Introduction to Cognitive Robotics

Module 11: Cognition-enabled Robot Manipulation with CRAM

Lecture 3: Error handling and recovery using different arms

David Vernon
Carnegie Mellon University Africa

www.vernon.eu

Simple Fetch and Place Plan

There are two ways to follow this lecture

1: To walk through the process, interactively adding and testing functionality

This follows the CRAM zero prerequisites demo tutorial here:

`http://cram-system.org/tutorials/demo/fetch_and_place`

2. To walk through the process with reference to complete implementation

This follows the version of the tutorial here:

`http://www.vernon.eu/wiki/Zero_Prerequisites_Demo_Tutorial:_Simple_Fetch_and_Place`

Since all of the code is provided, this approach is faster to complete

Simple Fetch and Place Plan

Demonstrate how to write a plan for a simple task

- Pick-and-place
  - Pick an object from one position
  - Place it in another position in the world.
- Error handling
- Recovery behaviors

To look in different places for the object

Environment Setup

Use `roslaunch` to instantiate the required ROS nodes

- PR2 robot
- Kitchen (specified just like a robot)
  - Doors have revolute (rotational) joints
  - Drawers have prismatic (translational) joints
  - Door handles have fixed joints

```
$ roslaunch cram_pick_place_tutorial world.launch
```

Command entered in a terminal

Configuration file

REPL Setup

(Read-Eval-Print Loop)

Use `roslisp_repl` to launch the Lisp compiler's interactive front-end: REPL

```
$ roslisp_repl &
```

Command entered in another terminal

REPL Setup

(Read-Eval-Print Loop)

And then, from REPL, at the Common Lisp User prompt `CL-USER>`

- Load the cram bullet world tutorial
- Make the cram-bullet-world-tutorial package current (this changes the prompt)

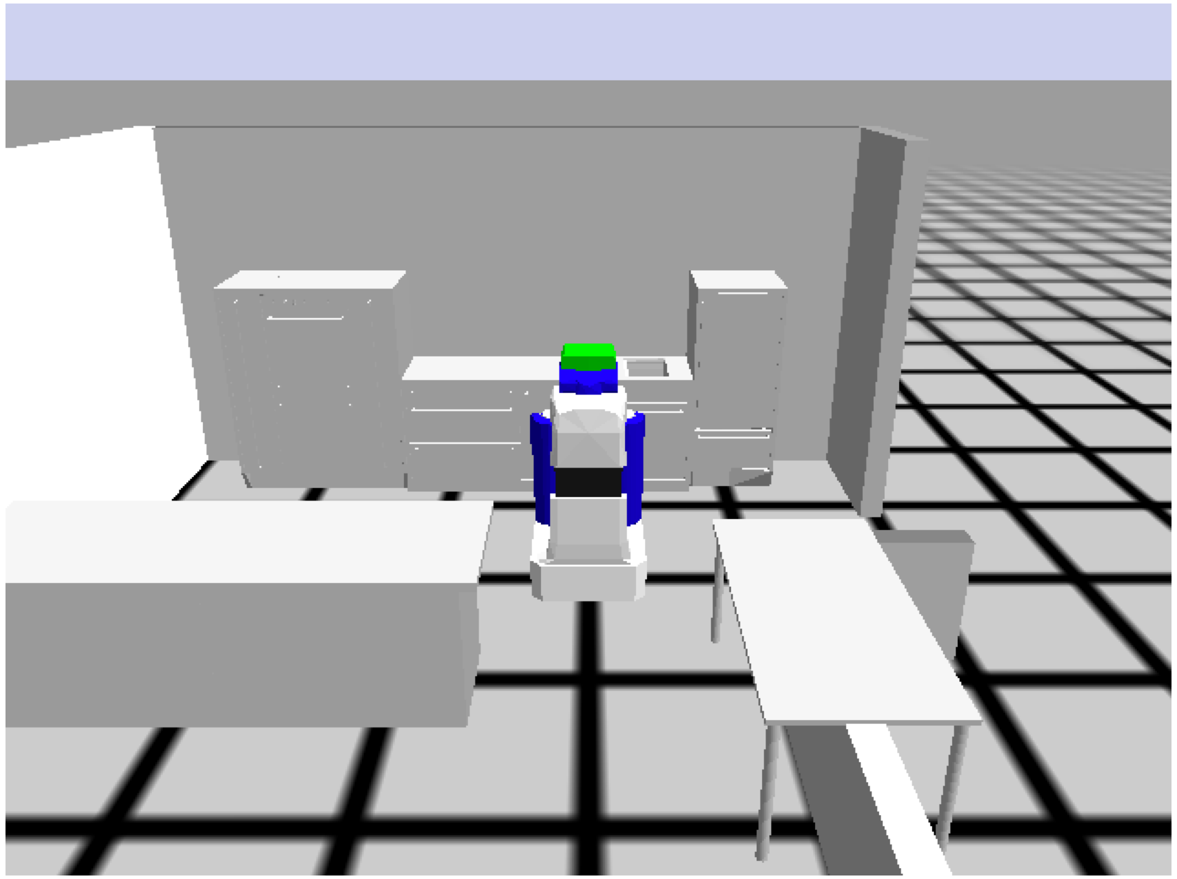
```
CL-USER> (ros-load:load-system "cram_pick_place_tutorial" :cram-pick-place-tutorial)
CL-USER> (in-package :cram-pick-place-tutorial)
```

- Start everything up

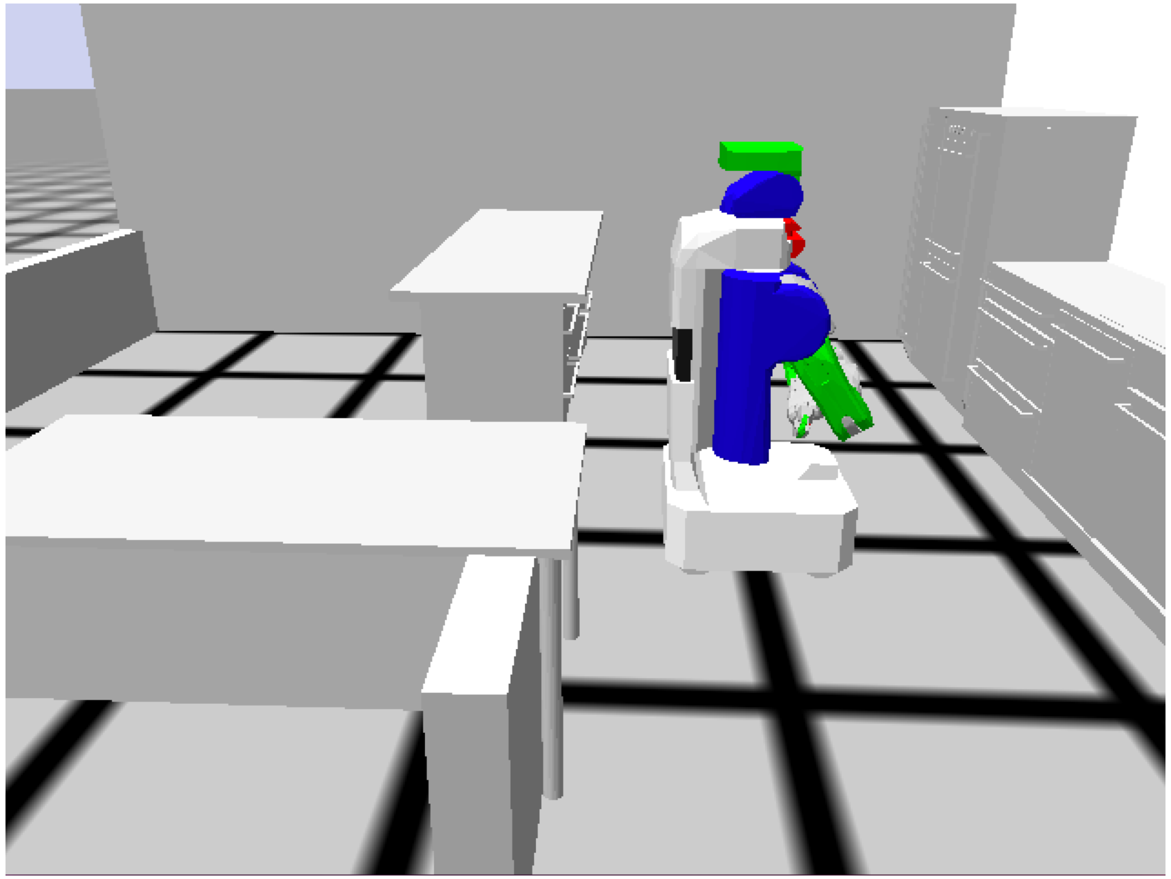
```
PP-TUT> (roslisp-utilities:startup-ros)
```

  - This will take some time (up to a minute)
  - It will launch a Bullet Real-Time Physics Simulation visualization window

Bullet Real-Time Physics Simulation

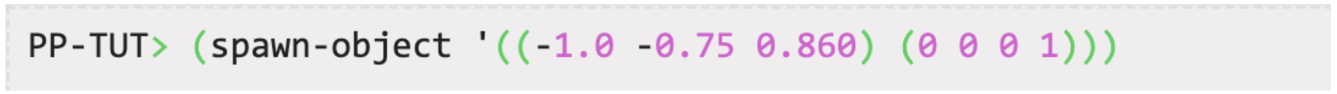


Bullet Real-Time Physics Simulation



Simple Fetch and Place Plan

- So far, so good for the scenarios envisaged so far
- What happens if the bottle is placed somewhere else …

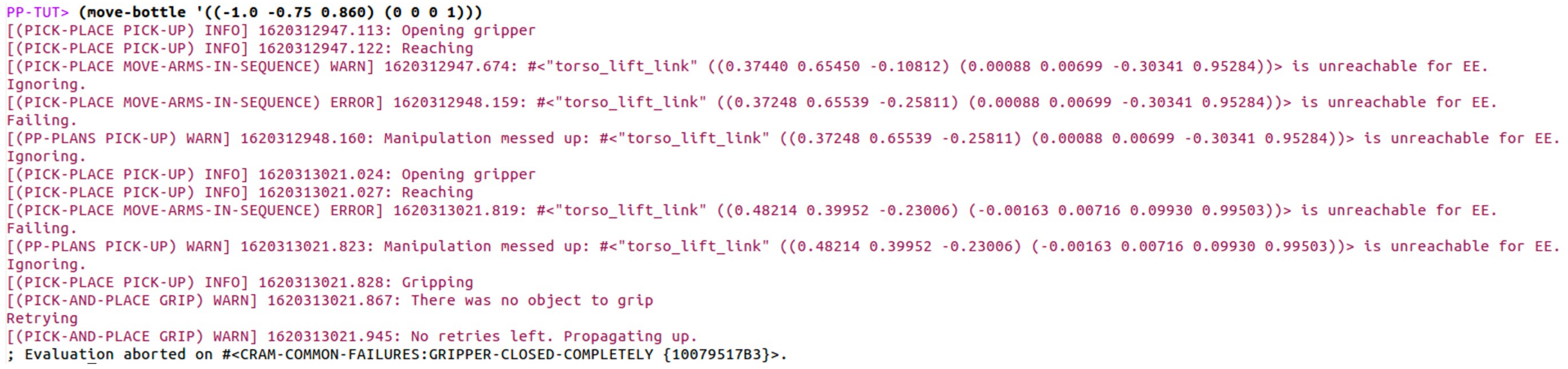


```
PP-TUT> (spawn-object '((-1.0 -0.75 0.860) (0 0 0 1)))
```

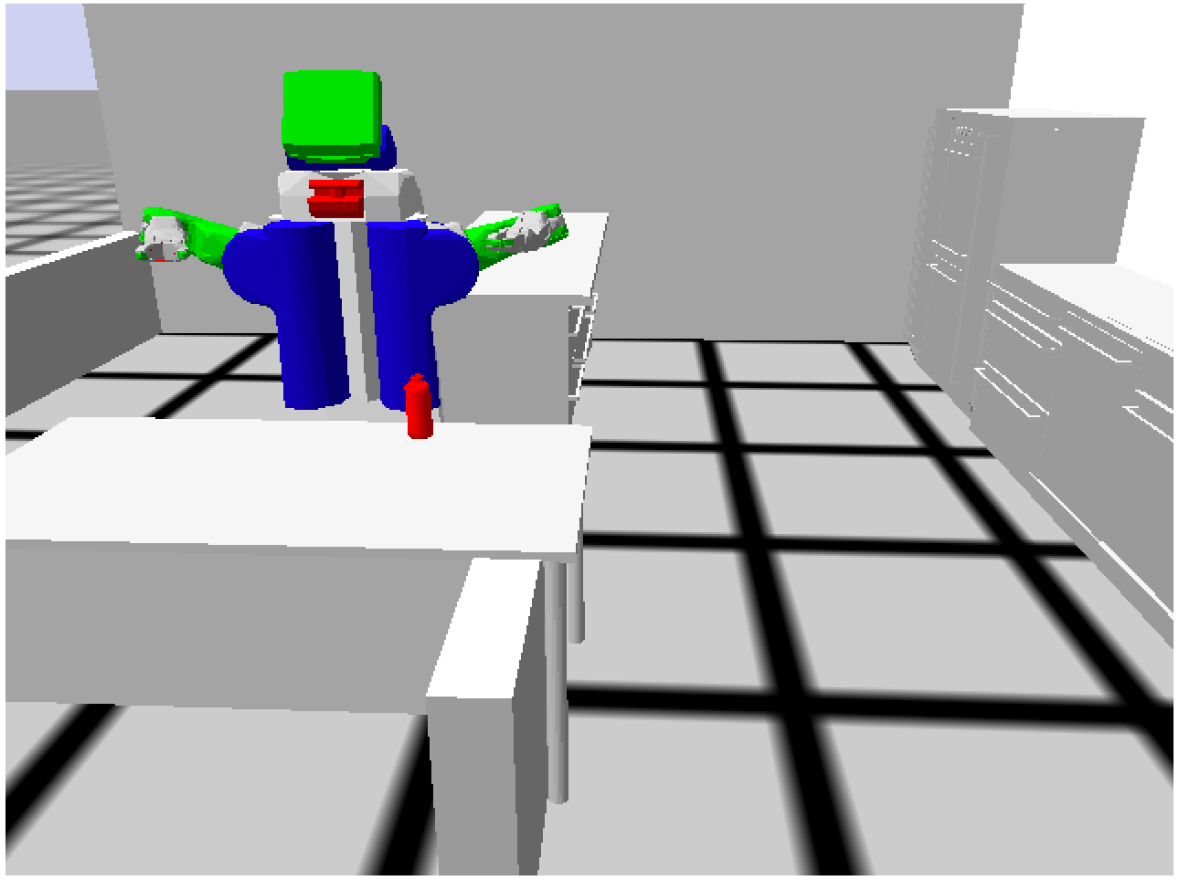
Was -1.6 and -2 in the previous two examples

