# KITTING WORLD MODEL

The Java Kitting Demo loopback CRCL server simulates the world model that is actually reported by the CRCL status. At a basic level, CRCL reports robot pose and joint position and command completion status. However, CRCL was extended to include world model status reporting which include the object instances (gears, trays, and kits) and instance inferences (slot state). In short, this means that each gear, supply tray and kit in the robot environment is reported: its name, type and location. Using kitting definitions for gears and trays and slots, inferences are about Kitting world are made about featur instances (e.g., slots in trays).

The World model is built from the CRCL and updated every time CRCL reports status (for both real and simulated loopback). Within Java, there is a CShape class definition which defines the basic object in the kitting world. It contains a name, type and centroid (which is a pose but centered within the object). In addition each shape contains a list of inferences about its feature instances, for example, slot features in a tray. Each inference contains a name (typically the slot name or gear name), its type (gear, slot) its location (for example, a slot has a position within a tray), a state (for example, whether the slot is open or contains a gear), parent and slot (used by a gear to indicate ownership – tray and slot names to indicate gear location). Within Java, there is also a CShapes class definition which contains two static lists:

1. instances derived from the CRCL status reporting and
2. definitions which gives specifics details about objects For example, each tray and kit contains the tray centroid, along with each slot name, type and location within the tray.



## CShape

CShape describes the basic attributes of the world model objects. Each object has a name, type and location. Slots can also be shapes. Trays also have a contains "slot" items. In addition, each shape has inferences about its state – for example if a slot is open or contains a gear. Depending on the parent (kit or supply tray) this can be a free gear or a filled slot.

So each CShape contains these basic variable definitions:

|  |  |
| --- | --- |
| Variable Type | Variable |
| List<[CShape](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html)> | [**\_contains**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#Z:Z_contains) slots in tray |
| PmPose | [**\_location**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#Z:Z_location) location of xyz bottom of object. |
| Vector<[CShape.inference\_type](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.inference_type.html)> | [**inferences**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#inferences) container for all inferences of a shape. |

Given this definition of a CShape, convenience methods are available to understand the type specifics of the CShape based on type as well as any inferences that are associated with the CShape.

|  |  |
| --- | --- |
| Return type | Method |
| boolean | [**isGear**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#isGear--)() method to determine if shape is gear. |
| boolean | [**isKit**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#isKit--)() method to determine if shape is kit. |
| boolean | [**isSkuPart**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#isSkuPart--)() does the model name contain sku which signals a kitting scene object. |
| boolean | [**isVessel**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#isVessel--)() method to determine if shape is supply vessel. |
| [CShape.inference\_type](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.inference_type.html) | [**findInference**](file:///C:\Users\michalos\Documents\agility%20performance%20metrics\Gwendolen\NetBeansProjects\GwendolynCrclClient\dist\javadoc\gwendolyncrclclient\CShape.html#findInference-java.lang.String-)**(String name)**  **find the inference with the given name in this shape.** |

CShape.inference\_type is an inference structure for describing a shape's inferences. For each object there can be multiple inferences. For example, a supply tray with four medium slots, will have 4 inferences about each of the slots. Likewise, for each gear in the supply tray, it will have an inferences describing its parent (tray name) and slot (slot name in parent). In summary, for a tray, each slot has its own inference, under the tray shape. For gears, the inference describes the tray and slot the in which the gear is located.

|  |  |
| --- | --- |
| **Variable** | Description of inference variable |
| **name** | name of the inference object - sku +type + numeric id |
| **type** | name of the inference object type - sku + type |
| **location** | pose of the inference object |
| **state** | state of the inference object occupied by name of gear or empty |
| **parent** | name of the inference object parent if slot |
| **slot** | name of the inference object slot if gear |

## CShapes

CShapes is a container for the shape instances in the kitting scene, the definitions defining the parts in the scene (gears, kits, supply vessel, as well as the slots in a tray).

Below are the methods (all static definitions) so an example use is CShapes.findFreeGear which then looks in the available supply trays for a free gear to transfer to kit open slot.

|  |  |
| --- | --- |
| Method | Description |
| findOpenKittingGearSlot | find an open slot in one of the kits. |
| findFreeGear | use the model inferences to find a free gear of type geartyp in the supply vessels. |
| AVAILBLE ROUTINES TO ACCESS INSTANCES AND PROPERTIES | |
| findDefinition | access method to find the associated definition for a type. |
| findInstance | accessor function to find a matching name in the instances vector supplied as a calling argument. |
| findSlot | given a tray with slot, find the shape definition for the given slot name (e.g., slot1, slot2, ... |
| initDefinitions | initialize all the shape attributes, and if a tray describes all the slots contained. |
| snapshotInstances | mutex clone of current instance. |
| storeInstance | simple store of a model instance given name and pose |
| storeInstanceProperties | store the instance properties for given model. |

## Kitting Algorithm

The kitting demo has a basic algorithm:

1. find an available gear (in supply tray) and kit open slot to place the

2-5) approach, move to gear, grasp gear, and retract

5-8) approach kit open slot move to near open slot, release gear, and then retract

To simplify higher level planning, this lower level CRCL centric robot control planning and control is replaced by a more abstract object instance interaction, as was described in the PDDL plans in the APRS lab:

1. Acquire part – return gear name of free gear
2. Take part(gear name)
3. Place part (kit slot name)

In the file: "GwendolynCrclClient.java" illustrates the simple use of the revised named-centric approach to kitting object manipulation.

while(true) {

CRCLClient.action\_type action= new CRCLClient.action\_type ();

if(crcl.acquire(action)<0)

break;

if(crcl.takepart(action.gear)<0)

break;

if(crcl.placepart(action.gear, action.kit, action.openslot)<0)

break;

Thread.sleep(50);

}

The action\_type Java class definition is used to pass the free gear, kitting tray and open slot names that are available to perform a kitting complete operation. Notice all the object instances are names, that at the lower level are translated into full functional specification of the CShape.