Practice Task

[*Last Updated on:* ***20th February 2021, 19:30 Hrs***]

NOTE:

- Before proceeding further, make sure you **have gone through the Rulebook**.
- This task will be useful for the teams to **determine their standing/rank in the competition**.

1. Problem Statement

Develop an algorithm **to navigate the ball through the maze** on top of **4 ball balancing platforms** and **deposit it in the mentioned collection box** of the given CoppeliaSim scene.

---

2. Given

1. CoppeliaSim's Scene File (`task_6_scene.ttt`)

- A scene file i.e. `task_6_scene.ttt` of CoppeliaSim software as shown in Figure 1.

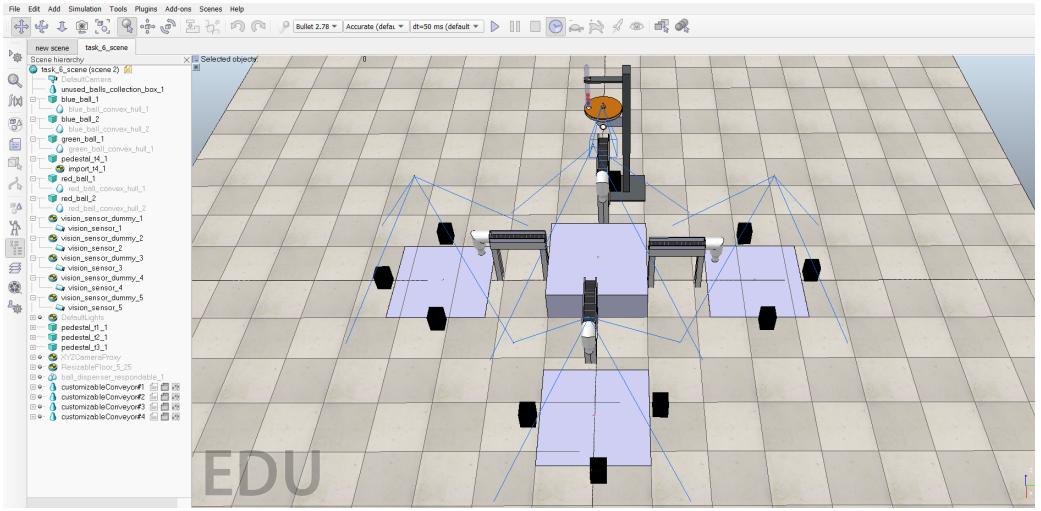

Figure 1: CoppeliaSim scene file for Practice Task and Task 6

- Objects along with their names and uses are provided in **Scene Details** document

- Important Points to **NOTE**:
  - You are **NOT** allowed to **remove** the objects mentioned in **Scene Details**.
  - You are **only ALLOWED to change the 'height/z-coordinate'** of the following objects:
    1. Customizable Conveyor
    2. Down Pipe

- As mentioned in Task 4B, teams **CAN decide the resolution of Vision Sensors**.
- Any other **change in the properties/position/orientation** of the above mentioned objects will lead to **POOR/NULL EVALUATION**.

- Teams will have to **import their ball balancing platform** from Task 5.

Note:

- This scene file **should be used for Task 6 as well unless stated otherwise**.

2. Ball Details (`ball_details.json`)

- This JSON file consists of the details of all balls that will be dispensed by the **Ball Dispenser (BD)**.

3. Maze Images for all the Tables (`maze_t1.jpg` , `maze_t2.jpg` , `maze_t3.jpg` , `maze_t4.jpg`)

- These are the maze images for all the Tables in the scene.

4. Grader Application

- The team **should run the grader Application ONLY after they have completed** writing the `task_6.py` script.

