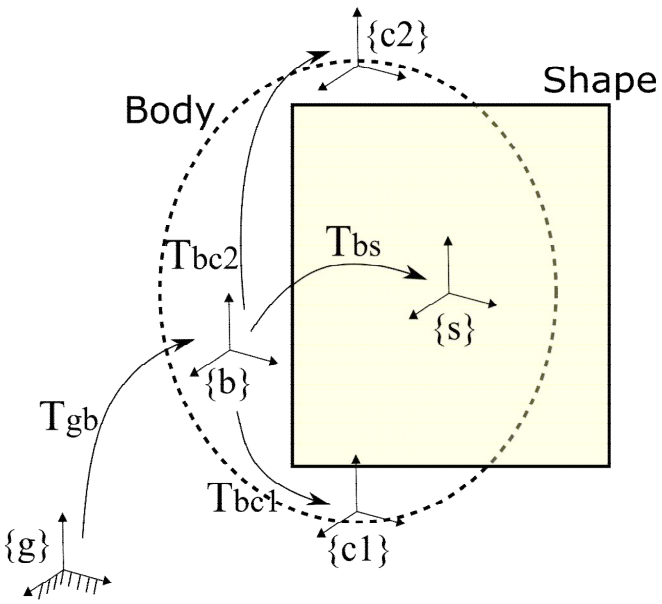
Romas (Robot Module Assembler) 설명서

version 0.1

1. Block의 주요구성요소
   가. Block의 개념

하나의 Block은 개념적으로 아래와 같이 구성된다.



- {g} : 고정좌표계
- Body : Block을 나타내는 강체(Rigid Body)이다. Body 자체는 아무런 형상을 가지고 있지 않다.
- {b} : block에 부착된 이동좌표계.
- T_{gb}: {g}에서 {b}로 향하는 동차변환행렬. 공간상에서 block의 위치와 자세를 표현한다.
- Shape : Block의 형상을 나타낸다.
- {s} : shape에 부착된 이동좌표계
- T_{bs}: {b}에서 {s}로 향하는 동차변환행렬. Shape이 그려지는 위치와 자세를 표현하여 Body에 대하여 상대적인 값이다.
- {c1}, {c2} : Body에 존재하는 Contact Point를 나타내는 이동좌표계이다. Contact Point는 이 Block이 다른 Block과 결합할 가능성이 있는 위치를 의미한다. 하나의 Block에서 Contact Point는 여러개 존재할 수 있으며, 위의 예에서는 2개가 존재하므로 각 {c1}, {c2}로 표시하였다.
- T_{bc1}: {b}에서 {c1}으로 향하는 동차변환행렬. Block에 존재하는 첫번째 Contact Point의 위치와 자세를 표현하며 Body에 대하여 상대적인 값이다.
- T_{bc2}: {b}에서 {c2}으로 향하는 동차변환행렬. Block에 존재하는 두번째 Contact Point의

위치와 자세를 표현하며 Body에 대하여 상대적인 값이다.

나. Contact Point

Contact Point란, Block과 Block을 결합할 수 있는 위치를 의미한다. 즉, 하나의 Block에 존재하는 여러개의 Contact Point에서 임의의 하나를 선택하고(이를 cp1이라 한다), 다른 Block에 존재하는 여러개의 Contact Point에서 또다른 임의의 하나를 선택하면(이를 cp2라 한다) cp1과 cp2를 표현하는 좌표계의 상대적인 배치에 따라서 두 개의 Block이 결합하는 것이다.

Contact Point는 'Assembly'과 'Joint'의 2가지 속성중의 하나를 반드시 가진다. 이 속성은 Contact Point가 다른 Block과 결합시에 상대적인 움직임을 가지는지 아닌지에 따라 결정된다. 서로 다른 Block에 속하는 두 개의 Contact Point가 결합되는 경우, 두 Contact Point의 속성에 따라서 결합방법의 선택범위가 달라진다.

