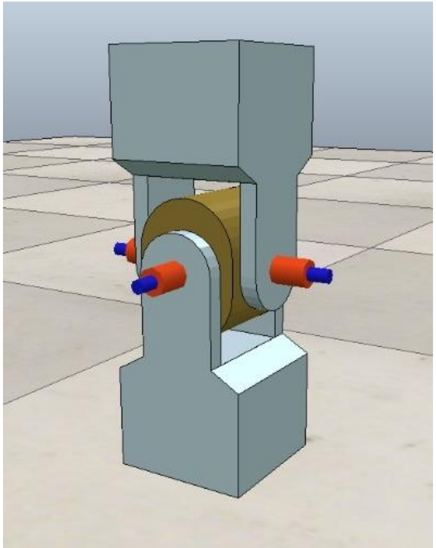
$\therefore$ The final score is 2550

Mechanism

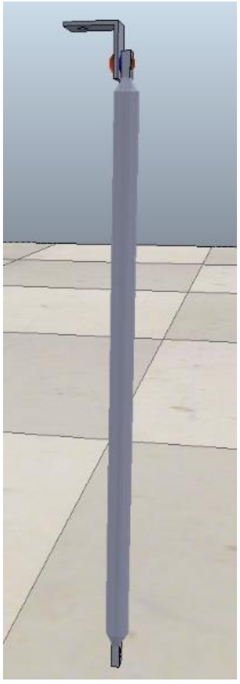
Q6. Explain the mechanism that you used for your ball balancing platform. (5)

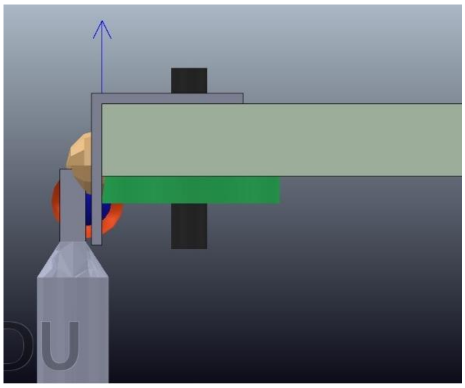
< You must explain the mechanical construction of your ball balancing platform and how have you connected all the different components provided to you. Make properly labelled diagrams to show the same. You may also use screenshots of the CoppeliaSim scene to demonstrate your mechanism.>

1. The mechanism used in our ball balancing platform consists of two manipulator assemblies which tilts the top plate in a given direction.
2. The top plate is connected to the base plate with a universal joint assembly that allows the top plate to tilt about both the x and y axes.
3. The universal joint assembly consists of two yokes and a universal joint pinion with each yoke having an independent revolute joint with the universal joint pinion. One of the yokes

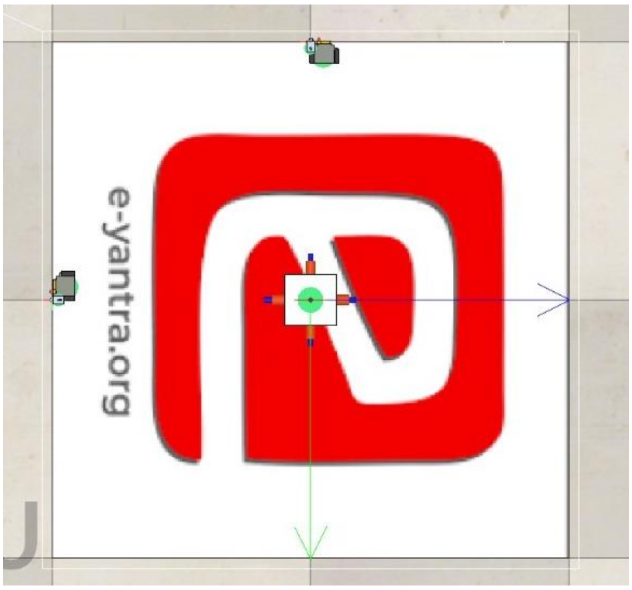


   is rigidly connected to the base plate while the other is rigidly connected to the top plate.
4. The arm assembly consists of a connecting rod and and l connector joint to each other with a spherical joint. The connecting rod is also connected to the servo fin with a revolute joint while the l connector is connected to the top plate rigidly.