---

3. Getting Started

- Download the following zip file **practice_task.zip** to get started with the task. **Right-click** on the hyperlink and select **Save Link As...** option to download.
  - You fill find the following files and folders in the zip file:
    - **task_6_scene.ttt**
    - **Set_1** - This folder contains json and jpg files that are provided as **1st Set**.
      1. **ball_details.json**
      2. **maze_t1.jpg**

3. **maze_t2.jpg**
4. **maze_t3.jpg**
5. **maze_t4.jpg**

- **Set_2** - This folder contains json and jpg files that are provided as **2nd Set**.
  1. **ball_details.json**
  2. **maze_t1.jpg**
  3. **maze_t2.jpg**
  4. **maze_t3.jpg**
  5. **maze_t4.jpg**
- **Extra_Set** - This folder contains json and jpg files that are provided as practice. These files **cannot** be used for testing your solution with the **Practice Task Grader App v1.1.0**.
  1. **ball_details.json**
  2. **maze_t1.jpg**
  3. **maze_t2.jpg**
  4. **maze_t3.jpg**
  5. **maze_t4.jpg**

- **task_6_scene_details**
- **Practice Task Grader App v1.1.0** (**NOTE: Updated version of Grader App needs to used.**)

NOTE: The browser might warn that the file can harm your PC, but it will not and you can safely download it.

- To work with one of the **set**, copy-paste the **five** (**one .json and four .jpg**) files from **Set_1** *OR* **Set_2** folder to your workspace.

> **Note:** The json and jpg files provided in **Extra_Set** folder are just for practice and should not be used for testing your solution with the **Grader App**.

- Make sure you add following files in the same directory:
  - `task_6.py` - This **must** be the name of your solution script.
  - `sim.py`
  - `simConst.py`
  - `remoteApi.dll` (***for Windows***) OR `remoteApi.so` (***for Linux***) OR `remoteApi.dylib` (***for Macintosh***)
  - You may add other necessary files to this workspace as required.
- Refer **Task 0 Test Setup** for further details.
- Now, read the following instructions carefully.

---

4. Understanding the Task

A. CoppeliaSim Scene (`task_6_scene.ttt`)

(i) General Scene Properties:

- Export the model made by your team in Task 5 as a **CoppeliaSim Model File**. You can refer Figure 10 of Task 1C- Design Ball Balance Platform documentation in order to do so.
- Paste the **.ttm** file in:
  - Ubuntu OS - `CoppeliaSim_Edu_V4_0_0_Ubuntu18_04/models/`
  - Windows OS - `C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\models\`
- Refer Figure 2 of Task 1C - Design Ball Balance Platform to understand better.
- After you open `task_6_scene.ttt` , you have to **add this model file**.
- Use ***import_t1_1***, ***import_t2_1***, ***import_t3_1*** and ***import_t4_1*** dummies to position your ball balancing platforms. Refer Figure 2 of Task 5.

NOTE:

- After positioning your model, make sure to **assign the entire model as a child of respective pedestal**.
- For example:
  - After positioning ***base_plate_respondable_t4_1***, you will have to assign it as a child of ***pedestal_t4_1***.

- You will have to **follow the naming convention as mentioned in Task 5** Document . Make sure to refer the Theme Description to understand the table naming convention.

- Teams are **ONLY ALLOWED to change the 'height/z-coordinate'** of ***customizable conveyor*** and ***down_pipe***.
- Refer the Figure 3 of Task 5 Document to learn how to modify the height of conveyor belt.

(ii) Collection and Collision Objects:

- For this task, you will have to **create collection and collision object** for **EACH** platform in the scene. Refer Table 1.

Sr. No.	Name of the Collection	Elements to be added in the collection
1	colliding_objects_t1	pegs_t1_1, walls_t1_1
2	colliding_objects_t2	pegs_t2_1, walls_t2_1
3	colliding_objects_t3	pegs_t3_1, walls_t3_1
4	colliding_objects_t4	pegs_t4_1, walls_t4_1
5	collection_balls	green_ball_convex_hull_1, red_ball_convex_hull_1, red_ball_convex_hull_2, blue_ball_convex_hull_1, blue_ball_convex_hull_2

Table 1: Naming convention for **Collections**.

- Click on **Calculation Module Properties**  and select **Add new collision object**. Refer Table 2.

Sr. No.	Name of the Collision Object	Element to be checked	Element to be checked against

Sr. No.	Name of the Collision Object	Element to be checked	Element to be checked against
1	ball_collision_t1	collection_balls	colliding_objects_t1
2	ball_collision_t2	collection_balls	colliding_objects_t2
3	ball_collision_t3	collection_balls	colliding_objects_t3
4	ball_collision_t4	collection_balls	colliding_objects_t4

Table 2: Naming convention for **Collision Objects**.