- 'Assembly'속성의 Contact Point와 'Assembly'속성의 Contact Point가 결합되는 경우 : 이 결합지점은 고정관절(Fixed Joint)가 된다. 즉, 두 개의 Block은 고정관절에 의해서 연결된다. 그 결과 두 개의 Block은 어떤 경우에도 상대적인 움직임이 없다.
- 'Joint'속성의 Contact Point와 'Assembly'속성의 Contact Point가 결합되는 경우 : 이 결합지점은 회전관절(Revolute Joint) 또는 미끄럼관절(Prismatic Joint)가 된다. 즉, 두 개의 Block은 회전관절이나 미끄럼관절에 의해서 연결되고 Robot Module Assembler의 사용자가 조립시에 회전관절 또는 미끄럼관절을 선택할 수 있다. 그 결과 두 개의 Block은 1자유도의 상대적인 움직임을 가질수 있다.
- 'Joint'속성의 Contact Point와 'Joint'속성의 Contact Point가 결합되는 경우 : 이러한 결합은 성립되지 않는다. Robot Module Assembler는 이렇게 두 개의 Contact Point가 선택되는 경우, 어떠한 작업도 하지 않는다.

2. Block 표현 파일의 형식

Block은 XML형식의 텍스트 파일에 의하여 표현된다. 아래는 그 예시와 각 요소(XML Element)및 속성(XML Attribute)에 관한 설명이다.

```
<Block id='006' name='BaseD' isBaseBlock='true' isToolBlock='false' tag='basic'>
    <shape pose="0;0;0;-90;0;0" value='body006.obj'/>
    <diffuse value='diffuse.png'/>
    <normal value='normal.png'/>
    <thumbnail value='default_thumb_pic006.png'/>
    <contactPoints>
        <contact pose='0;0;0.17;45;0;0' type='joint' rotAxis='z' zeroAxis='x'/>
    </contactPoints>
    <mass value='6.95'/>
    <com x='0' y='0' z='0.04'/>
    <inertia Ixx='0.02' Iyy='0.02' Izz='0.03' Ixy='0' Iyz='0' Ixz='0'/>
</Block>
```

- <Block> 요소 : 하나의 Block은 하나의 <Block>과 </Block>쌍 안에 존재한다. <Block>과 </Block> 내부에 Block의 특징이 모두 묘사된다.

- 속성

* id : Block의 고유 아이디를 의미한다.

* name : Block의 이름을 의미한다. Robot Module Assembler에서 Block을 불러들이는 경우 Block Candidate 화면에 이 이름이 표시된다.

* isBaseBlock : true인 경우 이 Block은 베이스로만 사용할 수 있다. false인 경우 이 Block은 베이스로는 사용할 수 없다.

* isToolBlock : true인 경우 이 Block은 툴로만 사용할 수 있다. false인 경우 이 Block은 툴로는 사용할 수 없다.

* tag : 여러개의 Block에 적힌 tag 속성이 동일한 경우, 이 Block들은 tag속성으로 묶인 Block들이 된다. 즉, Block이 속한 집단(Group)을 의미한다. 만일 이름과 아이디가 동일한 Block이 있는 경우 tag가 다르면 다른 Block으로 취급한다. tag마저 같은 경우, 에러메세지를 출력한다.

● <Shape> 요소 : Block의 형상을 표시하는데 필요한 정보를 담고 있다.

- 속성

* pose : 위 Block의 구조도에서 T_{bs}를 의미한다. 3개 또는 6개의 값을 가질 수 있다.

3개의 값을 가지는 경우 : 예를 들어 “1;0.5;1”인 경우 위치만을 3차원 벡터 (x, y, z)로 표현한 것이다. 위치의 단위는 미터(meter)이다.

6개의 값을 가지는 경우 : 예를 들어 “1;0.5;1;45;0;0”인 경우, 앞의 3개의 숫자는 위치를 3차원벡터(x,y,z)로 표현한 것이고 뒤의 3개의 숫자는 자세를 Roll, Pitch, Yaw로 표현한 것이다. 위치의 단위는 미터(meter)이고, Roll, Pitch, Yaw의 단위는 도(degree)이다.

* value : Block의 형상을 나타내는 3차원 모델링 파일의 이름을 적는다. 여기서는 obj 포맷의 모델링 파일형식이 사용되었다.

● <diffuse> 요소 : diffuse 광원에 반응하는 텍스쳐(texture)에 관한 정보이다.

- 속성

* value : 텍스처 파일의 이름을 가진다.

● <normal> 요소 : 광원에 대한 법선(normal)정보에 관여하는 텍스쳐(texture)에 관한 정보

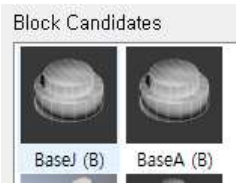
이다.

