Animation Foundations. Delivery 1

Introduction

You are asked to build a library to control an articulated robot. For security reasons, everything affecting the robot in a separate library. In addition, this library cannot have any dependency with the Unity libraries. For this purpose, we design a class called MyRobotController, which acts as an interface between the robot and Unity. The public methods of this class are the API to the robot.

Things to notice

Before you start implementing anything, please notice that: 1. In the repository there is a folder called *MyRobotLibrary*, which contains a C# solution that generates a library called *RobotController.dll*. Notice that this solution also contains a .cs file called *MyRobotController*. Notice also that this solution does **not** depend on the Unity Engine. Check that you can compile it well, otherwise check the pdf file called *library_setup_help.pdf* inside the folder *MyRobotLibrary*.

2. In the folder assets there is a copy of the *RobotController.dll* library. In the Unity project there is also a scene called *Test4unity.unity*. In this scene, the root of the robot has a script that calls the methods of class *MyRobotController*, as defined **inside** the library *RobotController.dll*. Pressing keys 1, 2 and 3 activate the calls to the methods associated with each of the exercises.
3. Pressing key 0 will activate exercise 0, which corresponds to a neutral case. Each time you press key 0 you will also leave the target object. In addition, pressing key "r" will reset the position of the target object.

Requirements (1 point)

Please make sure you comply with the following requiremenets:

1. In the readme file of the code repository, a section with title *Team Description*. Make sure in that section there is the description of your group, including the ID of the group, names of the team members, their emails, and their pictures.
2. The **only** assets committed are the source code files relevant for the library you are implementing, and one compiled .dll that can run directly with the compiled Unity project provided. Submitting files from folder *MyUnityProject* is not accepted.
3. The final commit of this delivery will be in a merge of the develop branch into the **master branch** with a tag containing **delivery 1*.
4. You have a .gitignore properly configured
5. You follow the git flow conventions. In particular, each exercise is in a separate branch.
6. It will be possible to move to each exercise in any order of key presses. For example, loading exercise 3, then exercise 1 and finally exercise 2 should work. Any other order should also work.
7. On function Hi() from the dll make the string contain both your names.

Requirements 1 and 2 are mandatory. This delivery will not be corrected if your repository does not satisfy them. Completing the exercises with the rest of requirements will give you one point.

Exercicse 1 (3 points)

In class *MyRobotController* implement the internal methods *Multiply* and *Rotate*. Once implemented, use these to implement the public method *PutRobotStraight*. This method will instantly move all the joints to place the robot in a position such as that the end effector touches the object called *Stud_target*. This functionality will have been developed in a branch called *feature/delivery1exercise1*, and a merge to the develop branch once completed.

Exercise 2 (4 points)

In class *MyRobotController* implement the method *PickStudAnim*. This method will interpolate a movement that will make the robot endEffector pick the object called *Stud_target* and leave it on object *Workbench_destination*. Notice that the project is set in such a way that, when the method returns false, the object is left and falls with gravity. When pressing key 2 in the main unity project, the robot will start from its current position and make a smooth movement to place the object in the targeted position. Adjust the movement in order the movement matches the physical constraints in the scene (mainly, other objects and walls), as well as the constraints of the robotic arm (mainly, the rotation axis).

Exercise 3 (2 points)

In class *MyRobotController* implement the method *PickStudAnimVertical*. This exercise is equivalent to the previous, with the difference that we want the main object to be placed horizontally, and carefully, on the *Workbench_destination*. For this purpose, notice the swing of *rot3* will be applied on the robotic joint *JOINT 3*, while the twist of *rot3* will be applied on the robotic joint *JOINT 4*, following the decomposition introduced in class.