Programming STAUBLI TX90XL

Dr. Kurtuluş Erinç Akdoğan

kurtuluserinc@cankaya.edu.tr

ÇANKAYA ÜNİVERSİTESİ

MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

Applications: Structure

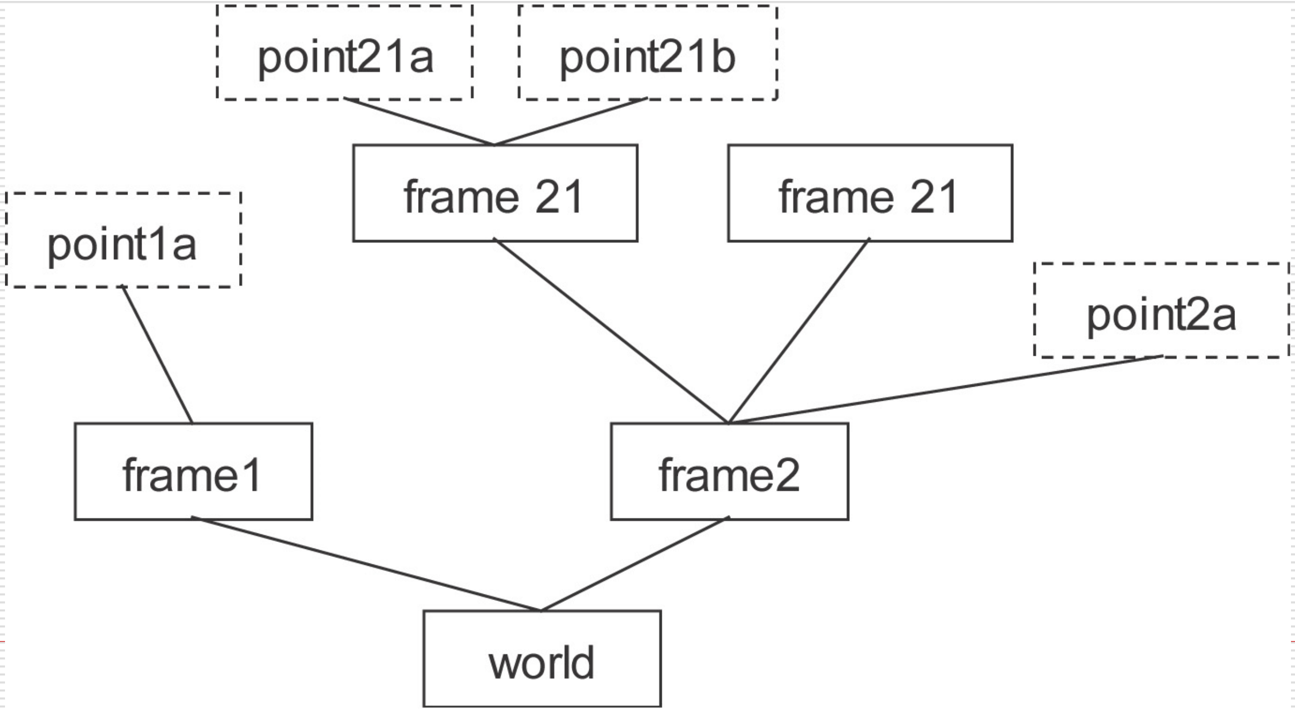
- Variable types:
  - pointRx: a point location in cartesian coordinates (p1, p2, p3, ...)
  - jointRx: a joint location in joint coordinates (jStart)
  - tool: a tool defined by the user (tTool)
  - trsf: a transformation (trAApl)
  - mdesc: motion descriptor (mNomSpeed)

Applications: Structure

- Functions:
  - movej(joint, tool, mdesc): move to a (point or joint) coordinate with specified tool and motion descriptor
  - movel(point, tool, mdesc): move linearly a point coordinate with specified tool and motion descriptor
  - appro(point, trsf): calculate a transformed point by using a point and a transformation
  - waitEndMove: wait for the current movement to end.

FRAME type

- The frame type is used to define the position of reference frames in the cell.
- a reference frame is linked to the world frame, either directly or via other frames.



Use

- To give a more intuitive view of the application points
  - The display of the cell's points is structured according to the hierarchical structure of the frames.
- To update the position of a set of points quickly
  - When an application point is linked to an object, it is advisable to define a frame for that object and link the VAL 3 points to the frame. If the object is moved, simply reteach the frame to allow all linked points to be corrected at the same time.
- To reproduce a trajectory in several places in the cell
  - Define the trajectory points relative to a working frame and teach a frame for each position in which the trajectory is to be reproduced. By assigning the value of a taught frame to the working frame, the entire trajectory "moves" to the taught frame.
- To make it easier to calculate geometrical movements
  - The **compose()** instruction allows geometrical movements expressed in any reference frame to be performed on any point. The **position()** instruction is used to calculate the position of a point in any reference frame.

TRSF type

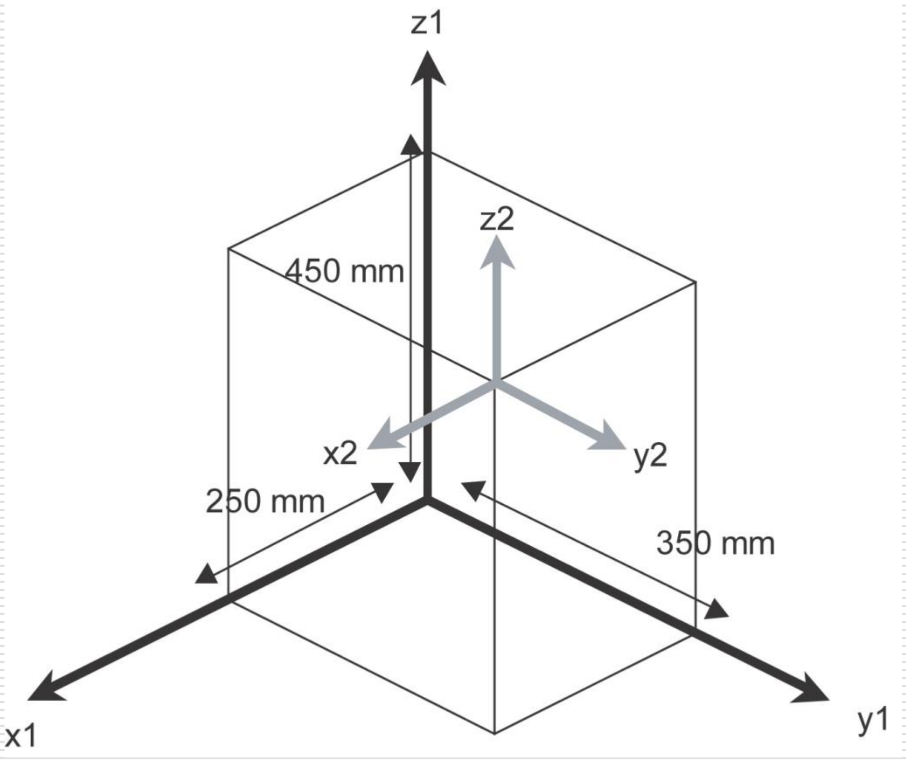
- A transformation (trsf type) defines a position and / or orientation change. It is the mathematical composition of a translation and a rotation.
- A transformation itself doesn´t represent a position in space, but can be interpreted as the position and orientation of a Cartesian point or frame relative to another frame.
- The trsf type is a structured type whose fields are, in this order:

