trinity

# DYNAMIC ROBOT TRAJECTORY GENERATION BASED ON INFORMATION FROM 3D CAMERA

INTEGRATION TUTORIAL

www.trinityrobotics.eu

## Hardware requirements

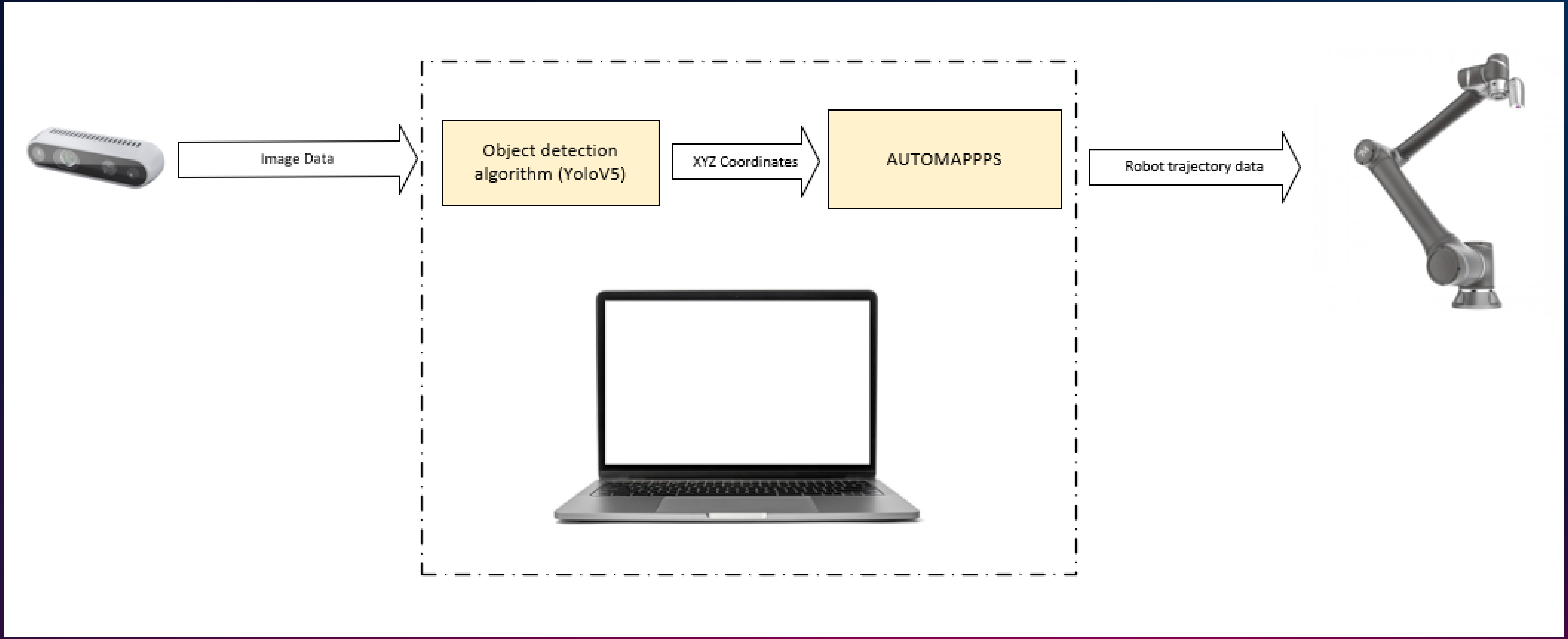
- Workstation PC
- GPU at least NVIDIA GeForce GTX 1060 or Quadro P5000, AMD Radeon Vega 56
- Memory at least 8 Gb
- Free disk space at least 1Gb
- USB 3.0 port
- Intel Realsense D435 camera
- Robot compatible with AutoMAPPPS

## Software requirements

- Windows 10 OS
- AutoMAPPPS
- AutoMAPPPS BinPicking and CellController modules
- Intel Realsense SDK (OS install)
- YoloV5

## Overview

- Software/Hardware infrastructure:



## Choosing the right camera

- Choosing the camera depends on environment and application requirements
  - Triggering, IP classification, connectivity, working principle, lighting…
- In this case, we need a high-resolution RGB+D image for object detection and trajectory planning
- → Intel Realsense D435i
  - Stereo camera with IR projector for enhanced accuracy

trinity ENGAGE WITH AGILE MANUFACTURING

## Positioning the camera

- Things to consider:
  - Good view of robot working area
  - Avoiding disturbance
    - Shaking, temperature, dust/fog/smoke, electromagnetic interference
  - Lighting
  - Connecting the device
  - Angle between camera and working area
    - Perpendicular is recommended to avoid positioning errors

## Installing the camera

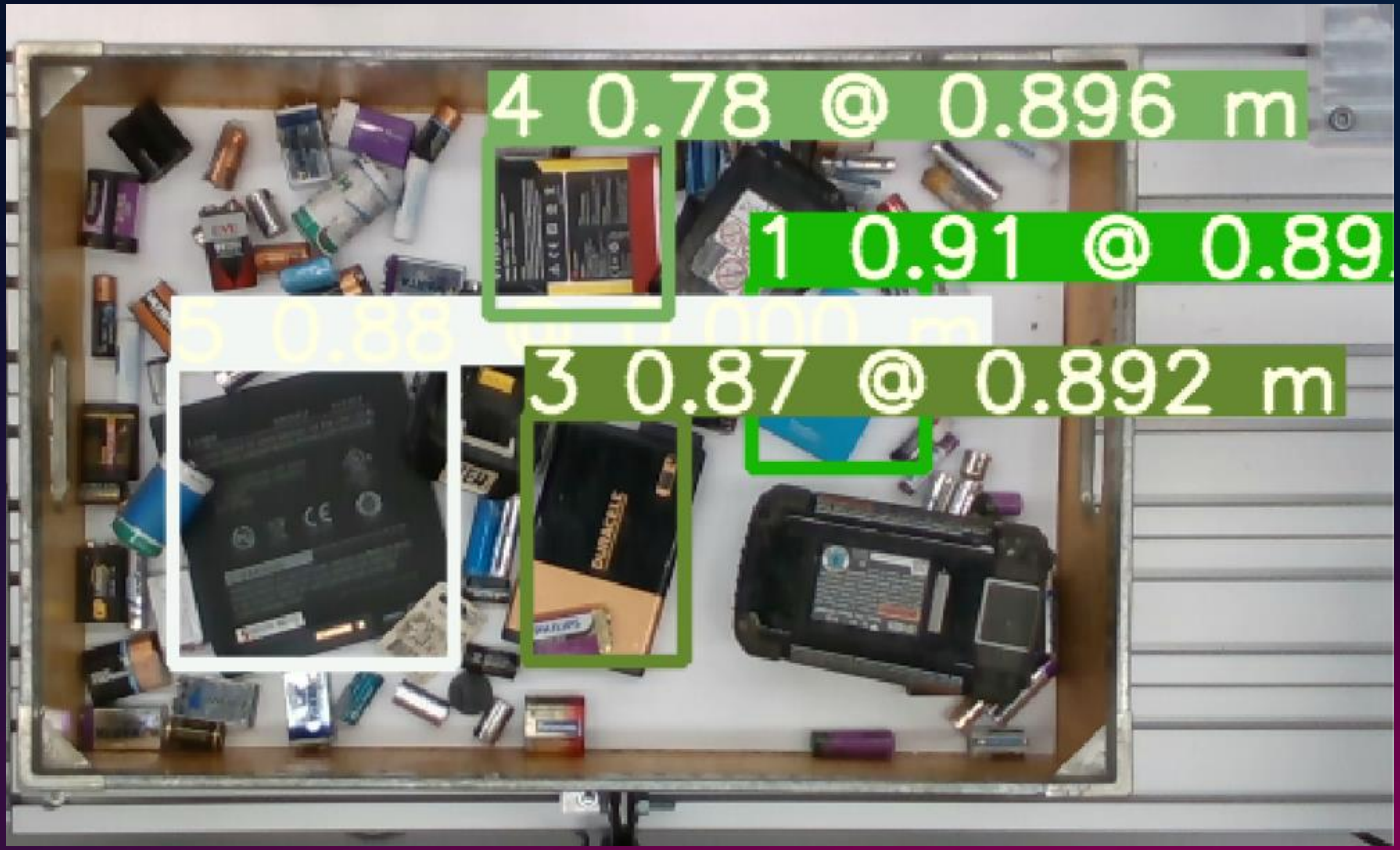
- Mounting guidelines may vary between equipment
  - Most manufacturers provide detailed screw pattern drawings, mounting instructions and mounting brackets
- Follow the manufacturer’s general instructions
- Ensure proper alignment

# Training YoloV5 model on custom data

- Creating a custom model for object detection requires:
  - A sufficient image dataset
  - Labeling the training data
  - Training the model
- Additional instructions:
  - https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data

trinity ENGAGE WITH AGILE MANUFACTURING

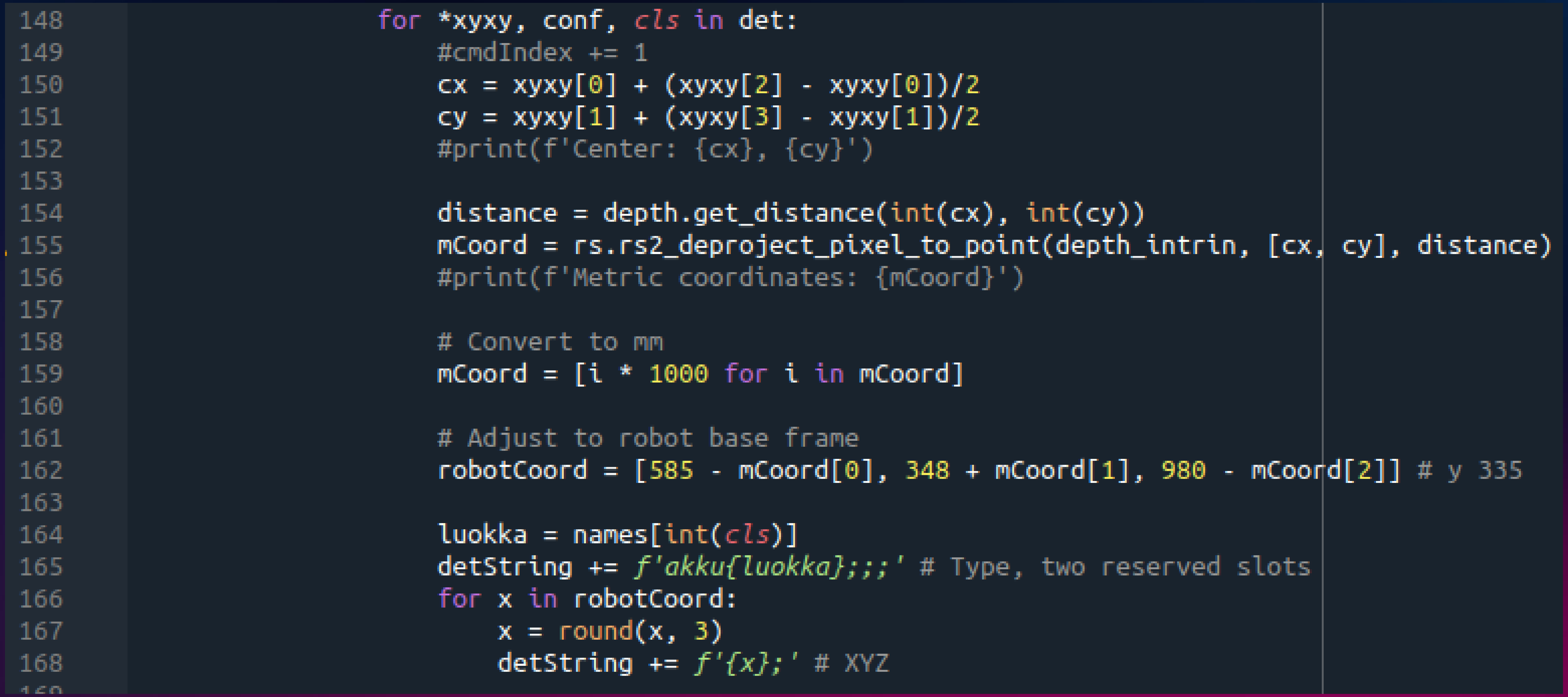
# Training YoloV5 model on custom data



trinity ENGAGE WITH AGILE MANUFACTURING

# Extracting workpiece coordinates

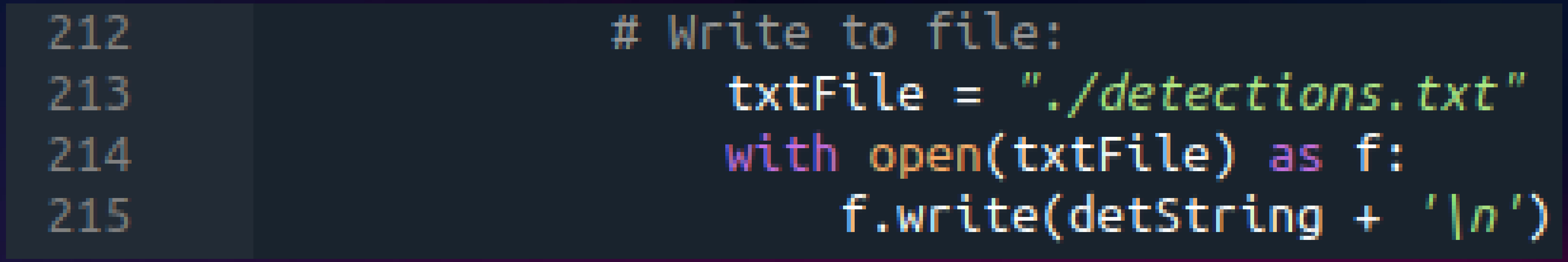
- We will calculate the workpiece coordinates using depth data from the camera



```
                for *xyxy, conf, cls in det:
                    #cmdIndex += 1
                    cx = xyxy[0] + (xyxy[2] - xyxy[0])/2
                    cy = xyxy[1] + (xyxy[3] - xyxy[1])/2
                    #print(f'Center: {cx}, {cy}')

                    distance = depth.get_distance(int(cx), int(cy))
                    mCoord = rs.rs2_deproject_pixel_to_point(depth_intrin, [cx, cy], distance)
                    #print(f'Metric coordinates: {mCoord}')

                    # Convert to mm
                    mCoord = [i * 1000 for i in mCoord]

                    # Adjust to robot base frame
                    robotCoord = [585 - mCoord[0], 348 + mCoord[1], 980 - mCoord[2]] # y 335

                    luokka = names[int(cls)]
                    detString += f'akku{luokka};;;' # Type, two reserved slots
                    for x in robotCoord:
                        x = round(x, 3)
                        detString += f'{x};' # XYZ
```

# Extracting workpiece coordinates

- The coordinates will be saved to a text file
  - *'objId x y z'*
- The text file will later be used for target input in AUTOMAPPPS



```
            # Write to file:
                txtFile = "./detections.txt"
                with open(txtFile) as f:
                    f.write(detString + '\n')
```

## CellController Configuration

- Communication of CellController is defined in file "CellControllerConfig.XML". It includes necessary definitions for robot controller to run

```
<ImageProcessingSystem
  mType = "TCPIP_9D"
  mHost = "169.254.221.9"
  mPort = "10000"                                    // this is the default value for TM
  mFile = "./CellController/data.txt"
  mPart = "akku"                                     // general name of objects to be picked
  mLocatedAllParts = "False"
  mUsePickPoseForObjectPose = "False"
>
</ImageProcessingSystem>
```

## Configuration

- The system supports picking of multiple types of workpieces which general name is given in parameter mPart.
- In the input file, the workpiece menu must contain workpieces with this name. If there are several types of workpieces, the names are akku0, akku1, akku2 etc.
- When online planner gets location of each part, it creates an own workflow for picking each part.
- Numbering of workflows goes as follows: for all types of akku0: akku0-0-Workflow, akku0-1-
- Workflow, akku0-2-Workflow, for types of akku1: akku1-0-Workflow, akku1-1-Workflow, etc.

## Example Configuration

```
<OnlinePlanner
  mHost = "localhost"
  mPort = "8501"
  mProjectFileIn = "./CellController/TMCentria-binPicking-3.0.16-real.ap"       // the name of input project file
  mProjectFileOut = "./CellController/resultCellController.ap"                  // the name of output project file
  mTemplateXML = "./CellController/template.xml"
  mOutputXML = "./CellController/job.xml"
  mObjectTag = "TEMPLATEPART"
  mDataObjectTag = "TEMPLATEWORKPIECE"
  mRefPointTag = "TEMPLATEREFPOINT"
  mStatus = "./CellController/jobStatus.txt"
  mDebug = "True"                                                               // True will echo all actions to console
/>
```

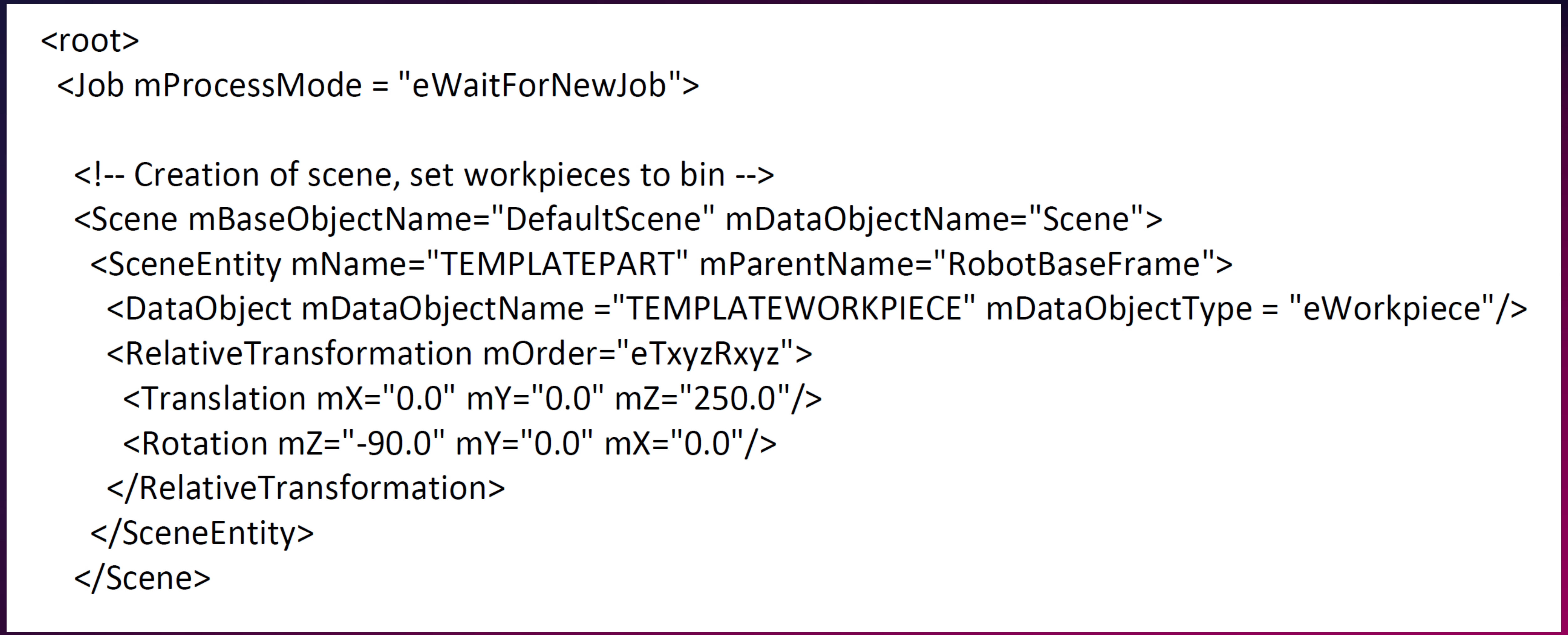
trinity ENGAGE WITH AGILE MANUFACTURING

# Example Configuration

```
  <RobotController
   mType = "TM_LISTEN_NODE"
   mHost = "169.254.221.10"                                              // IP address of robot controller
   mPort = "5890"                                                        // TM default port
   mUsername = ""                                                        // by default there is no username or password
   mPassword = ""
   mObjectTag = "TEMPLATEPART"                                           // String that is searched from template.xml file
   mProgramSourcePath = "./CellController/jobRobotProgramTEMPLATEPART.txt"      // folder where exported robots
                                                                                   will be searched for uploading
   mProgramDestinationPath = "/programs/citMotion.script"
   mMainSourcePath = "./CellController/citMotion.urp"
   mMainDestinationPath = "/programs/citMotion.urp"
   mPlay = "True"
   mPlayAll = "False"
   mWait = "True"
   mDebug = "True"                                                       // True will echo all actions to console
  />
</root>
```

## Template configuration

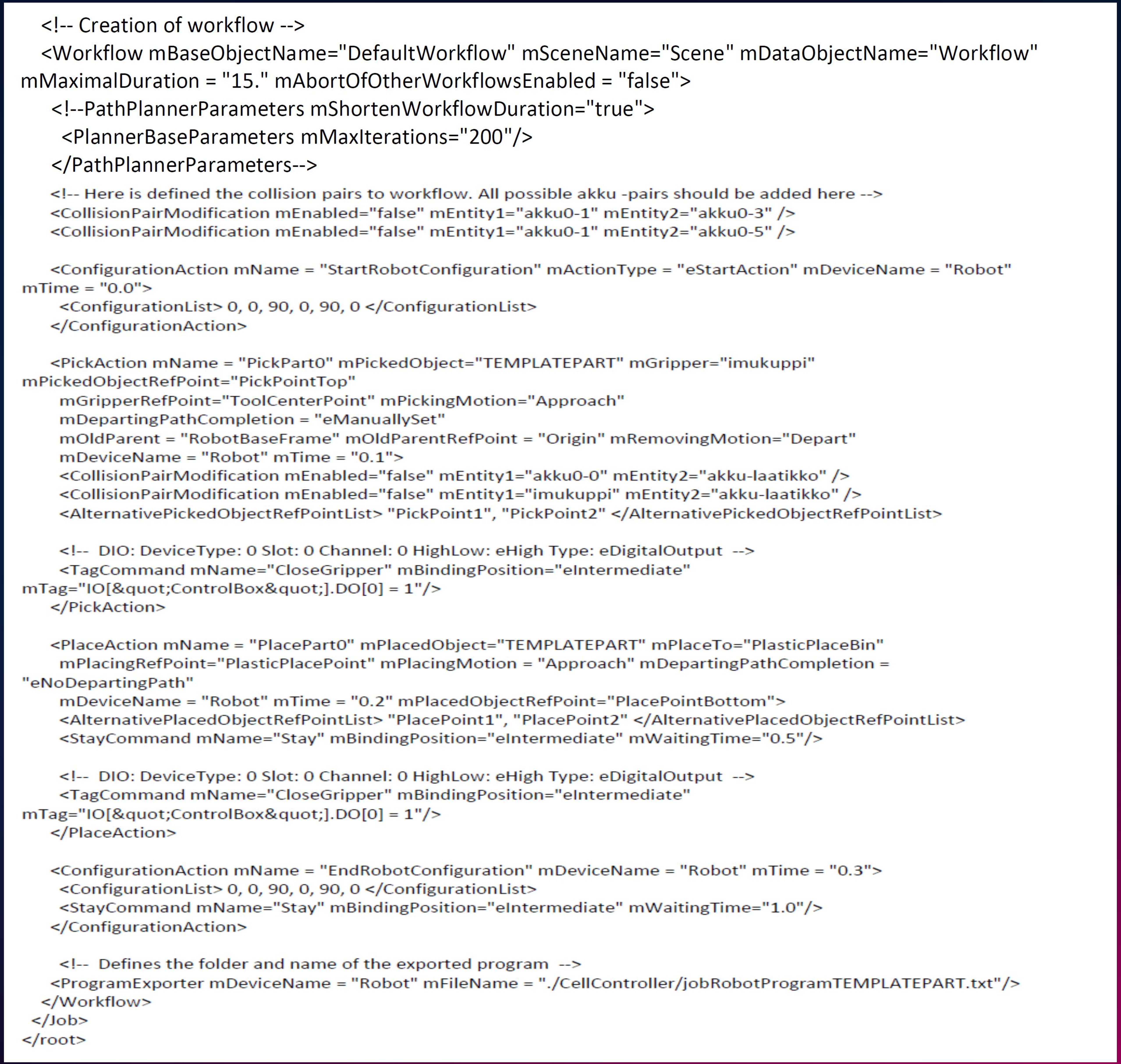
- Template.xml -file defines the tasks that will be done in the workflow and can be configured by the operator.
- Below defined scene is the template which will be used for creating job.xml file where each workobject creates a new entry to scene
- Here TEMPLATEPART and TEMPLATEWORKPIECE will be replaced by correct name as well as location of work object.



```xml
<root>
 <Job mProcessMode = "eWaitForNewJob">

  <!-- Creation of scene, set workpieces to bin -->
  <Scene mBaseObjectName="DefaultScene" mDataObjectName="Scene">
   <SceneEntity mName="TEMPLATEPART" mParentName="RobotBaseFrame">
    <DataObject mDataObjectName ="TEMPLATEWORKPIECE" mDataObjectType = "eWorkpiece"/>
    <RelativeTransformation mOrder="eTxyzRxyz">
     <Translation mX="0.0" mY="0.0" mZ="250.0"/>
     <Rotation mZ="-90.0" mY="0.0" mX="0.0"/>
    </RelativeTransformation>
   </SceneEntity>
  </Scene>
```

trinity ENGAGE WITH AGILE MANUFACTURING

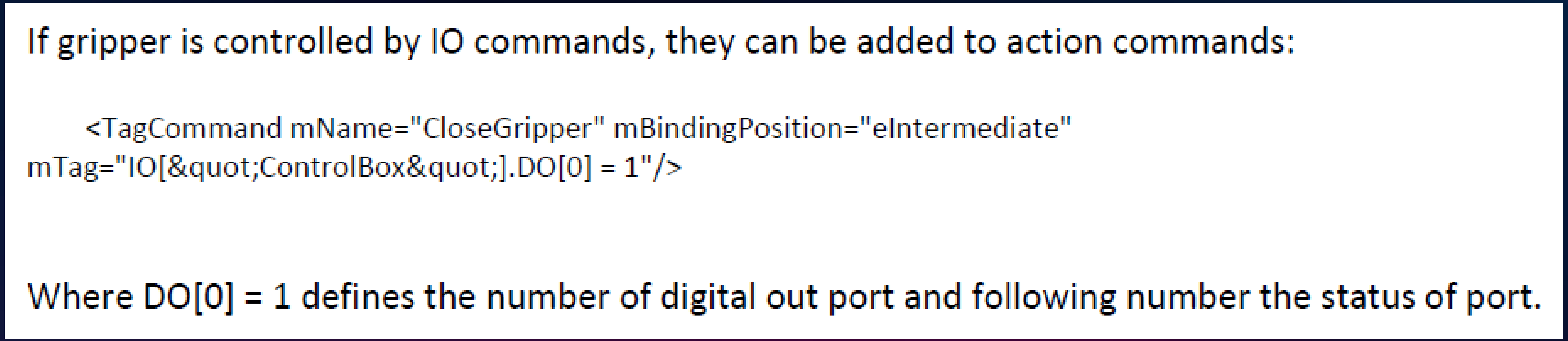
# Example Template Configuration



```
    <!-- Creation of workflow -->
    <Workflow mBaseObjectName="DefaultWorkflow" mSceneName="Scene" mDataObjectName="Workflow"
mMaximalDuration = "15." mAbortOfOtherWorkflowsEnabled = "false">
     <!--PathPlannerParameters mShortenWorkflowDuration="true">
       <PlannerBaseParameters mMaxIterations="200"/>
     </PathPlannerParameters-->
     <!-- Here is defined the collision pairs to workflow. All possible akku -pairs should be added here -->
     <CollisionPairModification mEnabled="false" mEntity1="akku0-1" mEntity2="akku0-3" />
     <CollisionPairModification mEnabled="false" mEntity1="akku0-1" mEntity2="akku0-5" />

     <ConfigurationAction mName = "StartRobotConfiguration" mActionType = "eStartAction" mDeviceName = "Robot"
mTime = "0.0">
       <ConfigurationList> 0, 0, 90, 0, 90, 0 </ConfigurationList>
     </ConfigurationAction>

     <PickAction mName = "PickPart0" mPickedObject="TEMPLATEPART" mGripper="imukuppi"
mPickedObjectRefPoint="PickPointTop"
       mGripperRefPoint="ToolCenterPoint" mPickingMotion="Approach"
       mDepartingPathCompletion = "eManuallySet"
       mOldParent = "RobotBaseFrame" mOldParentRefPoint = "Origin" mRemovingMotion="Depart"
       mDeviceName = "Robot" mTime = "0.1">
       <CollisionPairModification mEnabled="false" mEntity1="akku0-0" mEntity2="akku-laatikko" />
       <CollisionPairModification mEnabled="false" mEntity1="imukuppi" mEntity2="akku-laatikko" />
       <AlternativePickedObjectRefPointList> "PickPoint1", "PickPoint2" </AlternativePickedObjectRefPointList>

       <!--  DIO: DeviceType: 0 Slot: 0 Channel: 0 HighLow: eHigh Type: eDigitalOutput  -->
       <TagCommand mName="CloseGripper" mBindingPosition="eIntermediate"
mTag="IO[&quot;ControlBox&quot;].DO[0] = 1"/>
     </PickAction>

     <PlaceAction mName = "PlacePart0" mPlacedObject="TEMPLATEPART" mPlaceTo="PlasticPlaceBin"
       mPlacingRefPoint="PlasticPlacePoint" mPlacingMotion = "Approach" mDepartingPathCompletion =
"eNoDepartingPath"
       mDeviceName = "Robot" mTime = "0.2" mPlacedObjectRefPoint="PlacePointBottom">
       <AlternativePlacedObjectRefPointList> "PlacePoint1", "PlacePoint2" </AlternativePlacedObjectRefPointList>
       <StayCommand mName="Stay" mBindingPosition="eIntermediate" mWaitingTime="0.5"/>

       <!--  DIO: DeviceType: 0 Slot: 0 Channel: 0 HighLow: eHigh Type: eDigitalOutput  -->
       <TagCommand mName="CloseGripper" mBindingPosition="eIntermediate"
mTag="IO[&quot;ControlBox&quot;].DO[0] = 1"/>
     </PlaceAction>

     <ConfigurationAction mName = "EndRobotConfiguration" mDeviceName = "Robot" mTime = "0.3">
       <ConfigurationList> 0, 0, 90, 0, 90, 0 </ConfigurationList>
       <StayCommand mName="Stay" mBindingPosition="eIntermediate" mWaitingTime="1.0"/>
     </ConfigurationAction>

       <!--  Defines the folder and name of the exported program  -->
     <ProgramExporter mDeviceName = "Robot" mFileName = "./CellController/jobRobotProgramTEMPLATEPART.txt"/>
    </Workflow>
  </Job>
</root>
```