- 속성

* value : 텍스처 파일의 이름을 가진다.

- <thumbnail> 요소 : Block의 외형을 보여주는 작은 그림파일정보이다. Block Candidate에 여러개의 Block이 있는 경우, 원하는 형상의 Block을 손쉽게 선택하기 위해서 제공된다.

* value : 그림파일의 이름을 가진다.

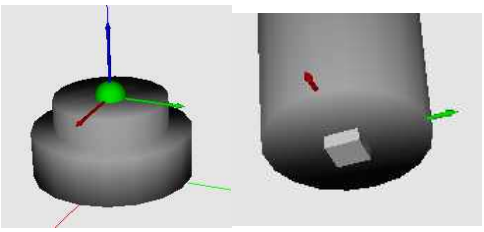


위 그림에서 block의 형상을 보여주는 작은 사각형의 그림들이 썸네일(thumbnail)이다.

- <contactPoints> 요소 : 여러개의 contact Point의 정보를 가지고 있다. <contactPoints>와 </contactPoints>의 사이에 있는 여러개의 <contact>요소가 각 Block이 가지고 있는 여러개의 contact Point를 의미한다.
- <contact> 요소 : 1개의 contact Point의 정보를 가지고 있다.

- 속성

* type 속성 : Contact Point가 'Assembly' 속성을 가지는지 'joint' 속성을 가지는지 보여준다. 'assembly'속성을 가지는 Contact Point의 경우 삼차원공간에서 정육면체로 표현되고, Joint 속성을 가지는 Contact Point의 경우 삼차원 공간에서 구로 표현된다.

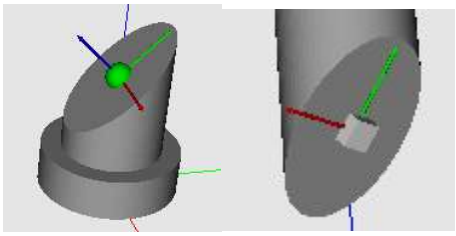


* pose : 위 Block의 구조도에서 T_{bc1}, T_{bc2}를 의미한다. 3개 또는 6개의 값을 가질 수 있다.

3개의 값을 가지는 경우 : 예를 들어 "1;0.5;1"인 경우 위치만을 3차원 벡터 (x, y, z)로 표현한 것이다. 위치의 단위는 미터(meter)이다.

6개의 값을 가지는 경우 : 예를 들어 "1;0.5;1;45;0;0"인 경우, 앞의 3개의 숫자는 위치를 3차원벡터(x,y,z)로 표현한 것이고 뒤의 3개의 숫자는 자세를 Roll, Pitch, Yaw로 표현한 것이다. 위치의 단위는 미터(meter)이고, Roll, Pitch, Yaw의 단위는 도(degree)이다.

* rotAxis 속성 : Contact Point의 좌표계 중에서 회전축으로 사용될 축을 지정한다. 'x', '-x', 'y', '-y', 'z', '-z'중 하나의 값을 입력하는데, 반드시 Block의 안에서 바깥쪽으로 나오는 방향으로 지정하여야 한다. rotAxis 속성은 Block간의 결합시에 Revolute Joint나 Prismatic Joint가 움직이는 축으로 사용된다.



왼쪽의 예에서는 z축(파란색 화살표)가 rotAxis가 되어야 하고, 오른쪽의 예에서는 x축(붉은색 화살표)와 y축(연두색 화살표)의 위치로 보아 -z축이 rotAxis가 되어야 한다.

* zeroAxis 속성 : Contact Point의 좌표계 중에서 원점의 기준으로 사용될 축을 지정한다. ‘x’, ‘-x’, ‘y’, ‘-y’, ‘z’, ‘-z’중 하나의 값을 입력하는데, rotAxis와 일치하거나 평행해서는 안된다.

● <mass> 요소 : Block의 질량정보를 보여준다.

- 속성

* value 속성 : kg단위로 Block의 질량을 기입한다.

● <com> 요소 : Block의 무게중심를 보여준다.

- 속성

* x, y, z 속성 : meter단위로 무게중심 벡터를 기입한다. 기준좌표계는 {b}이다.

● <inertia> 요소 : Block의 관성질량을 보여준다.