num x	Translation along the x axis
num y	Translation along the y axis
num z	Translation along the z axis
num rx	Rotation around the x axis
num ry	Rotation around the y axis
num rz	Rotation around the z axis

- The x, y and z fields are expressed in the unit of length of the application (millimeter or inch, see the chapter entitled Unit of length). The rx, ry and rz fields are expressed in degrees.
- The x, y and z coordinates are the Cartesian coordinates of the translation (or the position of a point or frame in the reference frame). When rx, ry and rz are zero, the transformation is a translation without change of orientation.
- When a trsf type variable is initialized, its default value is {0,0,0,0,0,0}.

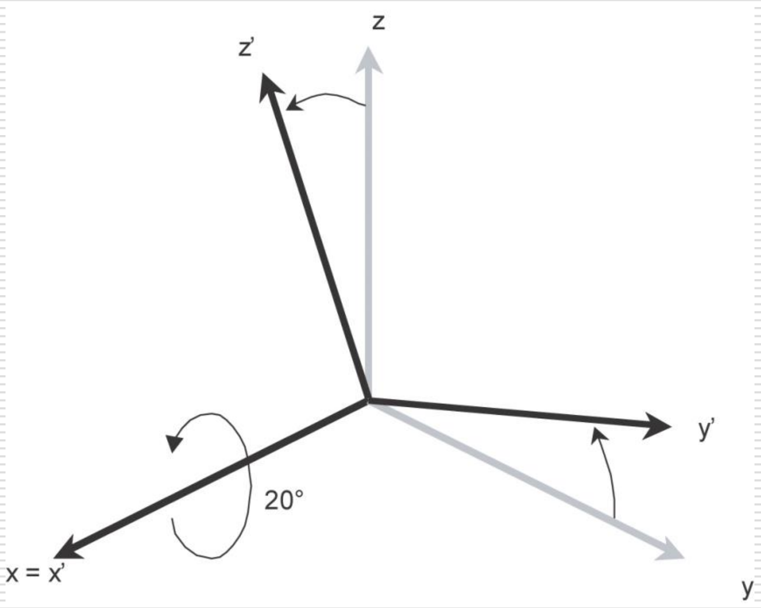
Translation

- The position of frame R2 (grey) relative to R1 (black) is:
- x = 250mm, y=350mm, z = 450mm,
- rx = 0°, ry = 0°, rz = 0°



Orientation

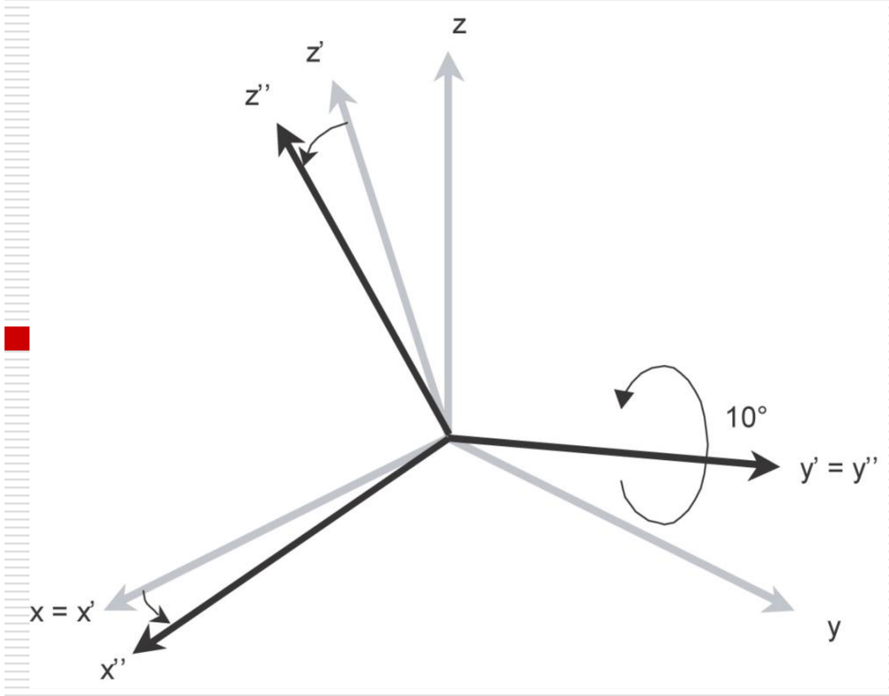
- orientation **rx = 20°, ry = 10°, rz = 30°** is obtained as follows.
- First, the frame **(x,y,z)** is rotated through **20°** around the **x** axis. This gives a new frame **(x’,y’,z’)**. The **x** and **x’** axis coincide.