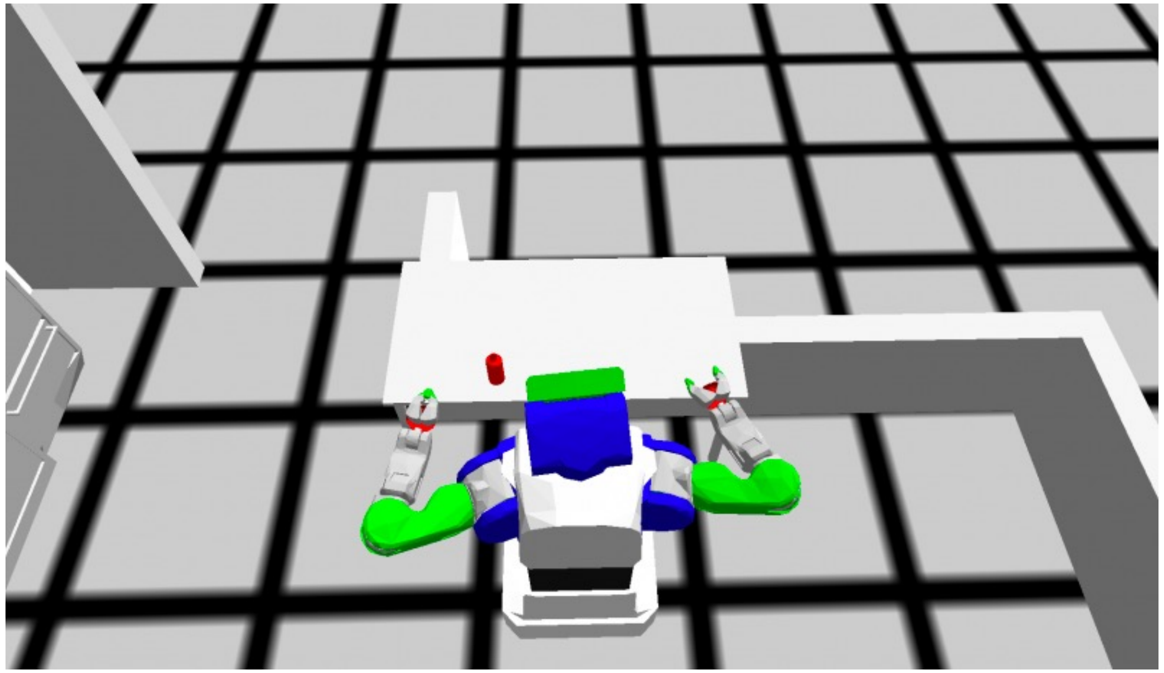
- and we run `(move-bottle)` again? We get a different error this time

Simple Fetch and Place Plan



```
PP-TUT> (move-bottle '((-1.0 -0.75 0.860) (0 0 0 1)))
[(PICK-PLACE PICK-UP) INFO] 1620312947.113: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1620312947.122: Reaching
[(PICK-PLACE MOVE-ARMS-IN-SEQUENCE) WARN] 1620312947.674: #<"torso_lift_link" ((0.37440 0.65450 -0.10812) (0.00088 0.00699 -0.30341 0.95284))> is unreachable for EE.
Ignoring.
[(PICK-PLACE MOVE-ARMS-IN-SEQUENCE) ERROR] 1620312948.159: #<"torso_lift_link" ((0.37248 0.65539 -0.25811) (0.00088 0.00699 -0.30341 0.95284))> is unreachable for EE.
Failing.
[(PP-PLANS PICK-UP) WARN] 1620312948.160: Manipulation messed up: #<"torso_lift_link" ((0.37248 0.65539 -0.25811) (0.00088 0.00699 -0.30341 0.95284))> is unreachable for EE.
Ignoring.
[(PICK-PLACE PICK-UP) INFO] 1620313021.024: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1620313021.027: Reaching
[(PICK-PLACE MOVE-ARMS-IN-SEQUENCE) ERROR] 1620313021.819: #<"torso_lift_link" ((0.48214 0.39952 -0.23006) (-0.00163 0.00716 0.09930 0.99503))> is unreachable for EE.
Failing.
[(PP-PLANS PICK-UP) WARN] 1620313021.823: Manipulation messed up: #<"torso_lift_link" ((0.48214 0.39952 -0.23006) (-0.00163 0.00716 0.09930 0.99503))> is unreachable for EE.
Ignoring.
[(PICK-PLACE PICK-UP) INFO] 1620313021.828: Gripping
[(PICK-AND-PLACE GRIP) WARN] 1620313021.867: There was no object to grip
Retrying
[(PICK-AND-PLACE GRIP) WARN] 1620313021.945: No retries left. Propagating up.
; Evaluation aborted on #<CRAM-COMMON-FAILURES:GRIPPER-CLOSED-COMPLETELY {10079517B3}>.
```

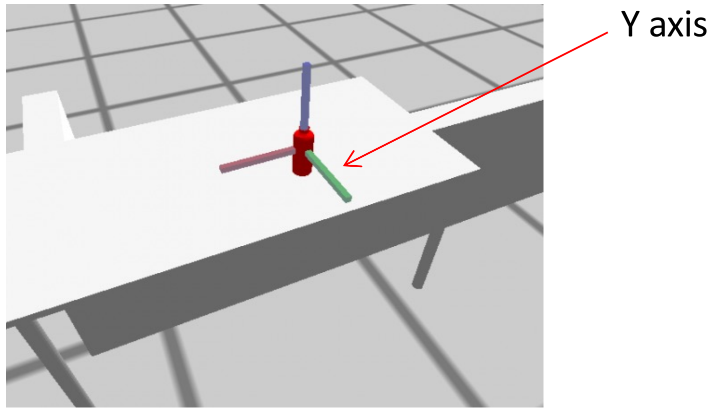




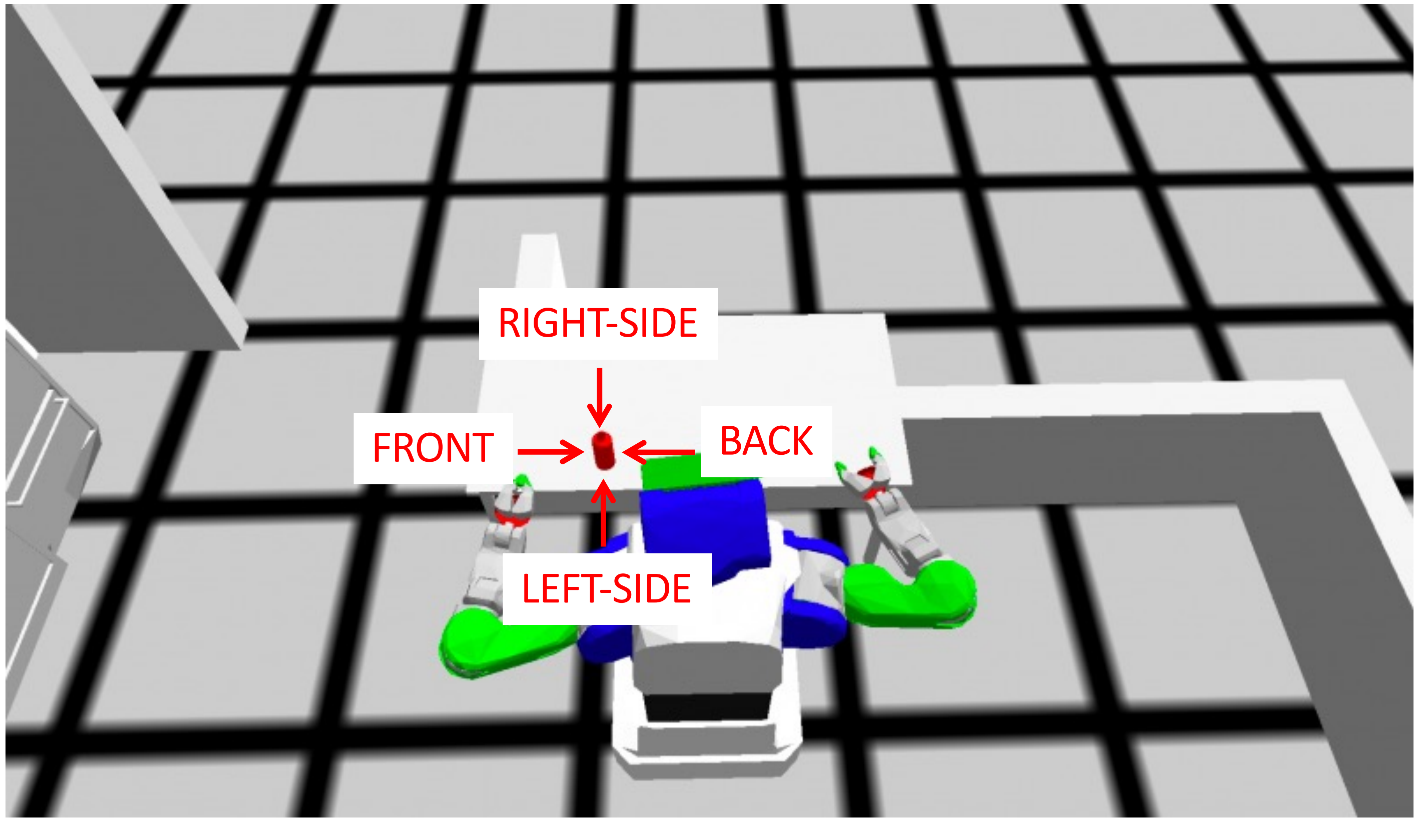
Simple Fetch and Place Plan

- So what went wrong?
- The problem we always use the right arm to grasp the object
- And it grasps from the left side, i.e. along the Y axis

```
(let ((?perceived-bottle (find-object :bottle))
      (?grasping-arm :right))
  (perform (an action
               (type picking-up)
               (arm ?grasping-arm)
               (grasp left-side)
               (object ?perceived-bottle))))
```



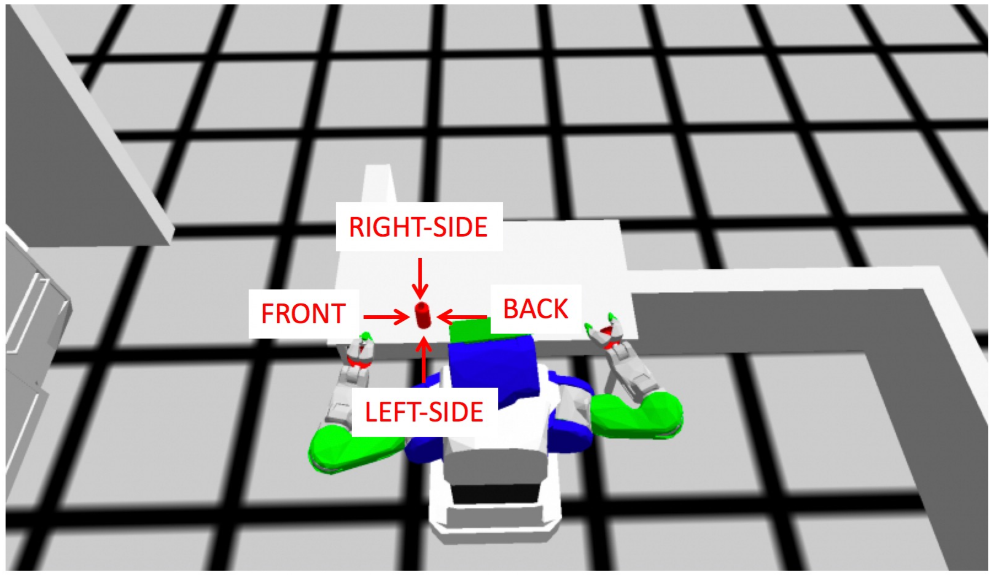
Recall: LEFT-SIDE, RIGHT-SIDE, BACK, FRONT grasp directions



Simple Fetch and Place Plan

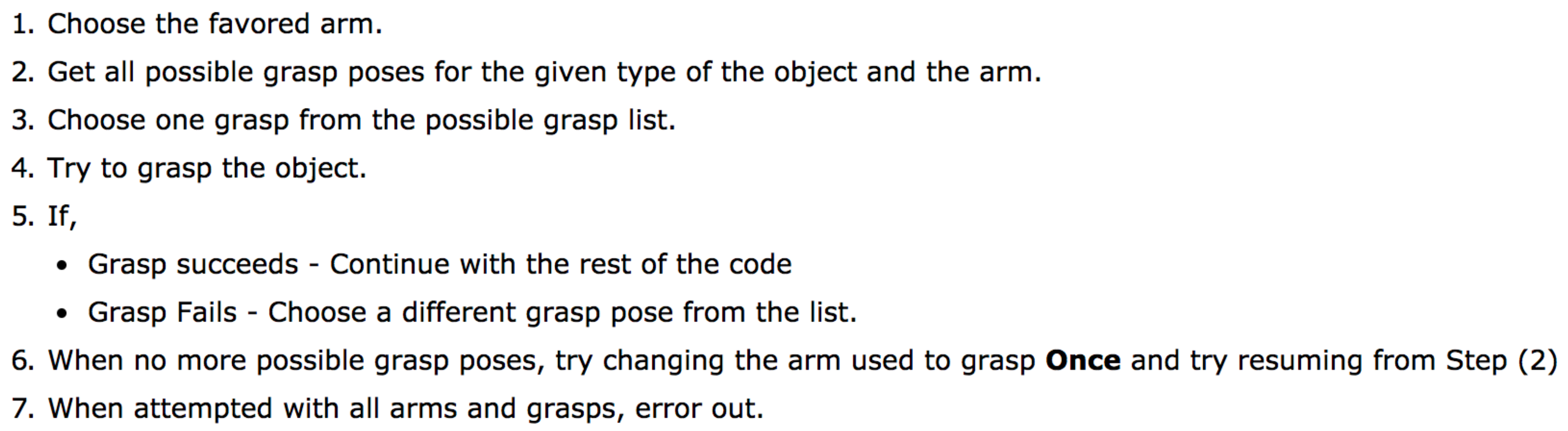
In this case, a better strategy would be either to

- Grasp with the left arm with left grasp
- Grasp with the right arm with back grasp



Simple Fetch and Place Plan

A more general strategy might be



1. Choose the favored arm.
2. Get all possible grasp poses for the given type of the object and the arm.
3. Choose one grasp from the possible grasp list.
4. Try to grasp the object.
5. If,
   - Grasp succeeds - Continue with the rest of the code
   - Grasp Fails - Choose a different grasp pose from the list.
6. When no more possible grasp poses, try changing the arm used to grasp **Once** and try resuming from Step (2)
7. When attempted with all arms and grasps, error out.