5. The two manipulator assemblies are mounted along two perpendicular edges of the base plate as shown in the picture.



Q7. In Task 1C, you were given the task to design the ball balancing platform while in Task 3, you were given the task to use this ball balancing platform to control the position of the ball on top of it. How did your ball balancing platform change between these tasks? (5)

< Explain in brief how your design changed in the subsequent tasks. If your design did not change, then justify your reasons for the same.
Answer format: Text - limit: 100 words. .>

1. In the first design there were four manipulator assemblies, one along each edge of the base plate. This was changed to there being only two manipulator assemblies mounted along two perpendicular edges of the base plate to simplify the operation of the mechanism and to eliminate its conflicting elements.
2. The joint between the connecting rod and the l connector was changed from it being a revolute joint to a spherical joint to facilitate freedom of motion needed to tilt the platform in the required direction.

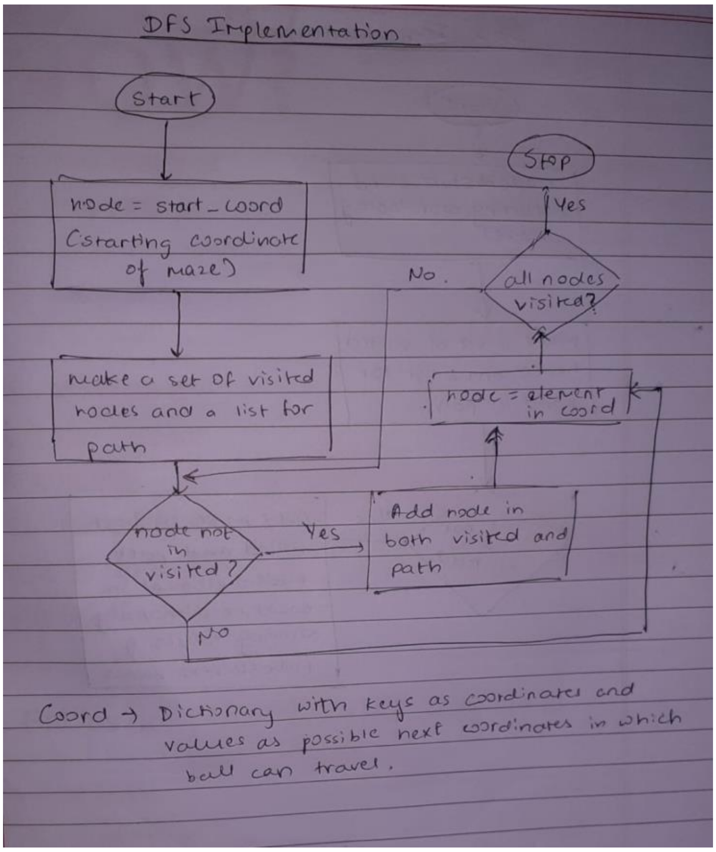
Path Planning

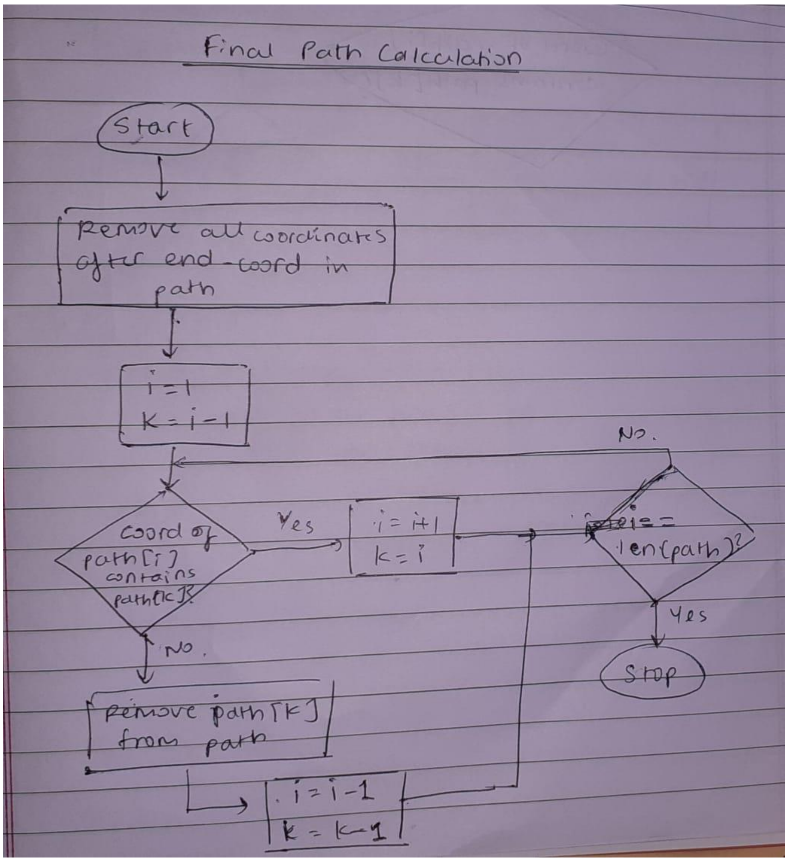
Q8. What kind of path planning algorithm did you use for finding the shortest path for the given maze images (in Task 4A)? (5)