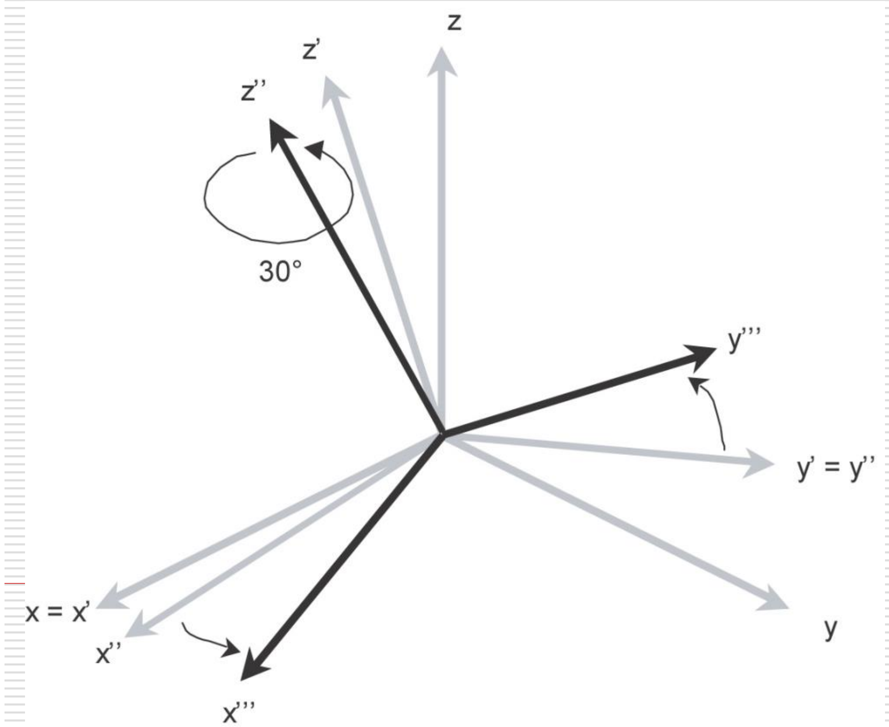


- Then the frame is rotated through 10° around the y′ axis of the frame obtained at the previous step. This gives a new frame (x′′,y′′,z′′). The y′ and y′′ axis coincide.

- Lastly, the frame is rotated through 30° about the z′′ axis of the frame obtained at the previous step. The orientation of the new frame obtained (x′′′,y′′′,z′′′) is defined by rx, ry, rz. The z′′ and z′′′ axis coincide

- The position of frame R2 (grey) relative to R1 (black) is:
  - x=250mm,
  - y=350 mm,
  - z=450mm,
  - rx = 20°,
  - ry = 10°,
  - rz = 30°



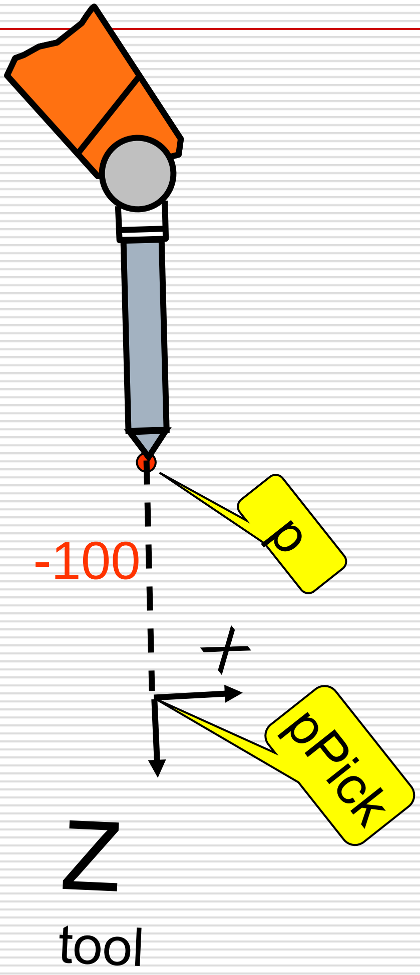


Approach with Compose

- **point compose**(**point** *pPosition*, **frame** *fReference*, **trsf** *trTransformation*)
- This instruction returns the *pPosition* to which the geometrical transformation *trTransformation* is applied relative to the *reference frame*
- Approach with Compose is according to coordinate frame

- `trShiftz={0,0,100,0,0,0}`

- **`p=compose(pPick,world,trShiftz)`**
- **`movel(p,tGrip,mFast)`**

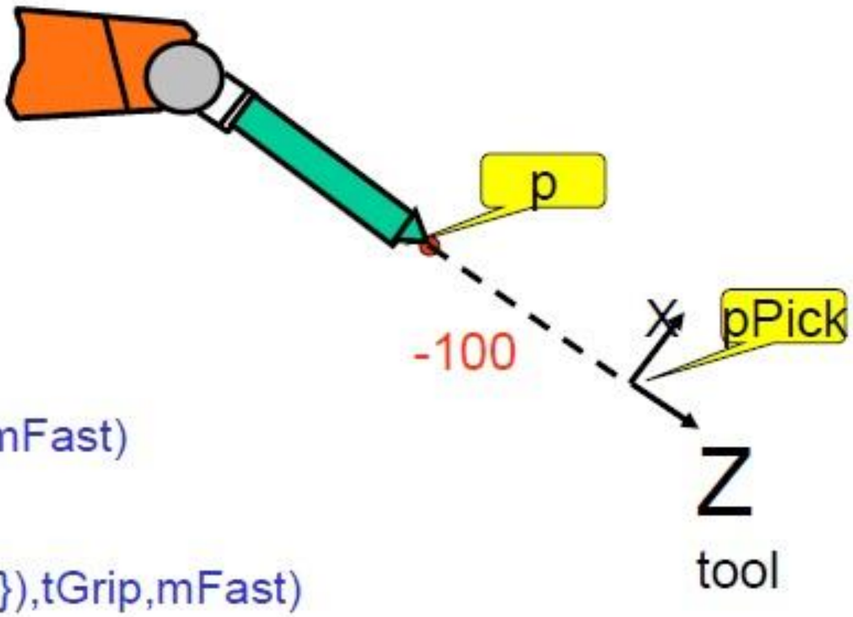


APPROACH ON POINT

POINT <== appro(POINT,TRSF)

APPRO computes a Cartesian point related to a point on which is applied a transformation

```
POINT p    POINT pPick  TRSF trShiftz    NUM nDistance=100   are defined
trShiftz={0,0,-nDistance,0,0,0}

p=appro(pPick,trShiftz)
movej(p,tGrip,mFast)
```



other writing :