Let's implement this in a function `pick-up-object`

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Two parameters:

1. the pose of the object
2. the preferred arm to be used

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Define a list of possible grasps

Make a working copy which we will work through, removing grasps as we try them

The initial grasp is the first in the list

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                      ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Two nested failure failure handling

one for object-unreachable

one for either
manipulation-pose-unreachable
OR
gripper-closed-completely (i.e. failed to grasp)

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

We set a limit on the number of re-tries in failure handling

(there is only one other arm to try)

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

This might confuse you ... didn't we already do this above in the let form?

No: we **defined** the variables in the let and gave them initial values

Here we are (re-)assigning values to these variables

Why do we do this?
Because we need to reinitialize the values in the outer handle-failure because they might have been changed in the inner failure-handle (in fact, they are ... see below)

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))            ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                         ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Define a list of possible grasps

Make a working copy which we will work through, removing grasps as we try them

The initial grasp is the first in the list

Remove the initial grasp from the list by setting the list to everything after the initial direction

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Perform an action of type picking-up to pick up an object passed as an argument
with the arm passed as an argument
with the grasp extracted from the list of grasps

```
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Execute this code if there is a failure
in the inner handle-failure

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                        ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                            ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Print an error message indicating the arm and the grasp that failed

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                            ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Check to see if there is a grasp left in the list of grasps and proceed if the result is true

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Get the first grasp from the list of remaining grasps

Update the list of remaining grasps

```
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Print a message indicating the grasp and the arm to be tried next

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Park the arms

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Park the arms

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                        ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Retry the action

```
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

} If there were no more grasps in the list, print a message and fail

```
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Execute this if there is a failure in the outer handle-failure

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Print a message to identify the arm that failed

```
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

loop through the number of remaining retries
(for the outer handle-failure)

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Choose the other arm to the one just tried

```
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Retry the code (for the outer handle-failure)

```
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))

                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :("))))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Print an error message if this (outer) handle-error fails

```lisp
(defun pick-up-object (?perceived-object ?grasping-arm)
  (let* ((?possible-grasps '(:left-side :right-side :front :back))  ;define all possible grasps
         (?remaining-grasps (copy-list ?possible-grasps))           ;make a copy to work though when trying each grasp
         (?grasp (first ?remaining-grasps)))                        ;this is the first one to try

      (cpl:with-retry-counters ((arm-change-retry 1))               ;there is one alternative arm if the first one fails

         ;; Outer handle-failure handling arm change
         (handle-failure object-unreachable
            ((setf ?remaining-grasps (copy-list ?possible-grasps))  ;make sure to try all possible grasps for each arm
             (setf ?grasp (first ?remaining-grasps))                ;get the first grasp to try
             (setf ?remaining-grasps (rest ?remaining-grasps))      ;update the remaining grasps to try

            ;; Inner handle-failure handling grasp change
            (handle-failure (or manipulation-pose-unreachable gripper-closed-completely)
               ;; Try to perform the pick up
               ((perform (an action
                             (type picking-up)
                             (arm ?grasping-arm)
                             (grasp ?grasp)
                             (object ?perceived-object))))

               ;; When pick-up fails this block gets executed
               (format t "~%Grasping failed with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
               ;;(format t "~%Error: ~a~%" e)                       ;uncomment to see the error message

               ;; Check if we have any remaining grasps left.
               ;; If yes, then the block nested to it gets executed, which will
               ;; set the grasp that is used to the new value and trigger retry

               (when (first ?remaining-grasps)                      ;if there is a grasp remaining
                  (setf ?grasp (first ?remaining-grasps))           ;get it
                  (setf ?remaining-grasps (rest ?remaining-grasps)) ;update the remaining grasps to try
                  (format t "Retrying with ~a arm and ~a grasp~%" ?grasping-arm ?grasp)
                  (park-arms)
                  (cpl:retry))
                ;; This will get executed when there are no more elements in the
                ;; ?possible-grasps list. We print the error message and throw a new error
                ;; which will be caught by the outer handle-failure
                (print  "No more grasp retries left :(")
                (cpl:fail 'object-unreachable)))

             ;; This is the failure management of the outer handle-failure call
             ;; It changes the arm that is used to grasp
             (format t "Manipulation failed with the ~a arm"?grasping-arm)
             ;; (print e)                                           ;uncomment if you want to see the error
             ;; Here we use the retry counter we defined. The value is decremented automatically
             (cpl:do-retry arm-change-retry
                ;; if the current grasping arm is right set left, else set right
                (setf ?grasping-arm (if (eq ?grasping-arm :right)
                                        :left
                                        :right))
                (park-arms)
                (cpl:retry))
             ;; When all retries are exhausted print the error message.
             (print "No more arm change retries left :(")))
  ?grasping-arm) ; function value is the arm that was used to grasp the object
```

Return the arm that was actually used!

Simple Fetch and Place Plan

Now, we use this function in a revised `move-bottle`

It is shorter than the previous versions
because we replaced the perform pick action with a call to `pick-up-object`

```
(defun move-bottle (bottle-spawn-pose)
  (spawn-object bottle-spawn-pose)
  (with-simulated-robot
    (let ((?navigation-goal *base-pose-near-table*))
      (cpl:par
        ;; Moving the robot near the table.
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?navigation-goal)))))
        (perform (a motion
                    (type moving-torso)
                    (joint-angle 0.3)))
        (park-arms)))

    (let ((?perceived-bottle (find-object :bottle))
          (?grasping-arm :right))
      ;; We update the value of ?grasping-arm according to what the method used
      (setf ?grasping-arm (pick-up-object ?perceived-bottle ?grasping-arm))
      (park-arm ?grasping-arm)
      ;; Moving the robot near the counter.
      (let ((?nav-goal *base-pose-near-counter*))
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?nav-goal))))))
       ;; Setting the object down on the counter
      (let ((?drop-pose *final-object-destination*))
        (perform (an action
                     (type placing)
                     (arm ?grasping-arm)
                     (object ?perceived-bottle)
                     (target (a location
                                (pose ?drop-pose))))))
      (park-arm ?grasping-arm))))
```

Same as previous version

Same as previous version

```
(defun move-bottle (bottle-spawn-pose)
  (spawn-object bottle-spawn-pose)
  (with-simulated-robot
    (let ((?navigation-goal *base-pose-near-table*))
      (cpl:par
        ;; Moving the robot near the table.
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?navigation-goal)))))
        (perform (a motion
                    (type moving-torso)
                    (joint-angle 0.3)))
        (park-arms)))

    (let ((?perceived-bottle (find-object :bottle))
          (?grasping-arm :right))
      ;; We update the value of ?grasping-arm according to what the method used
      (setf ?grasping-arm (pick-up-object ?perceived-bottle ?grasping-arm))
      (park-arm ?grasping-arm)
      ;; Moving the robot near the counter.
      (let ((?nav-goal *base-pose-near-counter*))
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?nav-goal))))))
       ;; Setting the object down on the counter
      (let ((?drop-pose *final-object-destination*))
        (perform (an action
                     (type placing)
                     (arm ?grasping-arm)
                     (object ?perceived-bottle)
                     (target (a location
                                (pose ?drop-pose))))))
      (park-arm ?grasping-arm))))
```

Call the function find-object to find the bottle and store the result

```
(defun move-bottle (bottle-spawn-pose)
  (spawn-object bottle-spawn-pose)
  (with-simulated-robot
    (let ((?navigation-goal *base-pose-near-table*))
      (cpl:par
        ;; Moving the robot near the table.
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?navigation-goal)))))
        (perform (a motion
                    (type moving-torso)
                    (joint-angle 0.3)))
        (park-arms)))

    (let ((?perceived-bottle (find-object :bottle))
          (?grasping-arm :right))
      ;; We update the value of ?grasping-arm according to what the method used
      (setf ?grasping-arm (pick-up-object ?perceived-bottle ?grasping-arm))
      (park-arm ?grasping-arm)
      ;; Moving the robot near the counter.
      (let ((?nav-goal *base-pose-near-counter*))
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?nav-goal))))))
       ;; Setting the object down on the counter
      (let ((?drop-pose *final-object-destination*))
        (perform (an action
                     (type placing)
                     (arm ?grasping-arm)
                     (object ?perceived-bottle)
                     (target (a location
                                (pose ?drop-pose))))))
      (park-arm ?grasping-arm))))
```

Set the (initial) grasping arm to be the right arm

```
(defun move-bottle (bottle-spawn-pose)
  (spawn-object bottle-spawn-pose)
  (with-simulated-robot
    (let ((?navigation-goal *base-pose-near-table*))
      (cpl:par
        ;; Moving the robot near the table.
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?navigation-goal)))))
        (perform (a motion
                    (type moving-torso)
                    (joint-angle 0.3)))
        (park-arms)))

    (let ((?perceived-bottle (find-object :bottle))
          (?grasping-arm :right))
      ;; We update the value of ?grasping-arm according to what the method used
      (setf ?grasping-arm (pick-up-object ?perceived-bottle ?grasping-arm))
      (park-arm ?grasping-arm)
      ;; Moving the robot near the counter.
      (let ((?nav-goal *base-pose-near-counter*))
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?nav-goal))))))
       ;; Setting the object down on the counter
      (let ((?drop-pose *final-object-destination*))
        (perform (an action
                     (type placing)
                     (arm ?grasping-arm)
                     (object ?perceived-bottle)
                     (target (a location
                                (pose ?drop-pose))))))
      (park-arm ?grasping-arm))))
```

Set the (possibly revised) grasping arm
to be the one that the pick-up-object returns

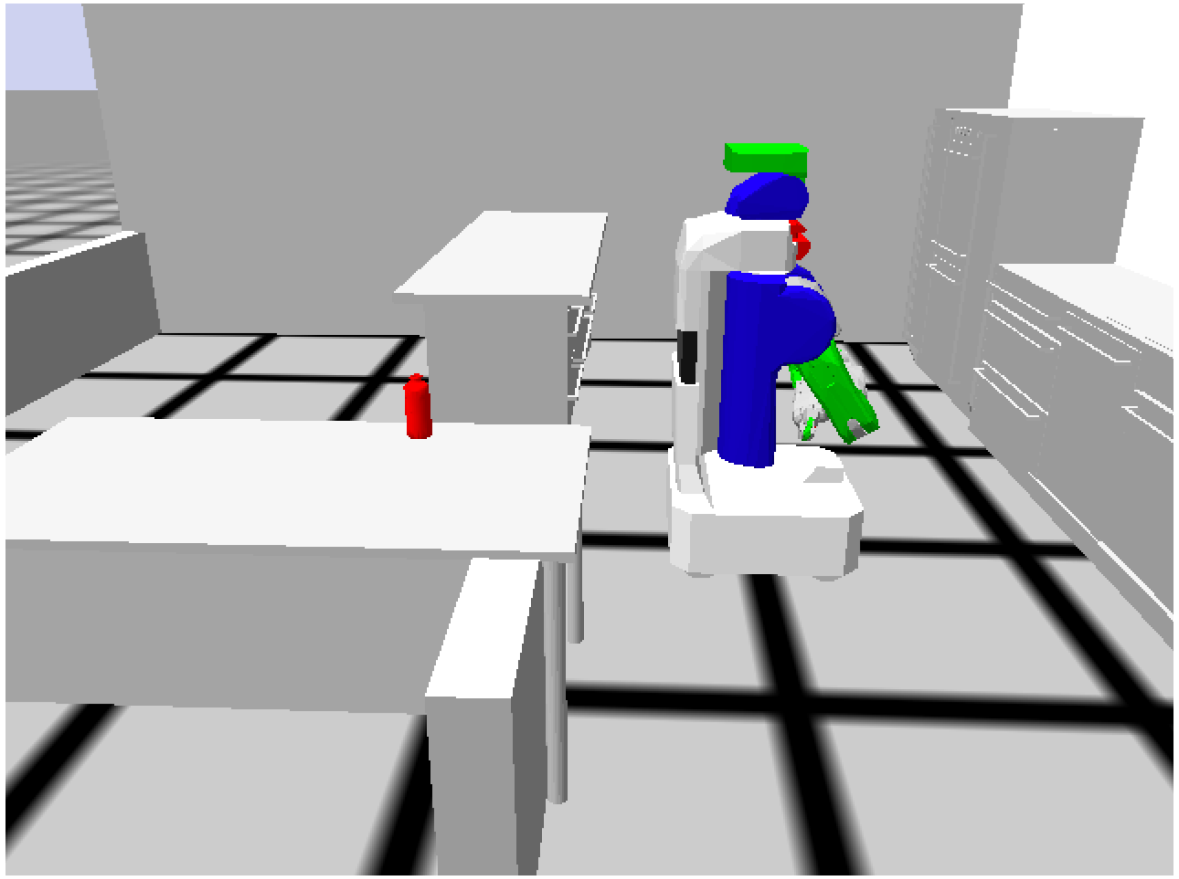
```
(defun move-bottle (bottle-spawn-pose)
  (spawn-object bottle-spawn-pose)
  (with-simulated-robot
    (let ((?navigation-goal *base-pose-near-table*))
      (cpl:par
        ;; Moving the robot near the table.
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?navigation-goal)))))
        (perform (a motion
                    (type moving-torso)
                    (joint-angle 0.3)))
        (park-arms)))