< Explain the logic behind the algorithm and the reason for your choice if any. You can use a pseudo-code and/or flowcharts to help elucidate your answer. >

We looked into several path planning algorithms such as BFS, DFS, Dijkstra's, A* etc. We realized that algorithms such as Dijkstra's and A* are very effective in path planning. However, theses algorithms require some sort of heuristics (cost of path from a particular node to final node) that are obtained from approximation methods or past experiences. Our first attempt was to implement an algorithm similar to A* which dynamically calculates the path by comparing the Euclidean distance of every possible node to the final node, and the node which has the least distance is chosen. This method is repeated for every node. The algorithm turned out to be very tedious to write and not very effective since the node with the least Euclidean distance was more than often not the node required for the correct path.

We thus discarded this algorithm and thought of implementing a simpler but effective algorithm like BFS or DFS. We studied both these algorithms and thought DFS was the best fit since it was faster and practically made sense since the approach used in DFS is similar to how humans think while solving a maze. A path is chosen and it is completely discovered. If this path is unsuccessful, the search is suspended and we explore a different path.


DFS Implementation
Start
node = start_coord
(starting coordinate of maze)
make a set of visited nodes and a list for path
node not in visited?
Yes
No
Add node in both visited and path
node = element in coord
all nodes visited?
No.
Yes
Stop
Coord → Dictionary with keys as coordinates and values as possible next coordinates in which ball can travel.