- Make sure to check **ONLY Collidee color changes**. Refer the instructions as mentioned in Figure 5 of the Task 5.
- Refer Table 5, Table 6 and Table 7 for **necessary respondable, collidable and renderable** objects.

B. Python Script (`task_6.py`)

- Teams can modify their `task_5.py` code and rename it as `task_6.py`
- A helper function (given below) **MAY be used** by the teams to **improve the Real Time Factor**

1. `invert_model_properties(object_handle)`

```
def invert_model_properties(object_handle):
    # NOTE: This function will be used to invert model properties such that the model w
    #           traversed in the current iteration can be partially disabled thereby sa
    #           resources.

    global client_id
    return_code, curr_model_prop = sim.simxGetModelProperty(client_id,object_handle,sim
    if(curr_model_prop == 0):
        # Overrides the required model props as NOT measureable, NOT dynamic, NOT colli
        # renderable, NOT detectable, NOT respondable and Invisible to other model's bc
        return_code = sim.simxSetModelProperty(client_id,object_handle,1135,sim.simx_op
    else:
        return_code = sim.simxSetModelProperty(client_id,object_handle,0,sim.simx_opmoc
```

NOTE:

- It is **MANDATORY** to **draw the current path** being traversed by the ball. After the path is traversed, teams should **delete the traversed path**.
- You should **not use** `sys.exit()` in **task_6.py** or any other dependent Python files, as it will stop the execution and will not generate any output files while testing your solution with **Grader Application**.

---

5. Flow of the task

- Teams are provided with a **maximum time limit** of **300 Simulation Seconds** for this task.

NOTE:

- Remember, the task is to ***navigate the ball from one table to another by balancing the ball on the platform tables***. So, make sure your solution doesn't let the ball slide along or touch each wall of the maze without caring about the collisions that will result.
- For the team's submission to be considered VALID, the **average Real Time Factor (RTF) of the entire run** of Task 6 should be **>=0.3 (greater than or equal to 0.3)**.
- Since the team will be implementing their algorithms on four tables, we have **reduced the minimum RTF from 0.5 (as in Task 5) to 0.3**.

- On running `task_6.py` , following should happen:

1. **Maze array** of all the tables should be **transferred to the lua's customization scripts**.
2. **Maze should be generated** on top of **all the tables** in the CoppeliaSim scene.
3. **Color of the ball** should be **determined from *vision_sensor_5*,** as soon as the ball is detected.
4. Color of the ball should now be compared with the contents of the ***ball_details.json*** to **find the correct Collection Box**.
5. Path **should be drawn on top of the maze** just before the ball is about to traverse it.
6. Path drawn on table **should be traversed**.
7. Path **should be deleted** after the ball has traversed the maze.
8. Ball should be **deposited in the designated collection box**.
9. **Points 3 to 8 should be repeated** for all the balls.

---

6. Testing the Solution And Submission Instructions

- Download the executable package from this page.
    - `NB_Task5_GUI-1.1.0_win.exe` (for **Windows OS**)
    - `NB_Task5_GUI-1.1.0_linux.AppImage` (for **Linux OS**)
    - `NB_Task5_GUI-1.1.0_mac.dmg` (for **Mac OS**)
- Refer the App for Grading of Progress Task document for installation and running in your respective OS.
- After the installation is complete, you will find the output as shown in Figure 2 when you run the Grader App.
- Now, download your favorite **Screen Recorder Application**. We recommend following software for recording:
    - ***Apowersoft*** and ***Microsoft Powerpoint*** (for **Windows**)
    - ***SimpleScreenRecorder*** (for **Linux**)
    - ***QuickTime*** (for **Mac**)
- Make sure to **start recording the entire screen** before proceeding to the next step.
- Enter your **Team ID** and **Path of Workspace/Directory** to begin.
- Select ***Set 1*** or ***Set 2***. **Make sure to paste the .jpg and .json files of the selected set in the current directory**.