    (let ((?perceived-bottle (find-object :bottle))
          (?grasping-arm :right))
      ;; We update the value of ?grasping-arm according to what the method used
      (setf ?grasping-arm (pick-up-object ?perceived-bottle ?grasping-arm))
      (park-arm ?grasping-arm)
      ;; Moving the robot near the counter.
      (let ((?nav-goal *base-pose-near-counter*))
        (perform (an action
                     (type going)
                     (target (a location
                                (pose ?nav-goal))))))
       ;; Setting the object down on the counter
      (let ((?drop-pose *final-object-destination*))
        (perform (an action
                     (type placing)
                     (arm ?grasping-arm)
                     (object ?perceived-bottle)
                     (target (a location
                                (pose ?drop-pose))))))
      (park-arm ?grasping-arm))))
```

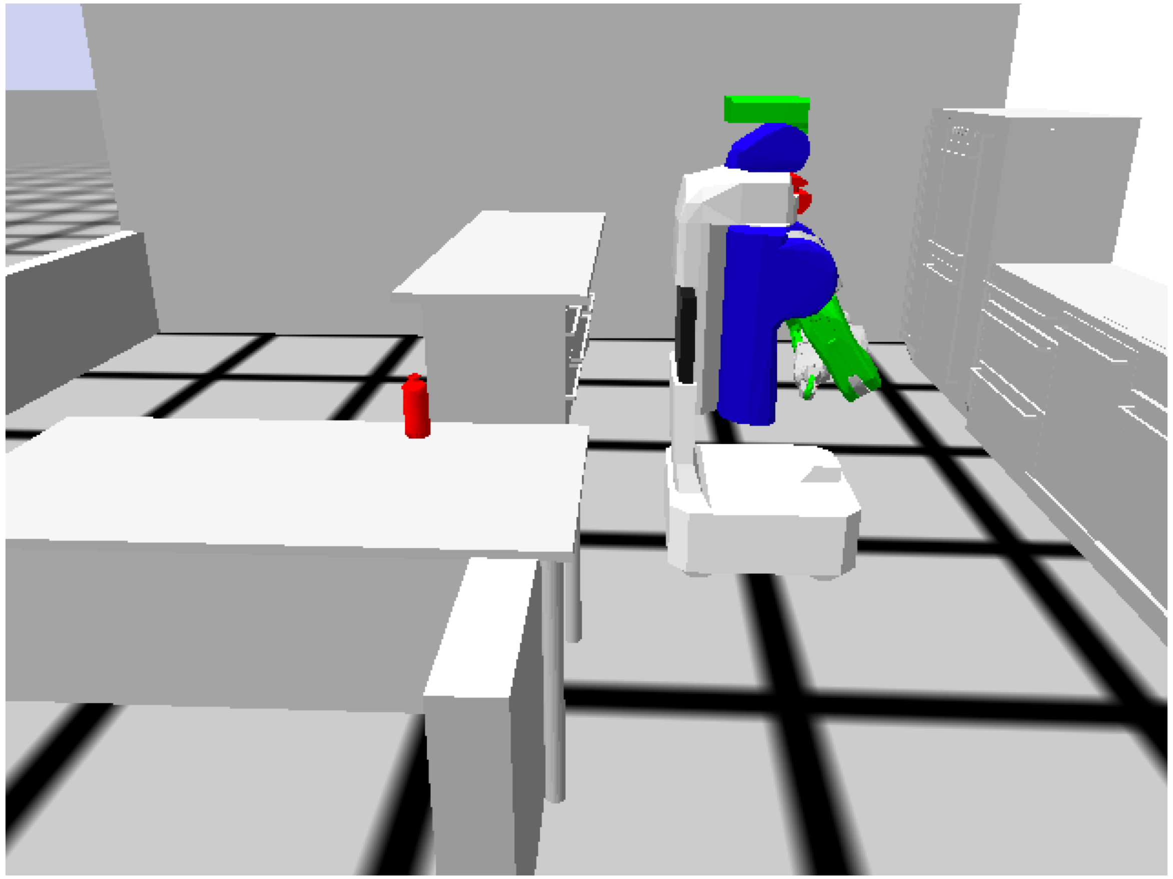
Park the grasping arm