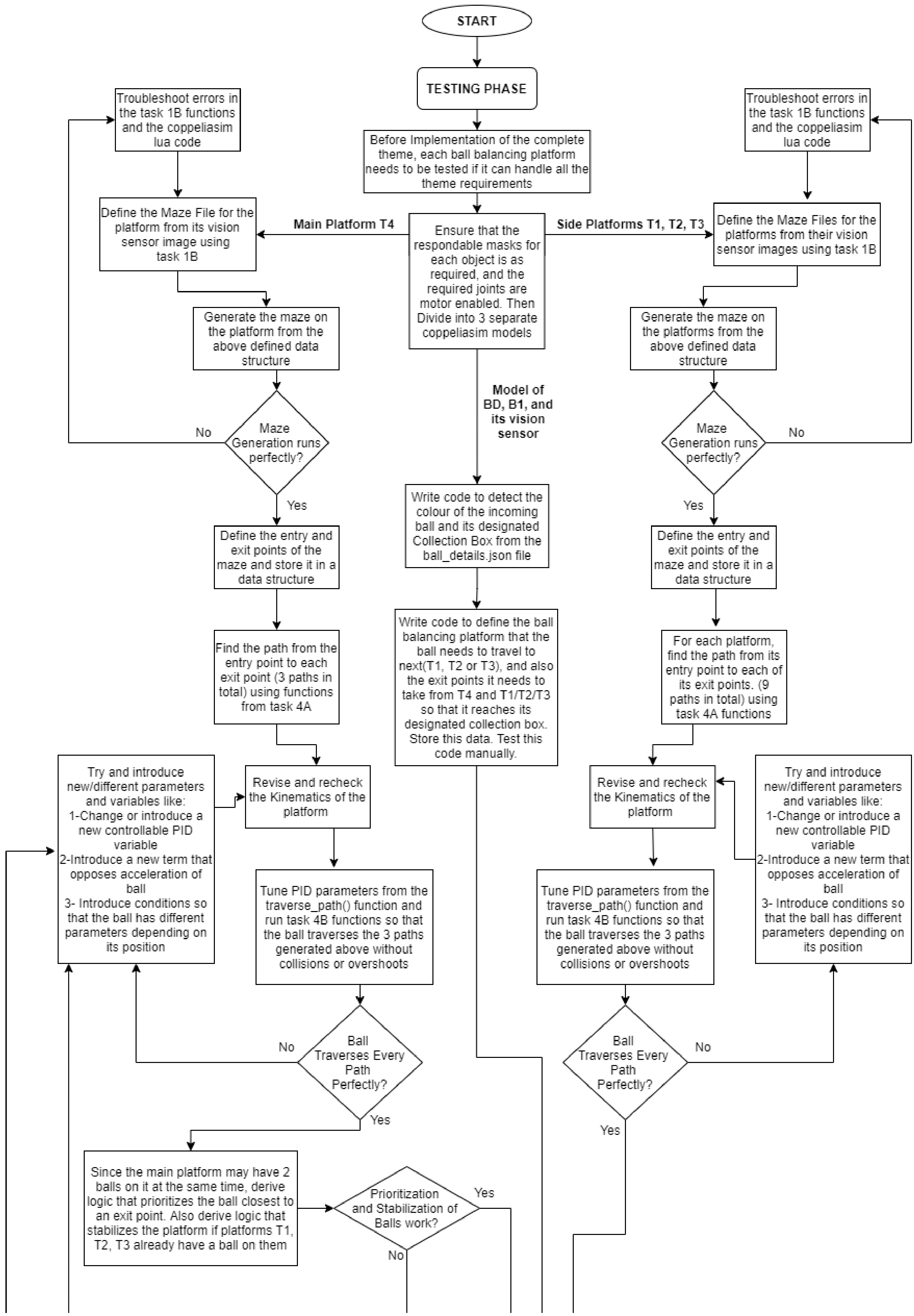

Final Path Calculation
Start
Remove all coordinates after end_coord in path
i = 1
k = i - 1
coord of path[i] contains path[k]?
Yes
i = i+1
k = i
No.
remove path[k] from path
i = i - 1
k = k - 1
i == len(path)?
No.
Yes
Stop