trinity ENGAGE WITH AGILE MANUFACTURING

## Setting up gripper

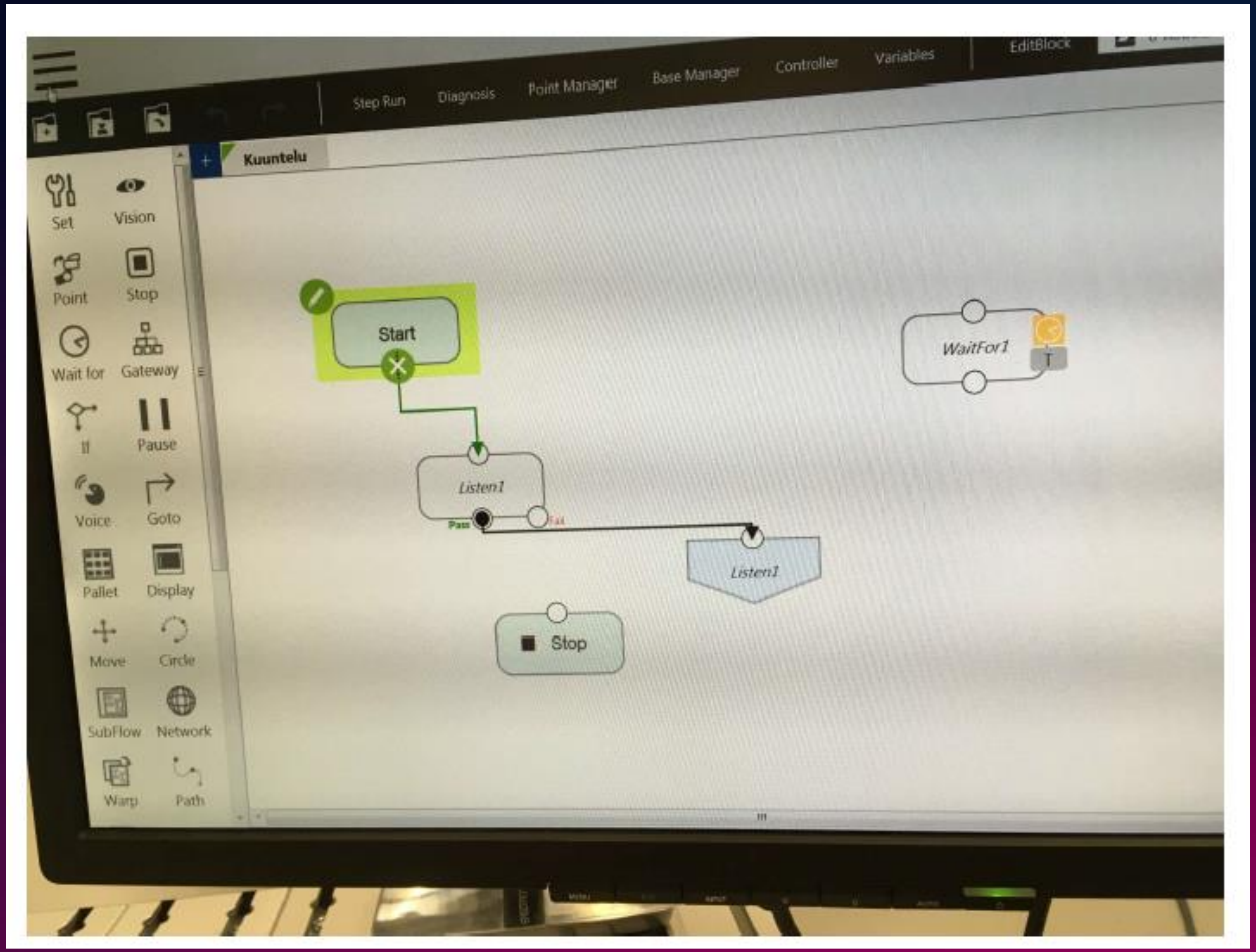

If gripper is controlled by IO commands, they can be added to action commands:

```
	<TagCommand mName="CloseGripper" mBindingPosition="eIntermediate"
mTag="IO[&quot;ControlBox&quot;].DO[0] = 1"/>
```

Where DO[0] = 1 defines the number of digital out port and following number the status of port.

## Set up robot listener

- To run AUTOMAPPPS CellController, listen node program created in TM Flow is needed

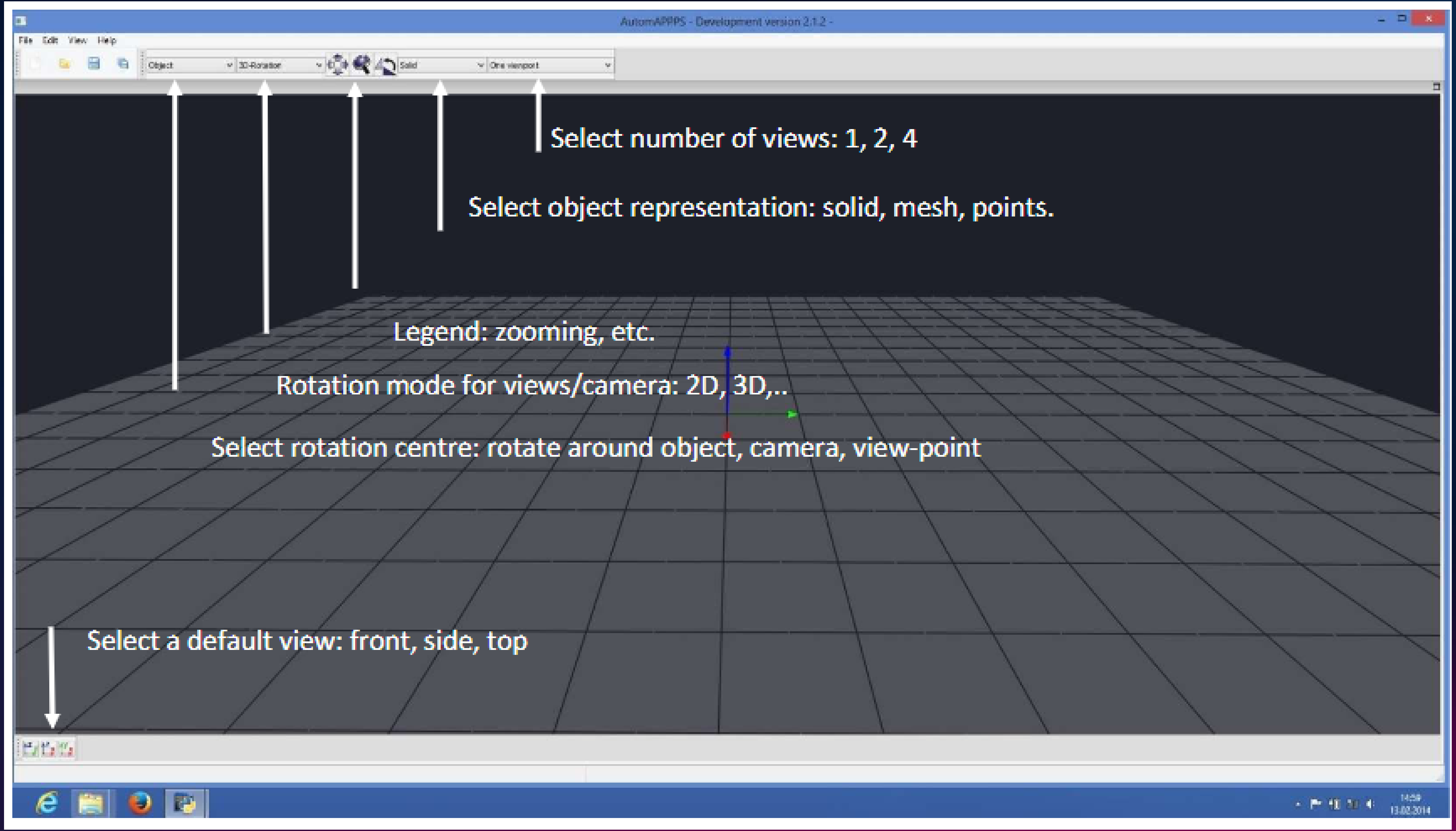


trinity ENGAGE WITH AGILE MANUFACTURING

## Starting data transfer

- To start the bin-picking, start first Listen node from TM robot
- Then vision software
- And finally CellController

# AUTOMAPPPS Basics



Select number of views: 1, 2, 4

Select object representation: solid, mesh, points.

Legend: zooming, etc.

Rotation mode for views/camera: 2D, 3D,..

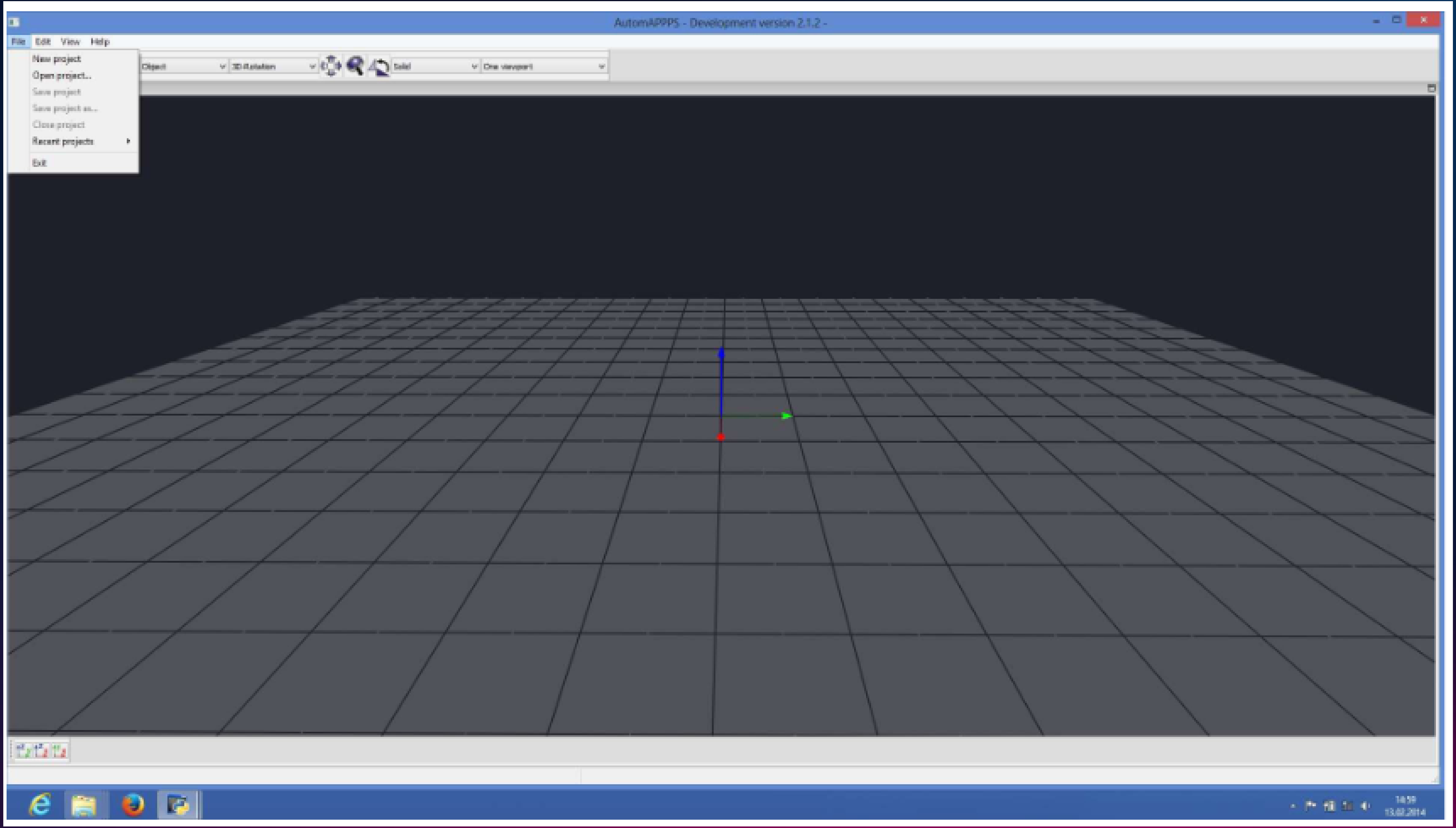
Select rotation centre: rotate around object, camera, view-point

Select a default view: front, side, top

Views can be changed with the mouse as highlighted in the legend. Tip: The same mouse movements with “CTRL” or “STRG” pressed refer to actions on the object or tool, or tool-paths.

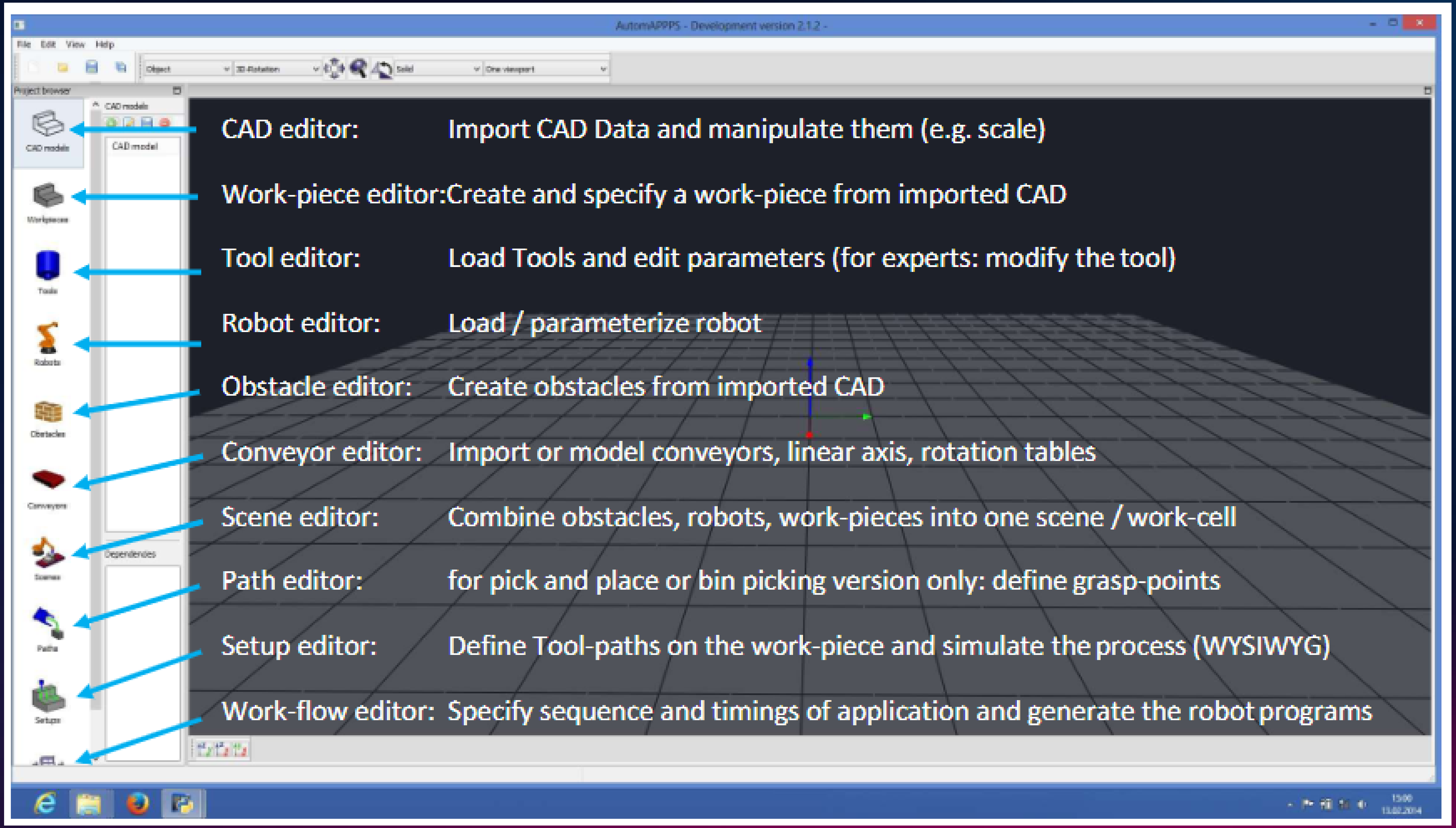
trinity ENGAGE WITH AGILE MANUFACTURING

# Create new project



trinity ENGAGE WITH AGILE MANUFACTURING

# Available editors

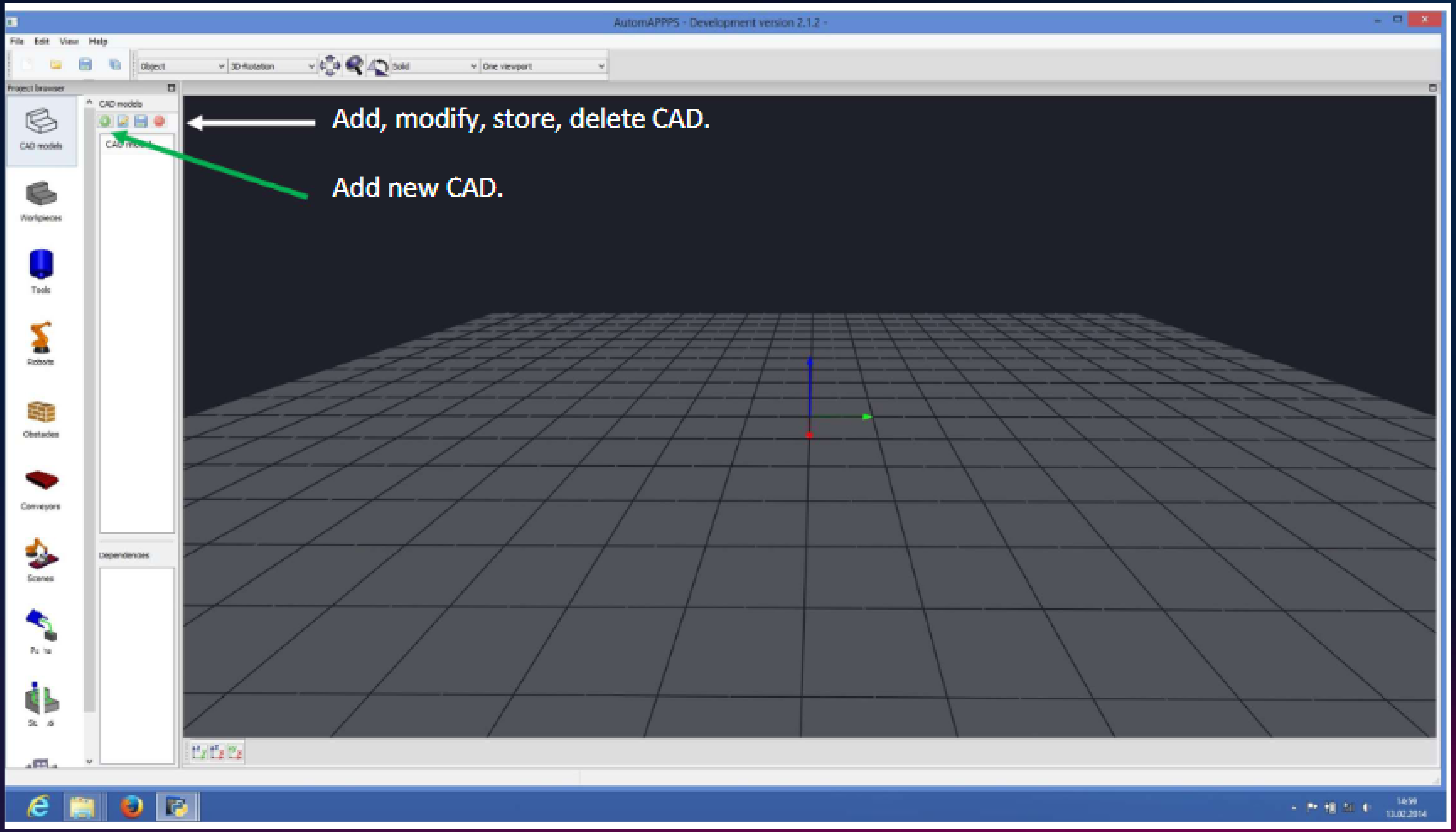


| | |
|---|---|
| CAD editor: | Import CAD Data and manipulate them (e.g. scale) |
| Work-piece editor: | Create and specify a work-piece from imported CAD |
| Tool editor: | Load Tools and edit parameters (for experts: modify the tool) |
| Robot editor: | Load / parameterize robot |
| Obstacle editor: | Create obstacles from imported CAD |
| Conveyor editor: | Import or model conveyors, linear axis, rotation tables |
| Scene editor: | Combine obstacles, robots, work-pieces into one scene / work-cell |
| Path editor: | for pick and place or bin picking version only: define grasp-points |
| Setup editor: | Define Tool-paths on the work-piece and simulate the process (WYSIWYG) |
| Work-flow editor: | Specify sequence and timings of application and generate the robot programs |

Available editors. The regular process of creating a workcell and its application is from top down. For ex- isting projects, commonly only the last 2 editors are needed for the process and the applications.

trinity ENGAGE WITH AGILE MANUFACTURING

# Launch CAD-Editor

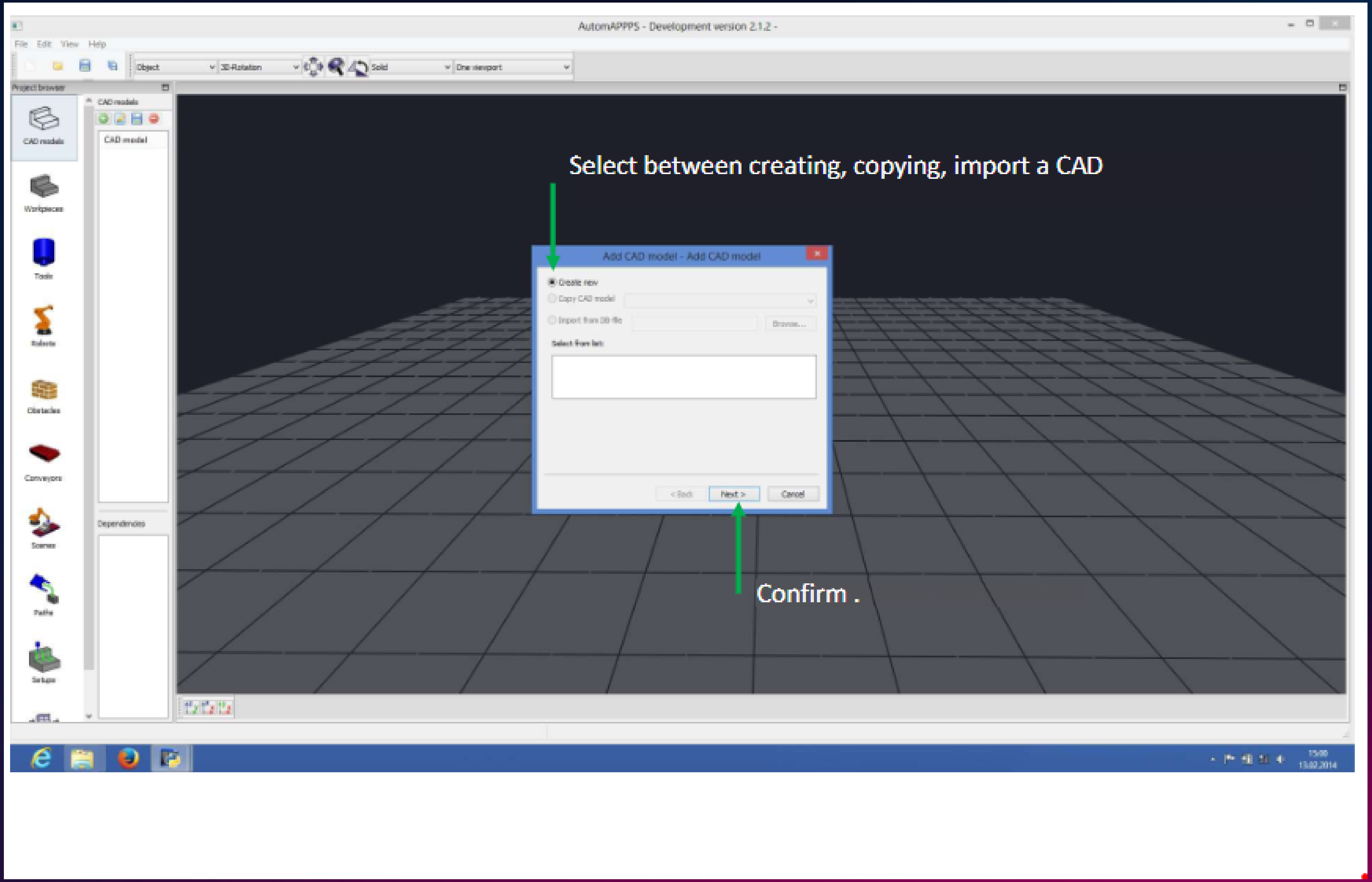


Add, modify, store, delete CAD.

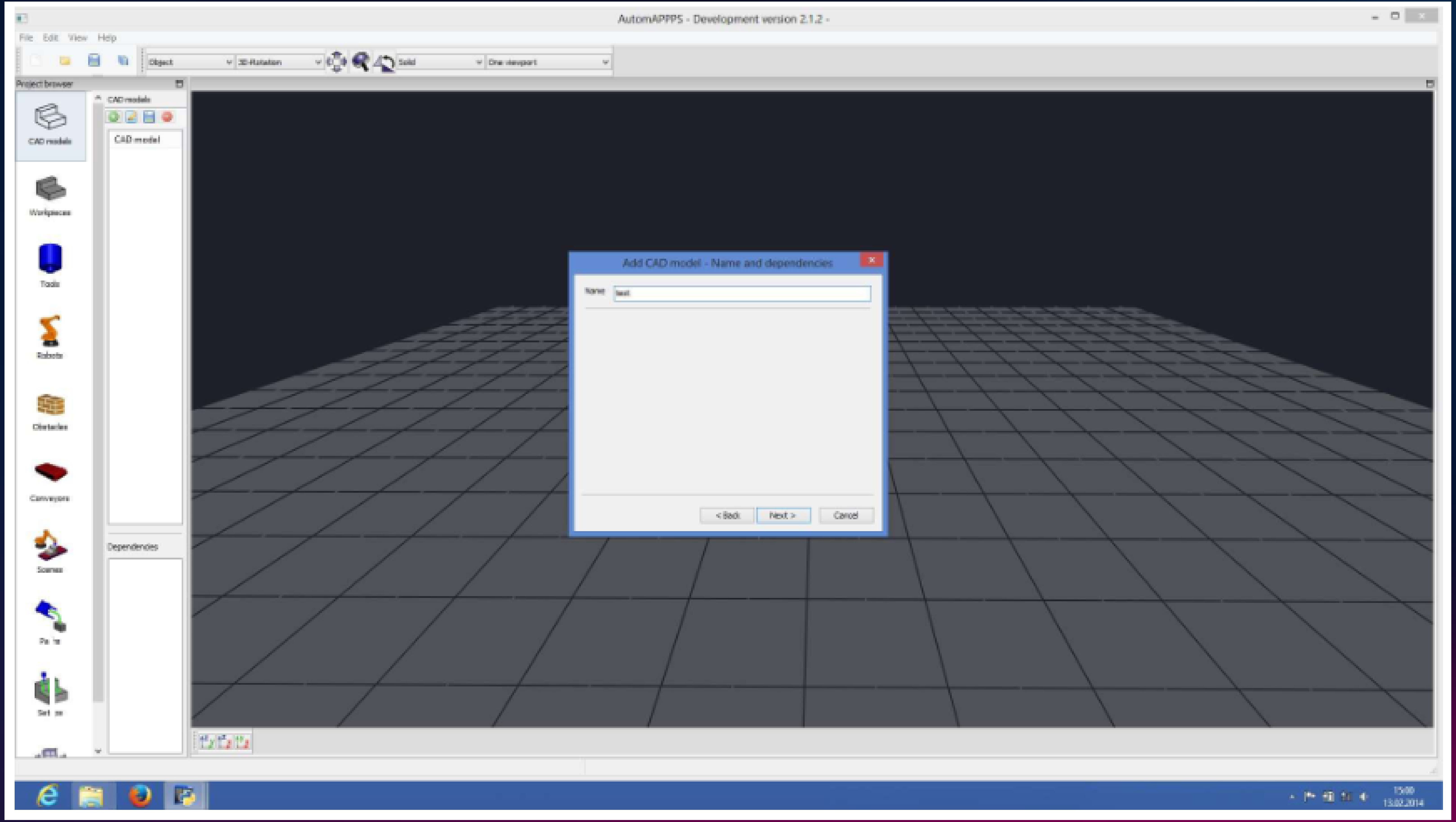
Add new CAD.

