Details about world 0.0.10 and robot Sakura

World:

* North direction changed for the compass usage.
* Contact properties changed (track and default, heave and default) to avoid sinking.
* Track: coulomb Friction: 500, force dependent slip:1, softCFM:1e-05.
* Food object might also need a specified contact property because it needs to slip on the robot.
* Texture background and light changed to “empty park” for a brighter world.
* Floor colour: 0.12, 0.12, 0.12

Task 1:

* Line: straight road segment and curved road segment with width of 0.1m. The gap between the line and floor is 0.0001m

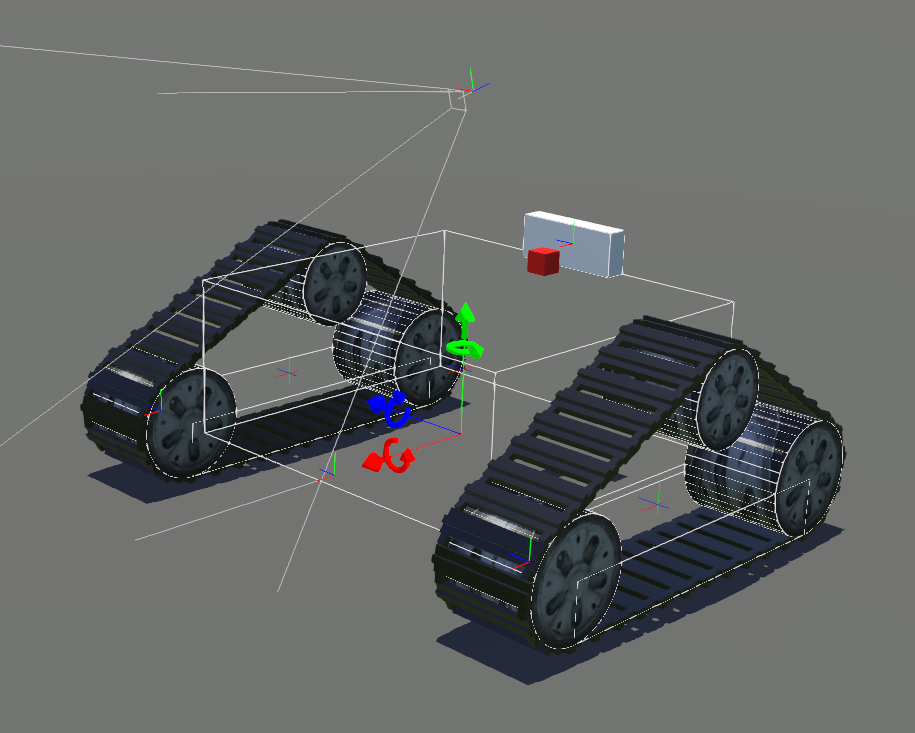
Task 2:

* Orange box: size is 0.36x0.12x0.12, colour is 1, 0.3, 0.

Task 3:

* Bridge: three straight road segments. Width is 1m. The width and height of its border is 0.1m
* The forest has no boundary, additional bounding objects are needed if the distance sensor is used.

Sakura:



Only the functional structure is designed yet. The appearance and skin will be designed by TMS.

Devices and their names used in Python:

* Two rotational motor for two tracks (“left motor”, “right motor”)
* One rotational motor for the rotating camera (“tower rotational motor”)
* One linear motor for releasing the fish food (“trigger motor”)
* One camera (“camera”)
* One compass for detection the vehicle orientation (“compass”)
* One distance sensor for detecting the obstacle in front of the vehicle (“distance sensor”)

It will be complicated to use those devices directly in the code, a better option is write them in *functions* to avoid repeating. For python, *class* is often used. Here is the *vehicle* class.

Vehicle

* getStage(), get the current working stage (which task)
* setStage(stage\_num), change the working stage (used when a task is completed)
* getCompass(), get compass value, the return value is the angle between the front direction of Sakura and North (0-2pi)
* getDistanceValue(), get the value of distance sensor.
* towerSeeLeft(), rotate the camera view 90 degree to left.
* towerSeeRight(), rotate the camera view 90 degree to right.
* towerRestore(), restore the camera view to front.
* releaseFood(), trigger the food motor and release the food.
* setSpeed(left, right), set the speed of two track motors.
* turnRound(diff), turn round, diff is the difference between the current direction and the target direction, positive for left and negative for right. Returns false if diff is small enough, return true otherwise
* linePatrol(), get the camera image, process it and set speed based on line patrolling algorithm return False if there is no line.
* boxPatrol(), get camera image and find orange box, then adjust the speed to go to the box return False if there is no box. Returns true when the box is at the center of the view and the vehicle is stopped.
* bridgePatrol(), get camera image and find bridge, then adjust the speed to go to the bridge return False if there is no bridge.
* archPatrol(), get camera image and find arch, then adjust the speed to go to the arch return False if there is no arch
* colourPatrol(), for task 5, parameters not determined yet.

The elements ending with “()” are functions in class vehicle. Note that functions can call other functions in the same class, for example, when calling the vehicle.linePatrol() function, the camera function vehicle.camera.getImage() (predefined by Webots, not shown here) is called, after processing the image, the robot determines the speed and set it by calling vehicle.setSpeed(left, right).

Working procedure (pseudocode):

0: Not relsead, 1: releasing, 2: released

releaseStatus = 0

0: not passed, 1: passing, 2: passed

passBridge = 0

task5Stage = 0

while 1:

switch (vehicle.getStage()):

case 0:

if (no green is detected):

vehicle.setStage(1)

else:

go ahead

case 1:

if not vehicle.linePatrol():

vehicle.setStage(2)

case 2:

if not released:

make sure the camera is looking its right (vehicle.towerSeeRight())

if vehicle.boxPatrol():

releaseStatus = 1

elif releaseStatus == 1:

set target to right

camera see front

if (vehicle.turnRound(target - vehicle.getCompass())):

pass

if vehicle.getDistanceValue() > 900:

vehicle.setSpeed(0.1, 0.1)

else:

vehicle.releaseFood()

releaseStatus = 2

vehicle.setStage(3)

case 3:

if not passBridge:

set target to front, camera look left

if vehicle.bridgePatrol():

passBridge = 1

elif passBridge ==1:

set target to left, camera see front

vehicle.turnRound(target - vehicle.getCompass())

if vehicle.getDistanceValue() > 700 (or smaller, depends on the value before reaching the bridge)

vehicle.setSpeed(0.1, 0.1)

else:

passBridge = 2

vehicle.setStage(4)

case 4:

set target to front, camera see left

if vehicle.archPatrol():

vehicle.setStage(5)

case 5:

if not task5Stage:

set target to left, camera see front

if vehicle.linePatrol():

task5Stage = 1

elif task5Stage == 1:

if colour is detected:

task5Stage =2

else:

vehicle.linePatrol()

elif task5Stage == 2:

not determined yet