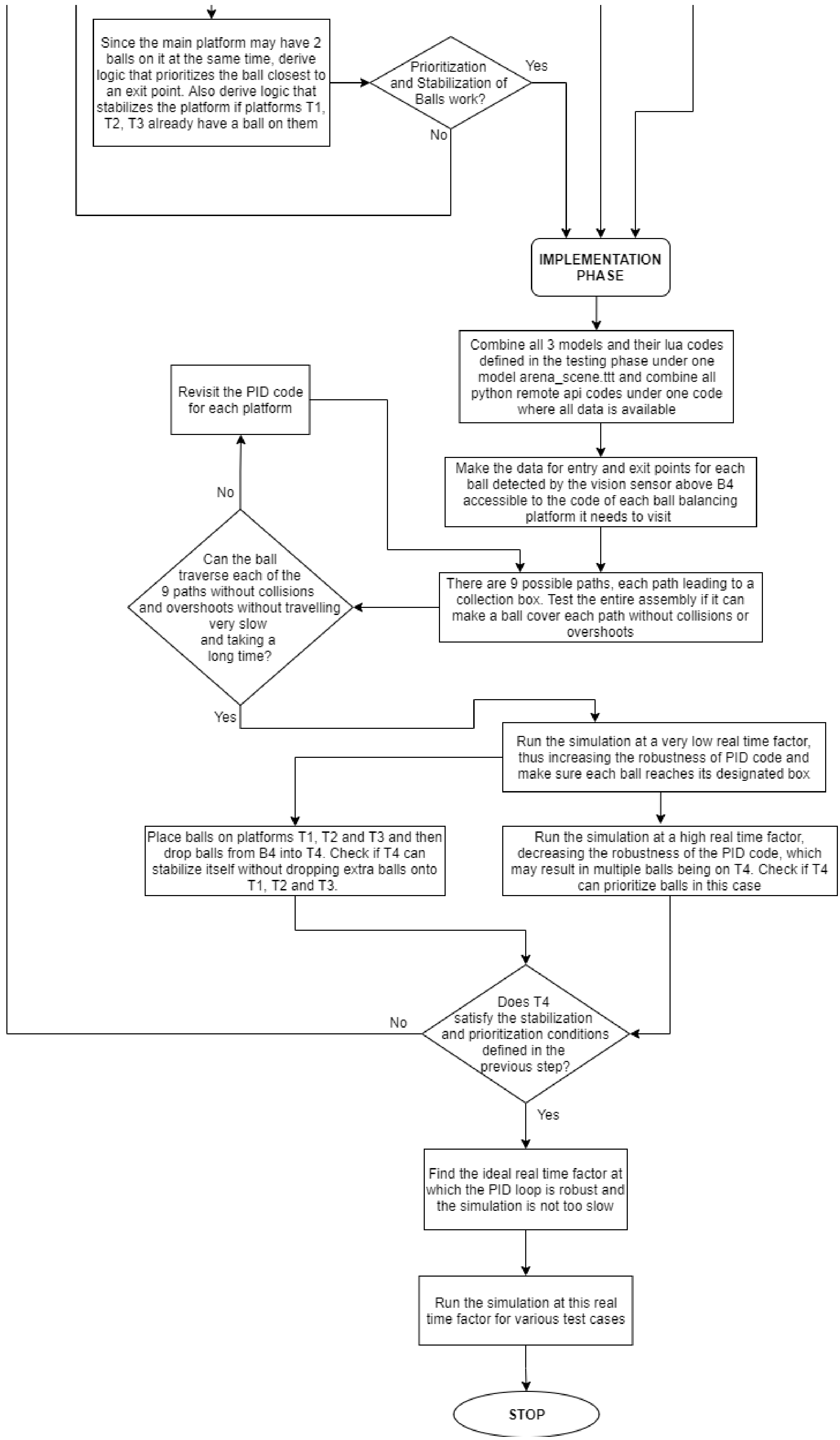
Algorithm Analysis

Q9. Draw a flowchart illustrating the algorithm / strategy you propose to use for theme implementation. (7)

< The flowchart should elaborate on every possible function that you will be using for completing all the Theme Run.

Follow the standard pictorial representation used to draw the flowchart. >


START
TESTING PHASE
Before Implementation of the complete theme, each ball balancing platform needs to be tested if it can handle all the theme requirements
Main Platform T4
Side Platforms T1, T2, T3
Ensure that the respondable masks for each object is as required, and the required joints are motor enabled. Then Divide into 3 separate coppeliasim models
Troubleshoot errors in the task 1B functions and the coppeliasim lua code
Define the Maze File for the platform from its vision sensor image using task 1B
Generate the maze on the platform from the above defined data structure
Maze Generation runs perfectly?
No
Yes
Define the entry and exit points of the maze and store it in a data structure
Find the path from the entry point to each exit point (3 paths in total) using functions from task 4A
Troubleshoot errors in the task 1B functions and the coppeliasim lua code
Define the Maze Files for the platforms from their vision sensor images using task 1B
Generate the maze on the platforms from the above defined data structure
Maze Generation runs perfectly?
No
Yes
Define the entry and exit points of the maze and store it in a data structure
For each platform, find the path from its entry point to each of its exit points. (9 paths in total) using task 4A functions
Model of BD, B1, and its vision sensor
Write code to detect the colour of the incoming ball and its designated Collection Box from the ball_details.json file
Write code to define the ball balancing platform that the ball needs to travel to next(T1, T2 or T3), and also the exit points it needs to take from T4 and T1/T2/T3 so that it reaches its designated collection box. Store this data. Test this code manually.
Try and introduce new/different parameters and variables like:
1-Change or introduce a new controllable PID variable
2-Introduce a new term that opposes acceleration of ball
3- Introduce conditions so that the ball has different parameters depending on its position
Revise and recheck the Kinematics of the platform
Tune PID parameters from the traverse_path() function and run task 4B functions so that the ball traverses the 3 paths generated above without collisions or overshoots
Ball Traverses Every Path Perfectly?
No
Yes
Revise and recheck the Kinematics of the platform
Try and introduce new/different parameters and variables like:
1-Change or introduce a new controllable PID variable
2-Introduce a new term that opposes acceleration of ball
3- Introduce conditions so that the ball has different parameters depending on its position
Tune PID parameters from the traverse_path() function and run task 4B functions so that the ball traverses the 3 paths generated above without collisions or overshoots
Ball Traverses Every Path Perfectly?
No
Yes
Since the main platform may have 2 balls on it at the same time, derive logic that prioritizes the ball closest to an exit point. Also derive logic that stabilizes the platform if platforms T1, T2, T3 already have a ball on them
Prioritization and Stabilization of Balls work?
Yes
No