- 속성

* Ixx, Iyy, Izz, Ixy, Iyz, Ixz 속성 : kgm^2단위로 관성모멘트를 기입한다. 기준좌표계는 {b}이다.

3. Assembly의 개념

Assembly란 여러개의 Block이 결합된 조립체이고 로봇의 다른 표현이라고도 할 수 있다. 물론 1개의 Block도 Assembly가 될 수 있다.

가. Base Block

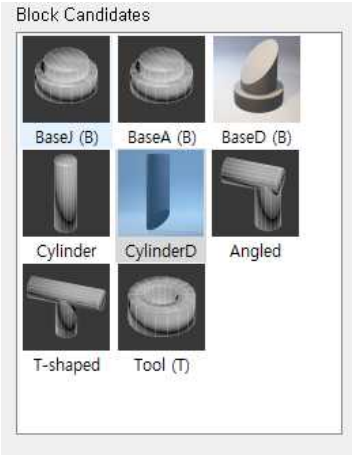
Assembly에는 반드시 1개의 Base Block이 존재한다. Base Block은 Block중에서 지면에 고정되는 Block을 의미하는데, Assembly내에서는 지면과 Block을 연결해주는 역할을 하는 Block이라고 볼수 있다.

Base Block은 Block 표현 파일(XML)에서 isBaseBlock 속성이 true로 설정된 Block이며, Robot Module Assembler에서는 아래와 같이 Block의 이름 옆에 (B)라고 표현된다는 점에 다른 Block과 다르다.

나. Tool Block

Assembly에는 반드시 1개의 Tool Block이 존재한다. Tool Block은 Assembly내에서 공구(Tool, End-Effector)가 장착될 수 있는 Block을 의미하며, Assembly 내에서는 가장 끝단에 존재한다.(즉, Base Block과는 가장 멀리 존재한다)

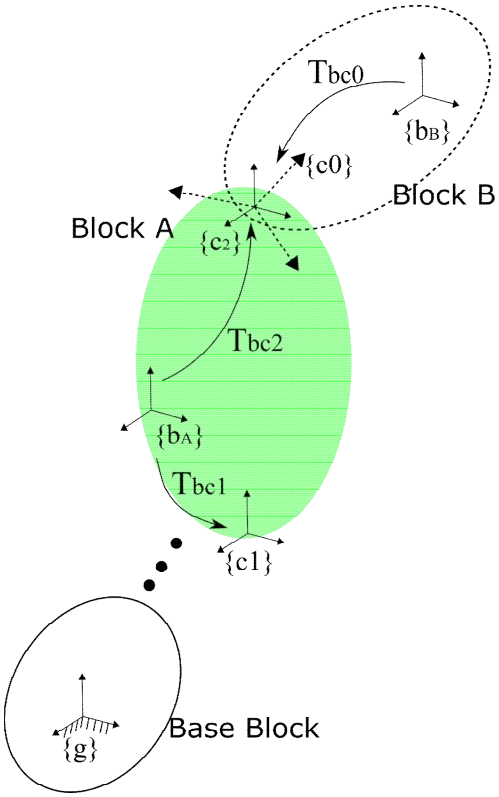
Tool Block은 Block 표현 파일(XML)에서 isToolBlock 속성이 true로 설정된 Block이며, Robot Module Assembler에서는 아래와 같이 Block의 이름 옆에 (T)라고 표현된다는 점에 다른 Block과 다르다.



Base Block에는 Block 이름 옆에 (B)가, Tool Block에는 Block 이름 옆에 (T)가 표시된다.

다. Assembly의 개념도

여러개의 Block이 연결된 Assembly는 개념적으로 아래와 같이 도시할 수 있다. 아래 그림은 Block A(연두색 타원)가 Block간의 결합을 통하여 Base Block까지 연결되어 있고, Block B (점선 타원)가 Block A와 결합되어 있는 Assembly를 보여준다.



- {g} : 공간상의 고정좌표계이자 Base Block의 위치와 자세를 표현하는 이동좌표계이다.
- $\{b_A\}$: Block A에 부착된 이동 좌표계
- $\{b_B\}$: Block B에 부착된 이동 좌표계
- Parent Block : 두 개의 결합된 Block에서 Base Block에 더 가까이 위치한 Block을

Parent Block이라고 한다. 위의 예에서는 Block A가 Parent Block이다.

