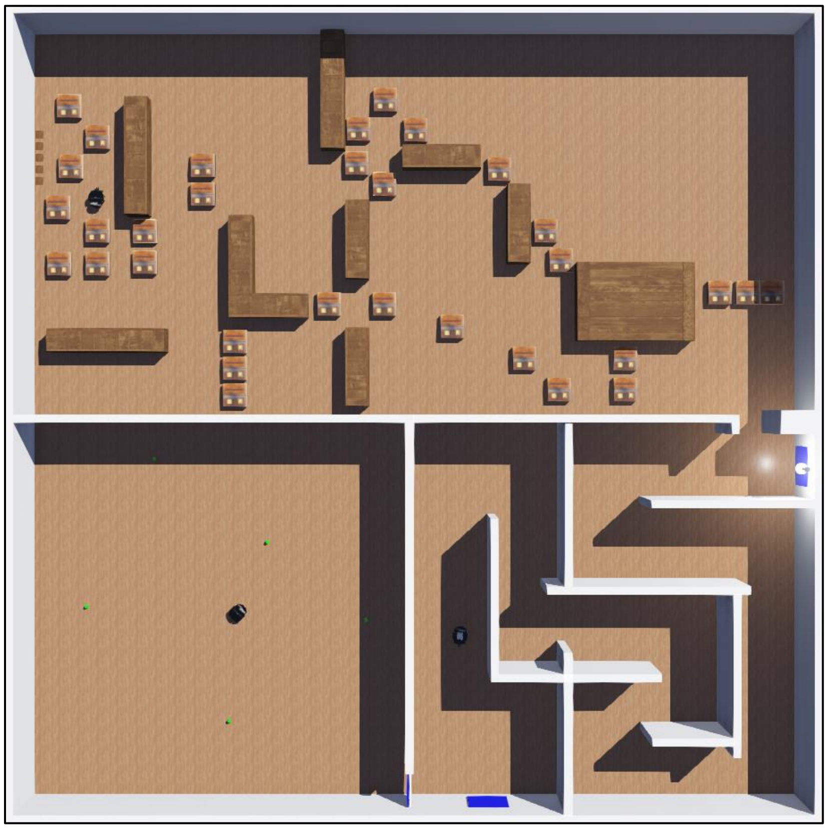
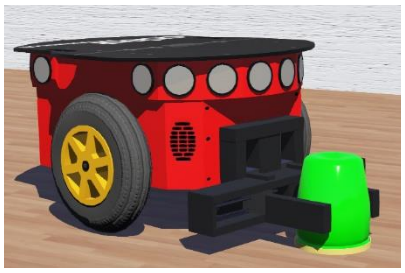
PROJECT – Due Sunday, Dec 10 at 11:59pm

(1) Download the **Project.zip** file and unzip it. Load up the **ProjectWorld** world:

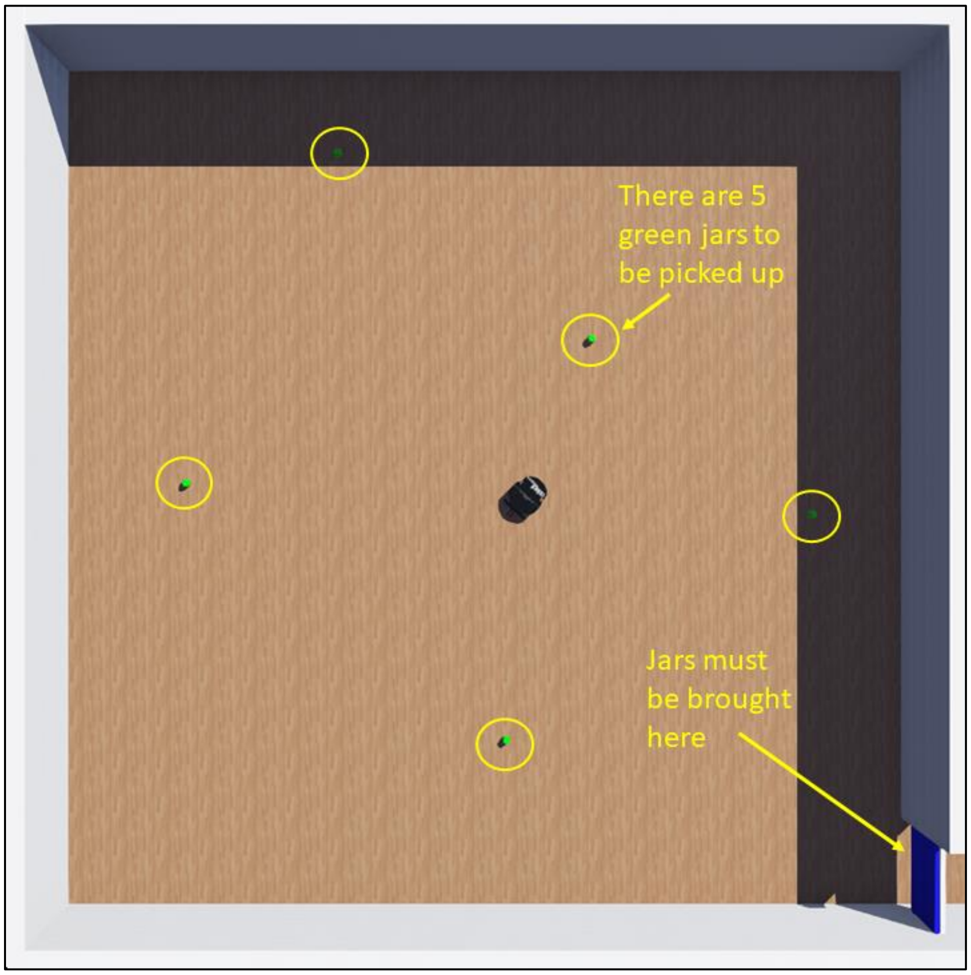




The overall goal of this project is to get **3 robots** to work together to deliver a set of **5 jars** from the bottom-left quadrant of the environment to specific locations in the top left quadrant. You MAY NOT modify the robot sensors/actuators nor the environment world file. You will work together as teams (chosen by the instructor) of three students to accomplish this … where each student writes the controller code for one of three environmental zones as the robots will pass the jars from one robot to the next. Each zone will have its own set of rules. You will need to decide which student will work on which zone. In my opinion, the first zone is the easiest, the last zone is the hardest. You may want to distribute tasks according to your humble programming abilities. Please do not argue with your teammates. Ultimately, you will all end up working together since all three robots must be working properly to get the task done. So, please spend time helping your teammates if they are having problems. Please realize that if the first robot fails, then the other robots will not be able to accomplish their task.

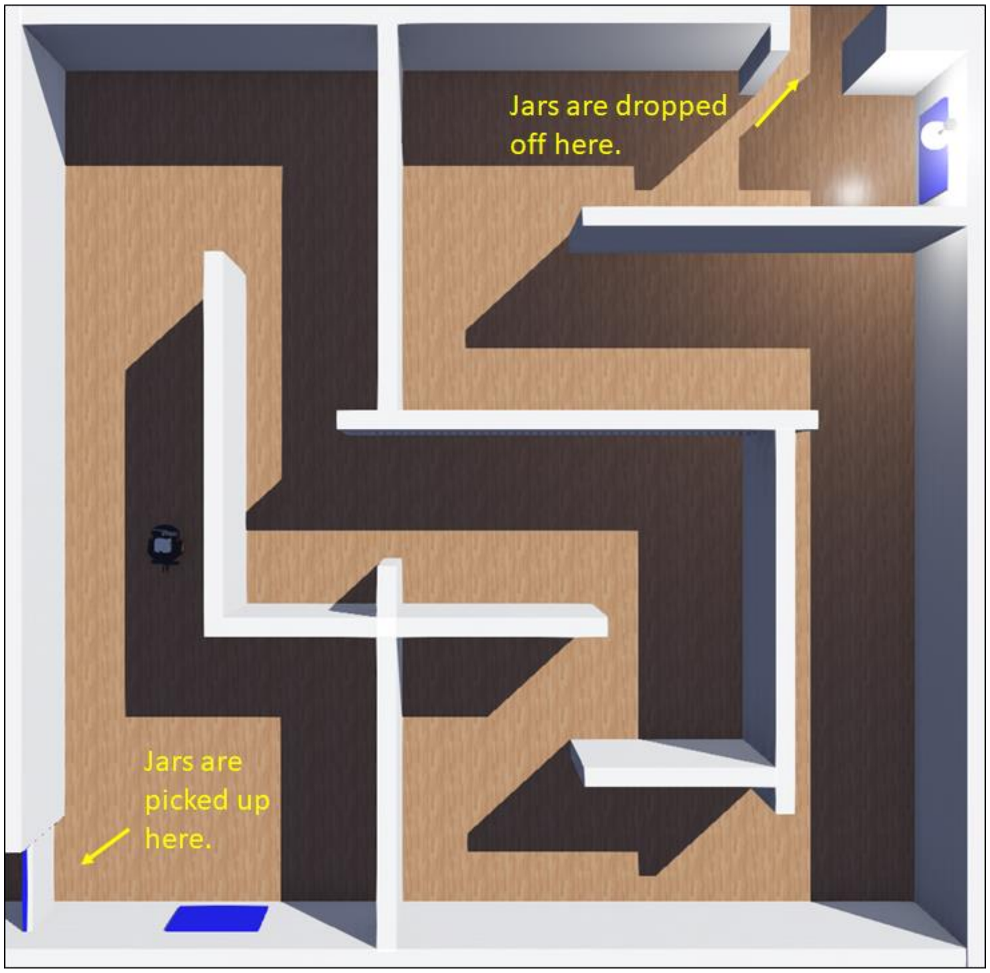
Each student will receive individual marks according to how well their robot accomplishes its task. The team itself will also receive marks according to how much was accomplished in the end.