```
PP-TUT> (move-bottle '((-1.0 -0.75 0.860) (0 0 0 1)))
…
[(PICK-PLACE PICK-UP) INFO] 1621250741.518: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621250741.522: Reaching
…
Grasping failed with RIGHT arm and LEFT-SIDE grasp
Retrying with RIGHT arm and RIGHT-SIDE grasp
[(PICK-PLACE PICK-UP) INFO] 1621250742.709: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621250742.709: Reaching
…
Grasping failed with RIGHT arm and RIGHT-SIDE grasp
Retrying with RIGHT arm and FRONT grasp
[(PICK-PLACE PICK-UP) INFO] 1621250744.157: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621250744.159: Reaching
…
Grasping failed with RIGHT arm and FRONT grasp
Retrying with RIGHT arm and BACK grasp
[(PICK-PLACE PICK-UP) INFO] 1621250745.425: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621250745.430: Reaching
[(PICK-PLACE PICK-UP) INFO] 1621250746.218: Gripping
[(PICK-PLACE PICK-UP) INFO] 1621250746.273: Assert grasp into knowledge base
[(PICK-PLACE PICK-UP) INFO] 1621250746.277: Lifting
[(PICK-PLACE PLACE) INFO] 1621250746.751: Reaching
[(PICK-PLACE PLACE) INFO] 1621250746.961: Putting
[(PICK-PLACE PLACE) INFO] 1621250747.064: Opening gripper
[(PICK-PLACE PLACE) INFO] 1621250747.142: Retract grasp in knowledge base
[(PICK-PLACE PLACE) INFO] 1621250747.185: Retracting
```

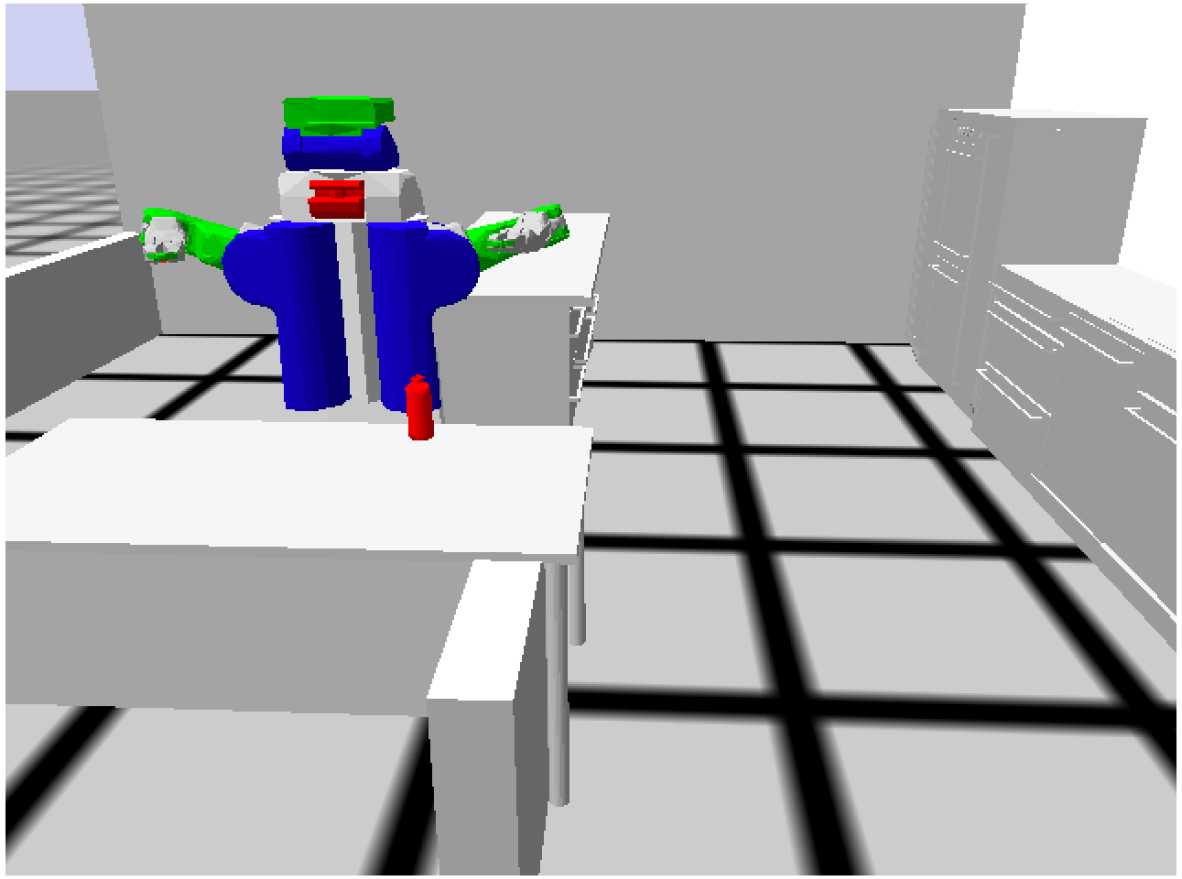
Spawn the bottle



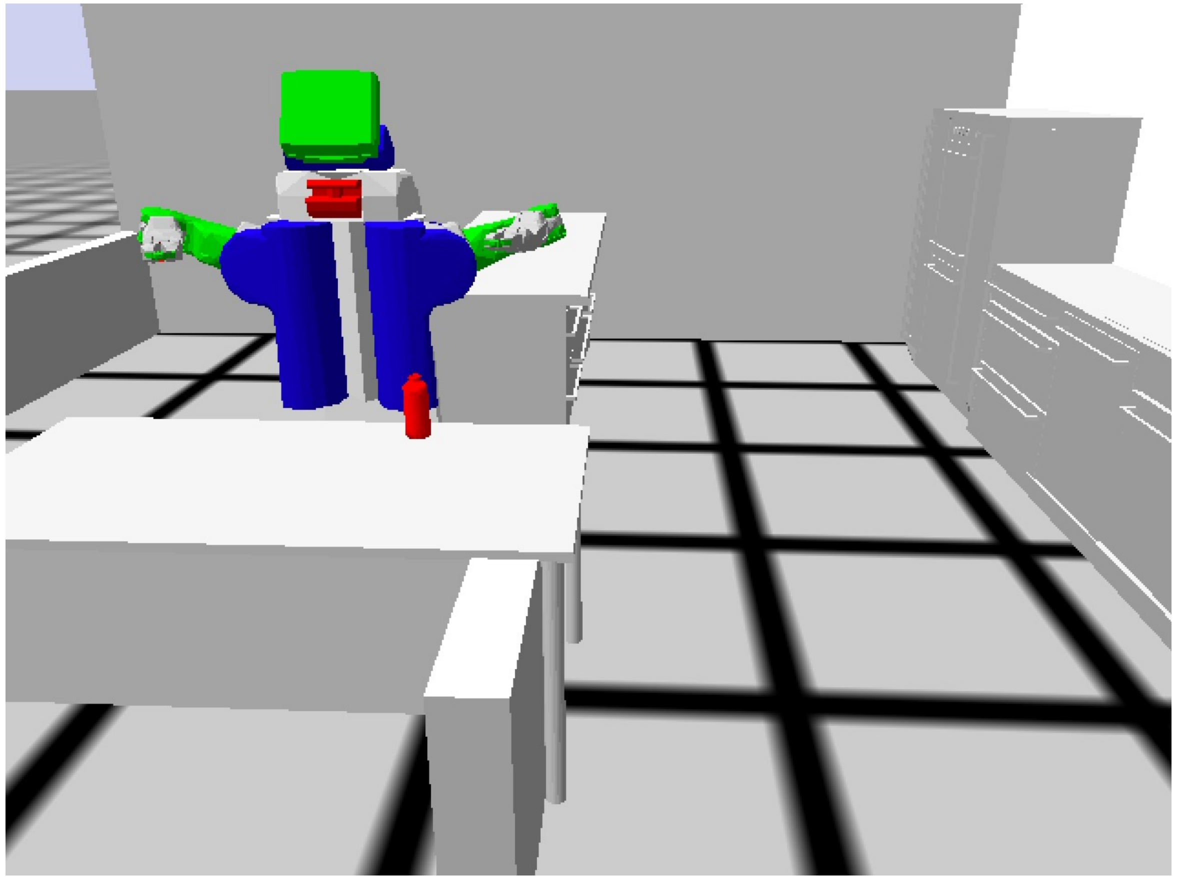
Perform motion
adjust torso



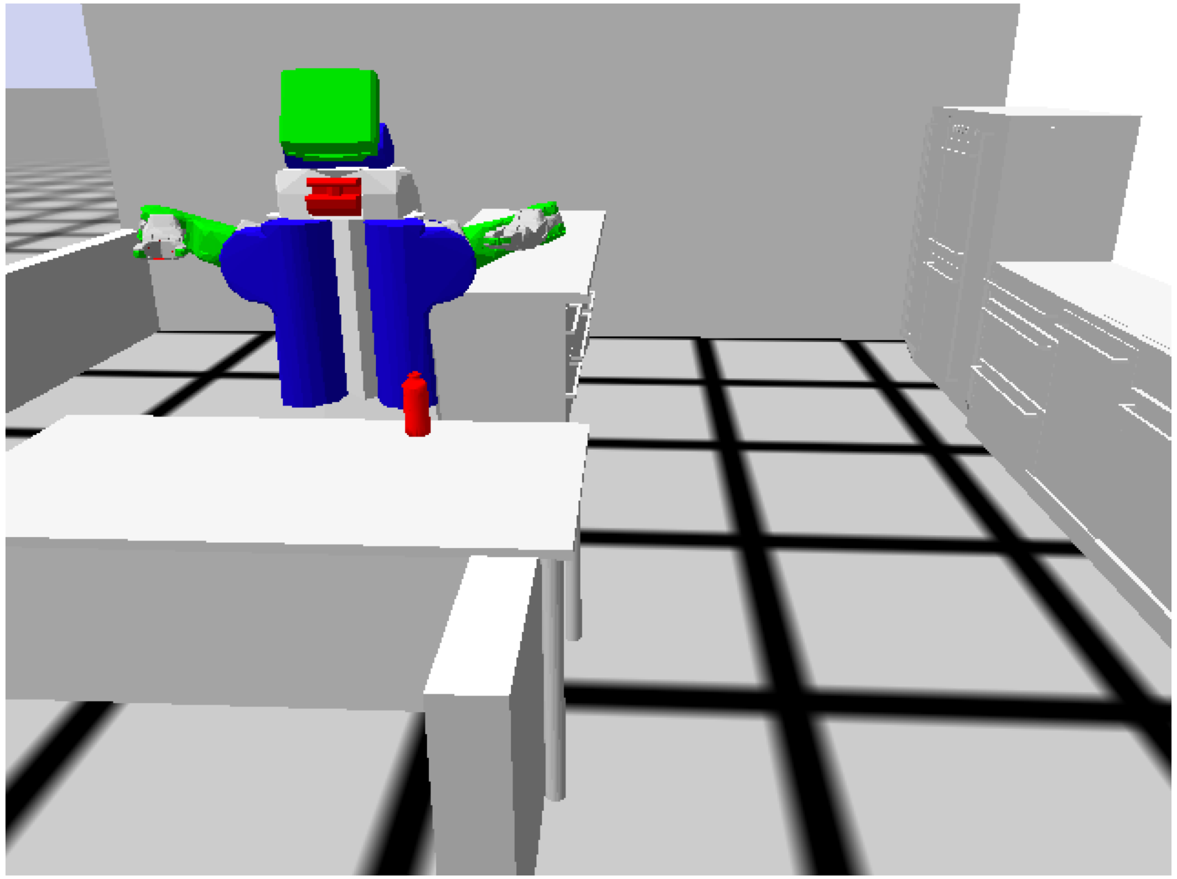
Perform action
move near the table



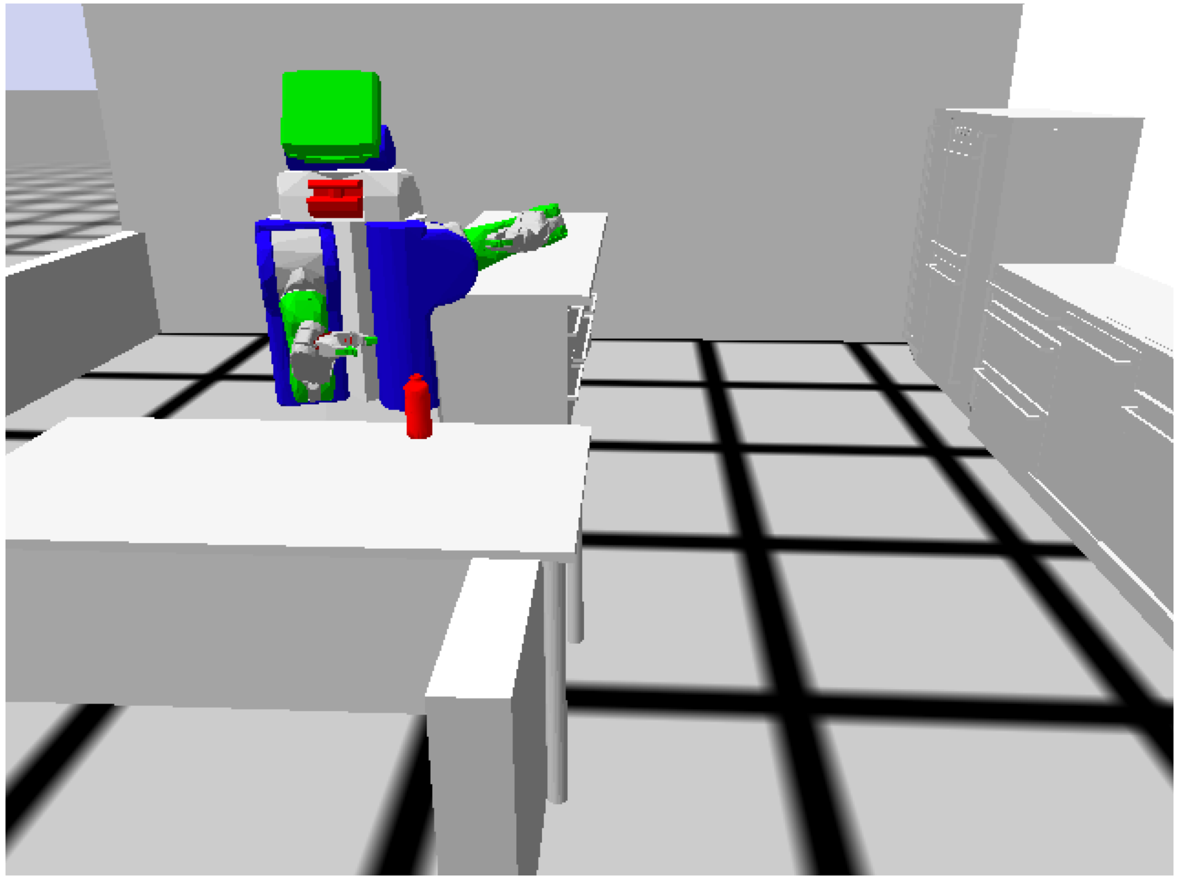
Perform action
look downwards



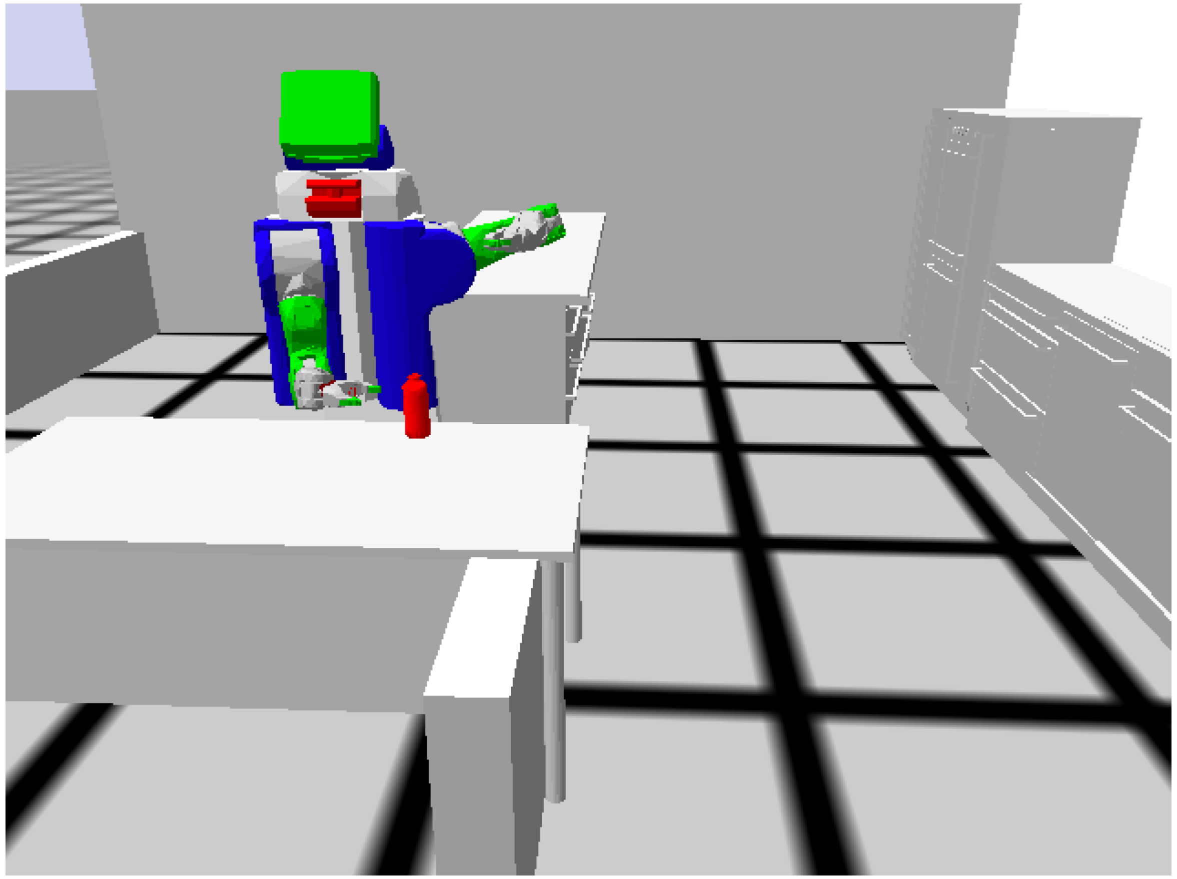
Perform action
pick-up: opening gripper



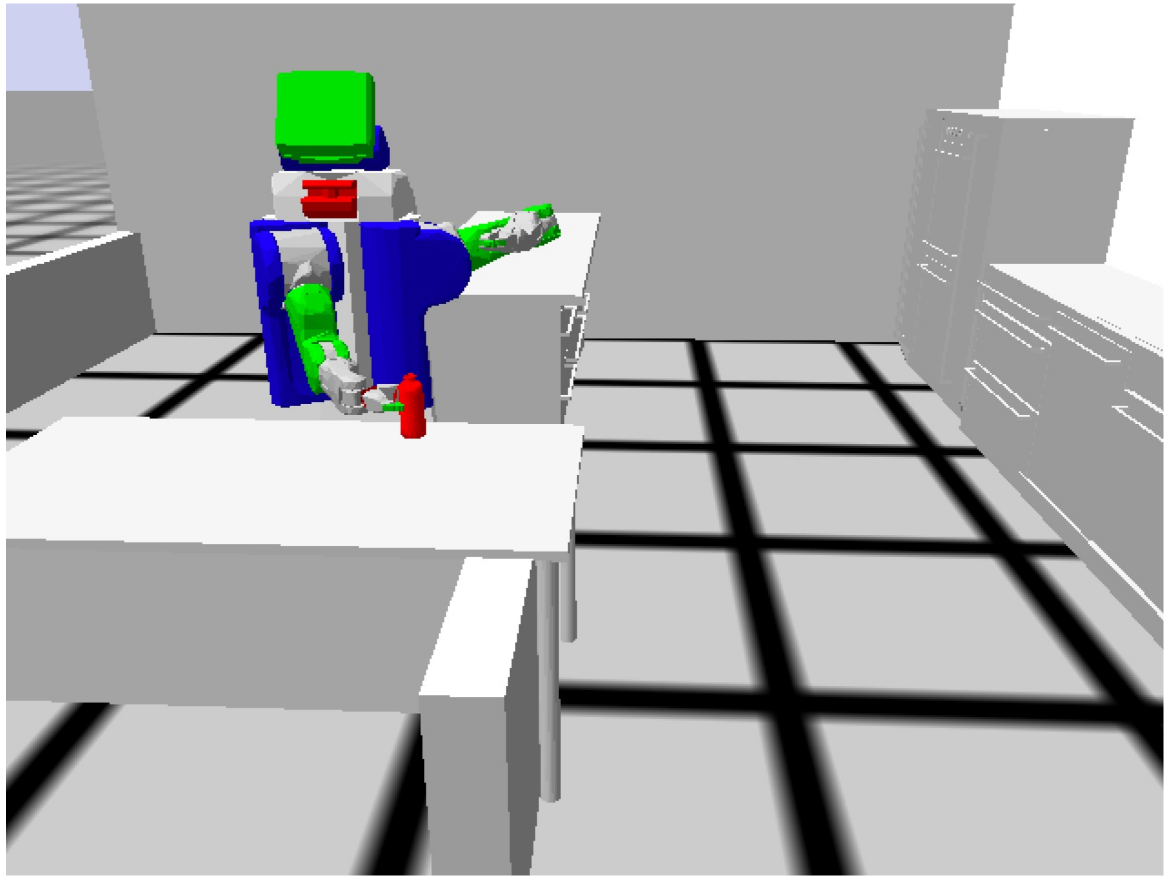
Perform action pick-up: reaching



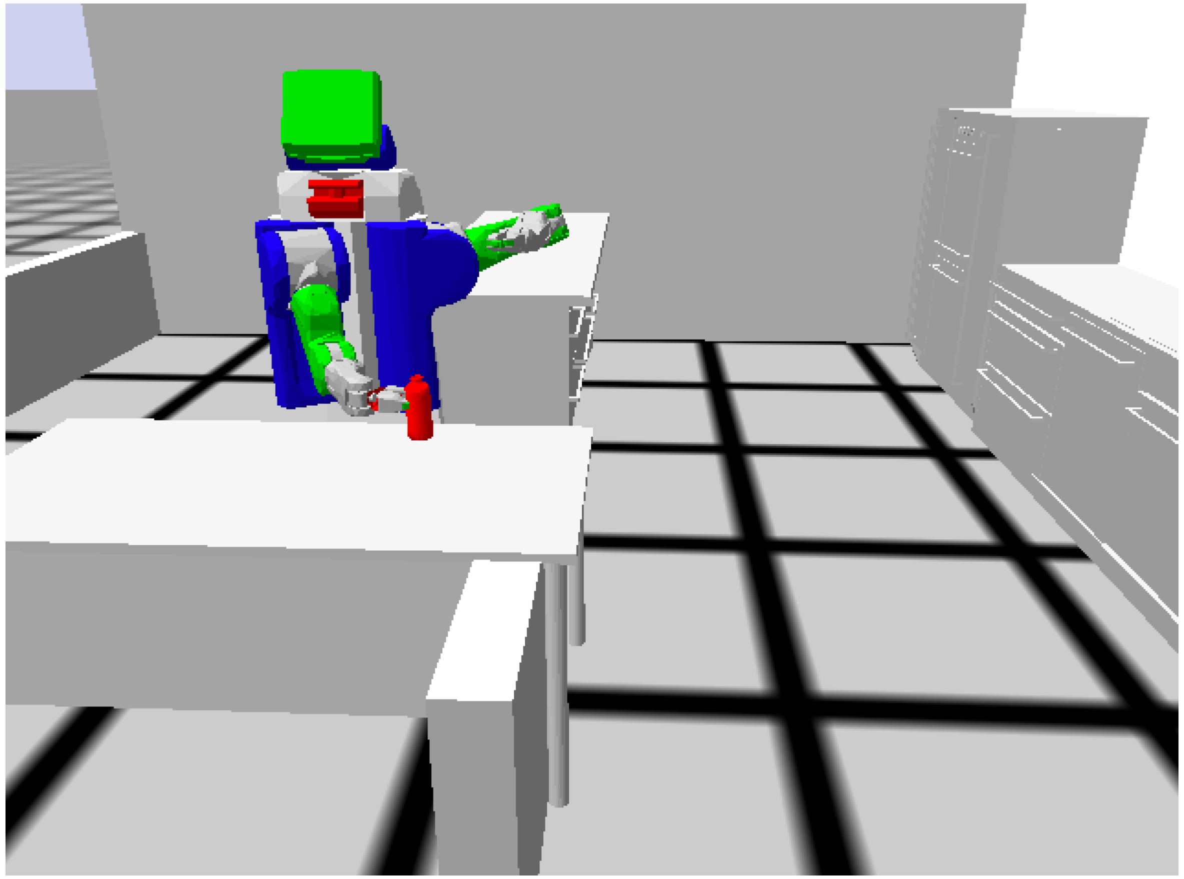
Perform action
pick-up: reaching



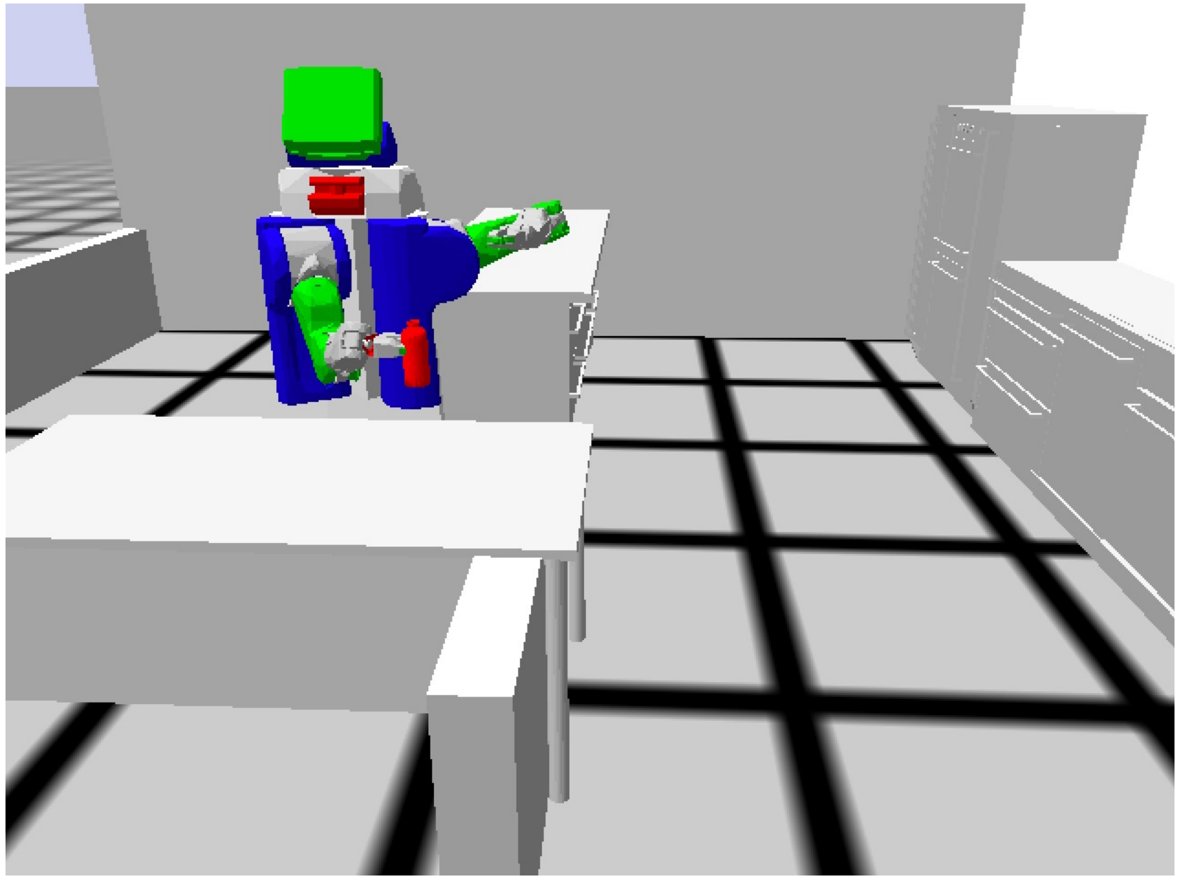
Perform action
pick-up: reaching



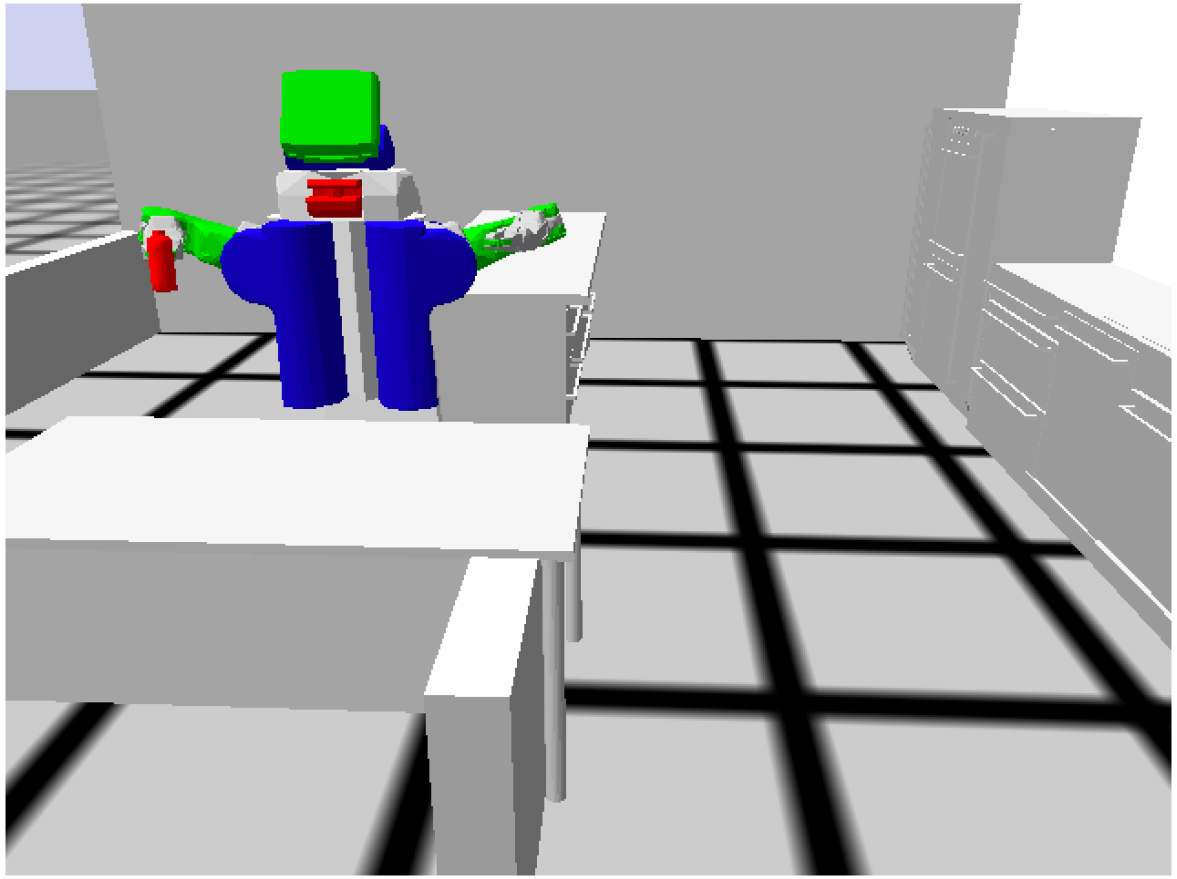
Perform action
pick-up: grasping