- Child Block : 두 개의 결합된 Block에서 Base Block과 더 멀리 위치한 Block을 Child Block이라고 한다. 위의 예에서는 Block B가 Child Block이다.
- $\{c_2\}$: Block A에 위치한 Contact Point중의 하나
- $\{c_0\}$: Block B에 위치한 Contact Point중의 하나. 위의 예에서는 Block A(Parent Block)의 $\{c_2\}$와 Block B의 $\{c_0\}$가 서로 만나서 두 개의 Block이 결합되었다. 여기서 보듯, Block이 결합될 때에는 두 개의 Contact Point는 원점이 일치한다.
- T_{bc2}: $\{b_A\}$에서 $\{c_2\}$으로 향하는 동차변환행렬.
- T_{bc0}: $\{b_B\}$에서 $\{c_0\}$으로 향하는 동차변환행렬. Block에 존재하는 두번째 Contact Point의 위치와 자세를 표현하며 Body에 대하여 상대적인 값이다.

라. Block의 연결절차

Block과 Block이 서로의 Contact Point에서 만나 결합하는 과정은 아래와 같다.

1. 두 Block에 각 존재하는 Contact Point는 원점이 일치한다. 따라서 Contact Point간의 위치와 자세 관계는 회전행렬(Rotation Matrix)로 표시된다.
2. Parent Block에 속한 Contact Point의 rotAxis 축과, Child Block에 속한 Contact Point의 rotAxis의 반대방향인 축을 서로 일치시킨다. 예를 들어 전자가 z축, 후자도 z축인 경우, Parent측과 Child측은 순서대로 z축과 -z축을 서로 일치시킨다.
3. Parent Block에 속한 Contact Point의 좌표축에서 1개의 축을 선택한다. 이 1개의 축은 rotAxis축과 같거나 평행해서는 안된다. 이 축을 Parent측의 Matched Axis라 한다. 마찬가지로 Child Block에 속한 Contact Point의 좌표축에서 1개의 축을 선택하는데 역시 rotAxis축과 같거나 평행해서는 안된다. 이축을 Child측의 Matched Axis라 한다. Parent 측의 Matched Axis와 Child 측의 Matched Axis를 일치시킨다.
4. 위 2.와 3.에서 좌표축 중 서로 일치하는 2개의 축을 지정하였으므로 두 좌표축간의 관계를 정할 수 있다. 이를 회전행렬로 표시한다.
5. 위에서 구한 회전행렬이 두 Block간의 상대적인 자세차이이다.
6. 두 Contact Point가 Joint - Assembly간의 결합인 경우 두 Contact Point의 원점이 일치하는 위치에 회전관절(Revolute Joint)또는 미끄럼 관절(Prismatic Joint) 구속조건을 추가한다. 회전관절의 회전축방향 또는 미끄럼관절의 미끄럼방향은 Parent측의 rotAxis를 사용한다.
7. 두 Contact Point의 zeroAxis간의 차이(각도)를 Difference로 정의하면, Difference를 계산할 수 있다.

4. Assembly 표현 파일의 형식

Assembly는 XML형식의 텍스트 파일에 의하여 표현된다. 아래는 그 예시와 각 요소(XML Element)및 속성(XML Attribute)에 관한 설명이다.

```
<Assembly>
   <Part id="0" parentId="-1" blockId="006" parentContactPointIndex="-1" childContactPointIndex="-1"
   parentMatchedAxis="x" childMatchedAxis="x" jointType="fixed"/>
   <Part id="1" parentId="0" blockId="007" parentContactPointIndex="0" childContactPointIndex="0"
   parentMatchedAxis="-y" childMatchedAxis="y" jointType="revolute" offset="0"/>
   <Part id="2" parentId="1" blockId="002" parentContactPointIndex="1" childContactPointIndex="0"
   parentMatchedAxis="y" childMatchedAxis="x" jointType="revolute" offset="0"/>
</Assembly>
```

- <Assembly> 요소 : 하나의 Assembly는 하나의 <Assembly>과 </Assembly>쌍 안에 존재한다. <Assembly>와 </Assembly> 내부에 Assembly의 특징이 모두 묘사된다.
- <Part> 요소 : Assembly에 속하는 하나의 Block에 관한 정보와 이 Block에 이웃하는 Block간의 결합관계가 서술된다.