Select: ADD new CAD (left click).

# Import CAD-data



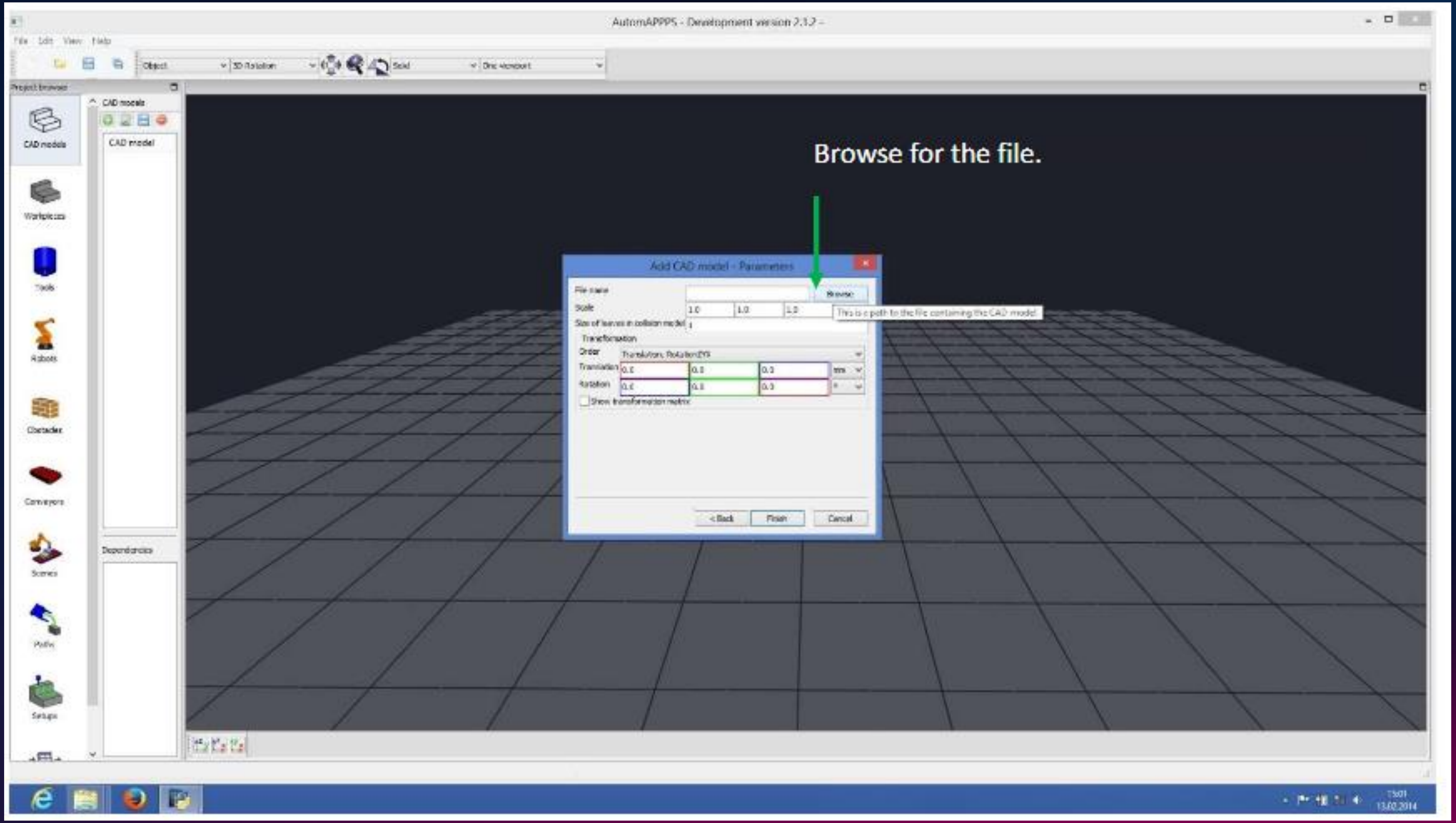
# Import CAD-data



Name: test

Assign a name and confirm with “next”.

# Import CAD-data



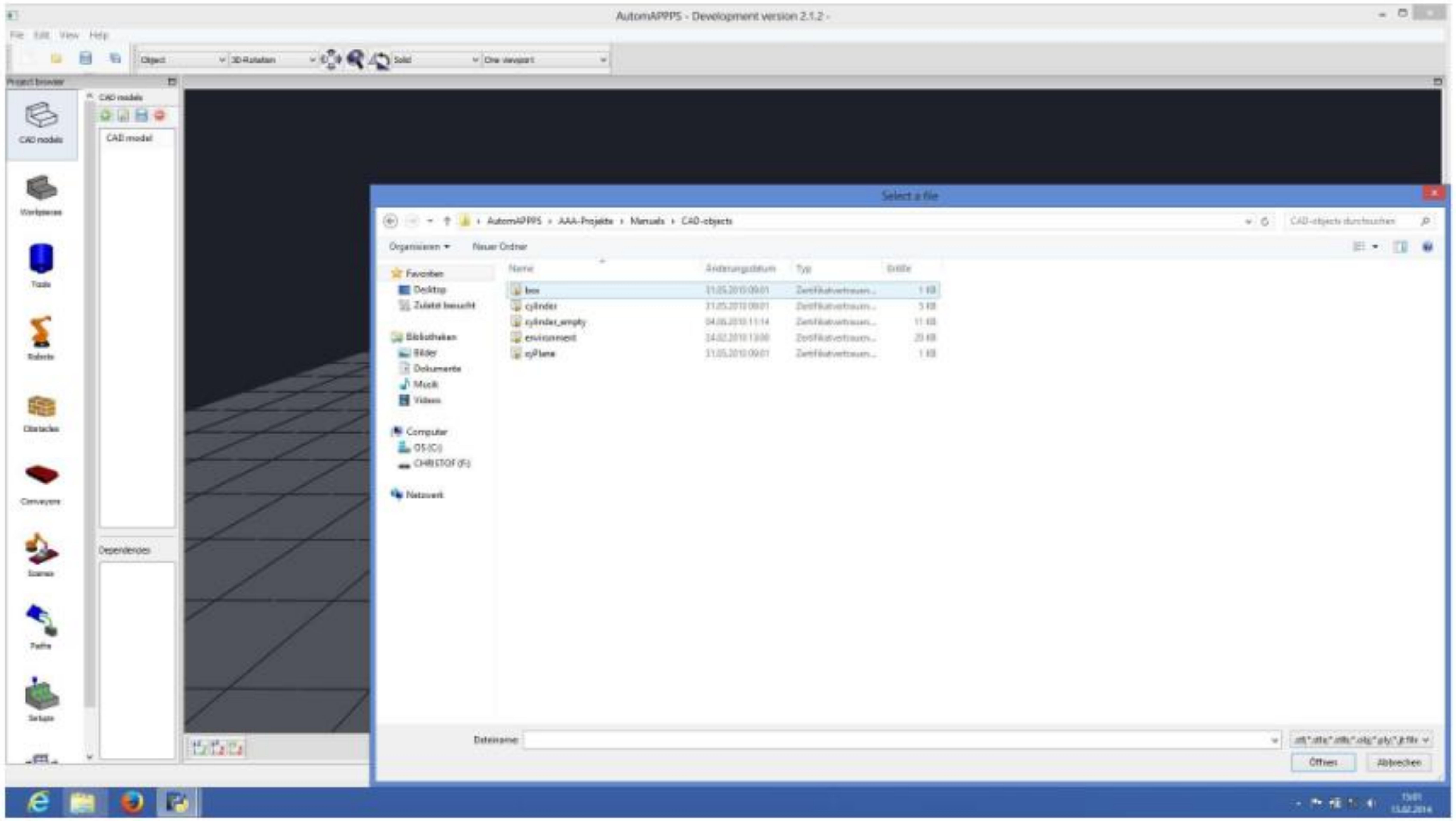
Browse for the file.

File name:
Scale: 1.0 1.0 1.0
Size of leaves in collision model: 1
Transformation
Order: Translation, RotationZYX
Translation: 0.0 0.0 0.0 mm
Rotation: 0.0 0.0 0.0 °
Show transformation matrix

This is a path to the file containing the CAD model

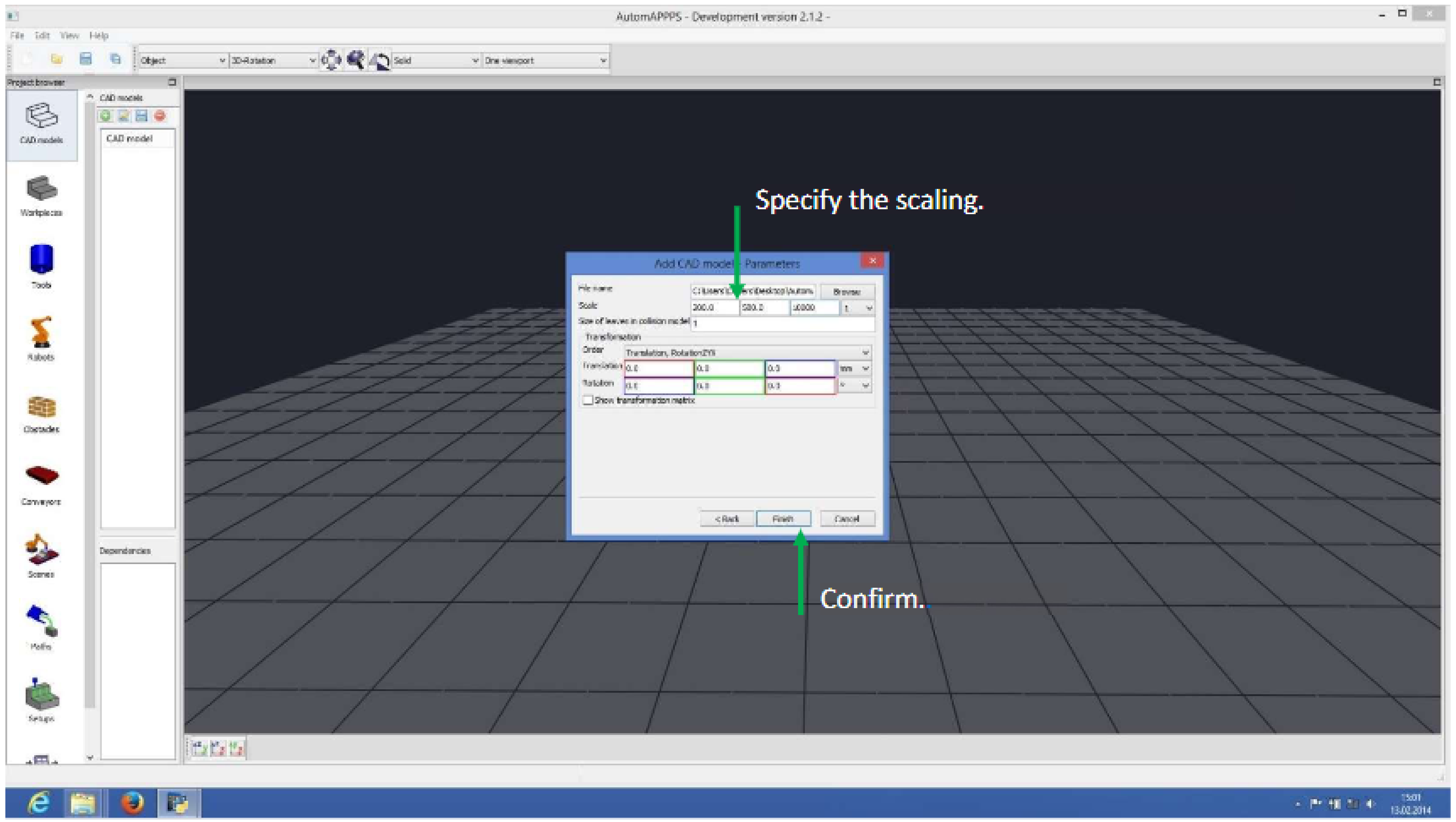
In case of loading a new CAD: browse for the file.

# Import CAD-data



Step 4: browse for the CAD file.

# Import CAD-data



Add CAD model - Parameters

File name: C:\Users\[illegible]\Desktop\Autom... [Browse]
Scale: 200.0 | 500.0 | 10000 | 1
Size of leaves in collision model: 1

Transformation
Order: Translation, RotationZYX
Translation: 0.0 | 0.0 | 0.0 | mm
Rotation: 0.0 | 0.0 | 0.0 | °
Show transformation matrix

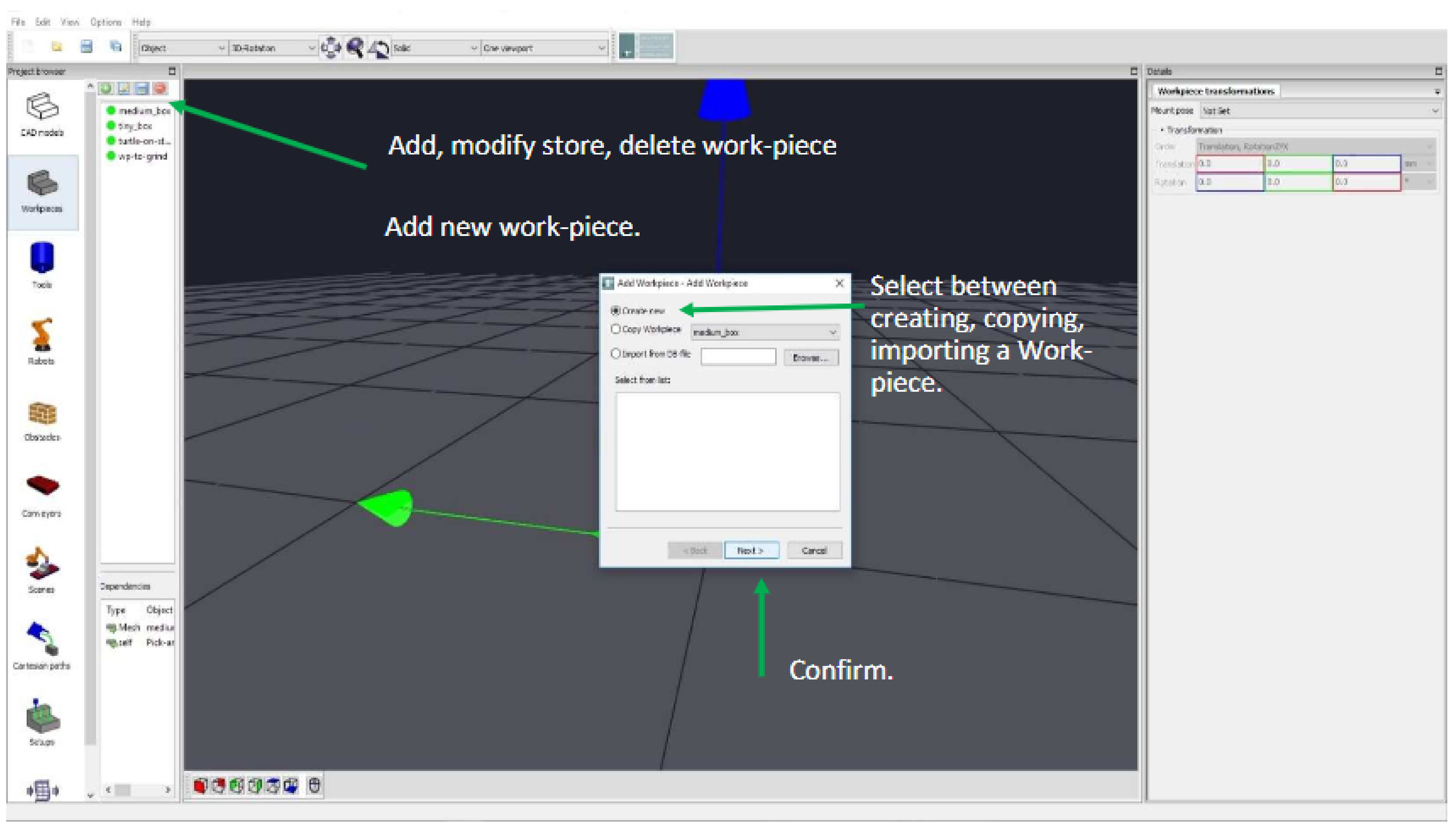
< Back | Finish | Cancel

Step 5: scale the CAD if needed. In this example, a standard box (1mm,1mm,1mm) was loaded and is scaled to get the desired shape of (200mm, 500mm, 10000mm). Confirm with Finish.

Tip: Also the CAD can be translated and rotated. However, we suggest to do this the scene editor when the instance is used (1 or n time).

trinity ENGAGE WITH AGILE MANUFACTURING

# Define Workpiece



Add, modify store, delete work-piece

Add new work-piece.

Add Workpiece - Add Workpiece

(● Create new) ○ Copy Workpiece: medium_box ○ Import from DB file: Browse...

Select from list:

< Back | Next > | Cancel

Select between creating, copying, importing a Work-piece.

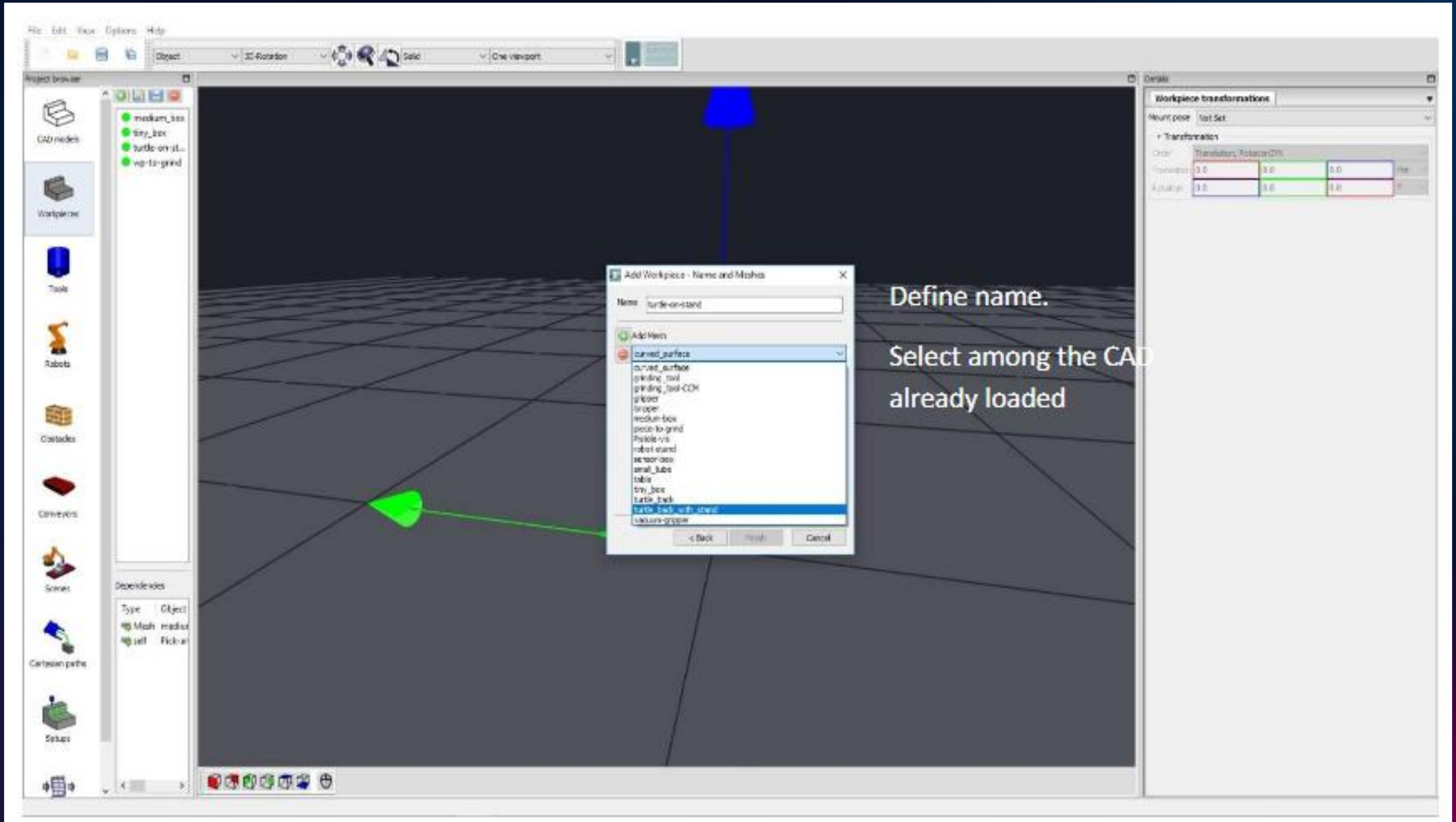
Confirm.

Step 0: the CAD of the work-piece was loaded in the CAD editor already (see before: CAD-Editor).

Step 1: create a new work-piece. Alternatives would be to copy one already existing in the same project/database or to import from another project / database.

Note: Each work-piece (type) can be used multiple times in the scene.

# Define Workpiece



Add Workpiece - Name and Meshes

Name: turtle-on-stand

Add Mesh

curved_surface

- curved_surface
- grinding_tool
- grinding_tool-COM
- gripper
- gripper
- medium-box
- piece-to-grind
- Pistole-vis
- robot-stand
- sensor-box
- small_tube
- table
- tiny_box
- turtle_back
- turtle_back_with_stand
- vacuum-gripper

< Back | Finish | Cancel

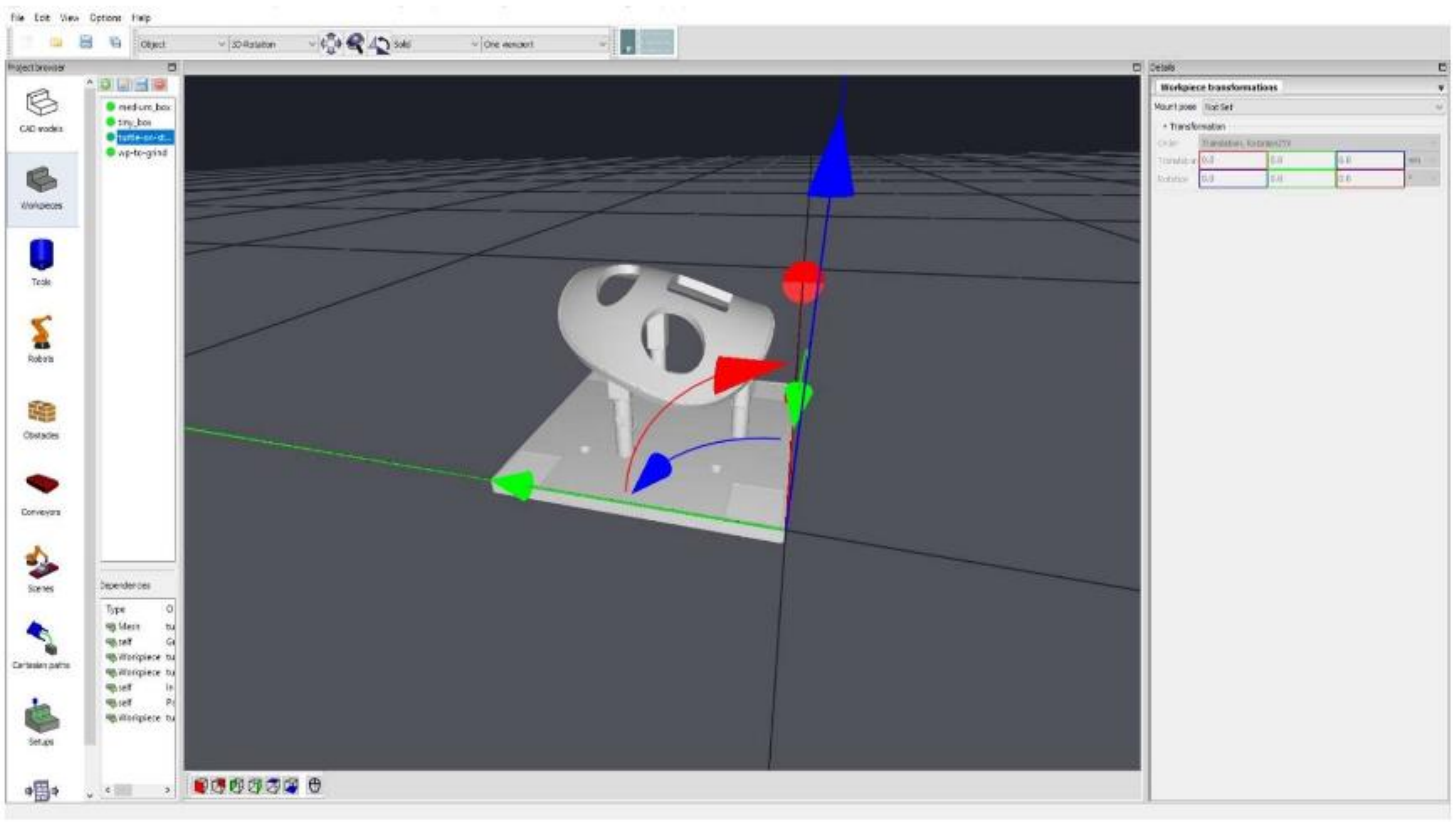
Define name.