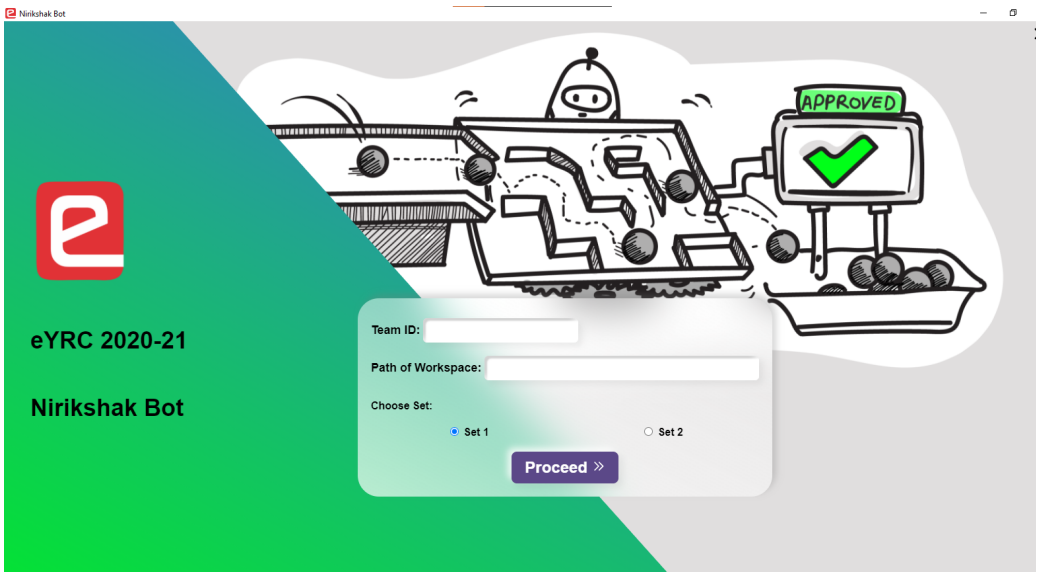

Figure 2: Initial screen of Practice Task GUI

- Now, click on ***Proceed***. A new screen as shown in Figure 3 will appear.

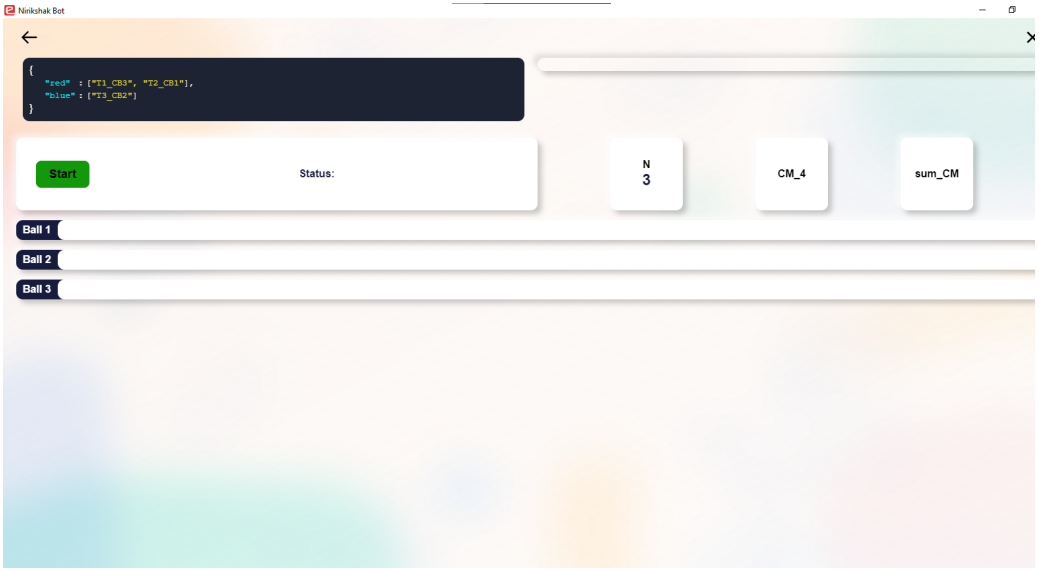

Figure 3: Testing screen of Practice Task GUI

- Now, open the scene file i.e. `task_6_scene.ttt` edited by your team.
- Click on ***Start*** button to begin the evaluation process.
- During the evaluation you will see the screen as shown in Figure 4.