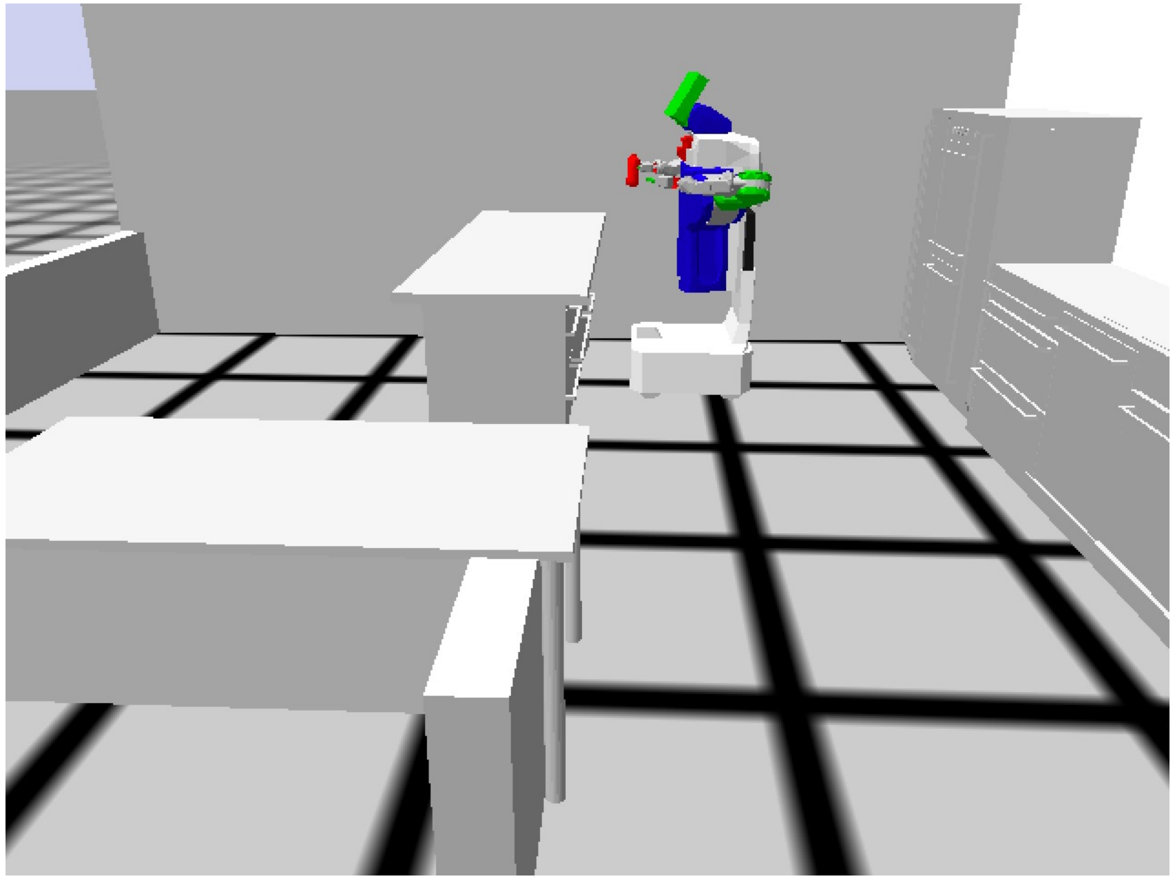
Perform action
pick-up: lifting



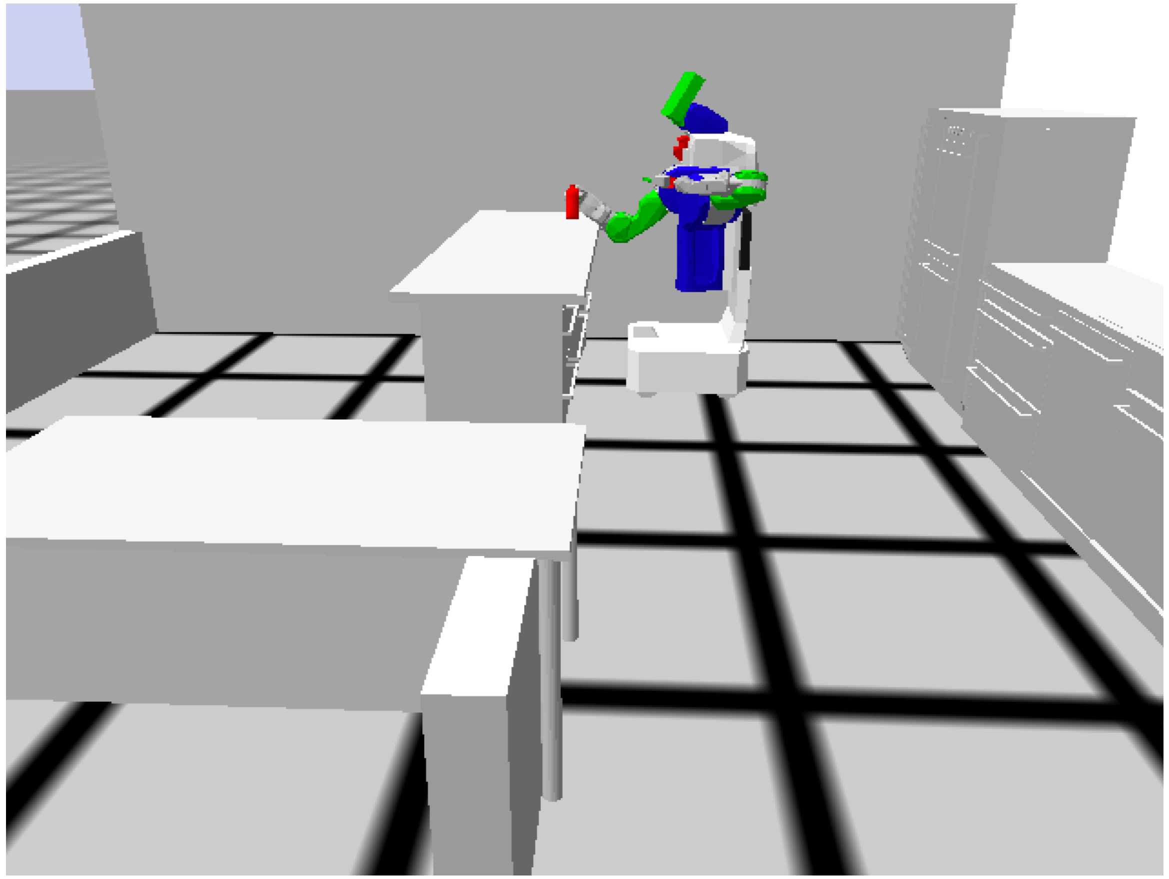
Perform action
pick-up: lifting



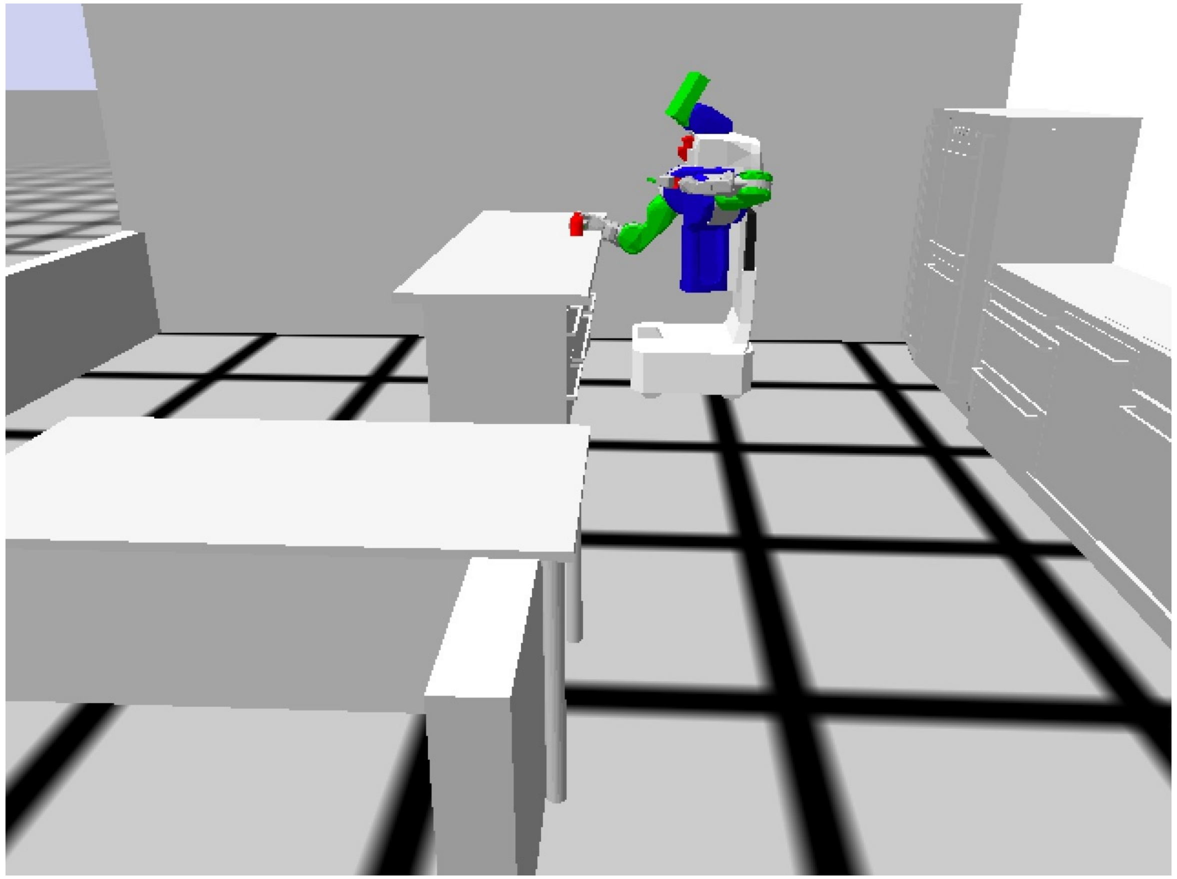
Perform action
move near the counter



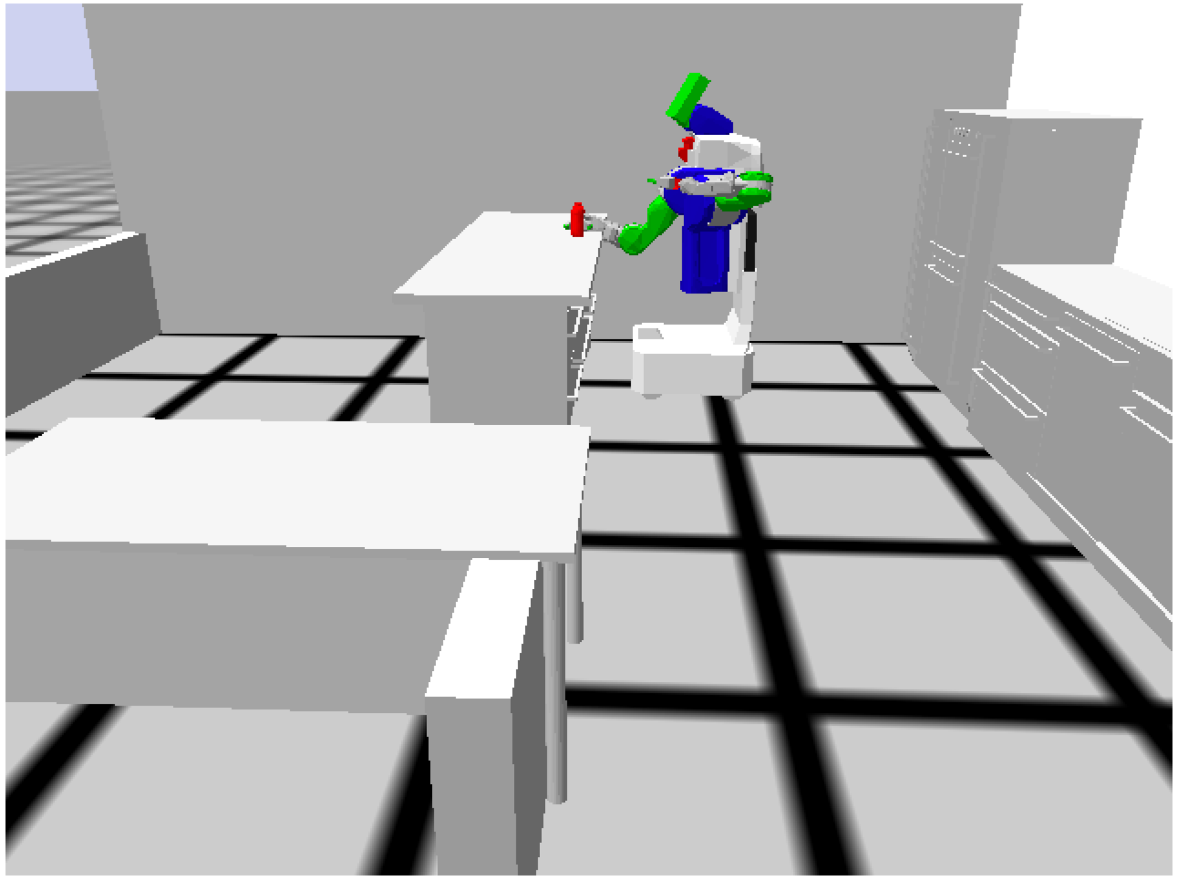
Perform action place: reaching



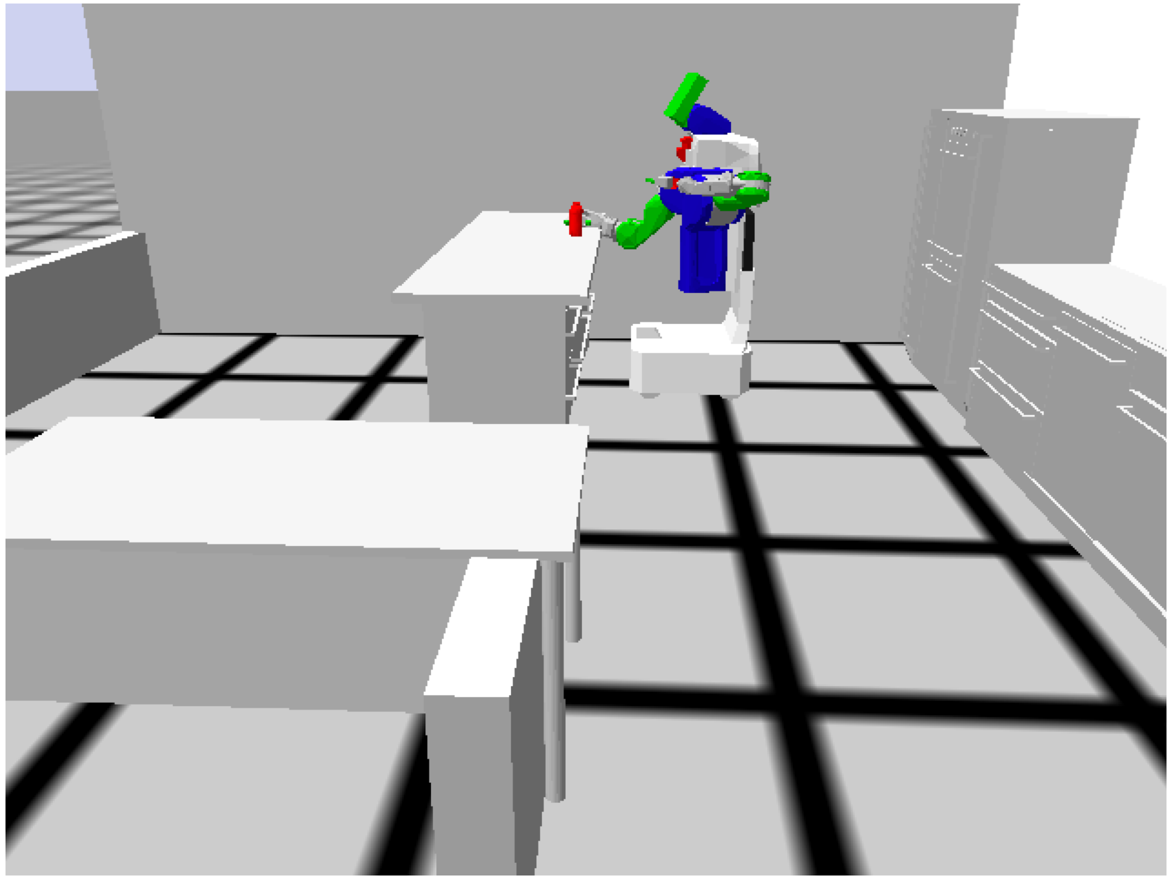
Perform action place: putting



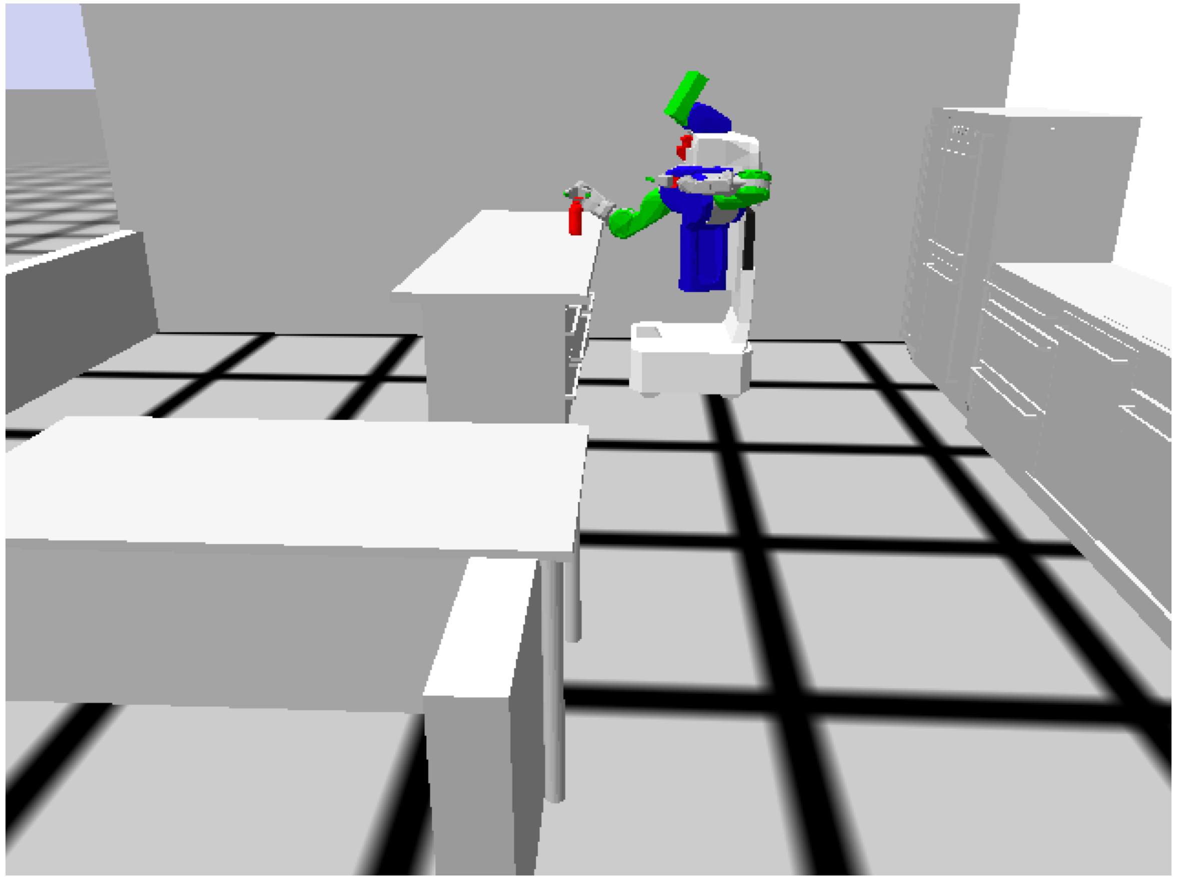
Perform action
place: opening gripper



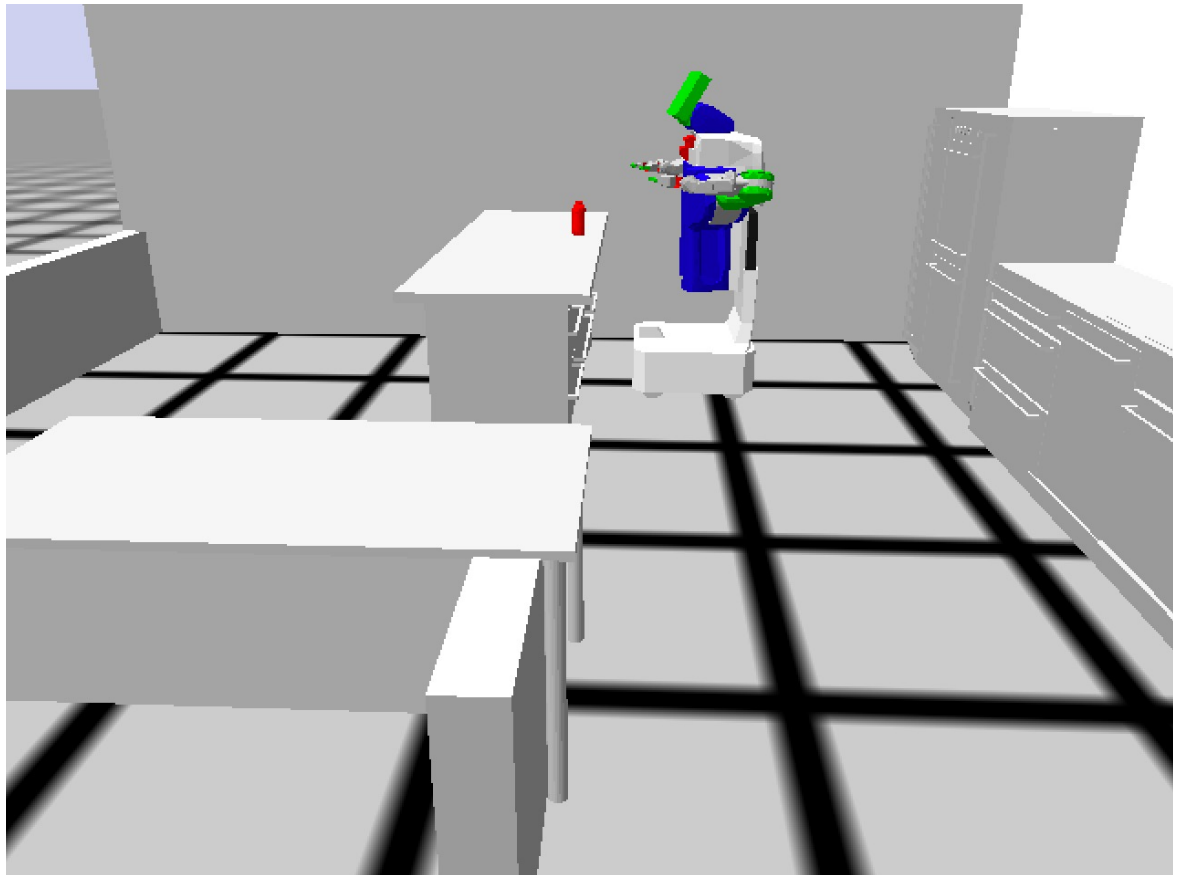
Perform action place: retracting



Perform action place: retracting



Perform action
place: retracting



Simple Fetch and Place Plan

With `(move-bottle '((-1.0 -0.75 0.860) (0 0 0 1)))`

the robot picks up the bottle with the right arm because it's still reachable.

What happens if we try with the bottle in a position that can't be reached with the right arm?

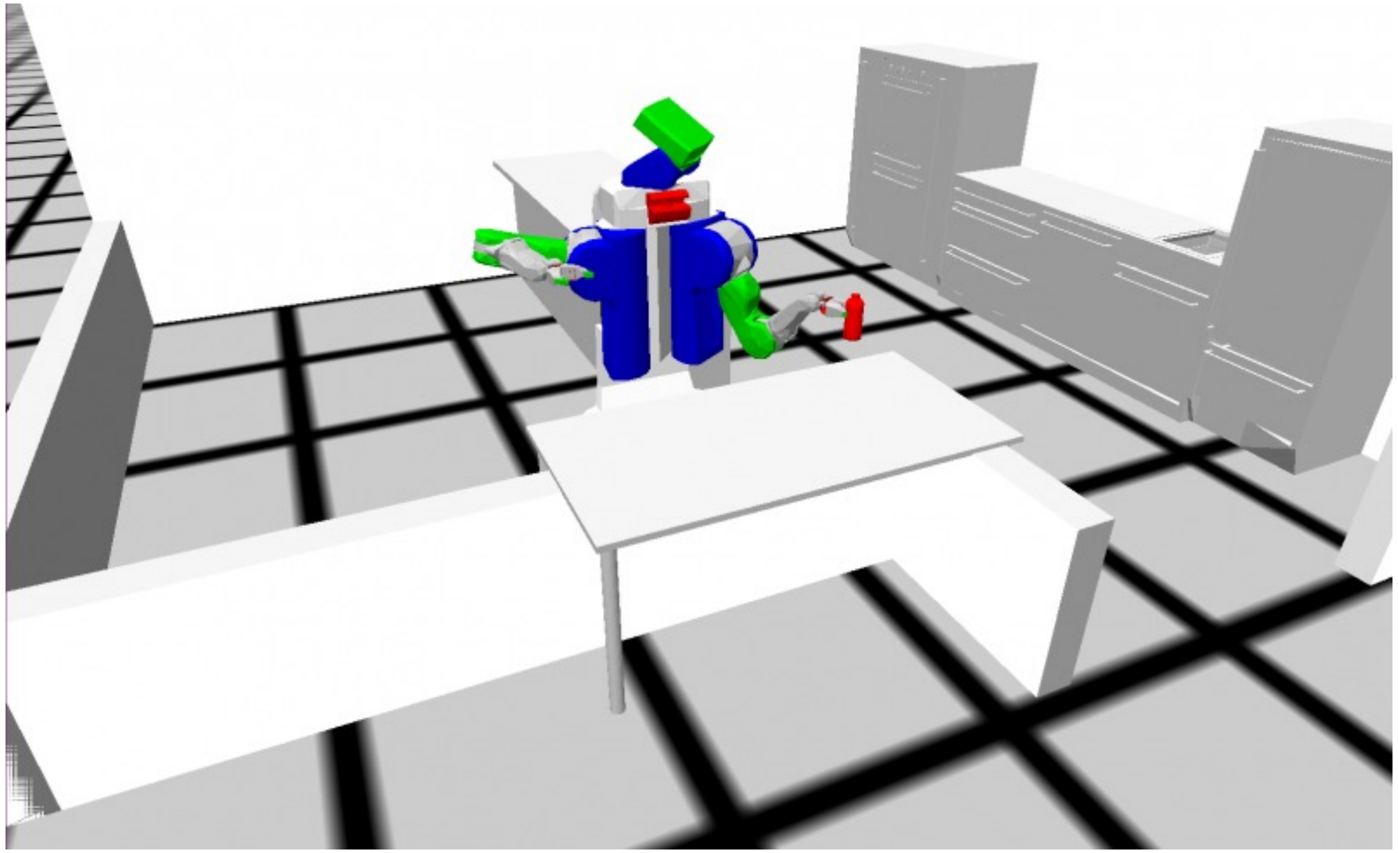
```
(move-bottle '((-0.9 -0.75 0.860) (0 0 0 1)))
```

Simple Fetch and Place Plan