Select among the CAD already loaded

Step 2: select the relevant CAD (loaded in the CAD Editor)

trinity ENGAGE WITH AGILE MANUFACTURING

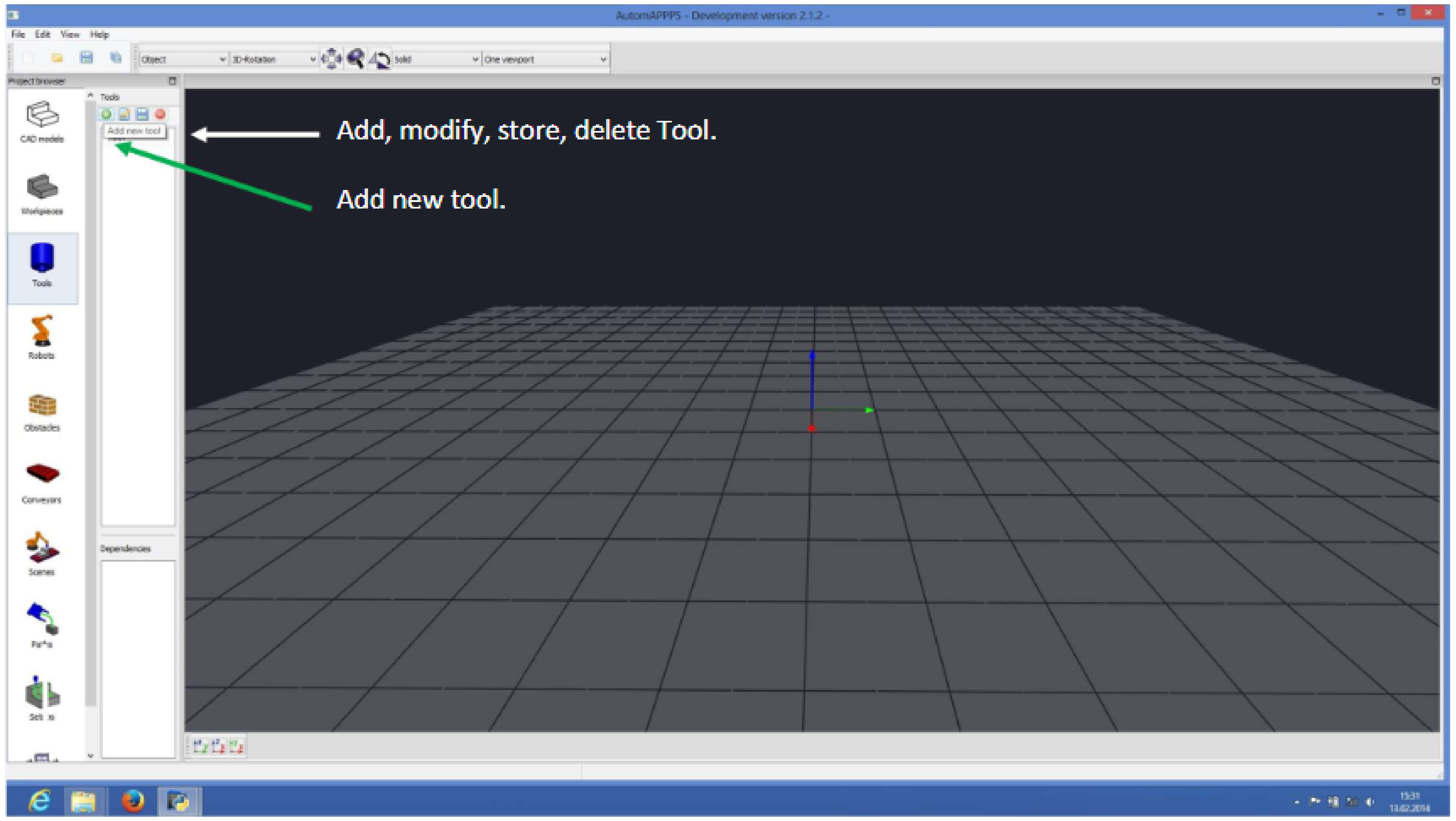
# Define Workpiece



Step 3: Work-piece created.

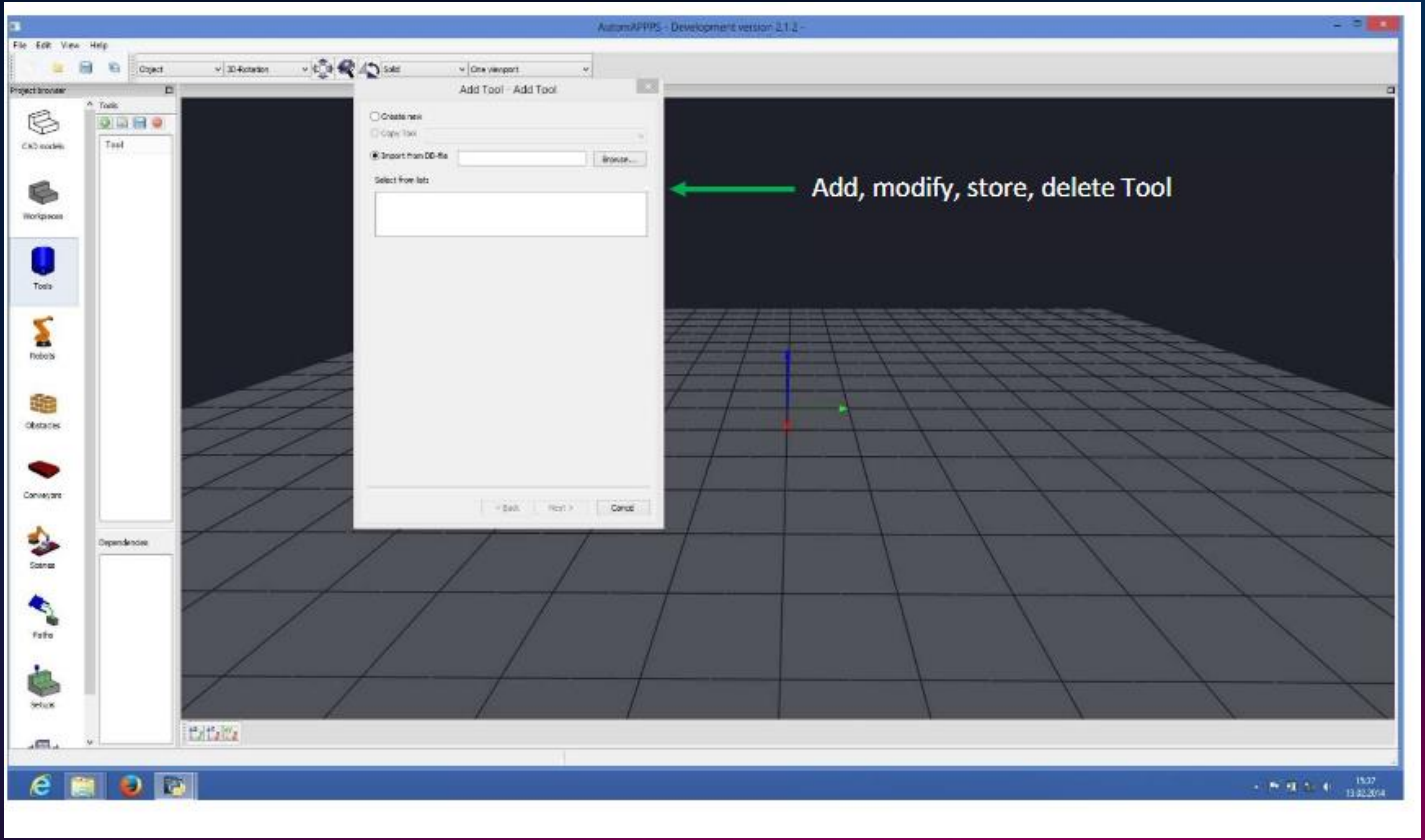
trinity ENGAGE WITH AGILE MANUFACTURING

# Define Tool



Tool Editor: Note: Creating a completely new tool is beyond scope of this guide. Also how the user can (easily) add a 3rd or 4th, camera to a inspection system is very specific and has nothing in common with how to include user-preference in micro-cleaning. Thus, for different processes (tools) supported and imported from an own Database there exist a own description, while this one is only presenting the basics.

## Define Tool



AutomAPPPS - Development version 2.1.2 -

Add Tool - Add Tool

( ) Create new

( ) Copy Tool

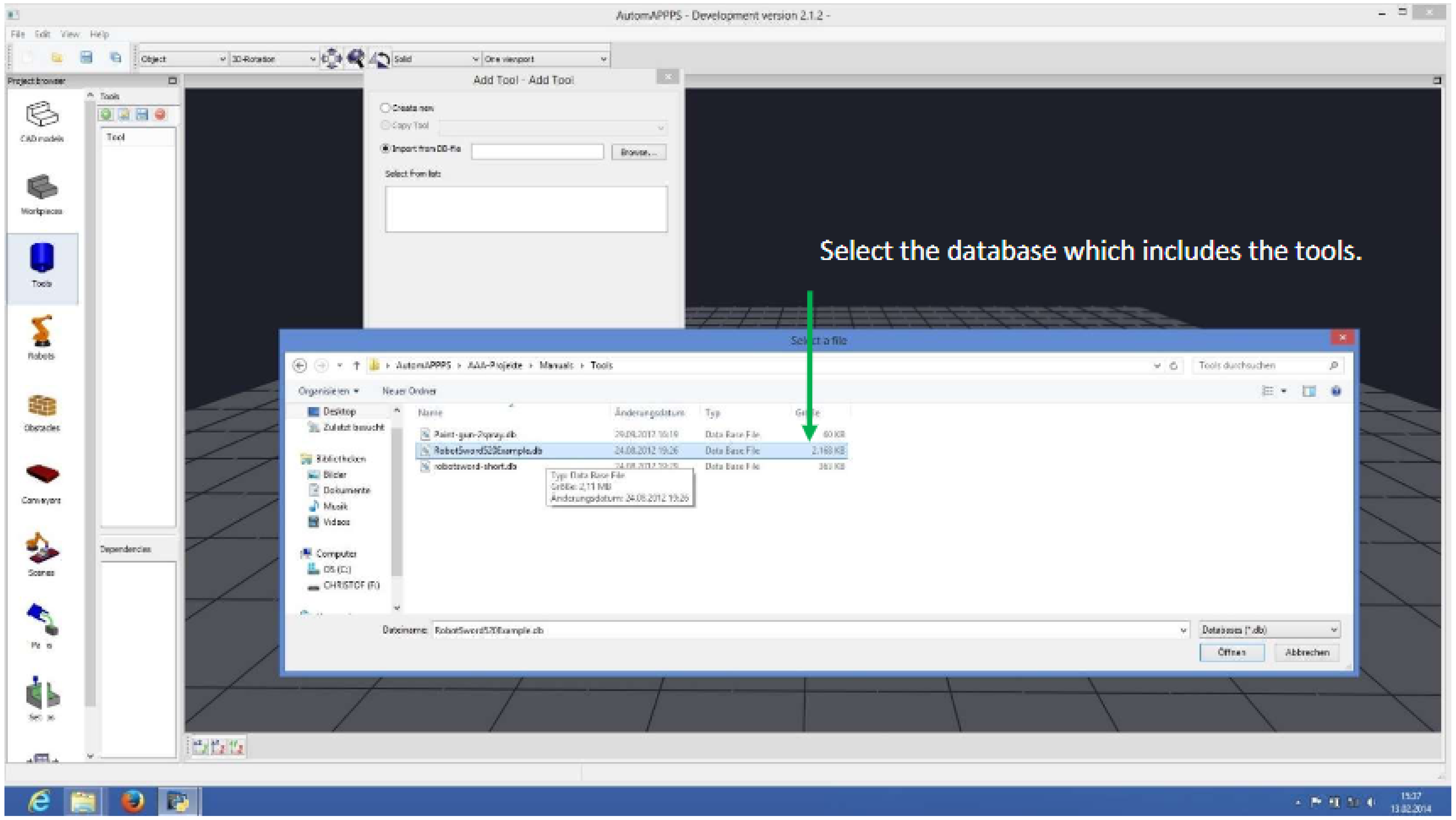
(●) Import from DB-file  Browse...

Select from list:

< Back  Next >  Cancel

Add, modify, store, delete Tool

# Define Tool



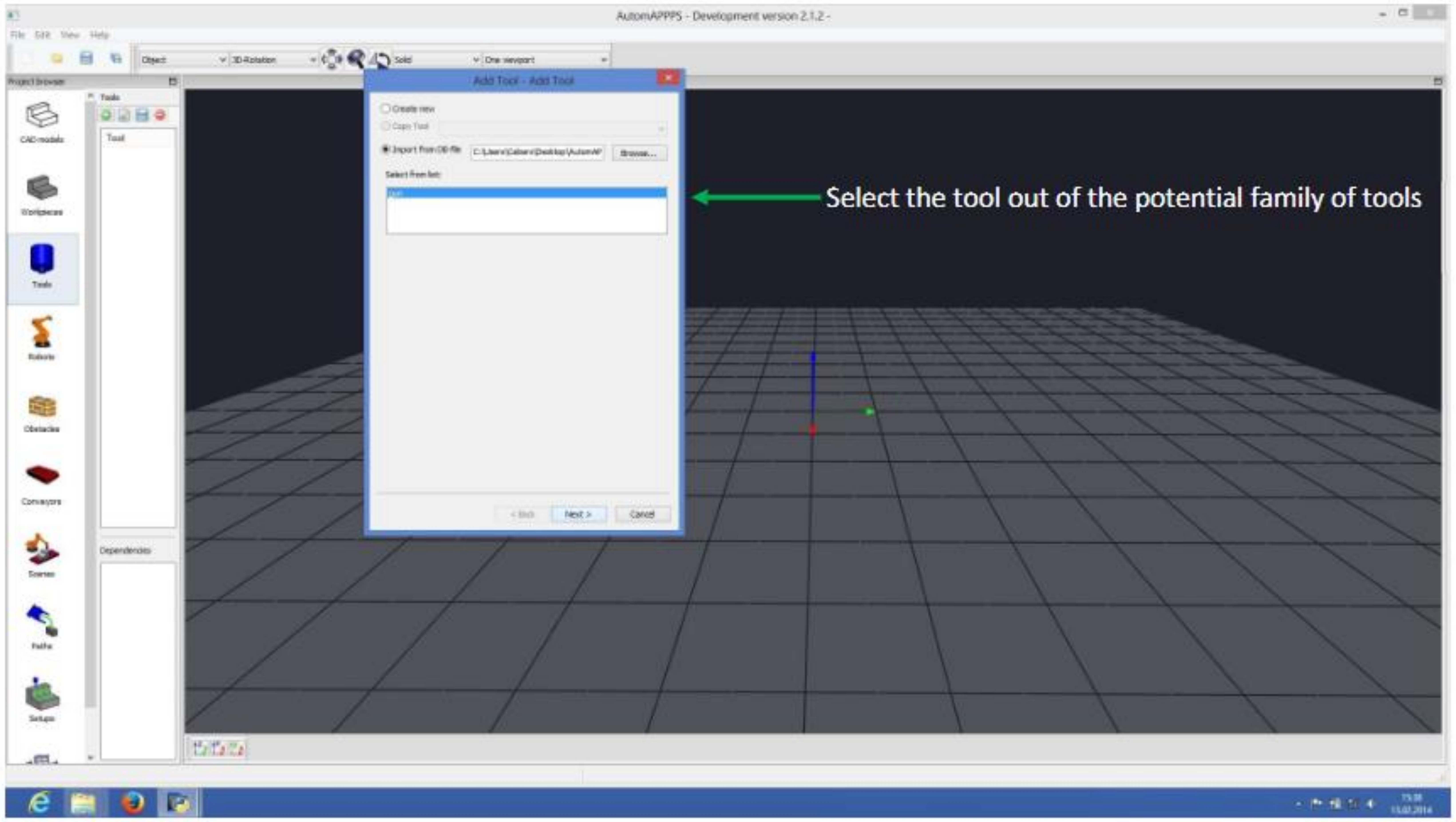
Select the database which includes the tools.

```
Name                       Änderungsdatum     Typ             Größe
Paint-gun-2spray.db        29.08.2012 16:19   Data Base File     60 KB
RobotSword520Example.db    24.08.2012 19:26   Data Base File  2.168 KB
robotsword-short.db        24.08.2012 19:26   Data Base File    363 KB

Typ: Data Base File
Größe: 2,11 MB
Änderungsdatum: 24.08.2012 19:26

Dateiname: RobotSword520Example.db      Databases (*.db)
```

Step 2: select the database (provided) in which the group or family of tools and simulation models are included.

trinity ENGAGE WITH AGILE MANUFACTURING

# Define Tool



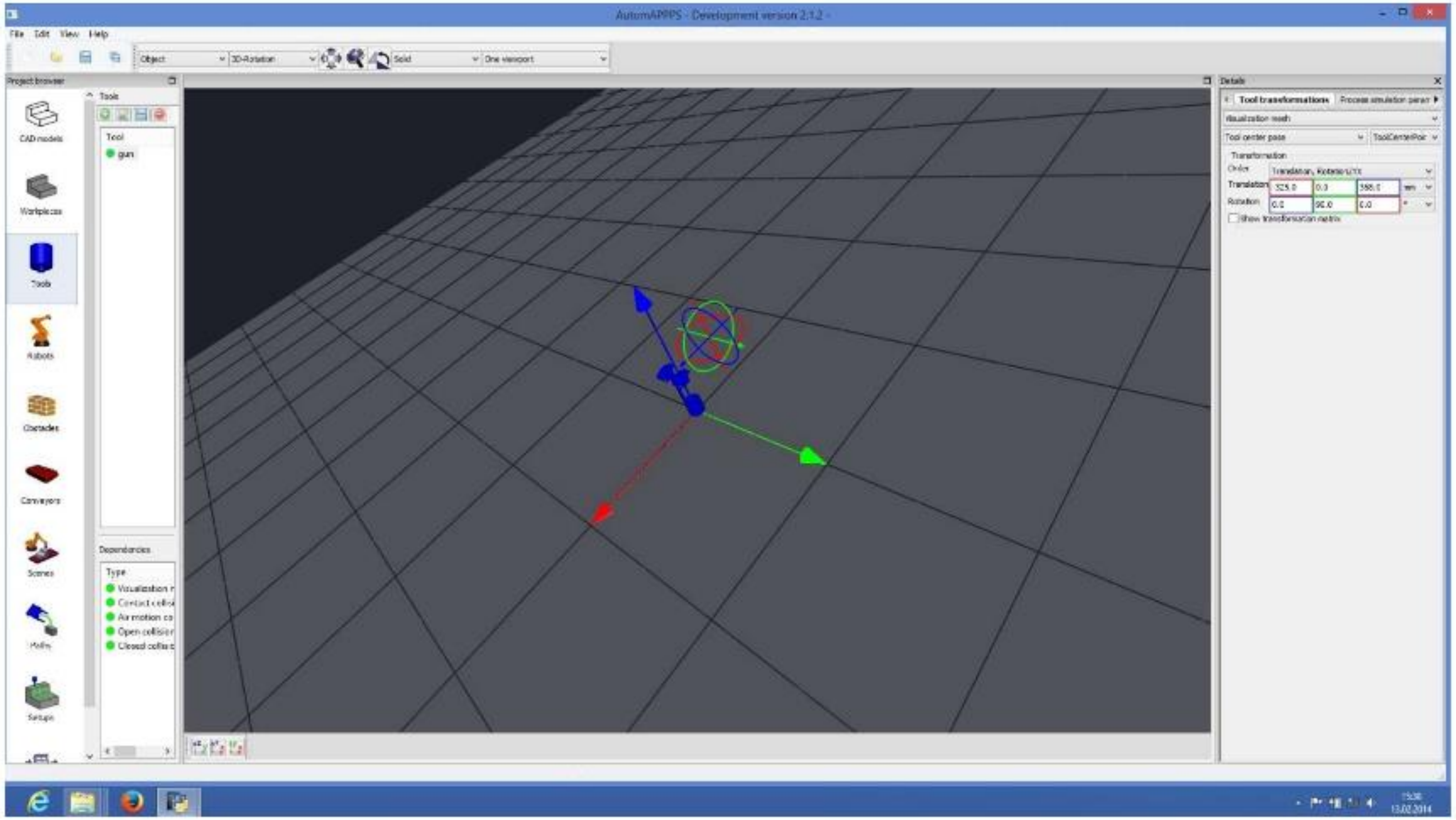
Import from DB file: C:\Users\Cabani\Desktop\AutomAP

Select the tool out of the potential family of tools

Step 3: Select the individual tool out of the family or group of modelled tools inside the database / DB file.

ENGAGE WITH
AGILE MANUFACTURING

## Define Tool



Visualization mesh

Tool center pose | ToolCenterPoi

Transformation

Order: Translation, Rotation z/y/x

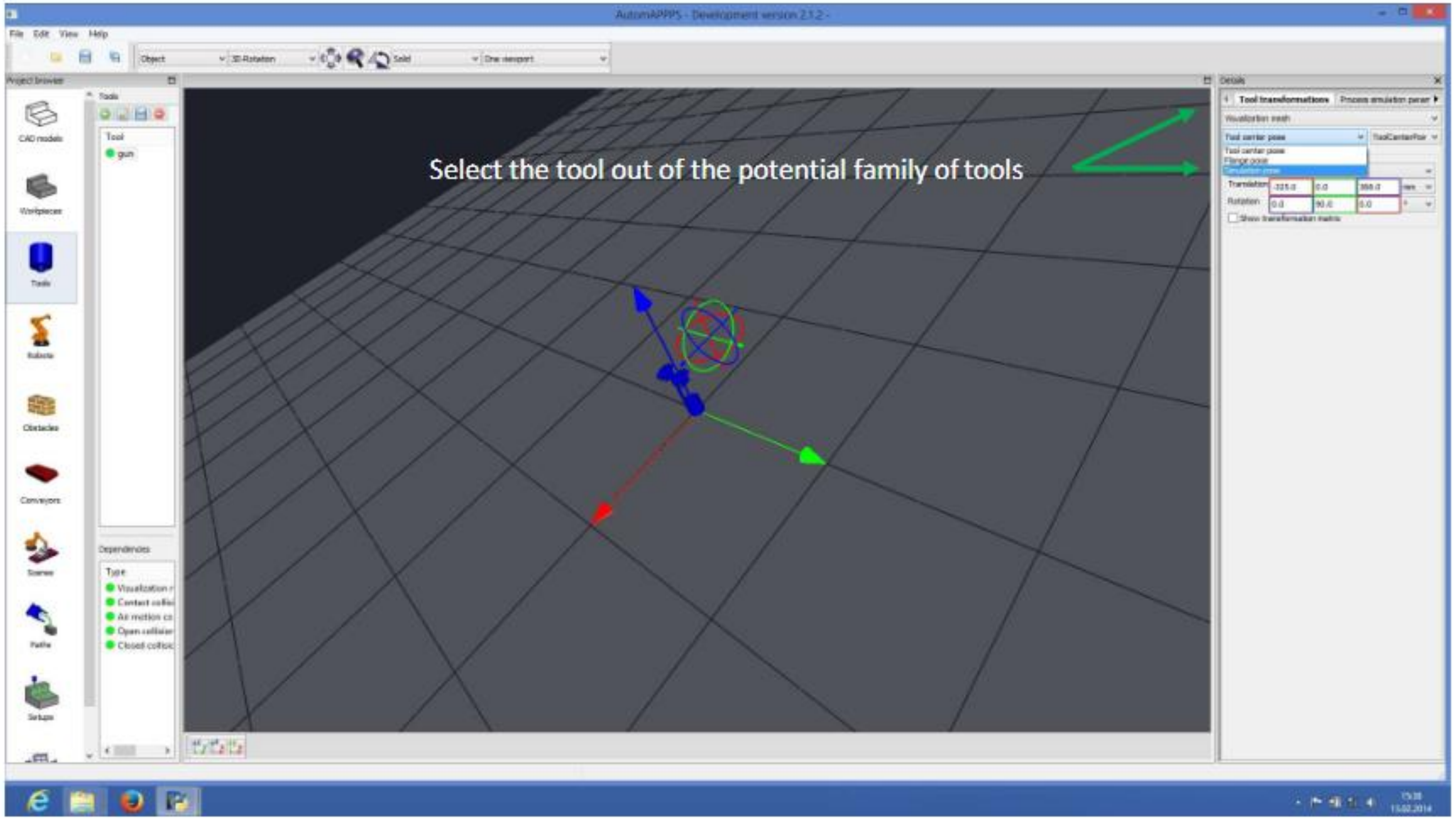
Translation: 325.0 | 0.0 | 398.0 | mm

Rotation: 0.0 | 90.0 | 0.0 | °

Show transformation matrix

Step 4: Imported tool.

# Define Tool

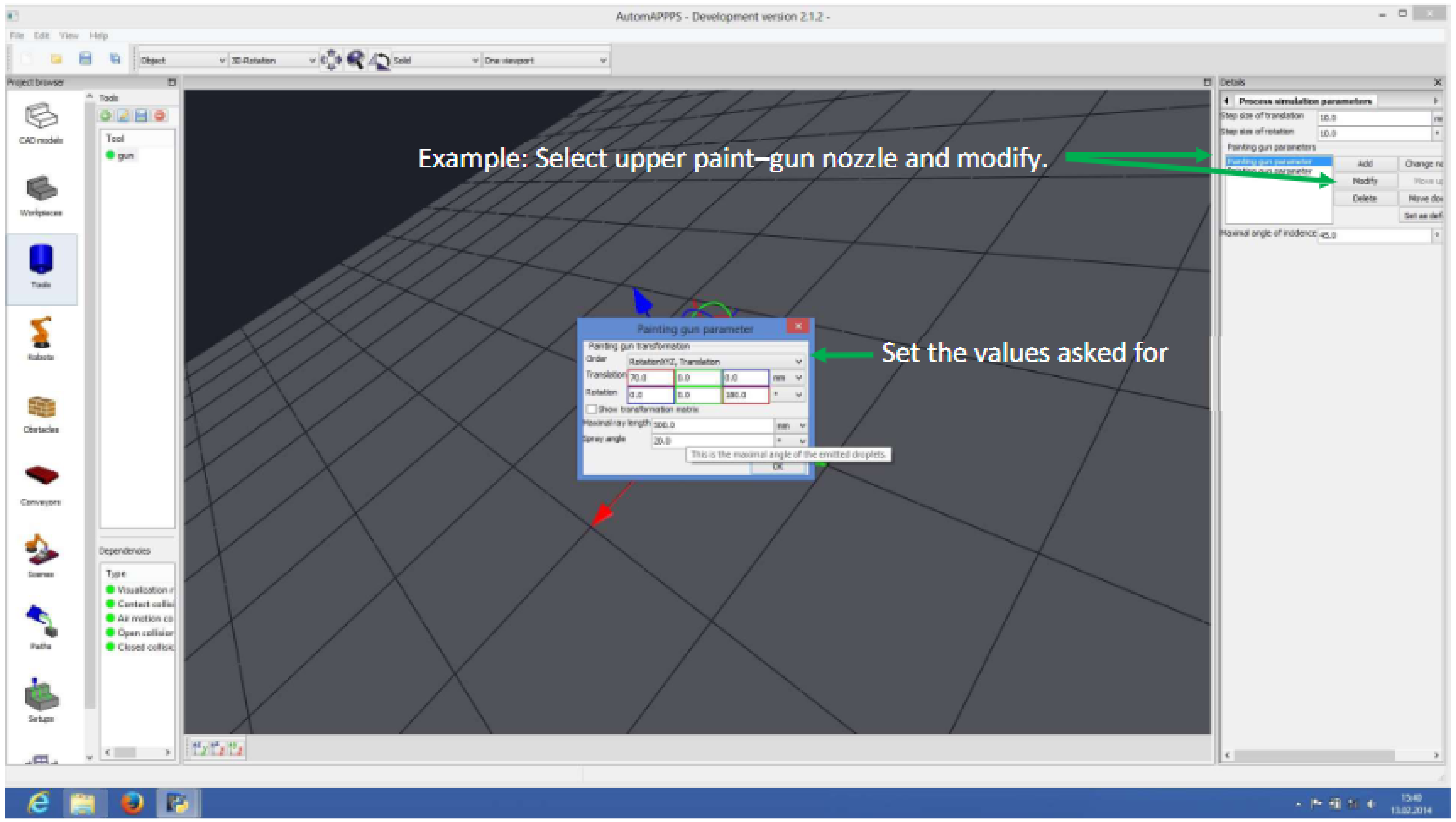


Select the tool out of the potential family of tools

Translation: -325.0 | 0.0 | 398.0 mm
Rotation: 0.0 | 90.0 | 0.0 °

Step 5: Optionally change the different transformations (TCP, Flange Pose, Simulation Pose)

# Define Tool



Example: Select upper paint–gun nozzle and modify.

Set the values asked for

Process simulation parameters

| | |
|---|---|
| Step size of translation | 10.0 |
| Step size of rotation | 10.0 |
| Maximal angle of incidence | 45.0 |

Painting gun parameters
- Painting gun parameter
- Painting gun parameter

Painting gun parameter

Painting gun transformation

| | | | | |
|---|---|---|---|---|
| Order | RotationXYZ, Translation | | | |
| Translation | 70.0 | 0.0 | 0.0 | mm |
| Rotation | 0.0 | 0.0 | 180.0 | ° |

☐ Show transformation matrix

| | | |
|---|---|---|
| Maximal ray length | 300.0 | mm |
| Spray angle | 20.0 | ° |

This is the maximal angle of the emitted droplets.