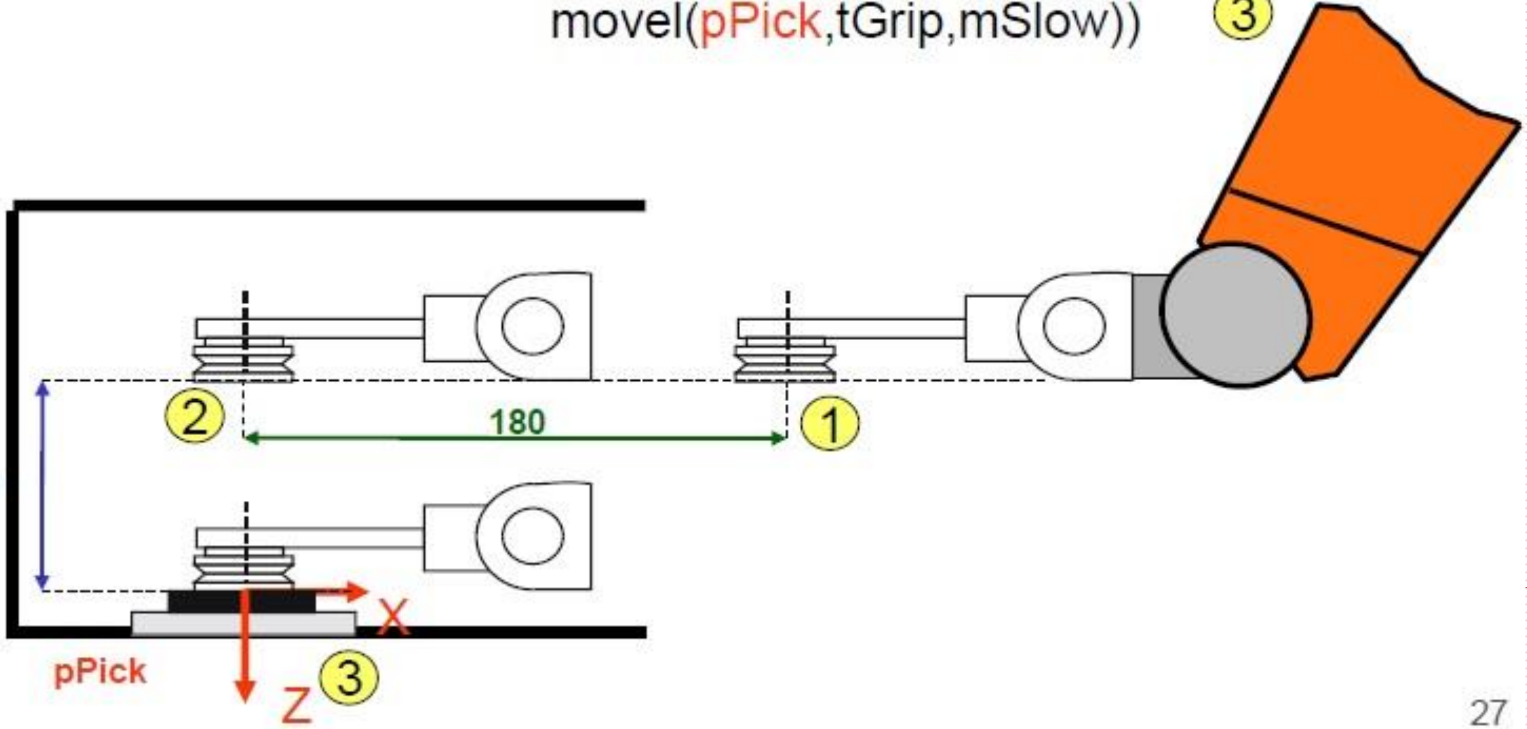
```
movej(appro(pPick,trShiftz),tGrip,mFast)
```

other writing :

```
movej(appro(pPick,{0,0,-100,0,0,0}),tGrip,mFast)
```

COMPLEX APPROACH - 1 -

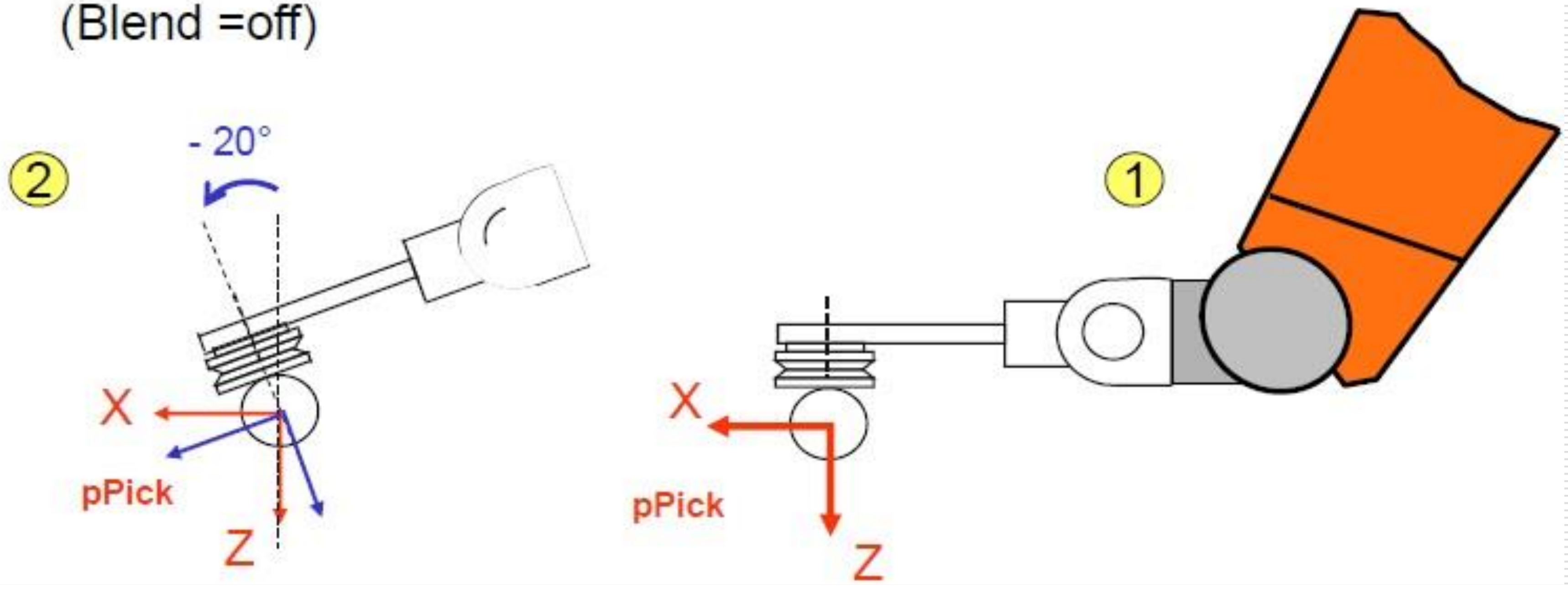
```
movej(appro(pPick,{180,0,-100,0,0,0}),tGrip,mSlow)        1

    movel(appro(pPick,{0,0,-100,0,0,0}),tGrip,mSlow)      2

                        movel(pPick,tGrip,mSlow))         3
```