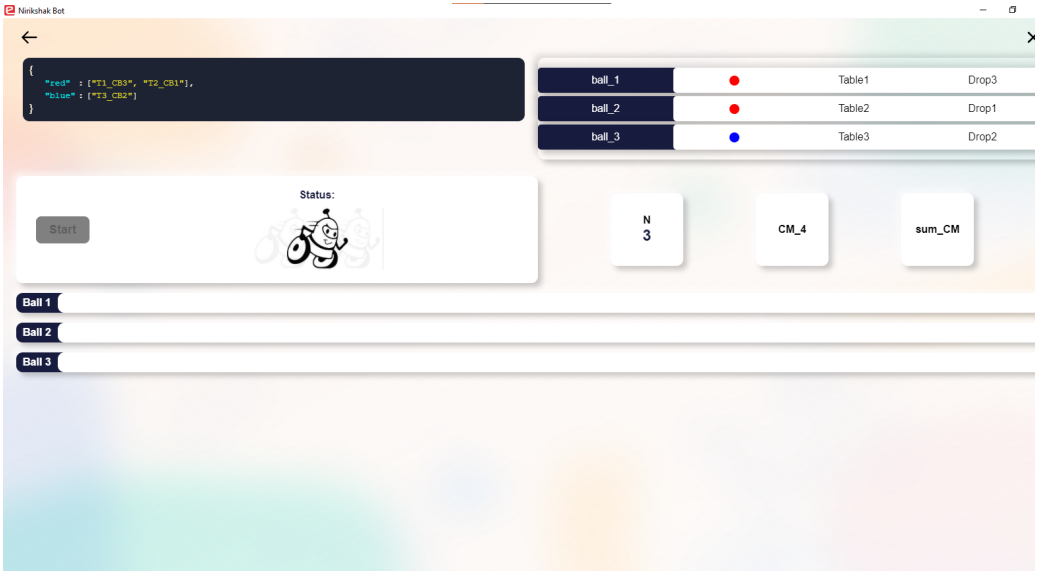

Figure 4: Evaluating the code and scene using GUI

- After testing the solution, you will see the status as one of the following:
  - **Ok Tested** - Team has **CORRECTLY** passed and dropped ***all balls specified in json file*** that are dispensed by BD to its designated CB.
  - **Valid Run** - Team has **CORRECTLY** passed and dropped ***at least one ball*** dispensed by BD to its designated CB.
  - **Invalid Run** - Team has **INCORRECTLY** passed and dropped at least one ball dispensed by BD to its designated CB.
  - **Error** - Evaluation failed. Check the log files as mentioned in App for Grading of Progress Task document. The location of the log file will be as follows:
    - **Windows**:

      ```
      C:\Users\{User_Name_of_Machine}\AppData\Roaming\NB_Practice_Task_GUI\logs
      ```

    - **Ubuntu**:

      ```
      ~/.config/NB_Practice_Task_GUI/logs/
      ```

    - **Macintosh**:

      ```
      ~/Library/Logs/NB_Practice_Task_GUI/
      ```

- **After opening the grader app to show the parameters calculated**, you can stop recording the simulation.

> **NOTE**: You have to use the above ***process of screen recording*** for **Task 6**. Hence the instructions are provided from now on.

- You will see the similar output on Grader App after the theme run is complete as in Figure 5.