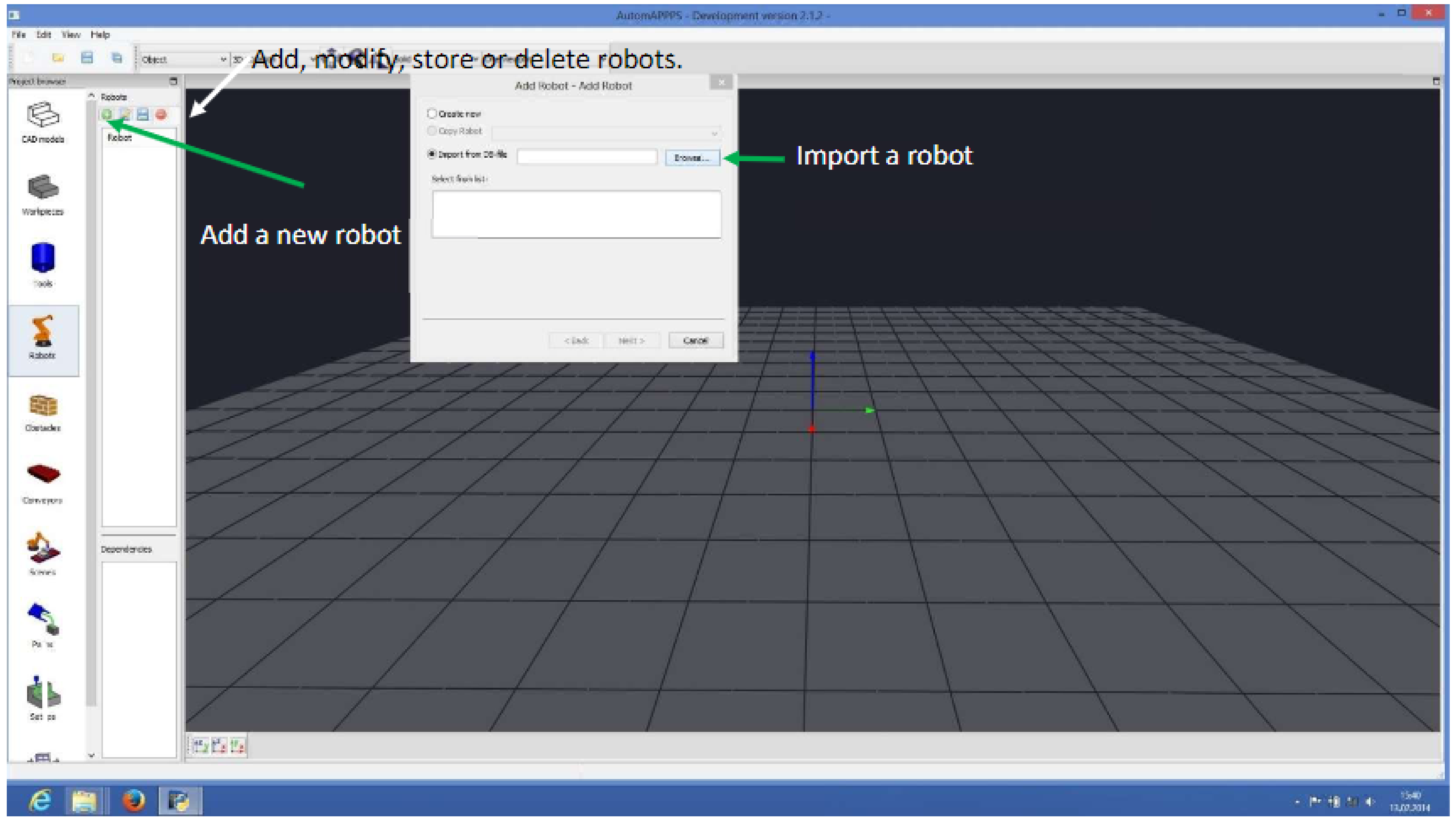
Step 8: Optionally modify the process simulation parameters:

Select if any further spray-nozzle shall be added (ADD) or if one (and which) of the two shall be modified (Note, the tool is a twin spray-gun).

Tip: the color coding of the transformation is the same as the arrows in the image.

trinity ENGAGE WITH AGILE MANUFACTURING

# Define Robot



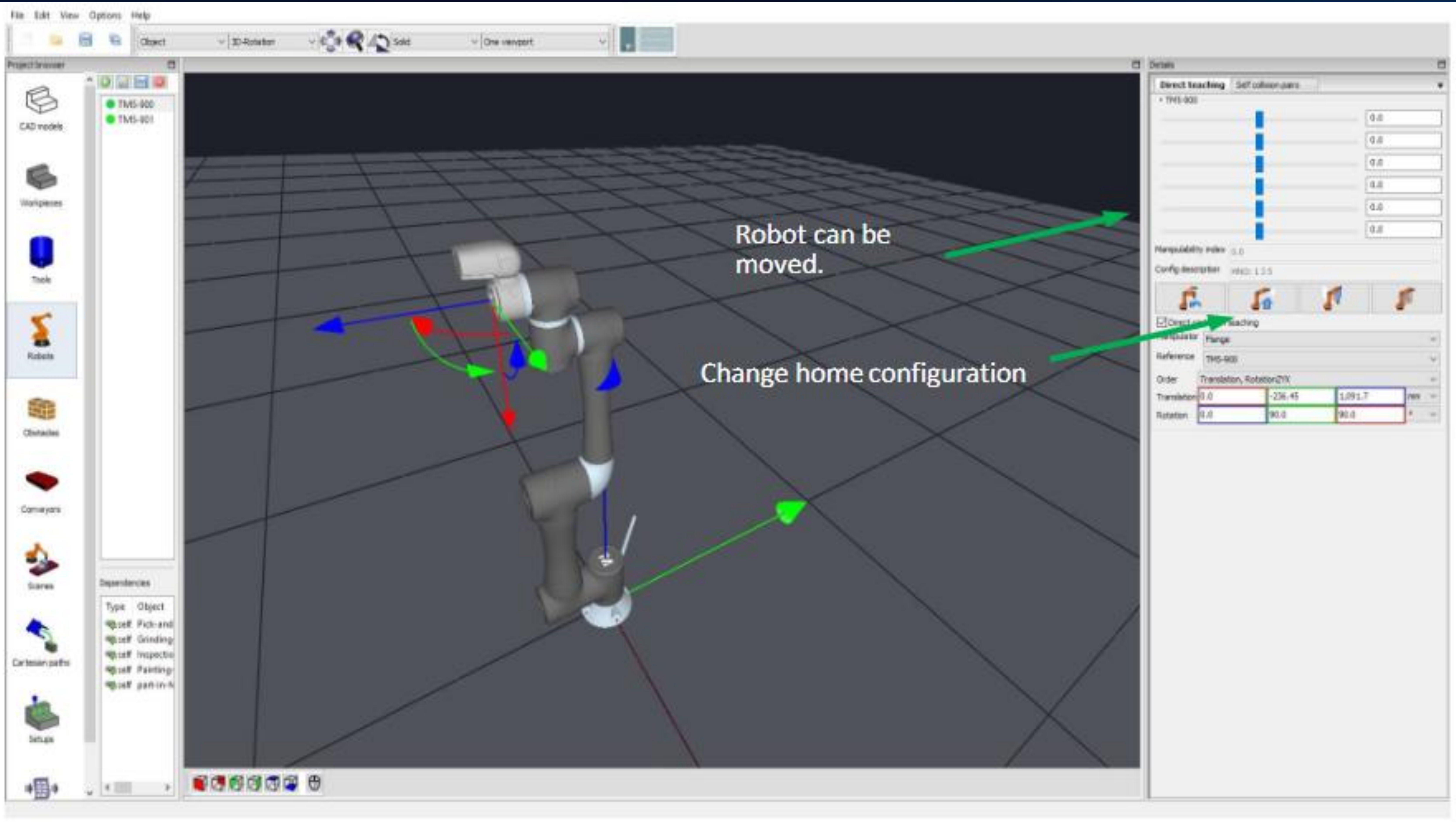
Step 0: Add a new robot (type). Step 1: Call “Import robot. Note: importing robots is the common way to extend the types that are available. To create a variance, already im- ported types can be copied and other constraints be added.

Step 2: Browse for the robot family or group (database) which includes the robot types wanted.

Step 3: Select the robot among the group or family of robots that is proposed after opining the robot .db

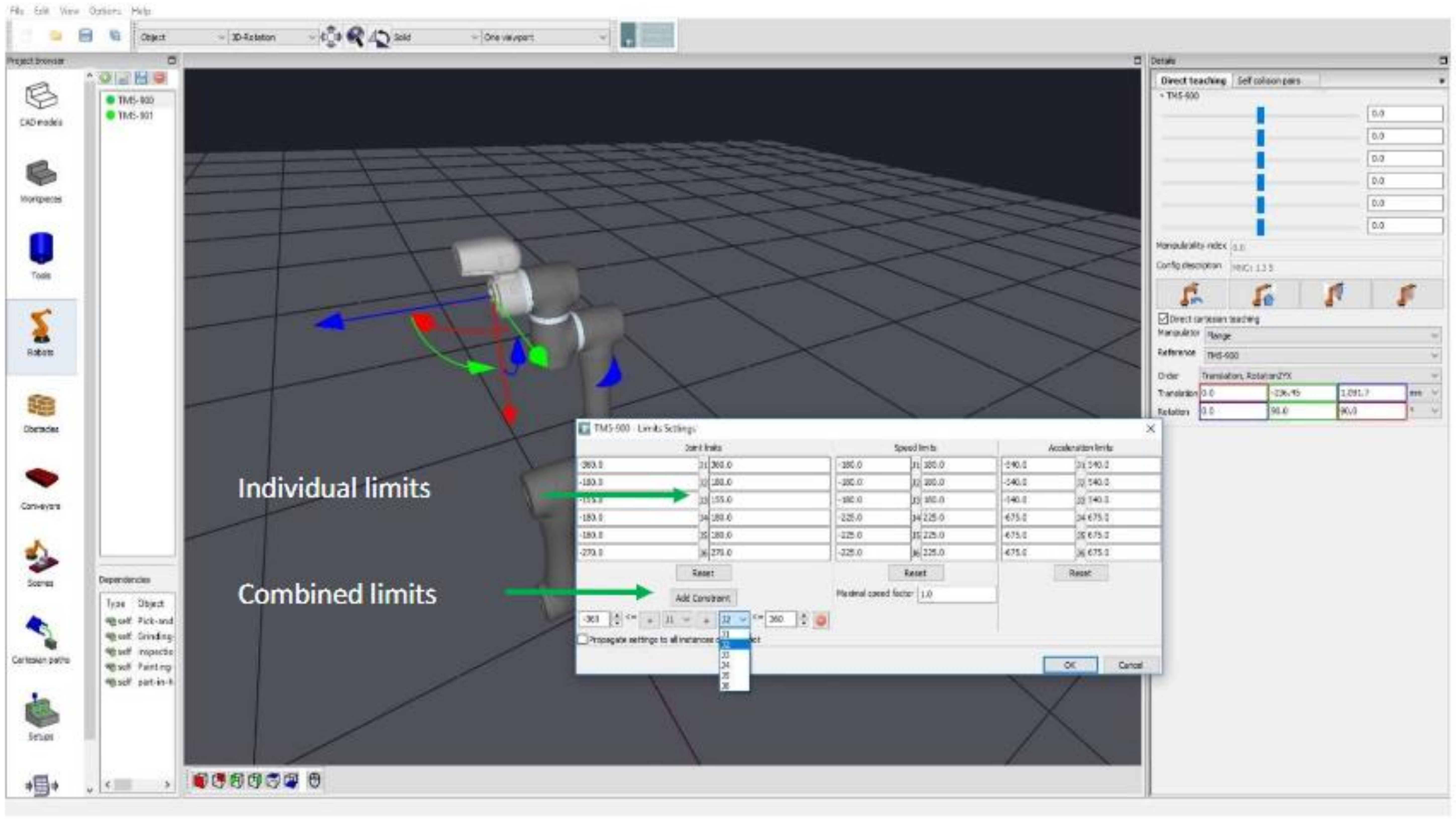
Step 4: Assign a name to the imported robot class

# Define Robot



Step 5: Optionally change the (predefined) HOME configuration

# Define Robot



TM5-900 - Limits Settings

| Joint limits | | | Speed limits | | | Acceleration limits | | |
|---|---|---|---|---|---|---|---|---|
| -360.0 | J1 | 360.0 | -180.0 | J1 | 180.0 | -540.0 | J1 | 540.0 |
| -180.0 | J2 | 180.0 | -180.0 | J2 | 180.0 | -540.0 | J2 | 540.0 |
| -155.0 | J3 | 155.0 | -180.0 | J3 | 180.0 | -540.0 | J3 | 540.0 |
| -180.0 | J4 | 180.0 | -225.0 | J4 | 225.0 | -675.0 | J4 | 675.0 |
| -180.0 | J5 | 180.0 | -225.0 | J5 | 225.0 | -675.0 | J5 | 675.0 |
| -270.0 | J6 | 270.0 | -225.0 | J6 | 225.0 | -675.0 | J6 | 675.0 |

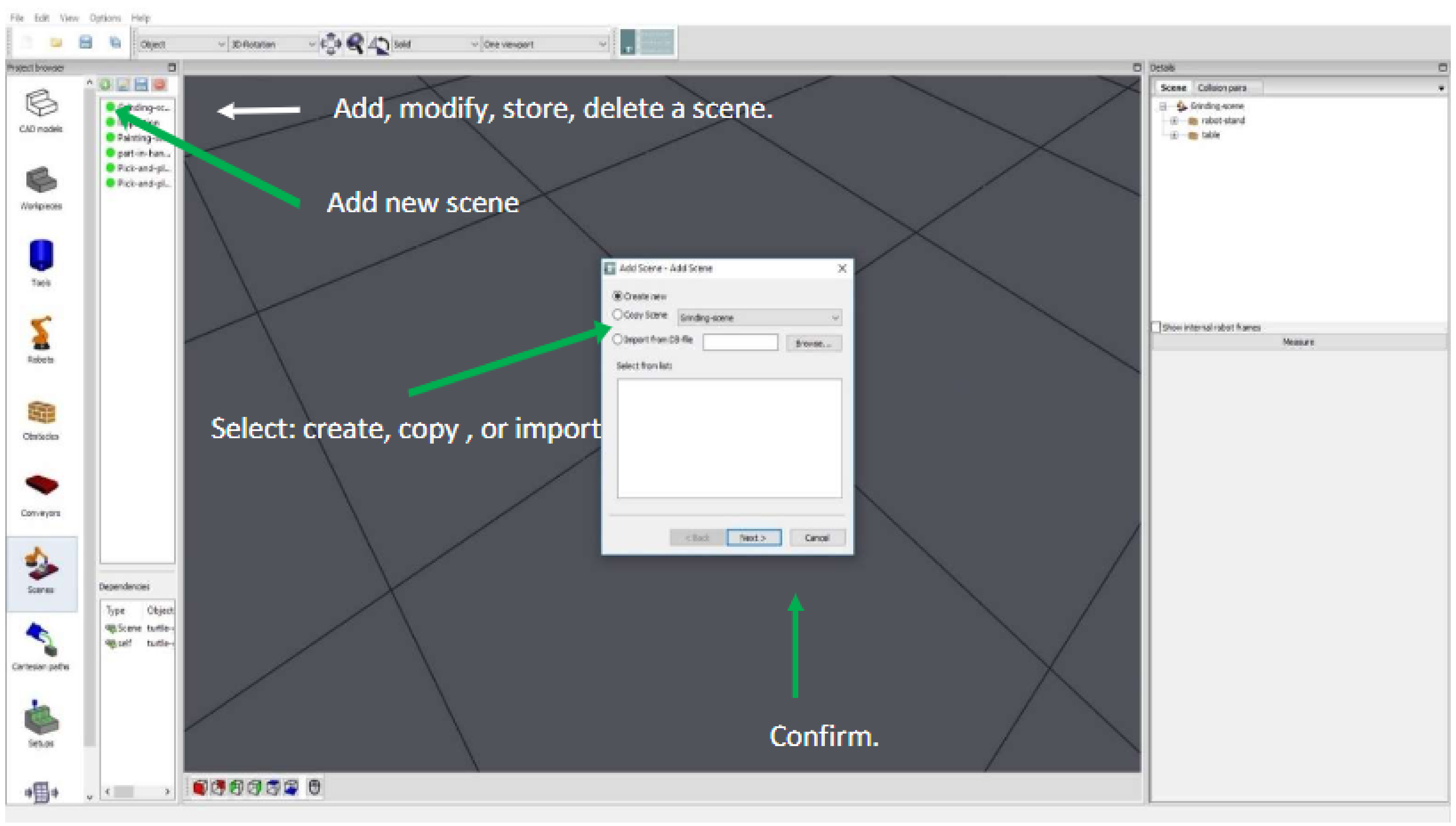
Maximal speed factor: 1.0

-360 <= + J1 + J2 <= 360

Propagate settings to all instances of object

Step 7: Optionally limit the individual joints in order to reduce allowed speed or to limit the range of motion (cables, free space). Additional constraints allow to include coupled constraints—common is e.g. to limit the combined rotation of axis 4 and 6 to protect cables. Tip: While walls are avoided automatically, reducing the range of the joints to not get close to them (as done in conventional robotics) reduces the time needed for collision free motion planning in the workflow.

# Define Scene



Add Scene - Add Scene

[✓ Create new] Copy Scene Grinding-scene Import from DB-file

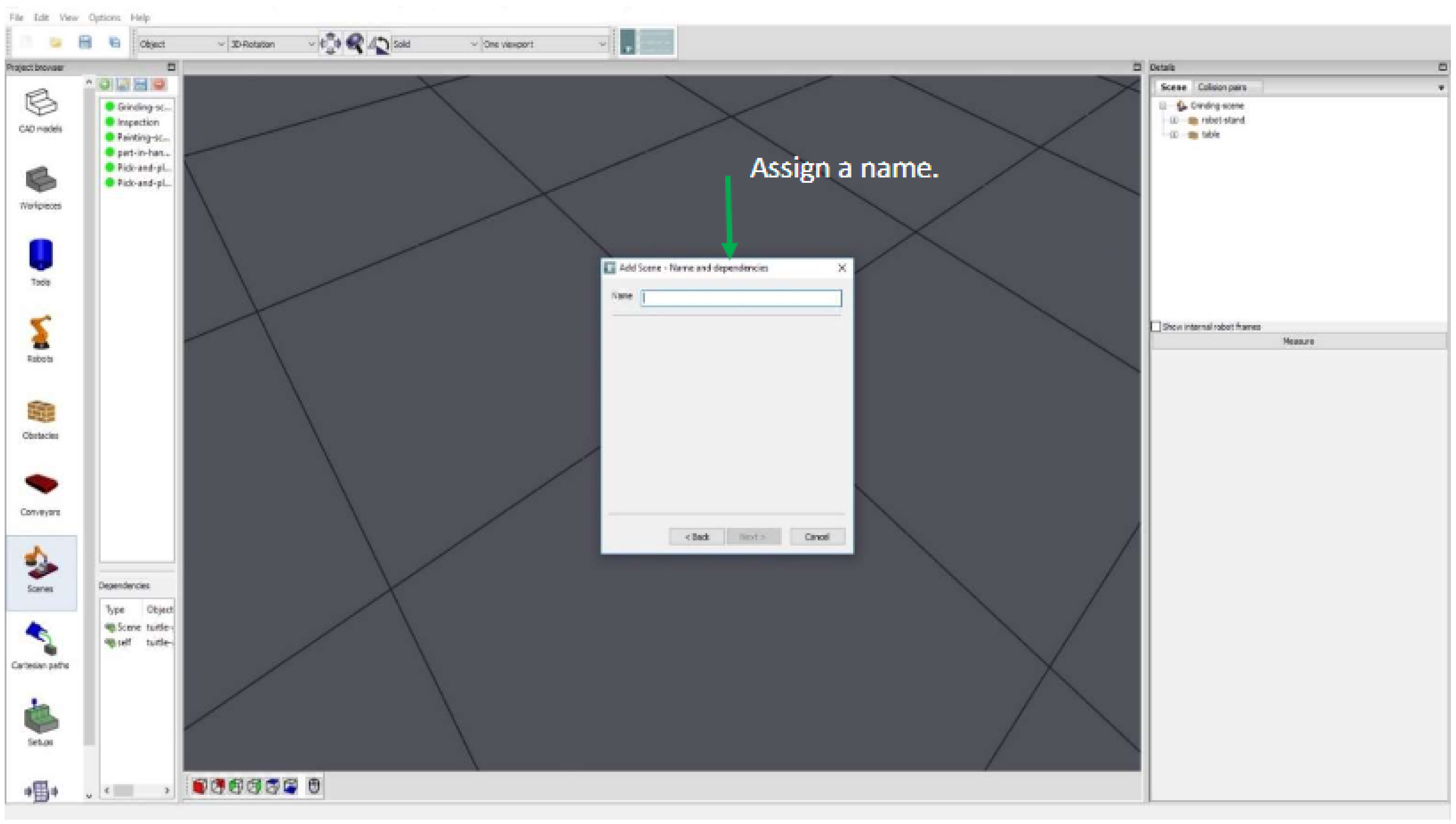
Select from list:

< Back | Next > | Cancel

Step 0: obstacles, robots, conveyors/axes, workpieces and tools have been created/imported

Step 1: create a new scene, i.e. commonly a workcell. Alternatives would be to copy one already existing in the same project/database or to import from another project / database.

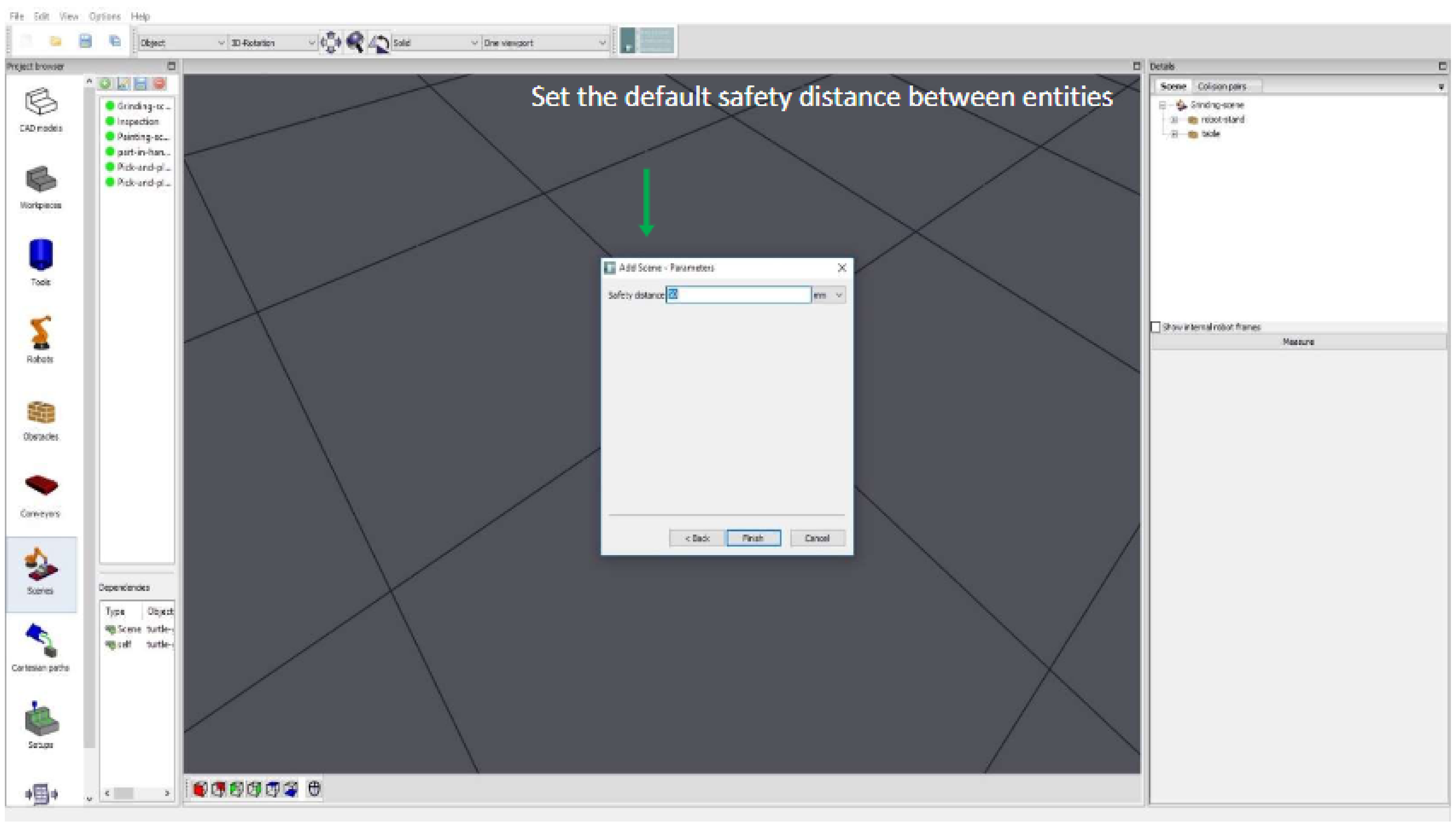
# Define Scene



Step 2: Assign a name to the scene.

trinity ENGAGE WITH AGILE MANUFACTURING

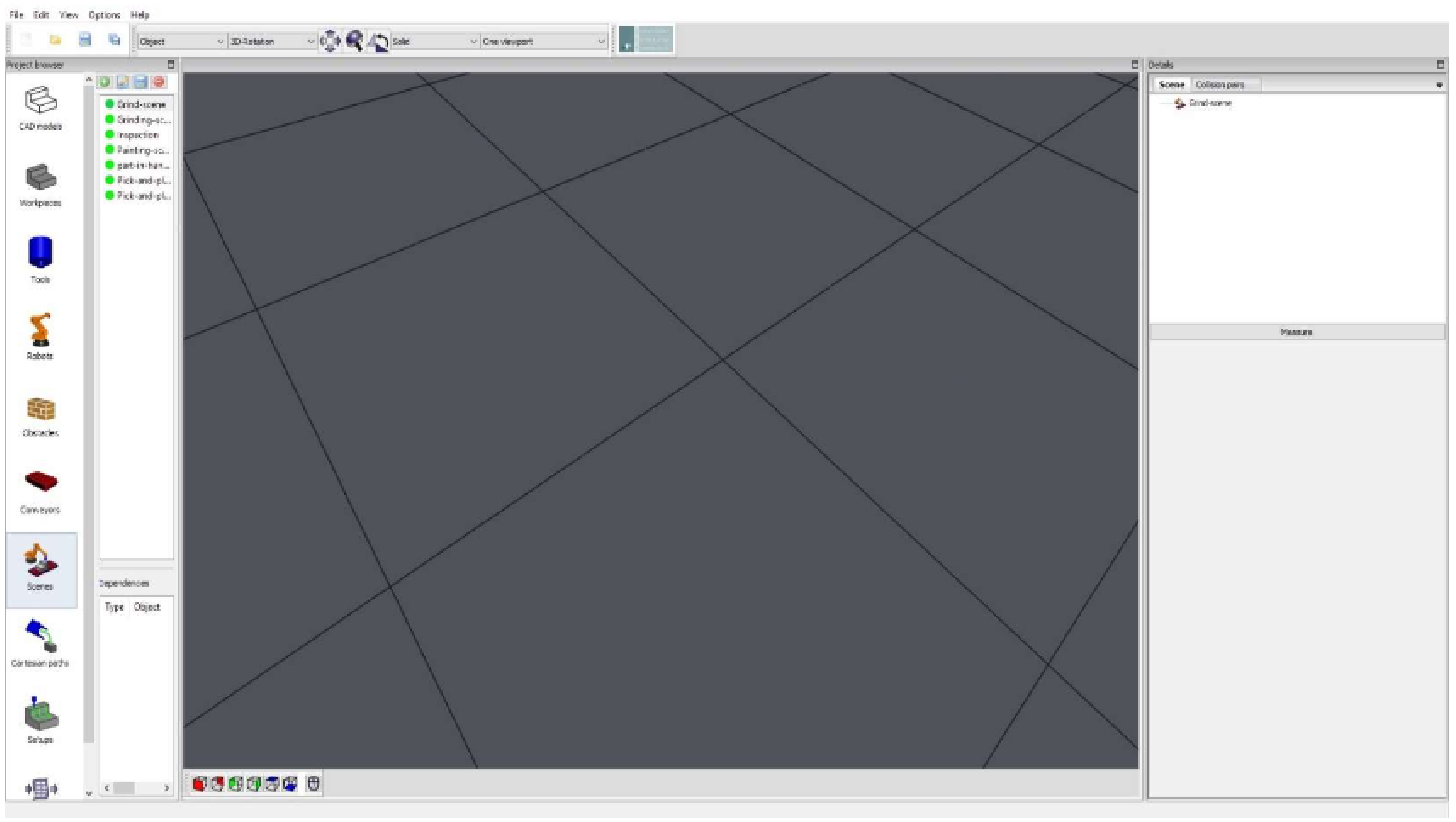
# Define Scene



Step 3: Define one initial safety distance between entities in the scene, such as tools, robots, workpieces, obstacles against each others. Safety distances for specific entities –like tool vs. work-piece, can be changed individually later. Note: collision pairs, that can not collide in the scene, such as joints or bases of the robot against (remote) obstacles, or static obstacles against other ones, are set to [void] automatically.