COMPLEX APPROACH - 2 -

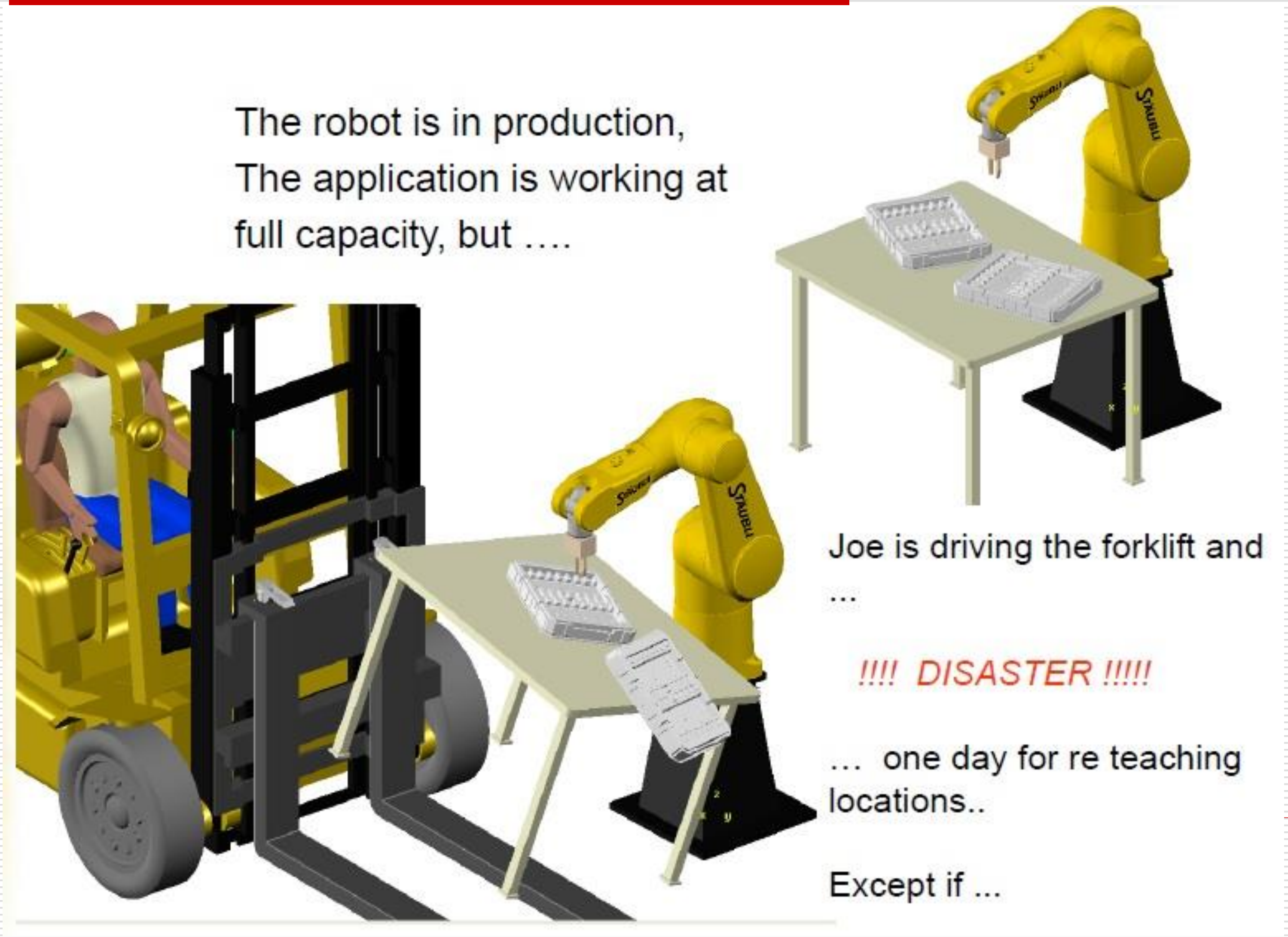
```
movel(pPick,tGrip,mSlow)                                1

movel(appro(pPick,{0,0,0,0,-20,0}),tGrip,mSlow)         2
```

(Blend =off)



Local Coordinate system & Palettization

WHY USING A FRAME ?



The robot is in production,
The application is working at full capacity, but ….

Joe is driving the forklift and …

!!!! DISASTER !!!!!

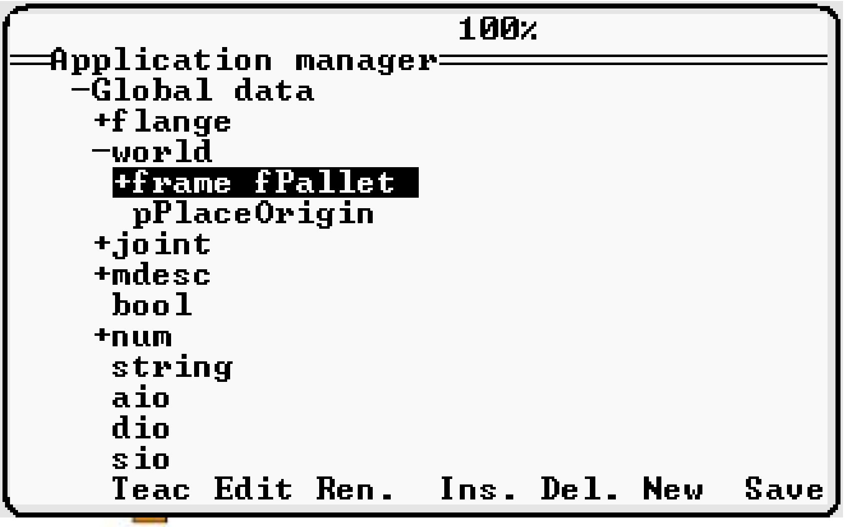
… one day for re teaching locations..

Except if ...