<u>ZONE 1:</u>


There are 5 green jars to be picked up
Jars must be brought here

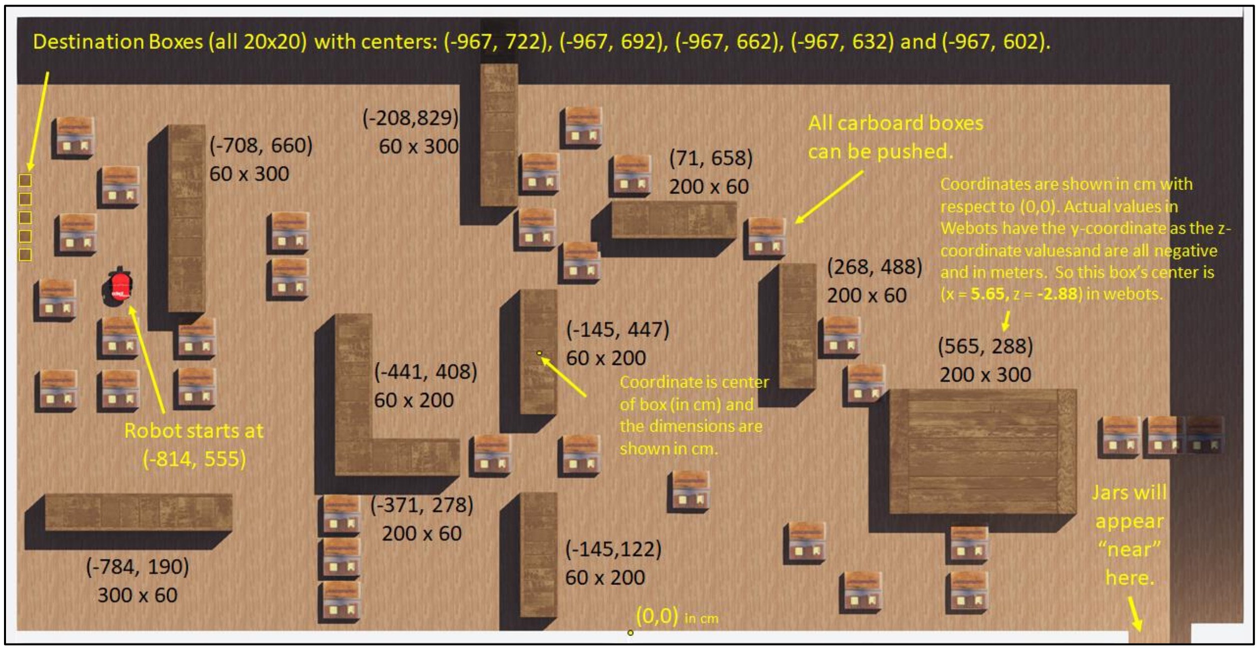
- The robot's code must be written in **ProjectController1.java** which has some code already written to get you started.
- The robot **MUST NOT** use a GPS nor a **Supervisor** class nor anything that gives the robot's location or orientation directly. If you want to know the (x,y) position at any time, you will have to compute it yourself. The compass can be used to obtain the orientation.
- The robot **MUST** find each jar and then drop it off at the bottom right corner so that the robot in Zone 2 can pick it up.
- The robot **MUST NOT** travel into Zone 2 at any time …its body is allowed to go into Zone 2 up to half its body length at most.
- Once the robot has completed dropping off 5 jars, it should stop running with a **System.exit(0);** call.