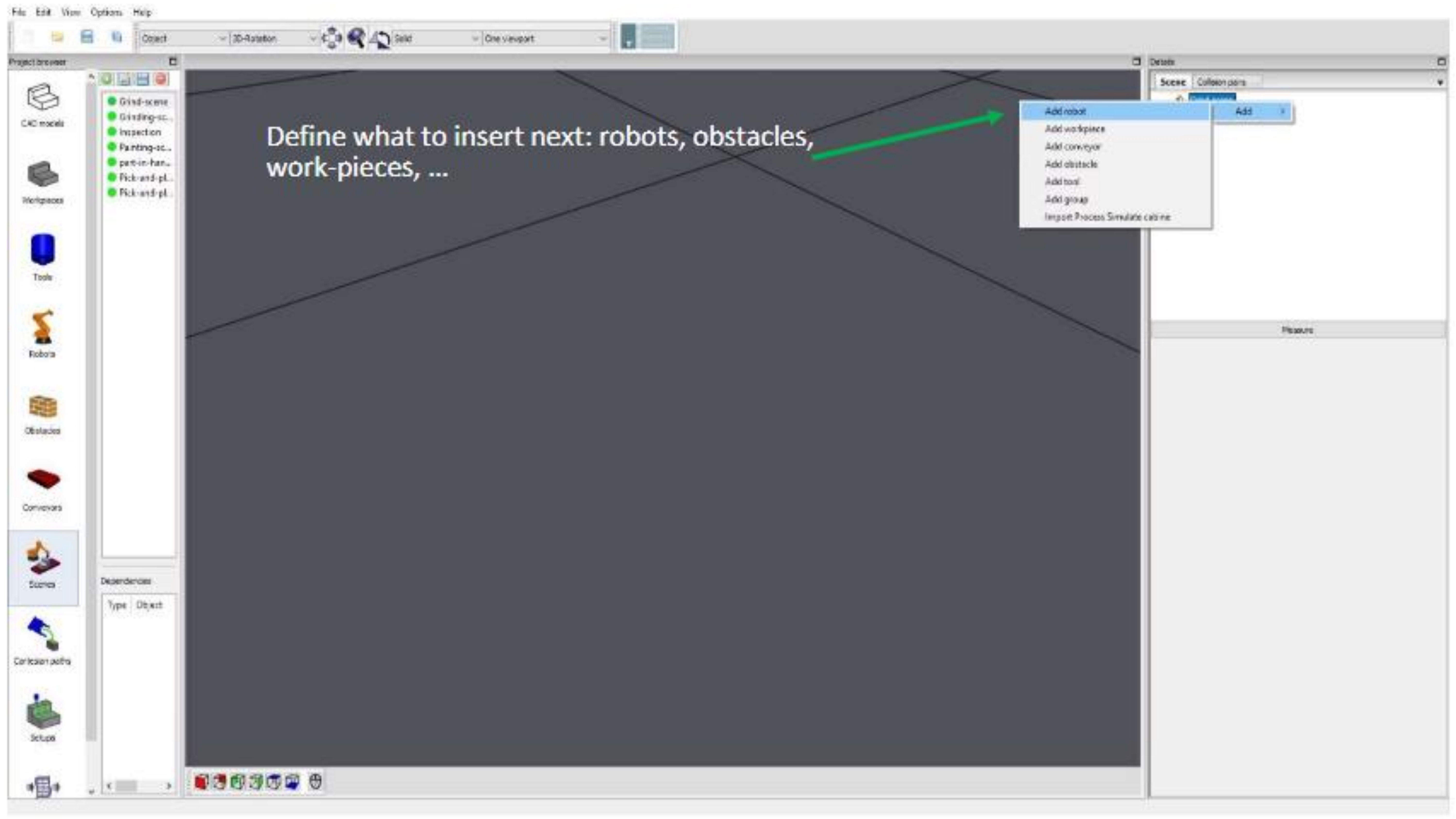
trinity ENGAGE WITH AGILE MANUFACTURING

# Define Scene



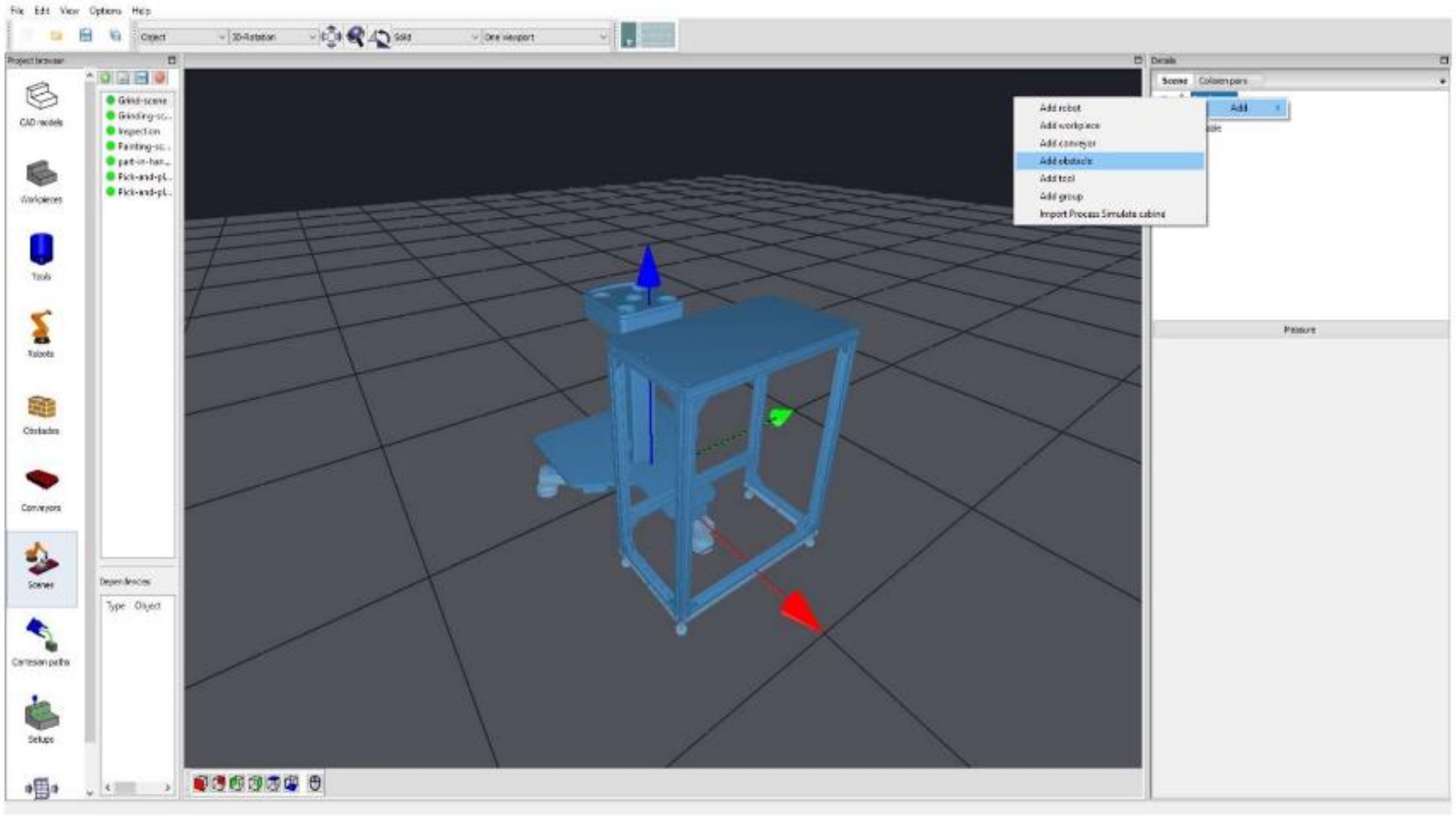
Step 4: Scene created (still empty).

# Define Scene


Define what to insert next: robots, obstacles, work-pieces, ...

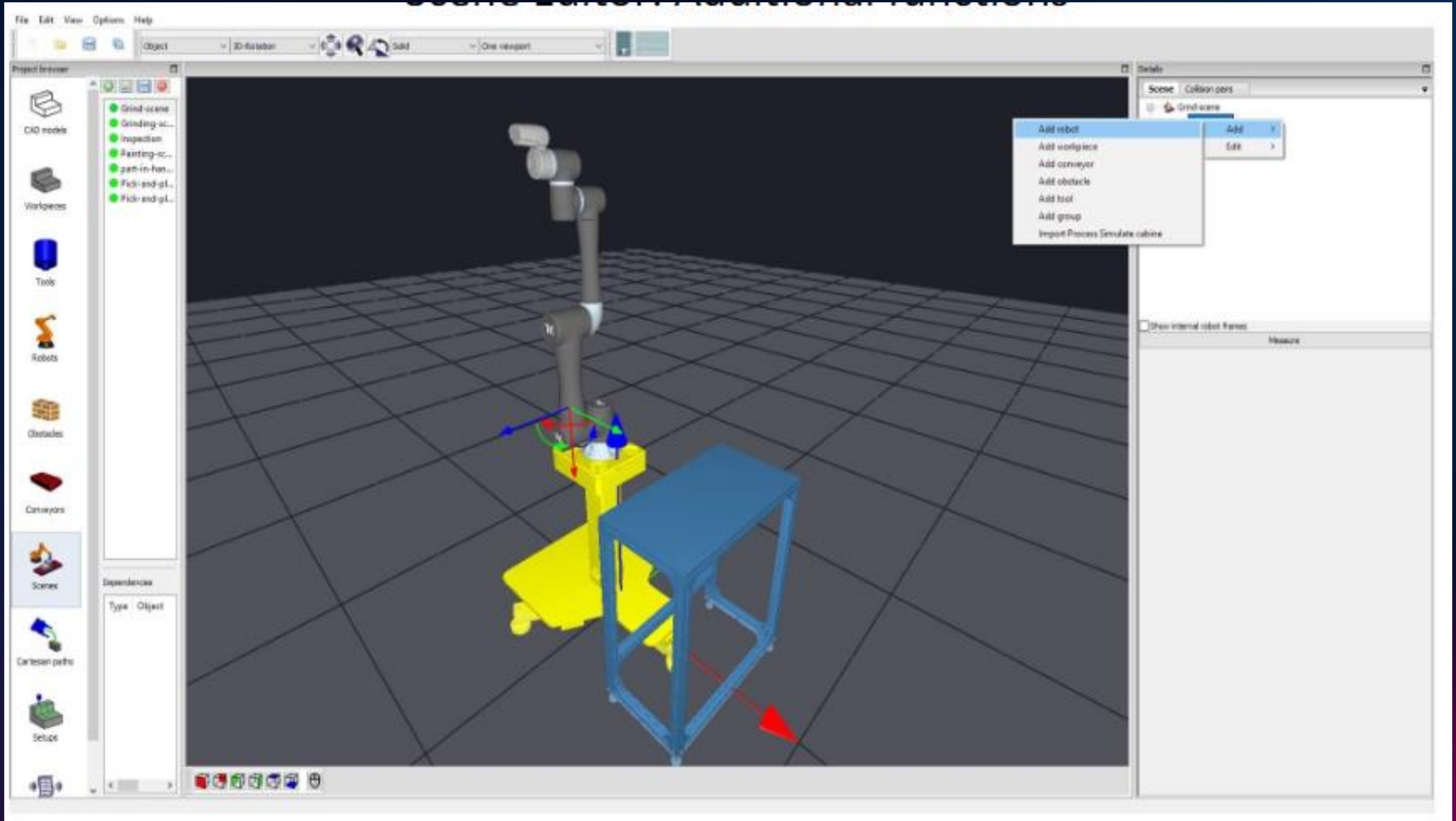
Step 5: Next step is to add elements to scene.

## Define Scene



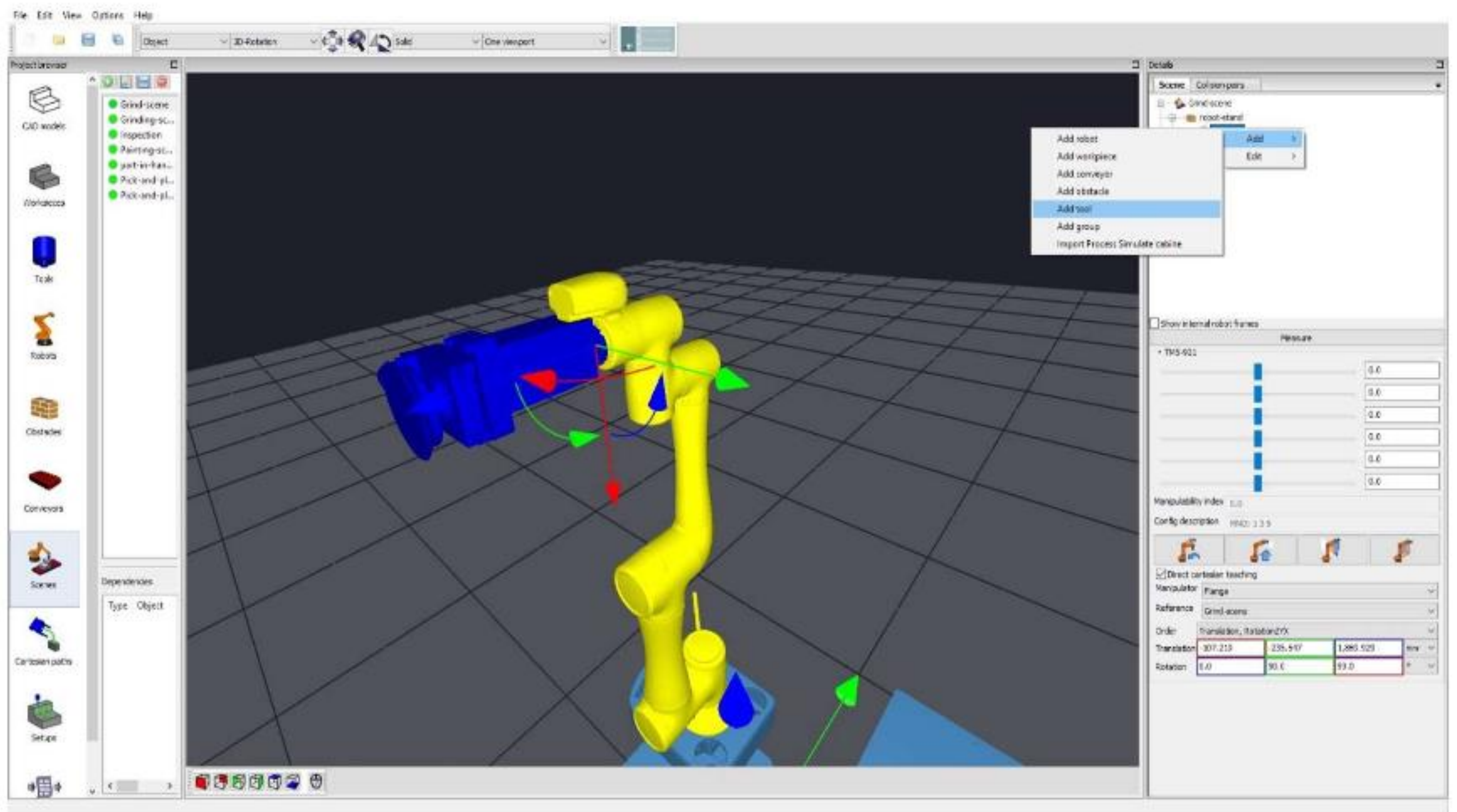
Step 6: Add first obstacles such as char for the robot and table for the workpiece.

## Define Scene



Step 7: Then add robot to the robot chair.

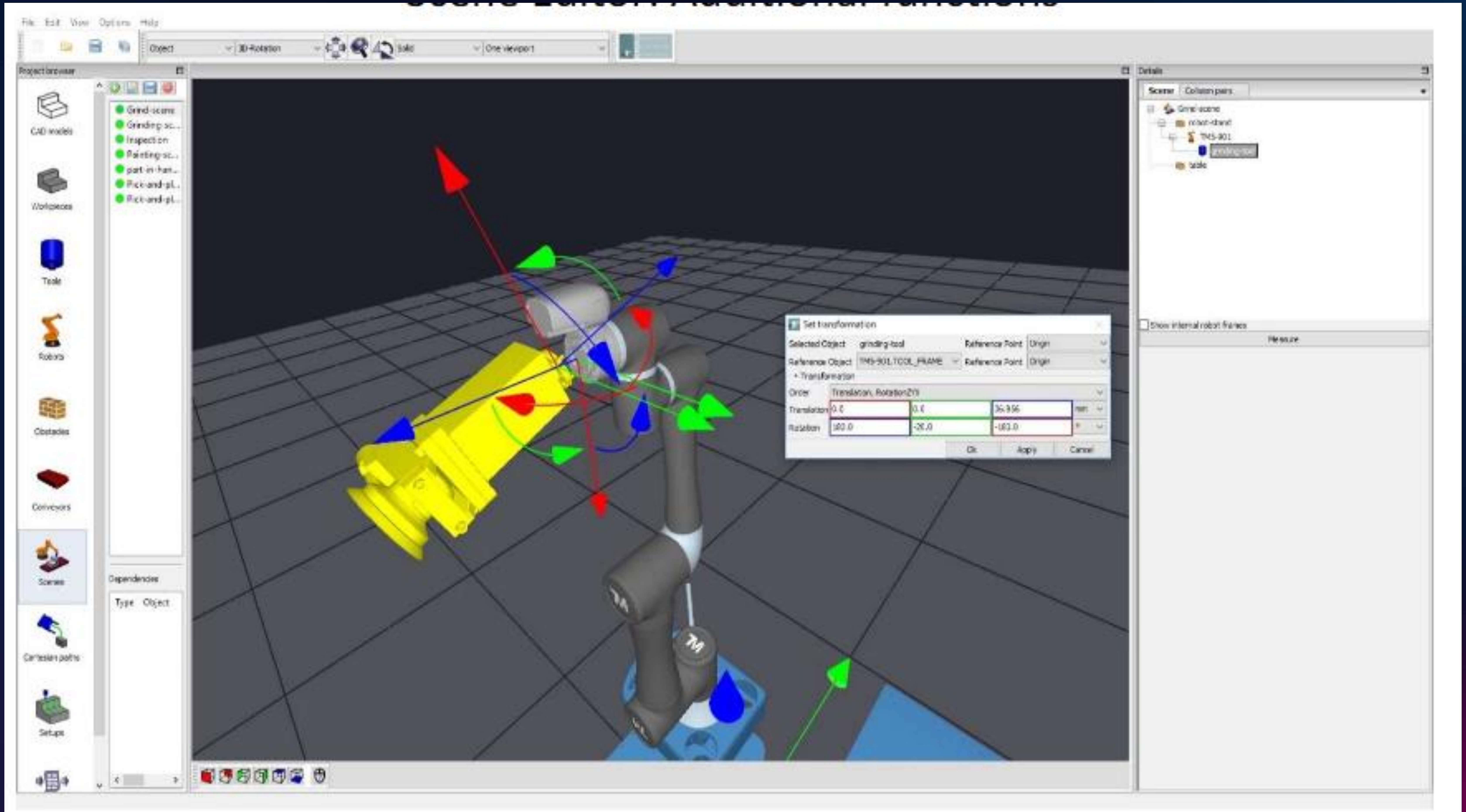
# Define Scene



Step 8: The tool is added to the robot in tha same way —from the list of tools in the project. Note: the tool will by default be connected with its Flange-pose (if define) to the robots-flange. You can modify this transformation afterwards.

trinity ENGAGE WITH AGILE MANUFACTURING

# Define Scene



Set transformation

Selected Object: grinding-tool — Reference Point: Origin

Reference Object: TM5-901.TOOL_FRAME — Reference Point: Origin

+ Transformation

Order: Translation, RotationZYX

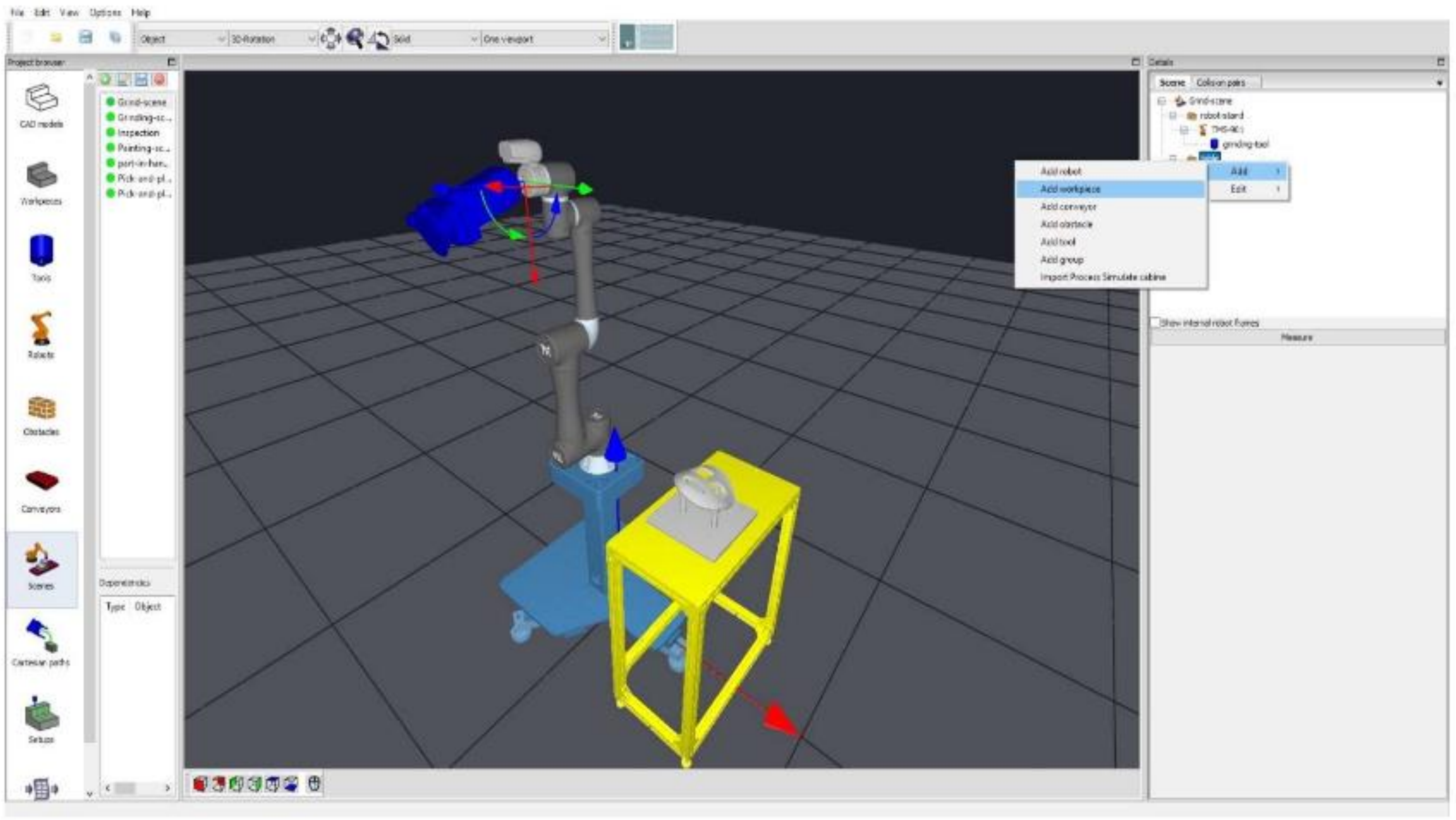
Translation: 0.0 | 0.0 | 36.956 | mm

Rotation: 180.0 | -20.0 | -180.0 | °

Ok | Apply | Cancel

Step 9: Orientation of the tool can be set by edit -> set transformation. Defined transformation visualized after "apply" (alternatively after OK).

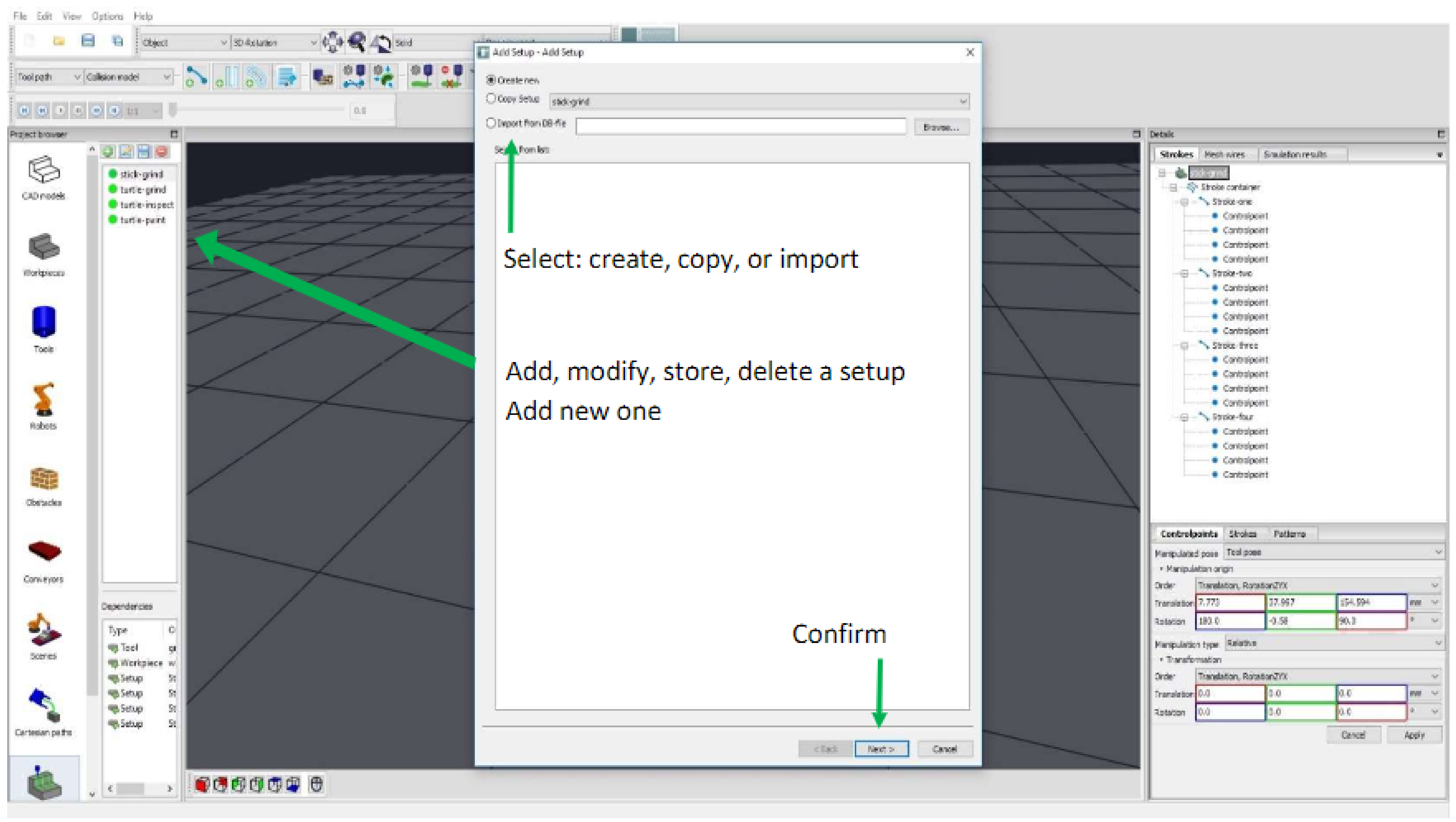
# Define Scene



Step 10: Select the work-piece Note the fist entity selected and added to the scene is initially attached to the scene-graph at the root. Assignments inside the hierarchy can be changed later in the scene graph by drag-and-drop.