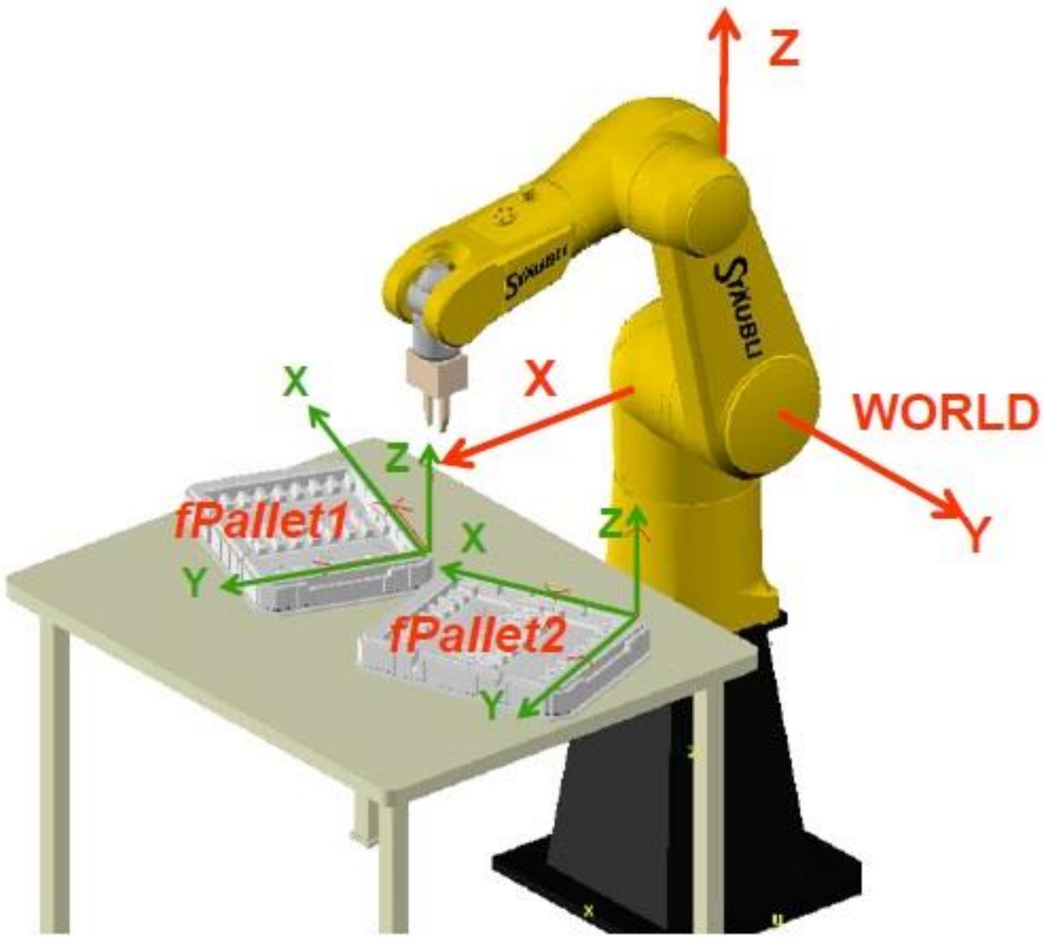
FRAME CREATION

Local Coordinate system :

- to make points re teaching easier
- used to duplicate locations
- shift of points in a pallet

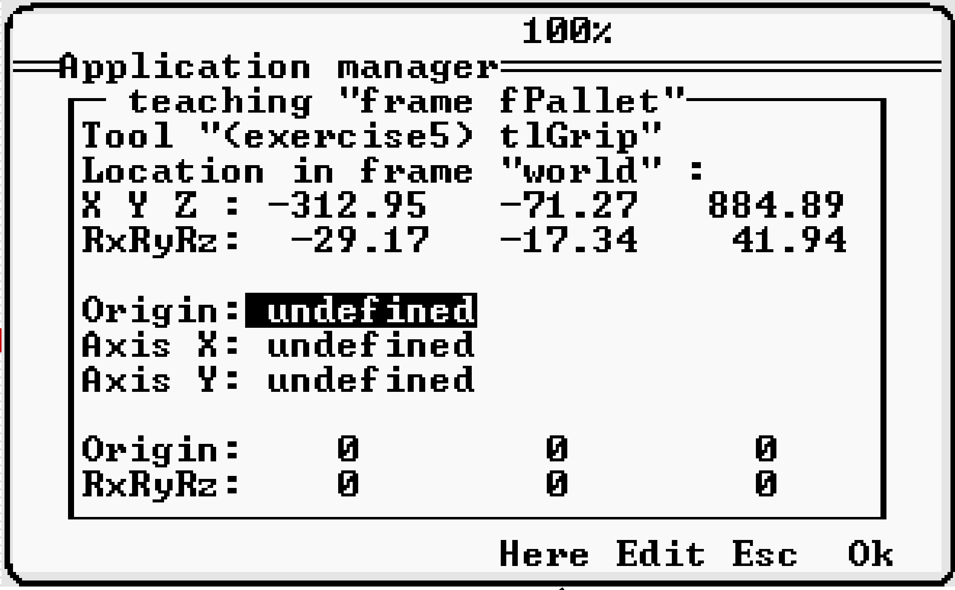


```
                    100%
=Application manager=====================
   -Global data
    +flange
    -world
     +frame fPallet
      pPlaceOrigin
    +joint
    +mdesc
     bool
    +num
     string
     aio
     dio
     sio
     Teac Edit Ren.  Ins. Del. New  Save
```