```
PP-TUT> (move-bottle '((-0.9 -0.75 0.860) (0 0 0 1)))
…
[(PICK-PLACE PICK-UP) INFO] 1621251445.989: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621251445.993: Reaching
…
Grasping failed with RIGHT arm and LEFT-SIDE grasp
Retrying with RIGHT arm and RIGHT-SIDE grasp
[(PICK-PLACE PICK-UP) INFO] 1621251447.290: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621251447.291: Reaching
…
Grasping failed with RIGHT arm and RIGHT-SIDE grasp
Retrying with RIGHT arm and FRONT grasp
[(PICK-PLACE PICK-UP) INFO] 1621251448.443: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621251448.454: Reaching
…
Grasping failed with RIGHT arm and FRONT grasp
Retrying with RIGHT arm and BACK grasp
[(PICK-PLACE PICK-UP) INFO] 1621251449.628: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621251449.650: Reaching
…
[(PICK-PLACE PICK-UP) INFO] 1621251450.450: Gripping
[(PICK-AND-PLACE GRIP) WARN] 1621251450.480: There was no object to grip
Retrying
[(PICK-AND-PLACE GRIP) WARN] 1621251450.491: No retries left. Propagating up.
#<CRAM-COMMON-FAILURES:GRIPPER-CLOSED-COMPLETELY {100CD890B3}>
Grasping failed with RIGHT arm and BACK grasp
"No more grasp retries left :("
#<CRAM-COMMON-FAILURES:OBJECT-UNREACHABLE {100B7A2583}> Manipulation failed with the RIGHT arm
[(PICK-PLACE PICK-UP) INFO] 1621251450.606: Opening gripper
[(PICK-PLACE PICK-UP) INFO] 1621251450.608: Reaching
[(PICK-PLACE PICK-UP) INFO] 1621251451.151: Gripping
[(PICK-PLACE PICK-UP) INFO] 1621251451.234: Assert grasp into knowledge base
[(PICK-PLACE PICK-UP) INFO] 1621251451.235: Lifting
[(PICK-PLACE PLACE) INFO] 1621251451.709: Reaching
[(PICK-PLACE PLACE) INFO] 1621251451.892: Putting
[(PICK-PLACE PLACE) INFO] 1621251451.988: Opening gripper
[(PICK-PLACE PLACE) INFO] 1621251452.027: Retract grasp in knowledge base
[(PICK-PLACE PLACE) INFO] 1621251452.074: Retracting
```

Fails with all the grasps with the RIGHT arm

Succeeds with the first (LEFT-SIDE) grasp with the LEFT arm



Recommended Reading

CRAM zero prerequisites demo tutorial: simple fetch and place

`http://cram-system.org/tutorials/demo/fetch_and_place`

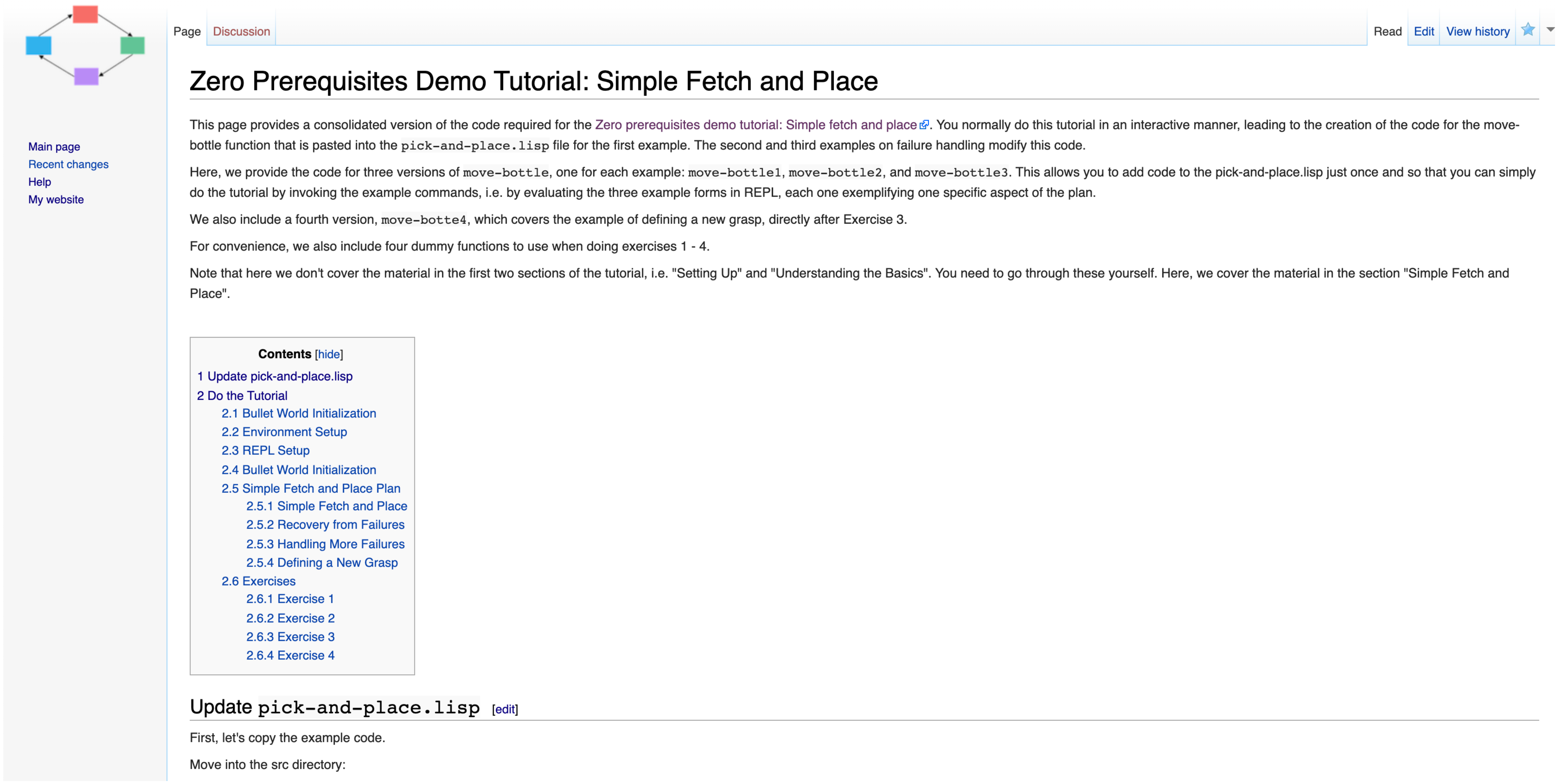
Implementation of a pick-and-place CRAM plan

Follow these instructions

“Zero Prerequisites Demo Tutorial: Simple Fetch and Place”

http://www.vernon.eu/wiki/Zero_Prerequisites_Demo_Tutorial:_Simple_Fetch_and_Place

to implement the pick-and-place example



Page Discussion Read Edit View history

Main page
Recent changes
Help
My website

Zero Prerequisites Demo Tutorial: Simple Fetch and Place

This page provides a consolidated version of the code required for the Zero prerequisites demo tutorial: Simple fetch and place. You normally do this tutorial in an interactive manner, leading to the creation of the code for the move-bottle function that is pasted into the `pick-and-place.lisp` file for the first example. The second and third examples on failure handling modify this code.

Here, we provide the code for three versions of `move-bottle`, one for each example: `move-bottle1`, `move-bottle2`, and `move-bottle3`. This allows you to add code to the pick-and-place.lisp just once and so that you can simply do the tutorial by invoking the example commands, i.e. by evaluating the three example forms in REPL, each one exemplifying one specific aspect of the plan.

We also include a fourth version, `move-botte4`, which covers the example of defining a new grasp, directly after Exercise 3.

For convenience, we also include four dummy functions to use when doing exercises 1 - 4.

Note that here we don't cover the material in the first two sections of the tutorial, i.e. "Setting Up" and "Understanding the Basics". You need to go through these yourself. Here, we cover the material in the section "Simple Fetch and Place".

Contents [hide]

Update `pick-and-place.lisp` [edit]

First, let's copy the example code.

Move into the src directory:

http://www.vernon.eu/wiki/Zero_Prerequisites_Demo_Tutorial:_Simple_Fetch_and_Place