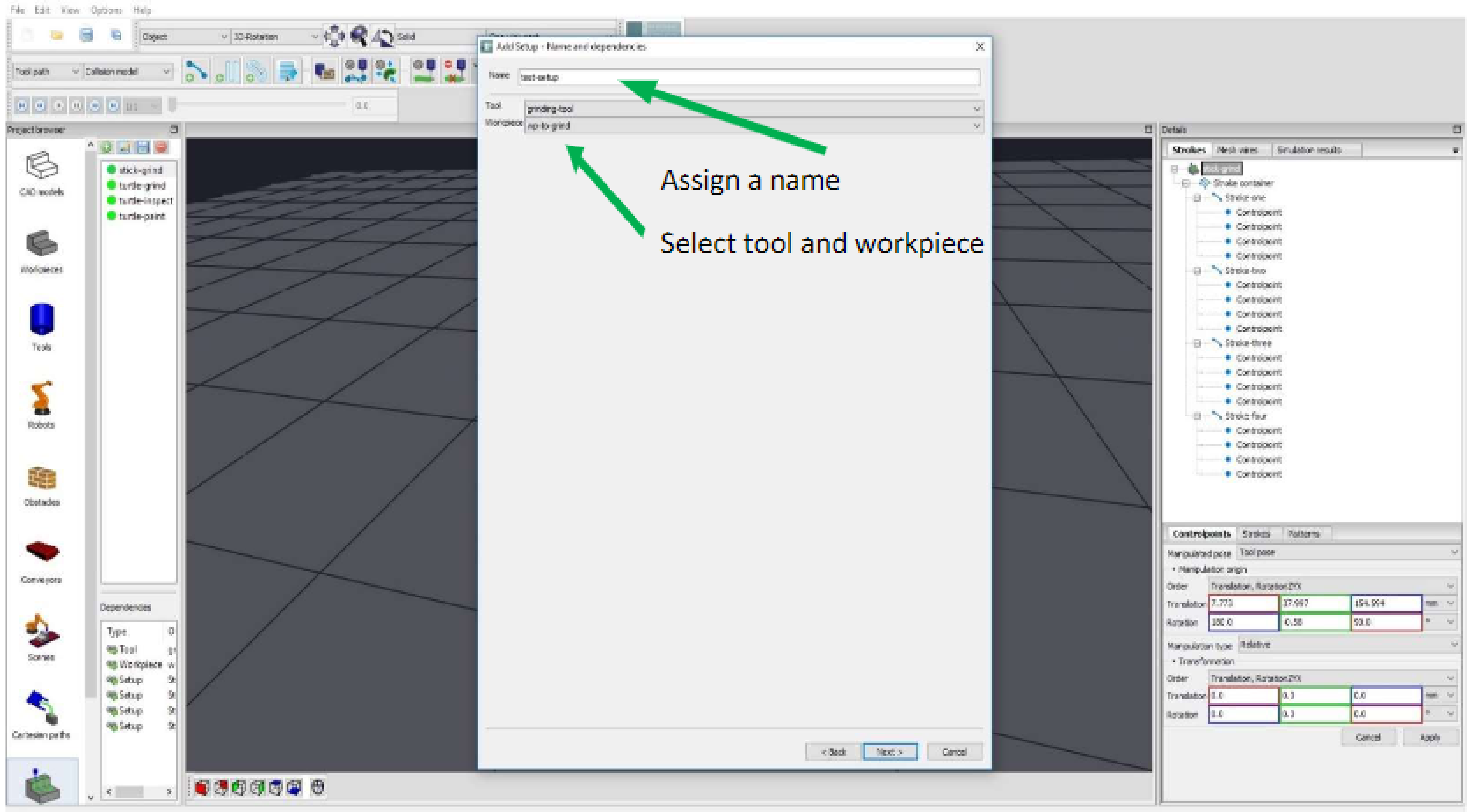
trinity ENGAGE WITH AGILE MANUFACTURING

## Define Paths



Step 1: Add a new setup. Alternatively another setup in the same project/database can be copied and modified afterward. Alternatively, a setup can also be imported that has been created within and exported out of another project/ database.

# Define Paths



Add Setup - Name and dependencies

Name: test-setup

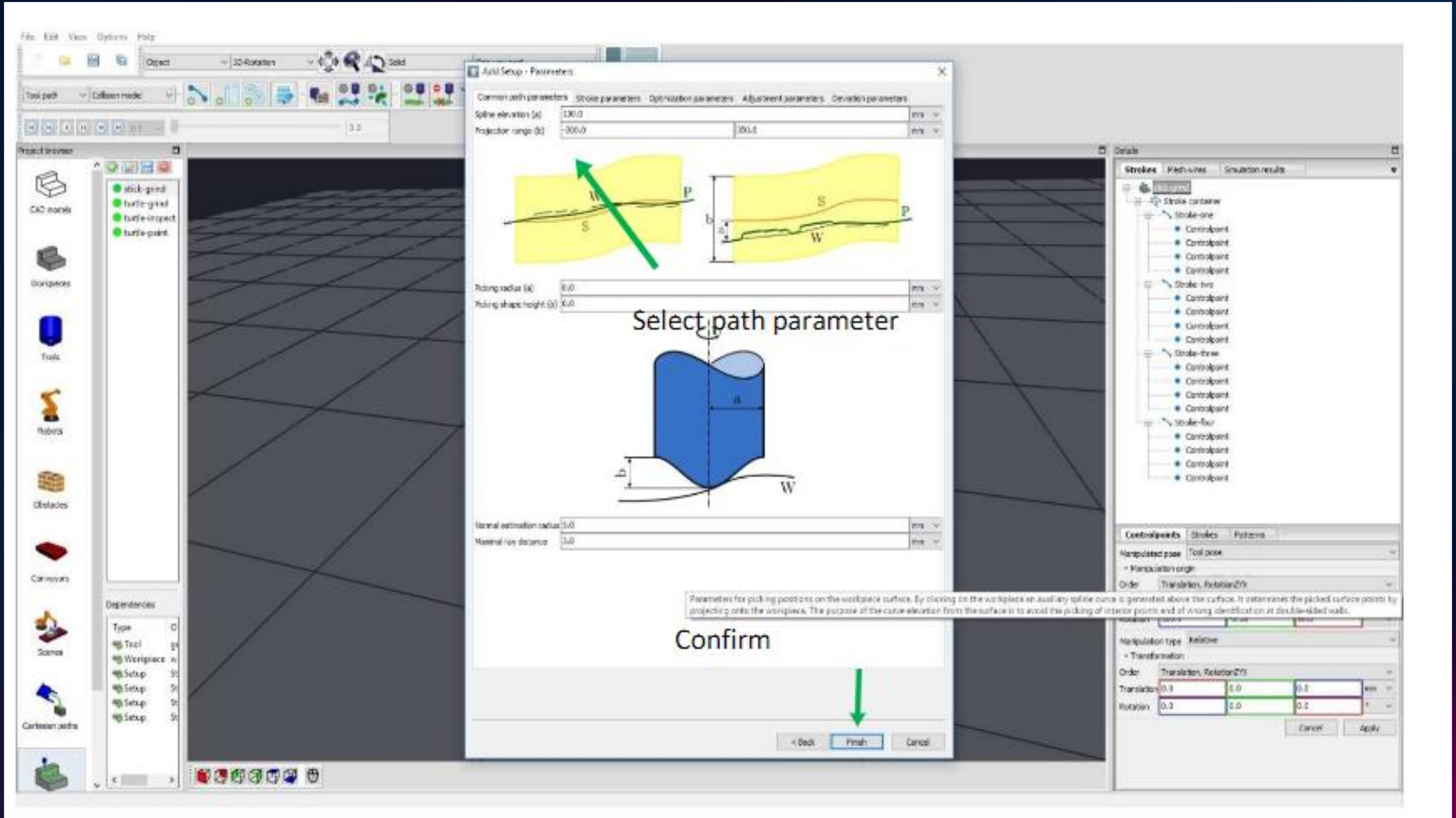
Tool: grinding-tool

Workpiece: wp-to-grind

Step 2: Assign a name. Select the work-piece to be processed and the tool for processing it.

Note: In case of multiple setups in one project, assign a meaningful name which can easily be identified when later, in the workflow (see next chapter) associating a setup to the robots to execute them.

# Define Paths

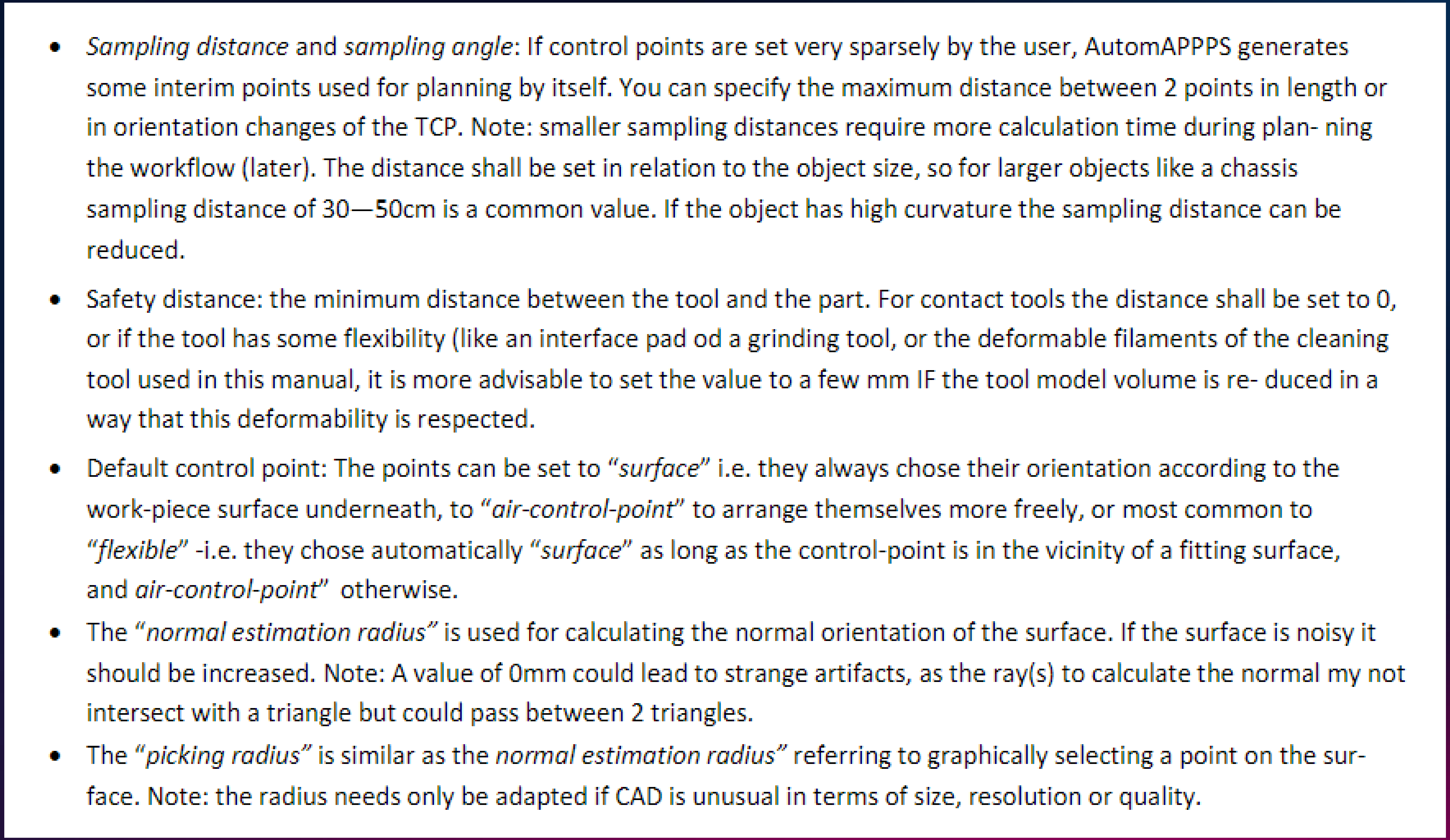


Spline elevation (a) 100.0 mm
Projection range (b) -200.0 200.0 mm
Picking radius (a) 0.0 mm
Picking shape height (b) 0.0 mm
Normal estimation radius 5.0 mm
Maximal ray distance 5.0 mm

Parameters for picking positions on the workpiece surface. By clicking on the workpiece an auxiliary spline curve is generated above the surface. It determines the picked surface points by projecting onto the workpiece. The purpose of the curve elevation from the surface is to avoid the picking of interior points and of wrong identification at double-sided walls.

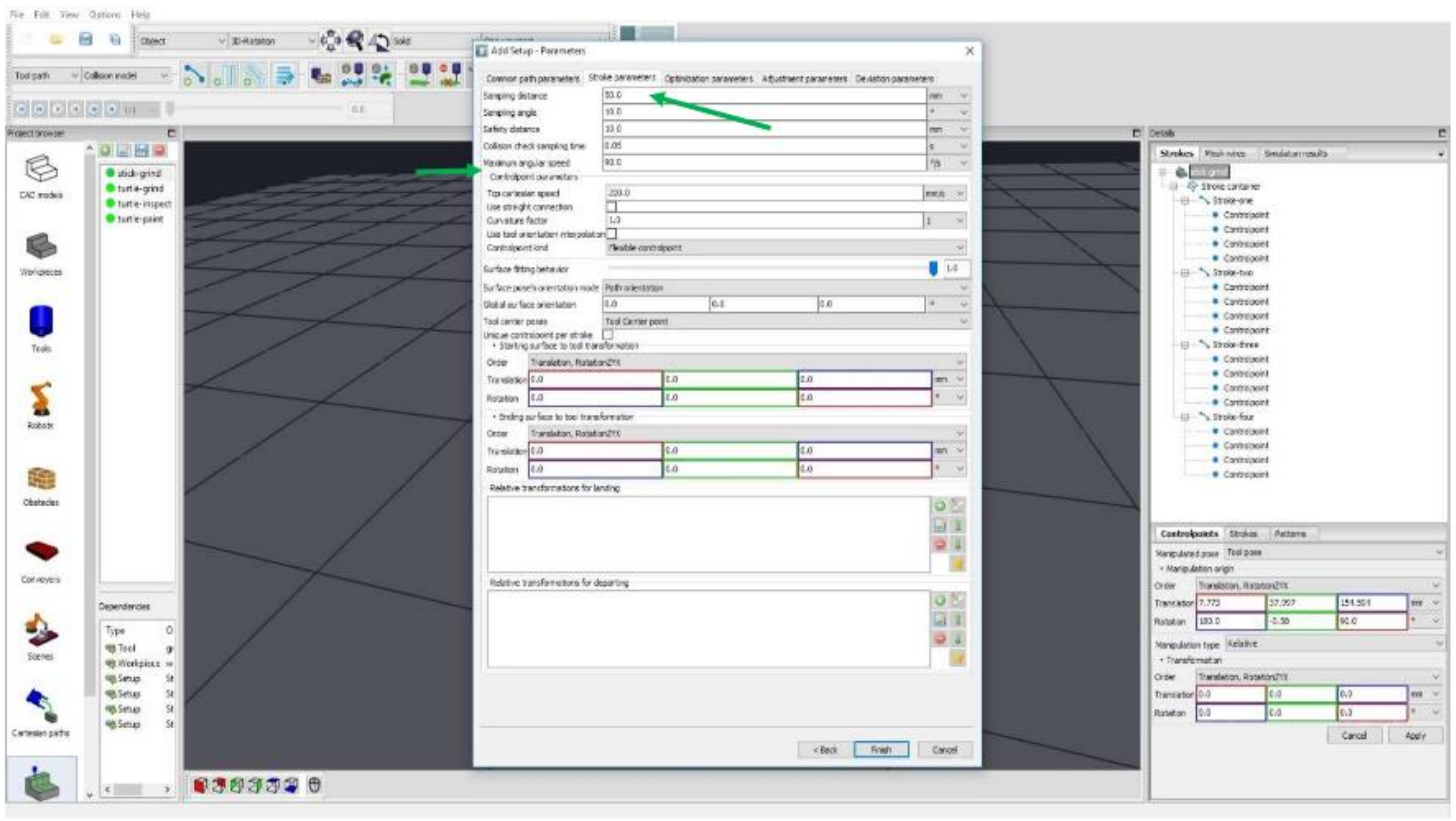
Step 3: *Common path parameters* is the menu that opens automatically. The settings should be fitting for normal use. For experts, the parameters are explained in more in the extended manual. The most important ones are:

# Define Paths



- *Sampling distance* and *sampling angle*: If control points are set very sparsely by the user, AutomAPPPS generates some interim points used for planning by itself. You can specify the maximum distance between 2 points in length or in orientation changes of the TCP. Note: smaller sampling distances require more calculation time during plan- ning the workflow (later). The distance shall be set in relation to the object size, so for larger objects like a chassis sampling distance of 30—50cm is a common value. If the object has high curvature the sampling distance can be reduced.
- Safety distance: the minimum distance between the tool and the part. For contact tools the distance shall be set to 0, or if the tool has some flexibility (like an interface pad od a grinding tool, or the deformable filaments of the cleaning tool used in this manual, it is more advisable to set the value to a few mm IF the tool model volume is re- duced in a way that this deformability is respected.
- Default control point: The points can be set to "*surface*" i.e. they always chose their orientation according to the work-piece surface underneath, to "*air-control-point*" to arrange themselves more freely, or most common to "*flexible*" -i.e. they chose automatically "*surface*" as long as the control-point is in the vicinity of a fitting surface, and *air-control-point*" otherwise.
- The "*normal estimation radius"* is used for calculating the normal orientation of the surface. If the surface is noisy it should be increased. Note: A value of 0mm could lead to strange artifacts, as the ray(s) to calculate the normal my not intersect with a triangle but could pass between 2 triangles.
- The "*picking radius"* is similar as the *normal estimation radius"* referring to graphically selecting a point on the sur- face. Note: the radius needs only be adapted if CAD is unusual in terms of size, resolution or quality.

# Define Paths



Add Setup - Parameters

Common path parameters | Stroke parameters | Optimization parameters | Adjustment parameters | Deviation parameters

| | | |
|---|---|---|
| Sampling distance | 50.0 | mm |
| Sampling angle | 10.0 | ° |
| Safety distance | 10.0 | mm |
| Collision check sampling time | 0.05 | s |
| Maximum angular speed | 90.0 | °/s |

Controlpoint parameters

| | | |
|---|---|---|
| Tcp cartesian speed | 220.0 | mm/s |
| Use straight connection | ☐ | |
| Curvature factor | 1.0 | 1 |
| Use tool orientation interpolation | ☐ | |
| Controlpoint kind | Flexible controlpoint | |
| Surface fitting behavior | 1.0 | |
| Surface pose's orientation mode | Path orientation | |
| Global surface orientation | 0.0, 0.0, 0.0 | ° |
| Tool center poses | Tool Center point | |
| Unique controlpoint per stroke | ☐ | |

Starting surface to tool transformation

| | | | | |
|---|---|---|---|---|
| Order | Translation, RotationZYX | | | |
| Translation | 0.0 | 0.0 | 0.0 | mm |
| Rotation | 0.0 | 0.0 | 0.0 | ° |

Ending surface to tool transformation

| | | | | |
|---|---|---|---|---|
| Order | Translation, RotationZYX | | | |
| Translation | 0.0 | 0.0 | 0.0 | mm |
| Rotation | 0.0 | 0.0 | 0.0 | ° |

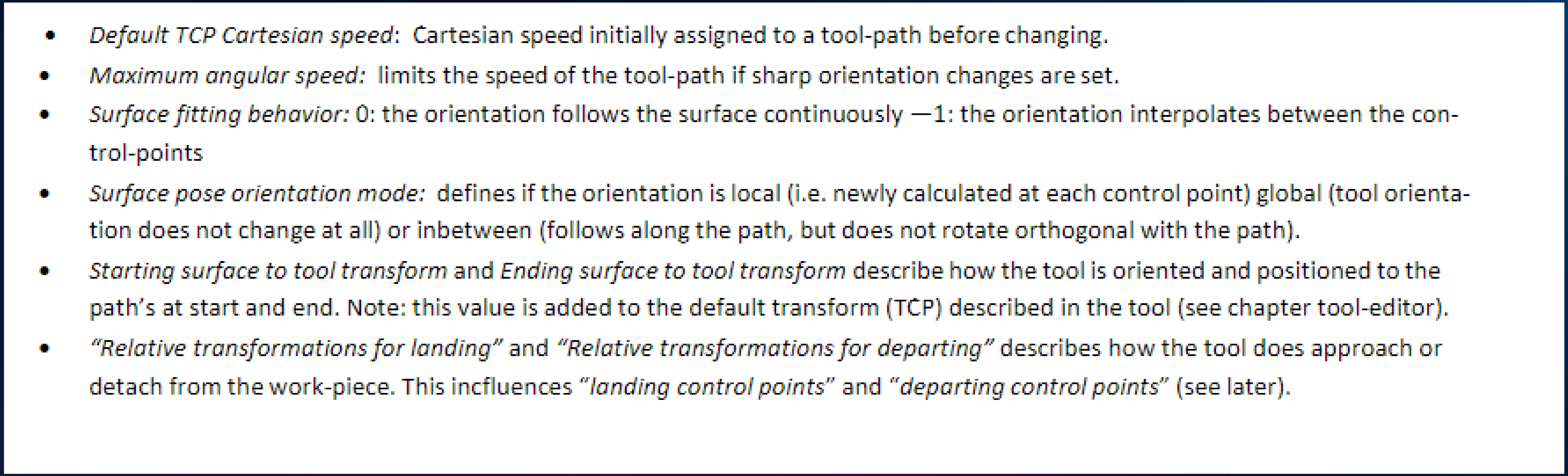
Relative transformations for landing

Relative transformations for departing

Step 4: Select Tool specific "*path parameters*". The settings should be fitting for normal use. For experts, the parameters are explained in more in the extended manual. The most important ones are:

trinity ENGAGE WITH AGILE MANUFACTURING

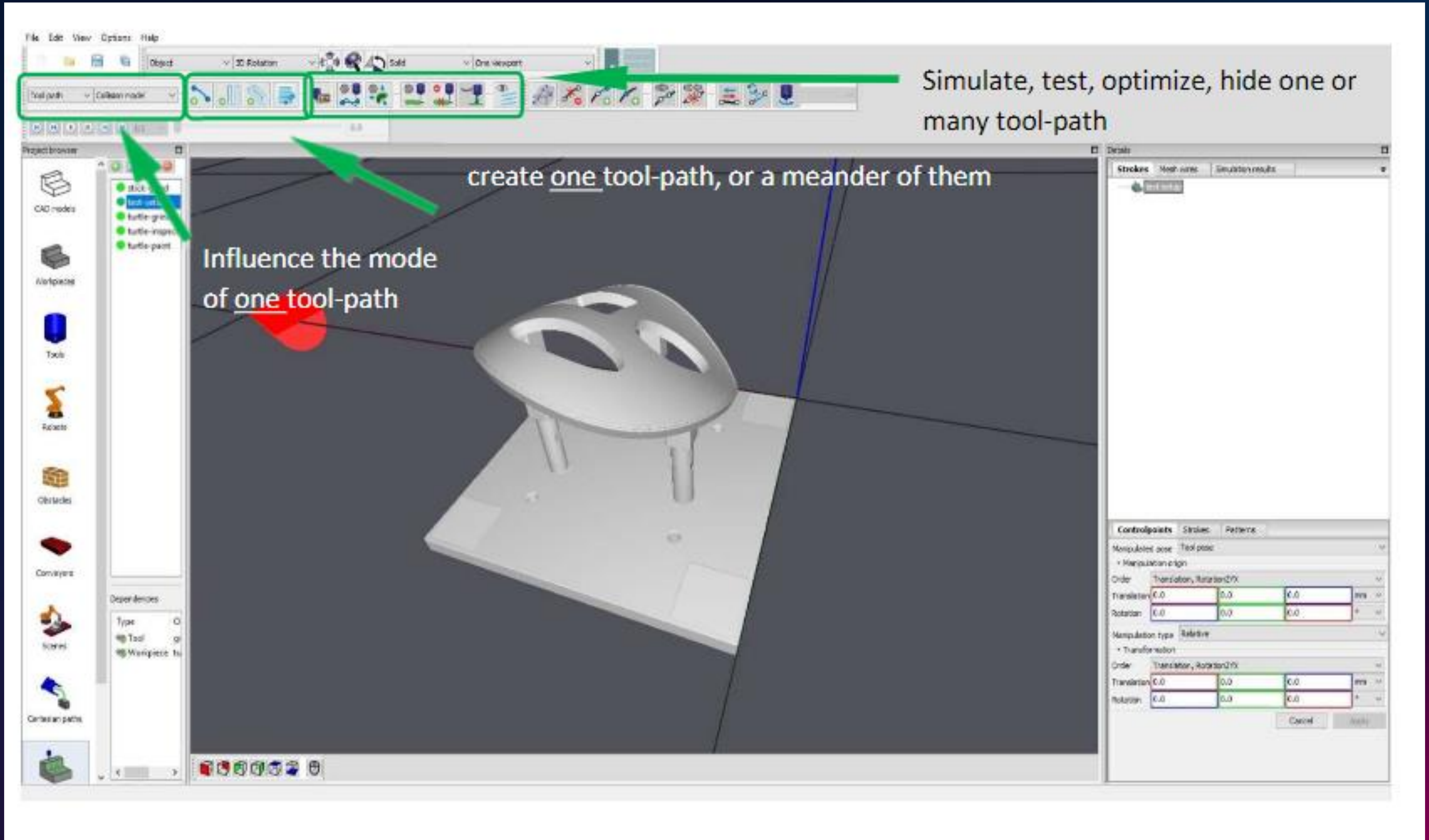
# Define Paths



- *Default TCP Cartesian speed*: Cartesian speed initially assigned to a tool-path before changing.
- *Maximum angular speed:* limits the speed of the tool-path if sharp orientation changes are set.
- *Surface fitting behavior:* 0: the orientation follows the surface continuously —1: the orientation interpolates between the con-trol-points
- *Surface pose orientation mode:* defines if the orientation is local (i.e. newly calculated at each control point) global (tool orienta-tion does not change at all) or inbetween (follows along the path, but does not rotate orthogonal with the path).
- *Starting surface to tool transform* and *Ending surface to tool transform* describe how the tool is oriented and positioned to the path's at start and end. Note: this value is added to the default transform (TCP) described in the tool (see chapter tool-editor).
- *"Relative transformations for landing"* and *"Relative transformations for departing"* describes how the tool does approach or detach from the work-piece. This incfluences *"landing control points"* and *"departing control points"* (see later).

trinity ENGAGE WITH AGILE MANUFACTURING

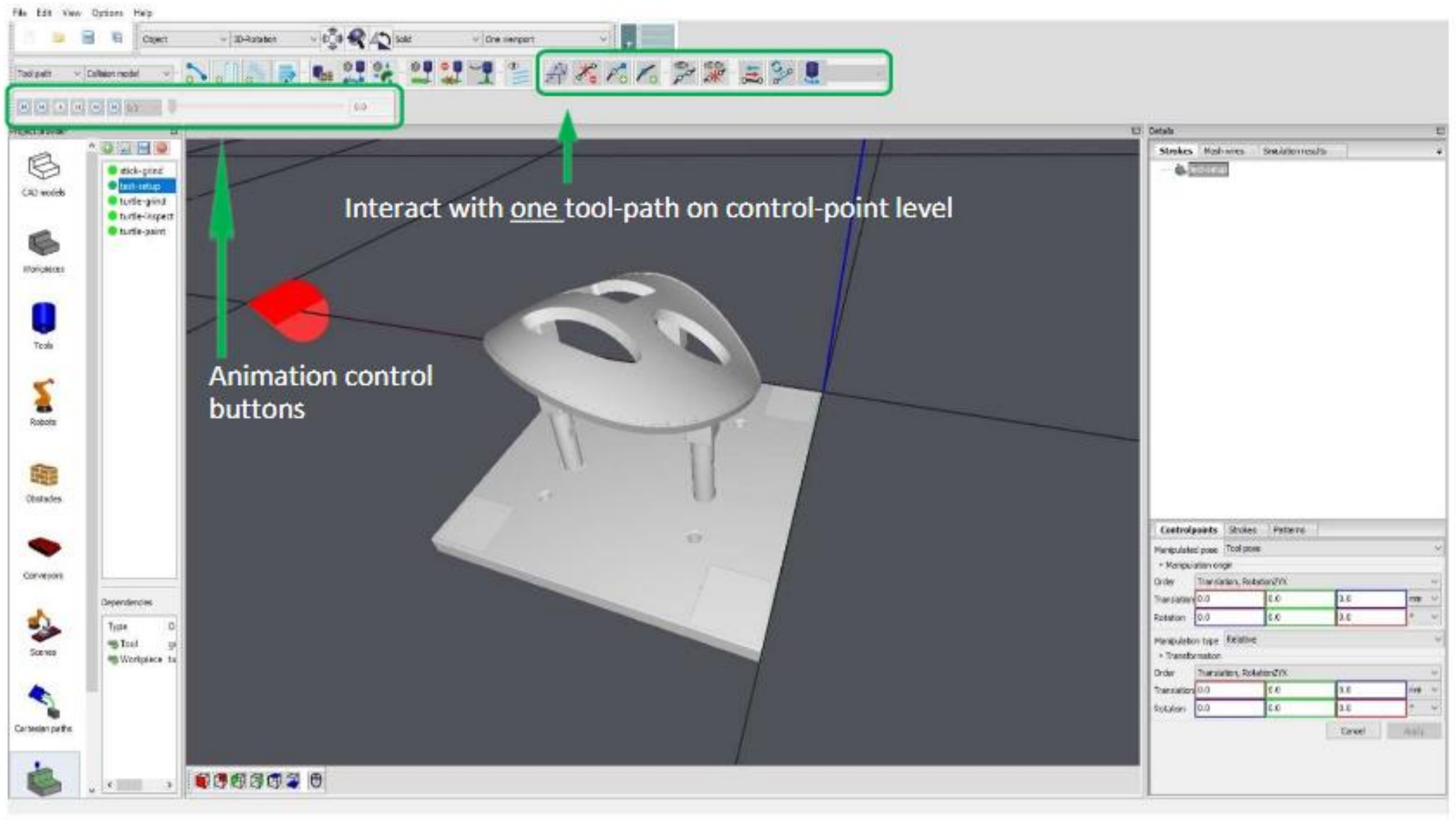
## Define Paths


Simulate, test, optimize, hide one or many tool-path
create one tool-path, or a meander of them
Influence the mode of one tool-path
File Edit View Options Help
Project browser
CAD models
Workpieces
Tools
Robots
Obstacles
Conveyors
Scenes
Cartesian paths
Details
Strokes
Controlpoints
Cancel
Apply