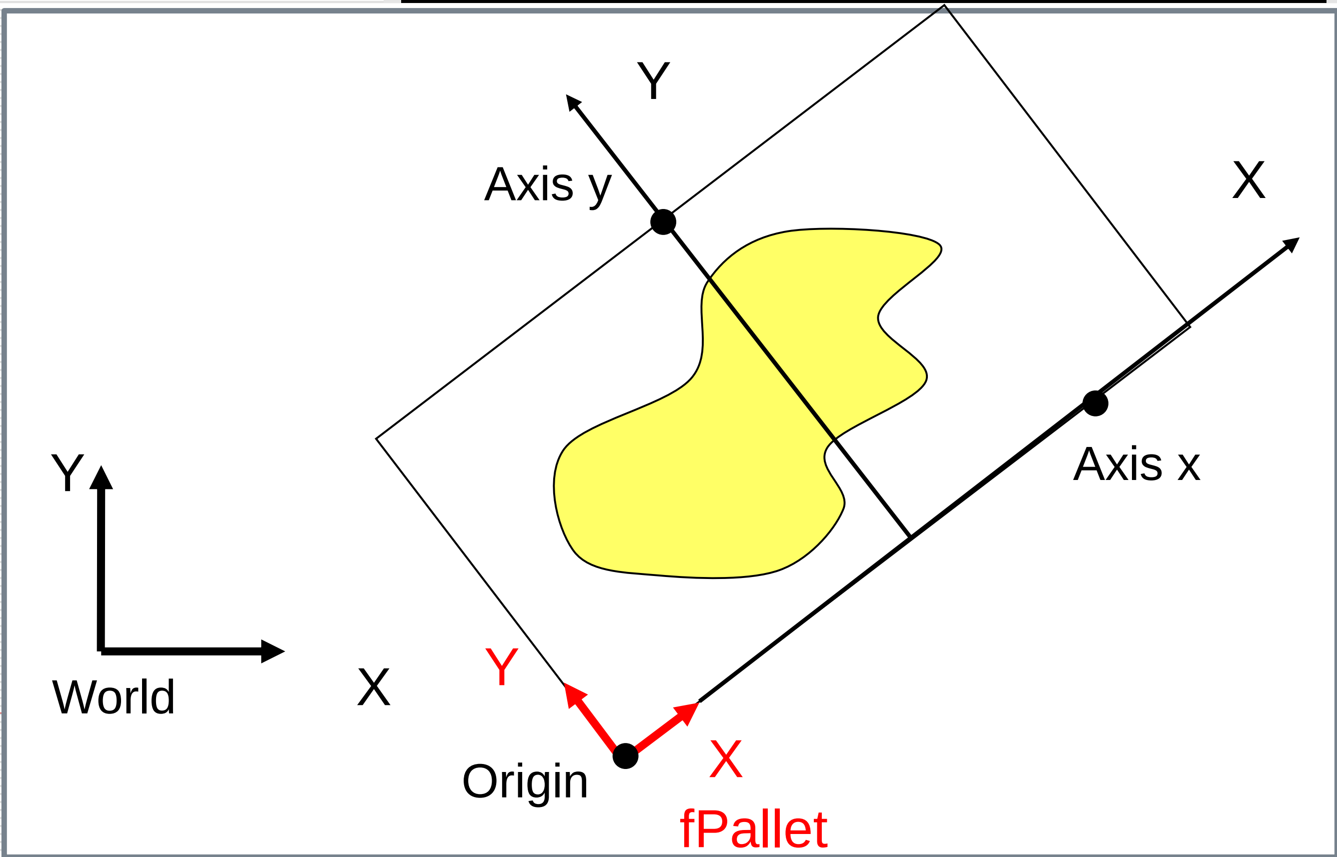


FRAME TEACHING

- Defined with 3 points to teach :
  - Use a precise tool : pointer
  - Define this tool as current
  - Teach points as far as possible each other (+ accurate)

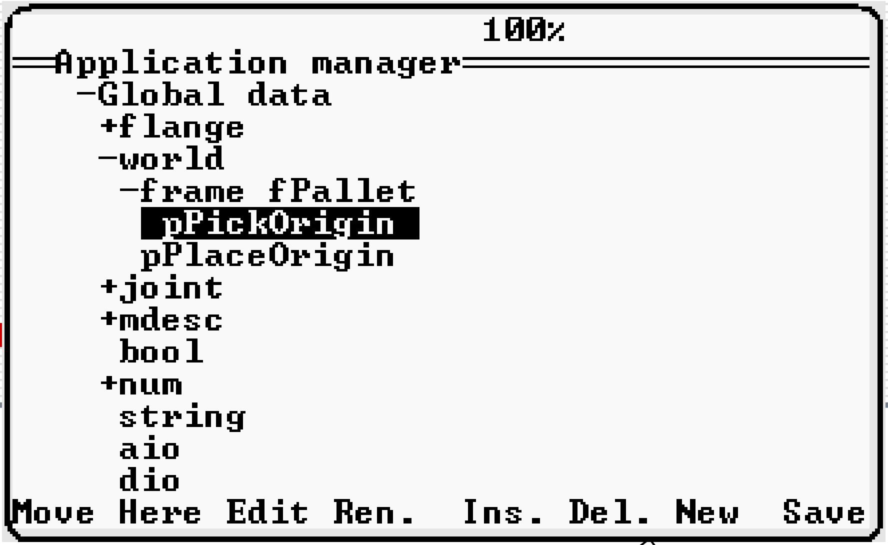


```
                              100%
Application manager
   teaching "frame fPallet"
Tool "(exercise5) tlGrip"
Location in frame "world" :
X Y Z : -312.95    -71.27    884.89
RxRyRz:  -29.17    -17.34     41.94

Origin: undefined
Axis X: undefined
Axis Y: undefined

Origin:     0         0         0
RxRyRz:     0         0         0

                      Here Edit Esc  Ok
```