Since the main platform may have 2 balls on it at the same time, derive logic that prioritizes the ball closest to an exit point. Also derive logic that stabilizes the platform if platforms T1, T2, T3 already have a ball on them
Prioritization and Stabilization of Balls work?
Yes
No
IMPLEMENTATION PHASE
Combine all 3 models and their lua codes defined in the testing phase under one model arena_scene.ttt and combine all python remote api codes under one code where all data is available
Revisit the PID code for each platform
Make the data for entry and exit points for each ball detected by the vision sensor above B4 accessible to the code of each ball balancing platform it needs to visit
No
Can the ball traverse each of the 9 paths without collisions and overshoots without travelling very slow and taking a long time?
There are 9 possible paths, each path leading to a collection box. Test the entire assembly if it can make a ball cover each path without collisions or overshoots
Yes
Run the simulation at a very low real time factor, thus increasing the robustness of PID code and make sure each ball reaches its designated box
Place balls on platforms T1, T2 and T3 and then drop balls from B4 into T4. Check if T4 can stabilize itself without dropping extra balls onto T1, T2 and T3.
Run the simulation at a high real time factor, decreasing the robustness of the PID code, which may result in multiple balls being on T4. Check if T4 can prioritize balls in this case
No
Does T4 satisfy the stabilization and prioritization conditions defined in the previous step?
Yes
Find the ideal real time factor at which the PID loop is robust and the simulation is not too slow
Run the simulation at this real time factor for various test cases
STOP

Challenges

Q10. What are the major challenges that you have faced till now and the ones that you can anticipate in addressing this theme and how do you propose to tackle them? (3)

< Answer format: Bullet points
1. Challenge 1
2. Challenge 2
3. Challenge 3, etc. >

1) Tuning the PID Loop for task 3, the dynamics of the ball balancing platform is very non-uniform in general, since the weight of the ball changes the system based on where it is, the ball moves slower when nearing the middle and moves faster when it is nearing an edge, in which case the differential term cannot keep up with the speed of the ball . Our design itself does not have a linear relationship between the angle of the servo and the angle of the top plate, which is a compromise we had to make for stability and independent control of each servo. This can be tackled by introducing code for forward and inverse kinematics to linearize the function.
2) The applyPerspectiveTransform() function is almost applied in all the tasks and used in different functions, it can act differently for different places its used in and can throw errors. This can be solved by fine tuning it or using different functions for different parts of the code.
3) Due to the limits of computational power, the PID Loop cannot be as robust as it should be in a real-life application, which leads to frequent overshoots of the ball because of the time complexity of each iteration. This forces us to reduce the stiffness of the PID Loop so that the ball travels slower and the code has enough time to run without the ball straying away, which may cause the simulation to exceed the time limit. This can be limited by running the simulation with as low a real time factor as possible, so that we can increase stiffness and make the ball move faster, therefore making the code more robust.

IMPORTANT:

- **The document you submit should be in YOUR OWN WORDS. To avoid any copyright violations, you must NOT copy phrases directly from manuals or the web.**
- **The team should NOT mail or upload the document anywhere else except on the portal.**
- **Teams failing to submit the document by the deadline will lose the marks for this task.**
- **e-Yantra WILL NOT entertain any request for extension of deadline for uploading the task.**
- **e-Yantra holds complete discretion to disqualify a team if any foul play is suspected.**