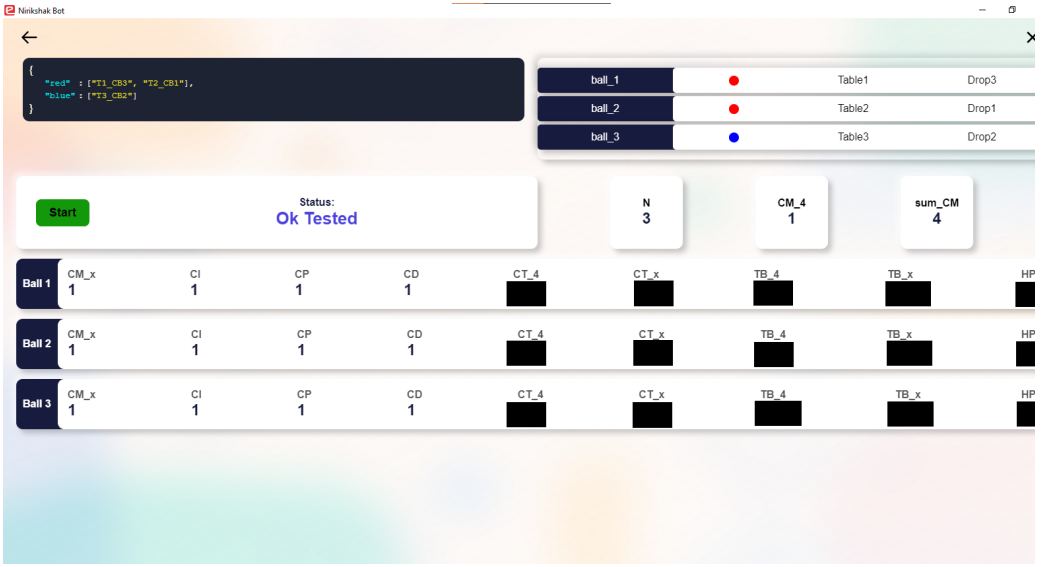

Figure 5: Final Output on Grader App of Practice Task

- The *Grader App* will generate two output files:
  - ***practice_task_ouput.txt*** - this file will be used for knowing your standing / rank from the Leader Board page.
  - ***debugging_practice_task.txt*** - this file can be used to share with Instructors for debuggir errors, if any that occur.
- Now, paste the contents of ***practice_task_ouput.txt*** in the text-area on Leader Board page of eYRC Portal. You will see the following output in Figure 6.

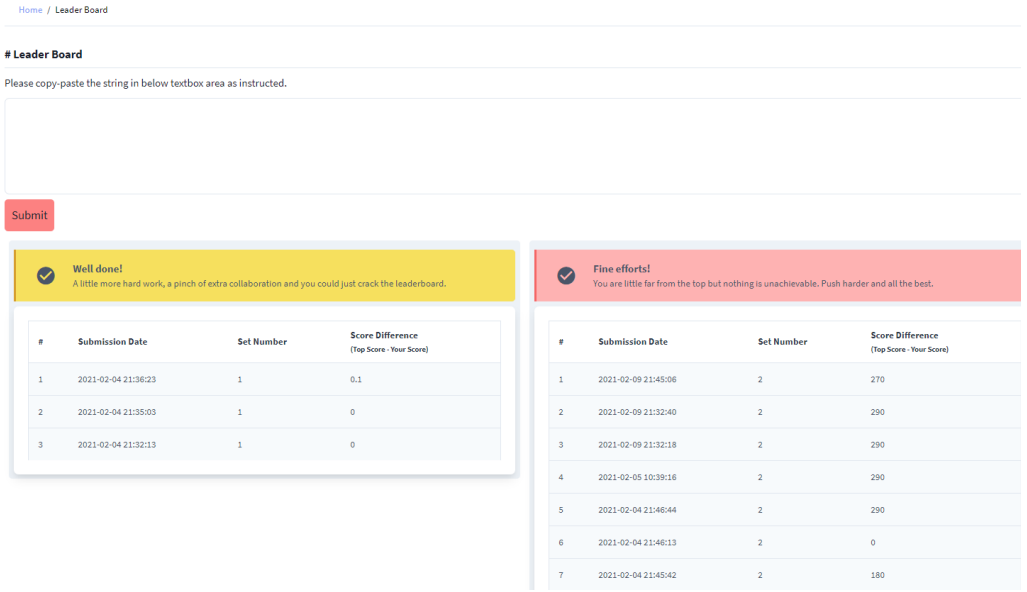

Figure 6: Output on Leader Board page of eYRC Portal

Important Points to consider

- Show the Grader App **ONLY at the beginning and the end of the video**.
- The video **should be recorded in one go and entire screen along with the date and time should be visible throughout**. Any edit made to the video will lead to disqualification.
- The **order of the balls will be randomized** whenever you will run your solution. Hence make sure your code is generic.
- Do **NOT include spaces** in the *workspace path* that you are going to provide to the Grader Ap
- Use contents of ***practice_task_ouput.txt*** in the Leader Board page.

NOTE:

Teams are not supposed to submit Video in this task. However since it will be required in Task 6, we recommend the teams to find the most suitable video recorder app from this task itself.

ALL THE BEST!!