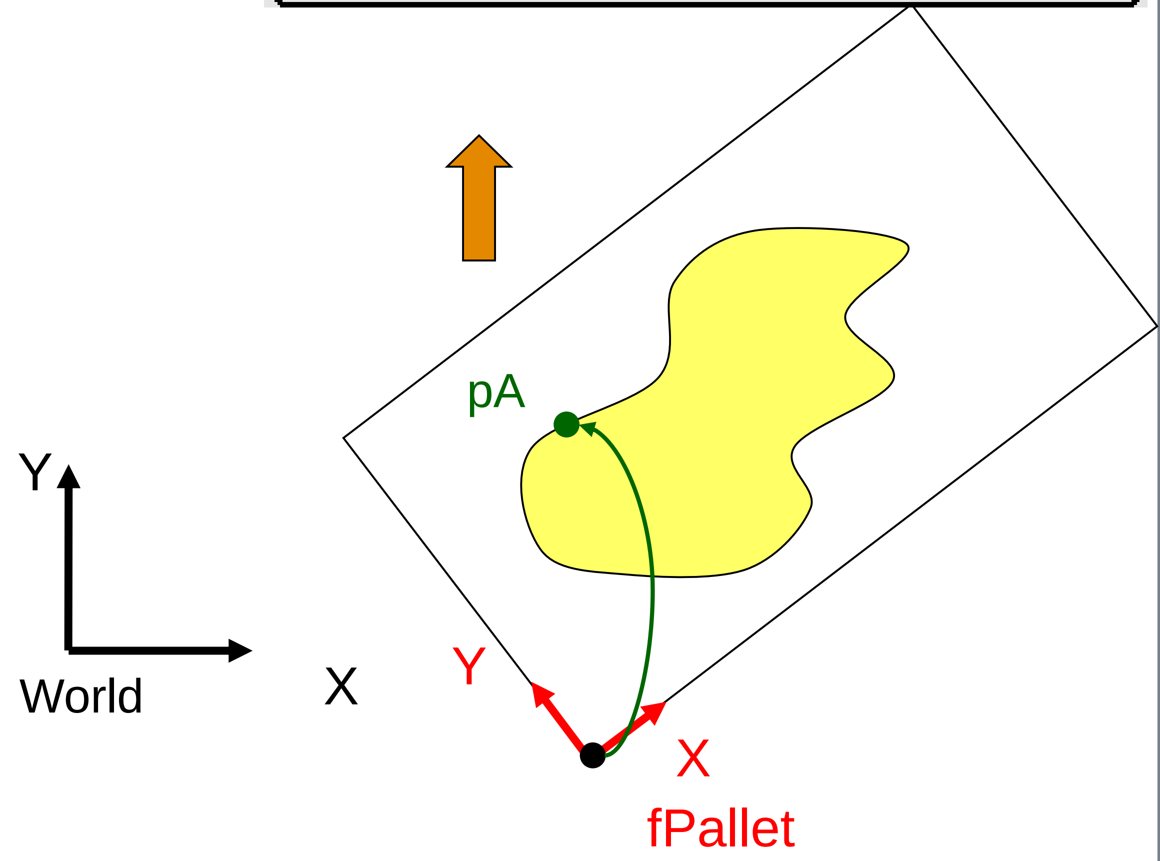


POINTS IN A FRAME

- Create in tree of frame, the needed points :
- During teaching coordinates are displayed in frame reference
- For the move instruction , it is not needed to specify the frame :
- movej(pA ,tGrip, mFast)



```
                    100%
Application manager
   -Global data
     +flange
     -world
       -frame fPallet
         pPickOrigin
         pPlaceOrigin
     +joint
     +mdesc
      bool
     +num
      string
      aio
      dio
Move Here Edit Ren.  Ins. Del. New  Save
```



COMPUTE A FRAME BY PROGRAM

```
nError = setFrame(pOrigin, pX,pY, fRef)
```

3 points O, +X, +Y

Frame to compute

Error Code :

0 : no error
-1 : ptX too closed from ptOrigin
-2 : 3 points are nearly aligned

```
setFrame(p0,px,py,fPallet)
```

PALETTIZATION IN A FRAME

`Compose(point,frame,trsf)` : compute a point by translation and rotation defined in trsf relative to menntioned frame in the function

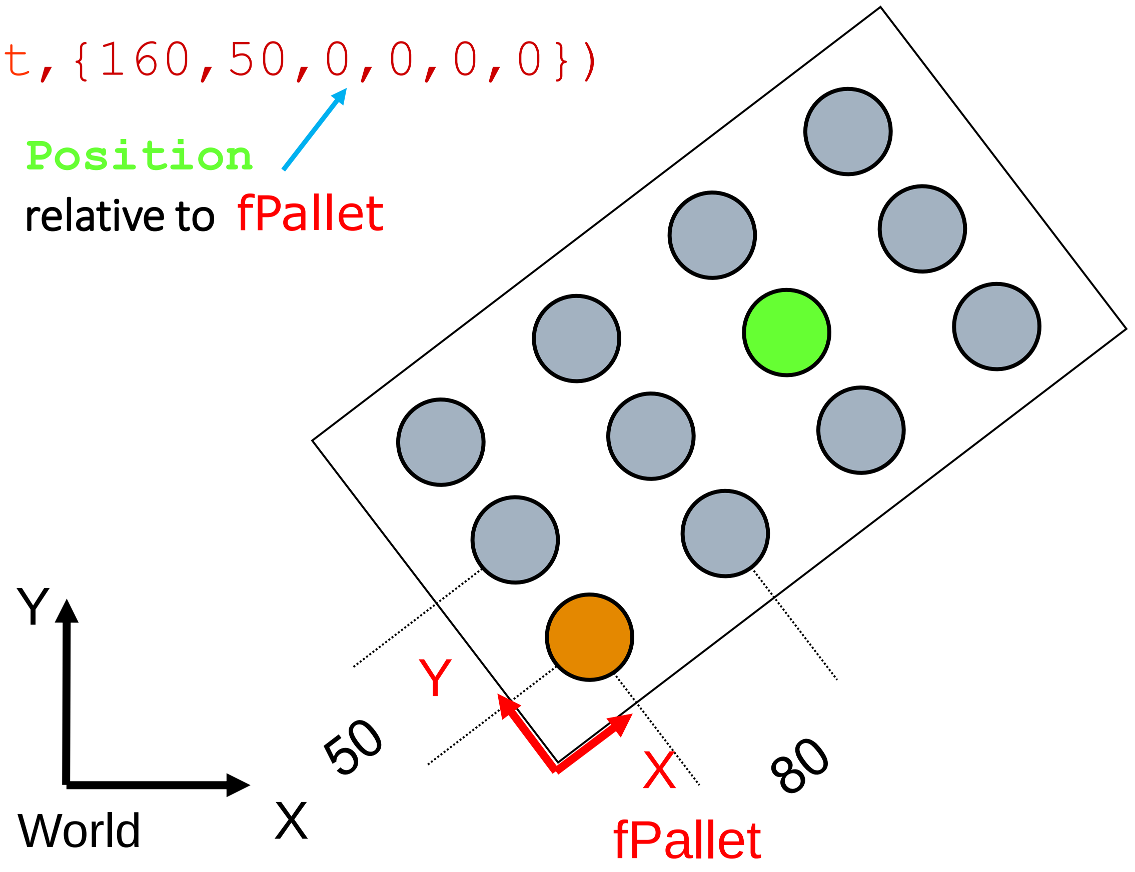
```
p=compose(pFirst,fPallet,{160,50,0,0,0,0})
```

Center of closest hole to the origin of fPallet

Frame defined as fPallet

Position relative to fPallet

```
movel(p,tGrip,mSlow)
```



void **open**(tool tTool)

Function

This instruction activates the tool (opening) by setting its digital output to **true**.
Before activating the tool, **open**() waits for the robot to reach the required point by carrying out the equivalent of a **waitEndMove**(). After activation, the system waits for **otime** seconds before executing the next instruction.
This instruction does not make sure that the robot is stabilized in its final position before the tool is activated.

When it is necessary to wait for complete stabilization of the movement, the **isSettled**() instruction must be used.
A runtime error is generated if the **tTool dio** is not defined or is not an output, or if a previously recorded motion command cannot be run.

Example

```
// the open() instruction is equivalent to:
waitEndMove()
tTool.gripper=true
delay(tTool.otime)
```

void **close**(tool tTool)

Function

This instruction activates the tool (closing) by setting its digital output to **false**.
Before activating the tool, **close**() waits for the robot to stop at the point by carrying out the equivalent of a **waitEndMove**(). After activation, the system waits for ctime seconds before executing the next instruction.
This instruction does not make sure that the robot is stabilized in its final position before the tool is activated. When it is necessary to wait for complete stabilization of the movement, the **isSettled**() instruction must be used.
A runtime error is generated if the **tTool dio** is not defined or is not an output, or if a previously recorded motion command cannot be run.

Example

```
// the close instruction is equivalent to:
waitEndMove()
tTool.gripper = false
delay(tTool.ctime)
```