Step 5: Situation after initiating the setup. The row of buttons in the middle does allow:

trinity ENGAGE WITH AGILE MANUFACTURING

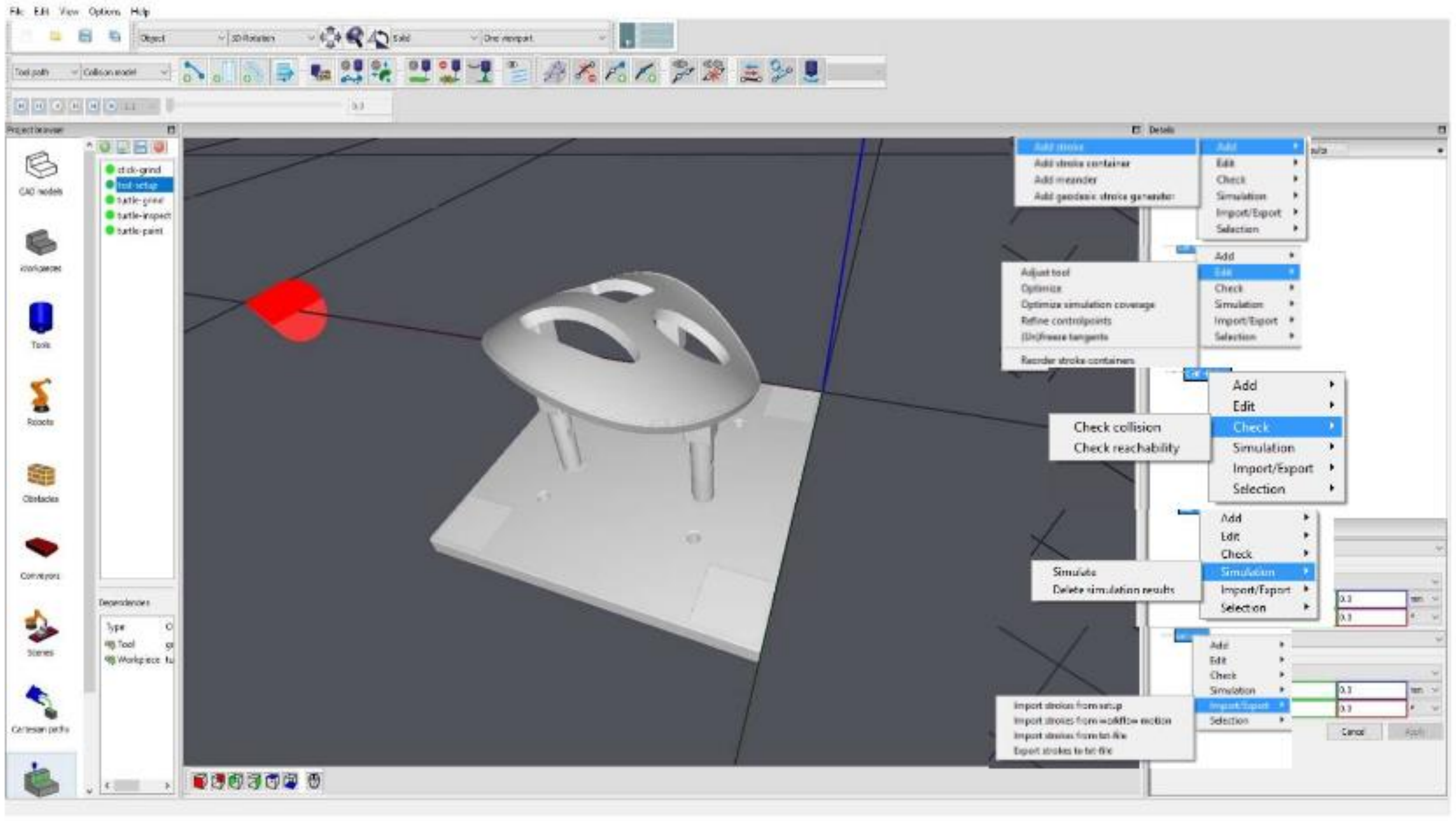
# Define Paths


Interact with one tool-path on control-point level
Animation control buttons
File Edit View Options Help
CAD models
Workpieces
Tools
Robots
Obstacles
Conveyors
Scenes
Cartesian paths
Details
Strokes
Controlpoints

Step 6: The interactions with 1 tool path. Selected row on the right up from left to right:

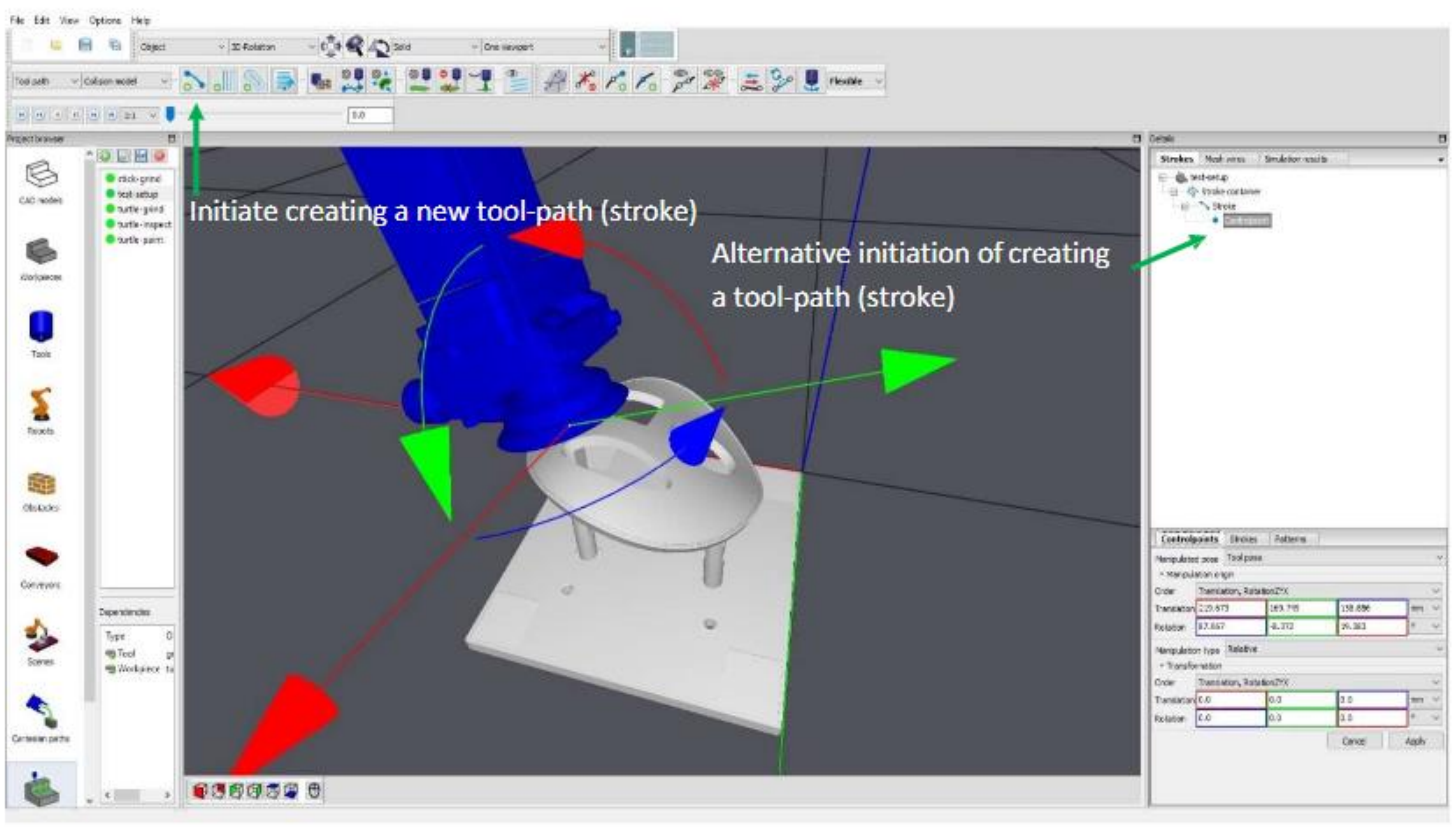
trinity ENGAGE WITH AGILE MANUFACTURING

## Define Paths



Step 7: Insert a tool-path and—if desired— a container for grouping tool-paths before. For the root "car-new" the following actions can be applied on all tool-paths within (linked to the root).

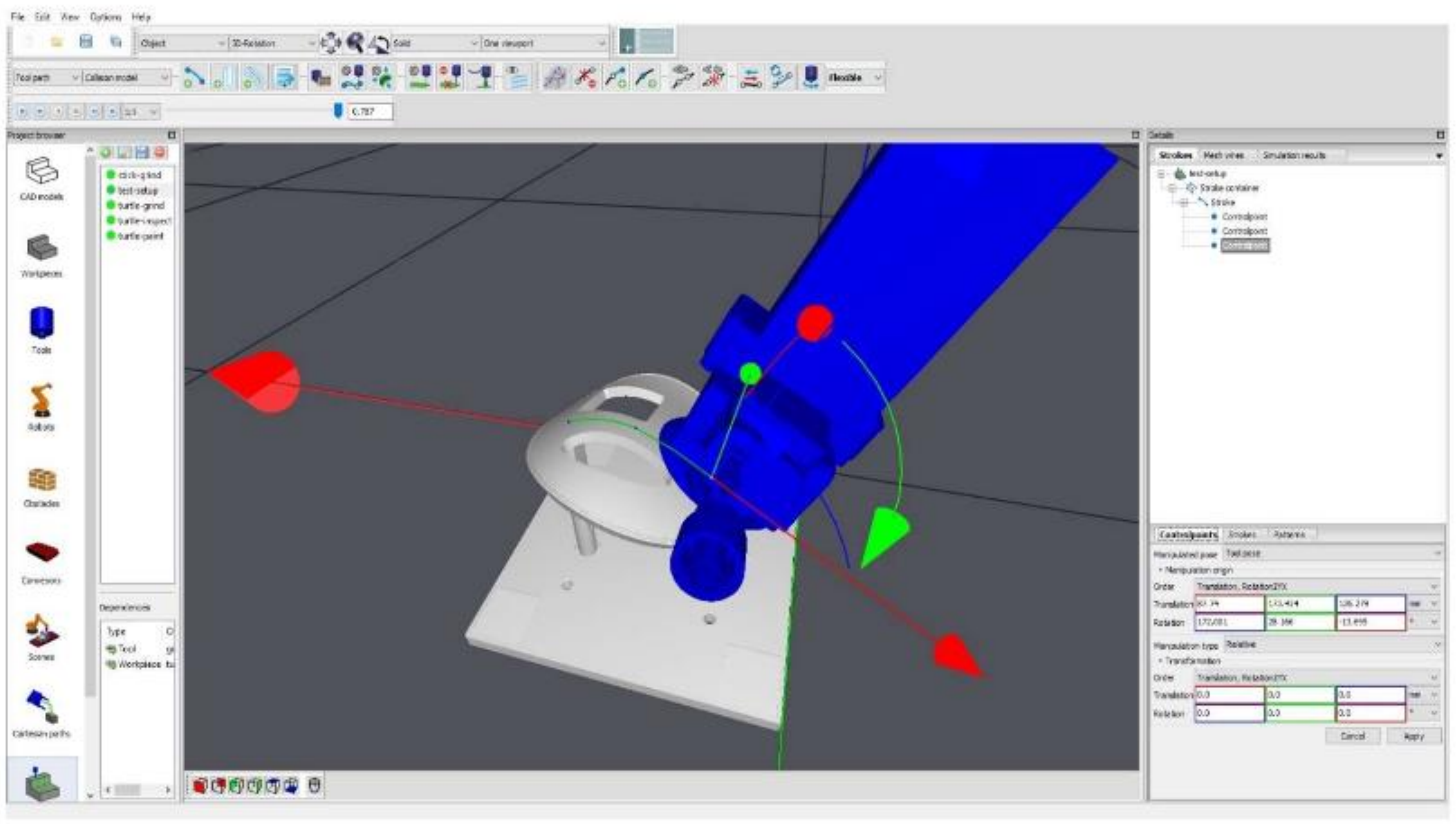
# Define Paths



Manipulated pose: Tool pose
Manipulation origin
Order: Translation, RotationZYX
Translation: 220.873 | 169.795 | 158.686 | mm
Rotation: 87.667 | -8.372 | 26.383 | °
Manipulation type: Relative
Transformation
Order: Translation, RotationZYX
Translation: 0.0 | 0.0 | 0.0 | mm
Rotation: 0.0 | 0.0 | 0.0 | °

Step 8: *create* a tool-path: use the button or right-mouse-click on the tree in the right side-bar.
Keeping *ctrl* pressed and with left-mouse-clicks, the control points are initially set to the work-piece-mesh one after the other (and visualized immediately according to the selected visualization mode (here collision model).
Note: the tool-path (stroke) can be inserted directly at the root (here: car) or after creating a container that groups them. The hierarchy (tool-paths and container) can be changed freely afterwards.

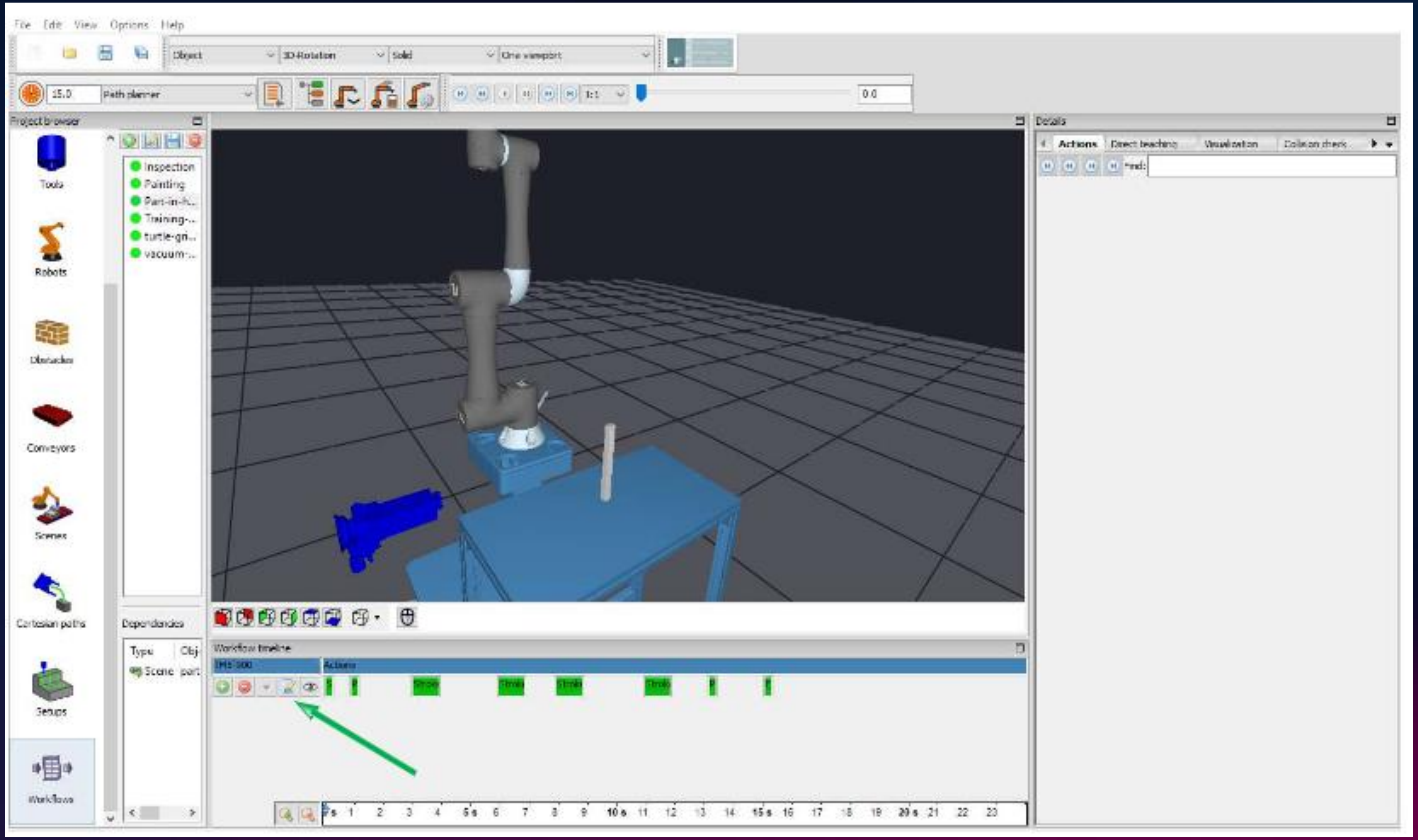
## Define Paths


Manipulated pose: Tool pose
Manipulation origin
Order: Translation, RotationZYX
Translation: [illegible] | 133.414 | 126.279 | mm
Rotation: 172.081 | 29.366 | -13.695 | °
Manipulation type: Relative
Transformation
Order: Translation, RotationZYX
Translation: 0.0 | 0.0 | 0.0 | mm
Rotation: 0.0 | 0.0 | 0.0 | °

Step 9: second and third control point of the tool-path inserted as described above. The tool-orientation adjusts itself as defined in the setup-parameters according to the orientation of the tool-path and/or its control-points.
Note: it is possible to rotate the view during setting of the control-points of a tool-path by releasing the *ctrl* button, changing the view, resume pressing *ctrl* and continuing setting new control-points.

trinity ENGAGE WITH AGILE MANUFACTURING

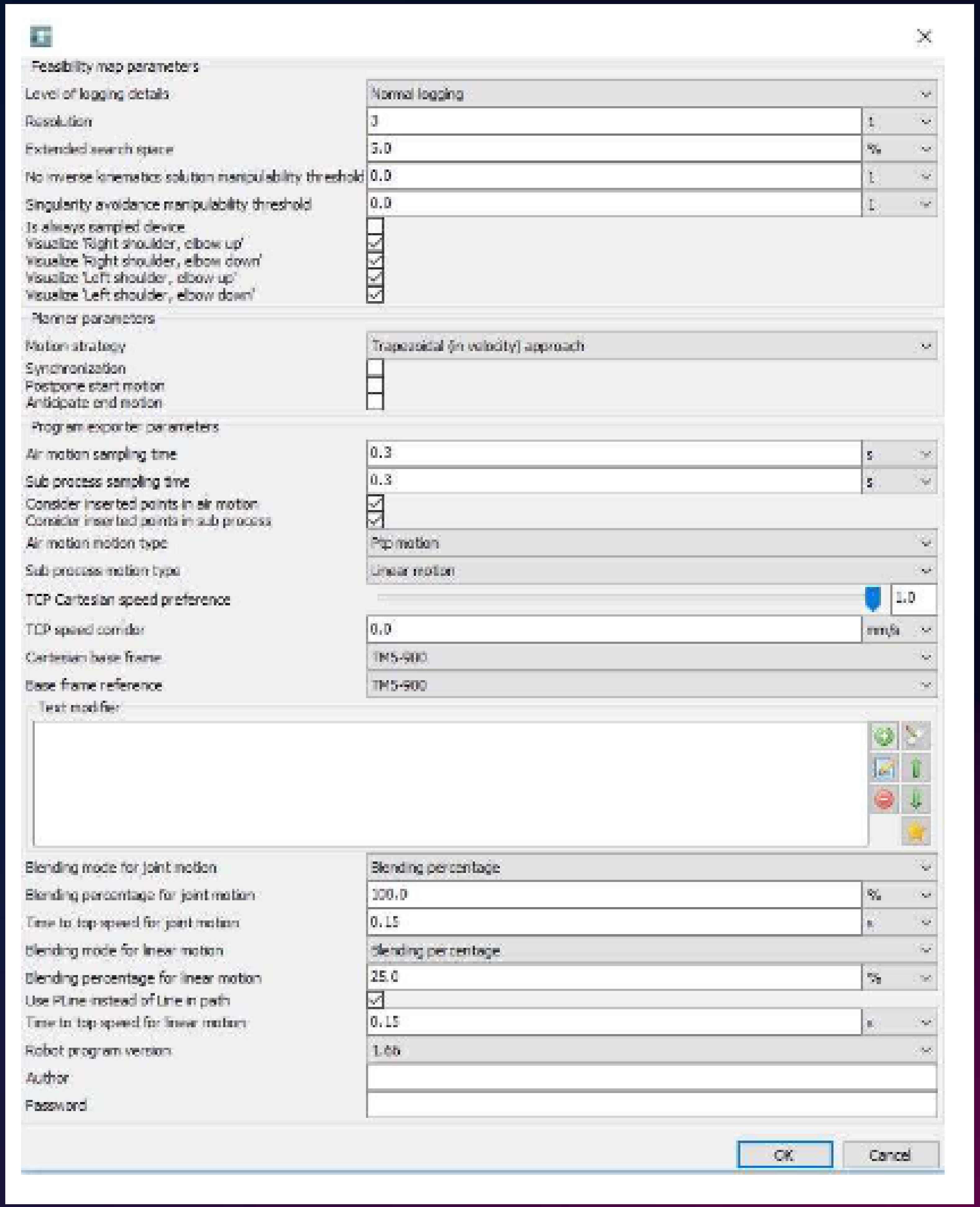
# Exporting trajectories



After the workflow is planned successfully, the executable program for the robot can be generated. There is a menu for setting parameters for the program exporter, it can be found from the bar down at the workflow.

trinity ENGAGE WITH AGILE MANUFACTURING

# Exporting trajectories



Feasibility map parameters

| | | |
|---|---|---|
| Level of logging details | Normal logging | |
| Resolution | 3 | 1 |
| Extended search space | 5.0 | % |
| No inverse kinematics solution manipulability threshold | 0.0 | 1 |
| Singularity avoidance manipulability threshold | 0.0 | 1 |
| Is always sampled device | ☐ | |
| Visualize 'Right shoulder, elbow up' | ✓ | |
| Visualize 'Right shoulder, elbow down' | ✓ | |
| Visualize 'Left shoulder, elbow up' | ✓ | |
| Visualize 'Left shoulder, elbow down' | ✓ | |

Planner parameters

| | |
|---|---|
| Motion strategy | Trapezoidal (in velocity) approach |
| Synchronization | ☐ |
| Postpone start motion | ☐ |
| Anticipate end motion | ☐ |

Program exporter parameters

| | | |
|---|---|---|
| Air motion sampling time | 0.3 | s |
| Sub process sampling time | 0.3 | s |
| Consider inserted points in air motion | ✓ | |
| Consider inserted points in sub process | ✓ | |
| Air motion motion type | Ptp motion | |
| Sub process motion type | Linear motion | |
| TCP Cartesian speed preference | 1.0 | |
| TCP speed corridor | 0.0 | mm/s |
| Cartesian base frame | TM5-900 | |
| Base frame reference | TM5-900 | |

Text modifier

| | | |
|---|---|---|
| Blending mode for joint motion | Blending percentage | |
| Blending percentage for joint motion | 100.0 | % |
| Time to top speed for joint motion | 0.15 | s |
| Blending mode for linear motion | Blending percentage | |
| Blending percentage for linear motion | 25.0 | % |
| Use PLine instead of Line in path | ✓ | |
| Time to top speed for linear motion | 0.15 | s |
| Robot program version | 1.66 | |
| Author | | |
| Password | | |

Parameters for the exporter will defined starting from the middle of the menu window. Most of the cases default parameters can be used for exporting programs and they are explained in next slide

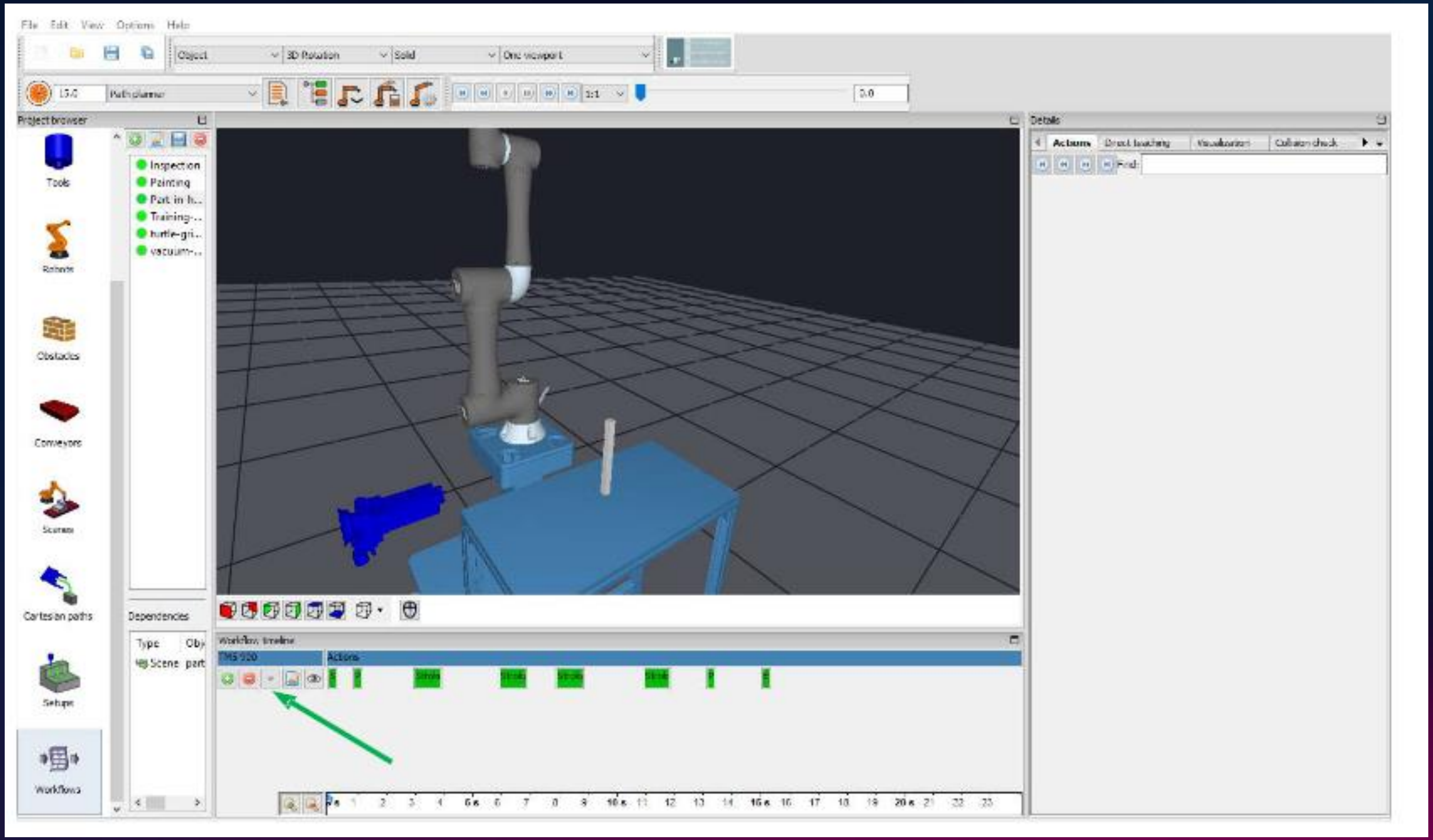
trinity ENGAGE WITH AGILE MANUFACTURING

# Export parameters

- **Air motion sampling time:** this is the sampling time for exported points in the robot program for motions in between actions. If the time is high (such as 1.0 or 2.0s), robot controller has more freedom to create the path.
- **Sub process sampling time:** this is the sampling time for exported points in the robot program for process motions such as strokes.
- **Air motion motion type:** Point-to-point or linear motion
- **Sub process motion type:** Point-to-point or linear motion
- **TCP cartesian speed preference:** when 0, predefined cartesian speed will be exported, when 1, planned cartesian speed will be exported. Planned cartesian speed means that it may vary from the set speed values in complex shapes like edges and corners.
- **TCP speed corridor:** This value defines the area speed can vary even so that it’s not changed for every point.
- **Cartesian base frame:** the frame where cartesian points are expressed
- **Base frame reference:** the frame where cartesian points are expressed
- **Blending mode for joint motion:** can be changed between “no blending”, “blending percentage” or “blending radius”. This will set the blending mode.
- **Blending percentage for joint motion:** if “Blending percentage” is selected above, the percentage will be defined here.
- **Time to top speed for joint motion:** time that can be used to accelerate to full speed.
- **Blending percentage for linear motion:** similar as for joint motions (see above)
- **Author and passwd:** file can be protected by giving information for that

trinity ENGAGE WITH AGILE MANUFACTURING

## Export



The robot program will be exported by pressing the button pointed by green arrow in the figure and selecting there “Export program”. A program folder opens and location where the program wanted to save have to be selected.

trinity ENGAGE WITH AGILE MANUFACTURING

## Summary

- At this point, we have exported trajectories for robot
- This concludes our training for module Dynamic Robot trajectory generation based on information from 3D camera

trinity