ZONE 2:



- The robot's code must be written in **ProjectController2.java** which has some code already written to get you started.
- The robot **MUST NOT** use a GPS nor a **Supervisor** class nor anything that gives the robot's location or orientation directly. If you want to know the (x,y) position at any time, you will have to compute it yourself. The compass can be used to obtain the orientation.
- The robot has a laser range finder that can be used, whereas the other two robots do not have this sensor.
- The robot **MUST** pick up each jar from the bottom left then drop it off at the top right corner so that the robot in Zone 3 can pick it up.
- The robot **MUST NOT** travel into Zone 1 **NOR** Zone 3 at any time …its body is allowed to go into either zone up to half its body length at most.
- Once the robot has completed dropping off 5 jars, it should stop running with a **System.exit(0);** call.

Zone 3:

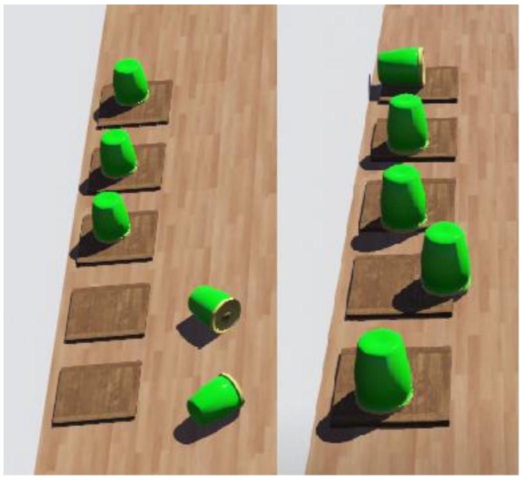

Destination Boxes (all 20x20) with centers: (-967, 722), (-967, 692), (-967, 662), (-967, 632) and (-967, 602).
(-708, 660)
60 x 300
(-208,829)
60 x 300
(71, 658)
200 x 60
All carboard boxes
can be pushed.
Coordinates are shown in cm with respect to (0,0). Actual values in Webots have the y-coordinate as the z-coordinate valuesand are all negative and in meters. So this box's center is (x = 5.65, z = -2.88) in webots.
(268, 488)
200 x 60
(-145, 447)
60 x 200
(565, 288)
200 x 300
(-441, 408)
60 x 200
Coordinate is center of box (in cm) and the dimensions are shown in cm.
Robot starts at
(-814, 555)
(-371, 278)
200 x 60
(-784, 190)
300 x 60
(-145,122)
60 x 200
Jars will appear "near" here.
(0,0) in cm

- The robot's code must be written in **ProjectController3.java** which has some code already written to get you started.
- The robot **MAY** use a **Supervisor** class to give the robot's location or orientation directly so that you have the (x,y) location at all times. You may use the compass to obtain the orientation.
- The robot **MUST** pick up each jar from the bottom right then drop them off onto the 5 Destination boxes at the left size of the zone.
- The robot **MUST NOT** travel into Zone 2 at any time …its body is allowed to go into either zone up to half its body length at most.
- Once the robot has completed dropping off 5 jars, it should stop running with a **System.exit(0);** call.
- There are many cardboard boxes in the environment. These can be pushed out of the way at any time.
- Coordinates (and dimensions) of all wooden-box obstacles are given with respect to their centers. The origin (0,0) is shown at the bottom center of the zone.

Marking for Individual Students (15 marks)

If the robot uses a Supervisor class (when not allowed to), or the student has modified the robot by adding/removing sensors/actuators … the student gets **0** on the project!!! If the student copied any portions of the code from a student on another team, the student gets **0** on the project and the case is sent to the Dean's Office for investigation of plagiarism. You are allowed to get help writing your code from your teammates … but your teammates have their own robot to work on, so please do not use up their time too much. The following marking scheme is used:

- **5 marks** – one mark for each jar robot finds and grabs successfully
- **1 mark** – robot quits after delivering (or believing that it has delivered) 5 jars
- **1 mark** – robot avoids collisions
- **5 marks** – for each jar robot drops off to the "correct" location. For zones 1 and 2 … this will be the doorway to the next zone. For zone 3, each jar must be **on** a separate destination box, dropped off in any order. For example, the image on the left is worth 3 marks … the image on the right is worth 5 marks.



- **-0.5 marks** – for each jar that your robot causes to fall over (this might happen when a jar is already there and gets pushed over by the next jar delivery or it could happen if the robot loses the jar while travelling). For zone 3, jars that fall over on a destination box will not lose marks.
- **-0.5 marks** – each time the robot runs into a wall
- **3 marks** – Your ranking on your 3-person team. If you have the highest points on your team, you get 3 marks, 2nd highest gets 2 marks and lowest gets 1 mark. If a tie, then all students get the higher mark.

Overall Team Performance (5 marks)

Each team will perform to see who can accomplish the task the fastest. Be careful though … the faster your robots travel, the more likely you will have collisions, drop a jar or knock over jars! Each team will be sorted by their performance time. Each student on a team that is in the top 20% will receive 5 marks. For the next 20%, the team members will receive 4 marks, etc… where the slowest 20% of the teams will receive 1 mark.

NOTE: Your code may work fine when you run it … but it may fail when running with your teammate's code. Even the instructor's solution code does not work perfectly 100% of the time. So, your grade will result from how well your code works when the TA runs it. Keep in mind that it may not run as well as when you ran it. TAs WILL NOT be re-running your code a second time, so please be prepared (emotionally) to lose a couple of more marks than you expected.

PROJECT DELIVERABLES

- Each student must submit his/her **controller code** on Brightspace, along with **all other .java files needed**. Make sure that your **name** and **student number** is in the first comment line of your controller code. If your code does not compile, you will unfortunately get zero on the project. If you hand in the wrong version of your code, you will unfortunately get 0 on your project. So please, after submitting your code, download it again and make sure that the version you submitted actually works. DO NOT submit any newer versions of your code after the deadline. If you do, your code will be considered late and you will get 0.

- You are NOT to organize your code into any kind of java packages since the TA must be able to compile and run your source code as submitted. If you submit the **wrong version** of your file, then it is bad news for you … there is no re-submission of files after the due date. Your team will be very angry if you make a mistake on this as they will lose the Team Performance marks if your robot doesn't work!!!!

- There must be a **README** file indicating the **members on the team** (use full proper names that match the name of the registered student) and **which zone** you were responsible for.

- You will want to prove that your portion of the project works properly. To do so, you should **record a video** (use the movie button in the simulator) of your code working. For zones 2 and 3, you will likely have to manually place the jars at the pickup locations during the recording. Have a couple of them placed singly and have a case where 3 are placed very close together … two of them touching… as this is a realistic test case). When recording, chose a reasonable resolution so that it is clear what is happening, but not too high of a resolution that the file is huge. After recording, it seems to take a minute or so for the file to be saved. You will need to **post this video on YOUTUBE and provide a link to it** (that is NOT password protected … double-check this) in your README file so that the TA can access it. Your video should not be posted publicly, but privately. Here is a short video explaining how to do this:

  https://www.youtube.com/watch?v=JOr7JluzEOM

- **REMINDER:** YOU MUST **NOT POST ANY SOURCE CODE** ONLINE … NOT EVEN ON GITUB. YOU DO NOT HAVE PERMISSION. IT WOULD THEREFORE BE COPYRIGHT INFRINGEMENT.