-속성

* id : Part의 아이디이다. 이 id는 Base Block에서부터 Tool Block을 향하여 0으로 시작하여 1씩 증가한다.

* parentid : 현재 Part에 속하는 Block의 Parent Block의 아이디이다. 이 아이디가 -1인 경우 지표면을 의미한다.

* blockid : 현재 Part에 속하는 Block의 아이디이다.

* parentContactPointIndex : Parent Block의 Contact Point중에서 선택된 것의 인덱스를 의미한다. 0부터 시작하는 정수이며, 인덱스가 -1인 경우 사용되지 않아 의미가 없다. 예를 들어 Parent Block의 Contac Point 정보가 아래와 같은 경우, parentContactPointIndex가 2이면 pose='0.25;0;0.3'의 속성을 가지는 세 번째 contact point를 의미한다.

```
<contactPoints>
   <contact pose='0;0;0' type='assembly' rotAxis='-z' zeroAxis='x'/>
   <contact pose='-0.25;0;0.3' type='joint' rotAxis='-x' zeroAxis='y'/>
   <contact pose='0.25;0;0.3' type='joint' rotAxis='x' zeroAxis='y'/>
</contactPoints>
```

* childContactPointIndex : 현재 part에 속하는 Block의 Contact Point중에서 선택된 것의 인덱스를 의미한다. 0부터 시작하는 정수이며, 인덱스가 -1인 경우 사용되지 않아 의미가 없다.

* parentMatchedAxis : Parent Block의 선택된 Contact Point의 좌표축에서 일치시키려고 하는 축을 의미한다.

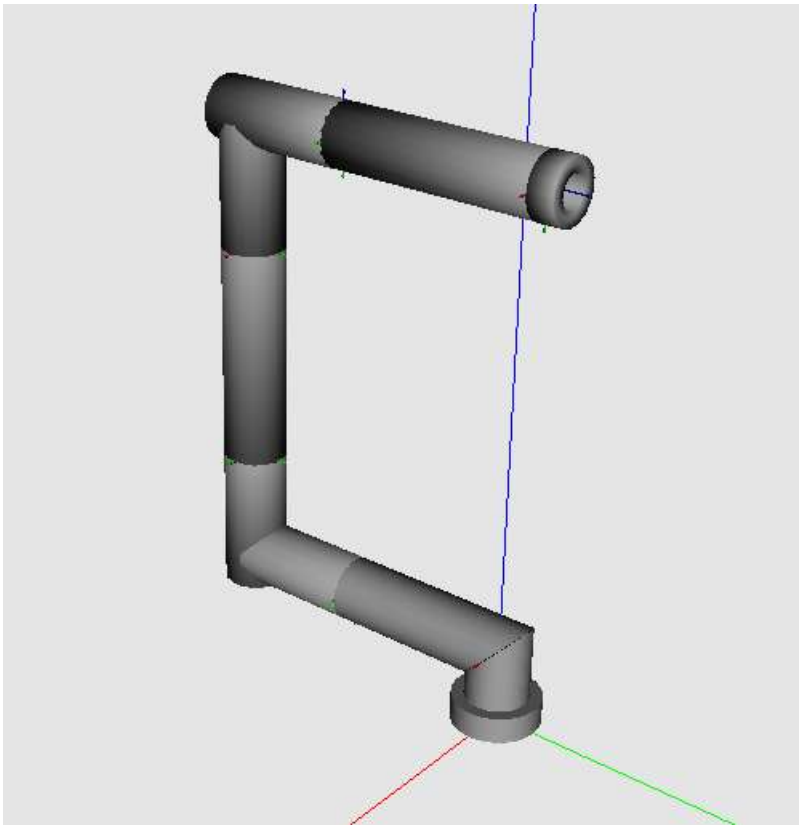
* childMatchedAxis : 현재 part에 속하는 Block의 선택된 Contact Point의 좌표축에서 일치시키려고 하는 축을 의미한다. parentMatchedAxis와 childMatchedAxis가 서로 일치하는 방향으로 두 개의 Block이 결합한다.

* jointType : 'fixed', 'revolute', 'prismatic'중의 하나의 값을 가진다. 순서대로 고정관절, 회전관절, 미끄럼관절이라는 구속조건으로 두 개의 Block을 결합함을 의미한다.

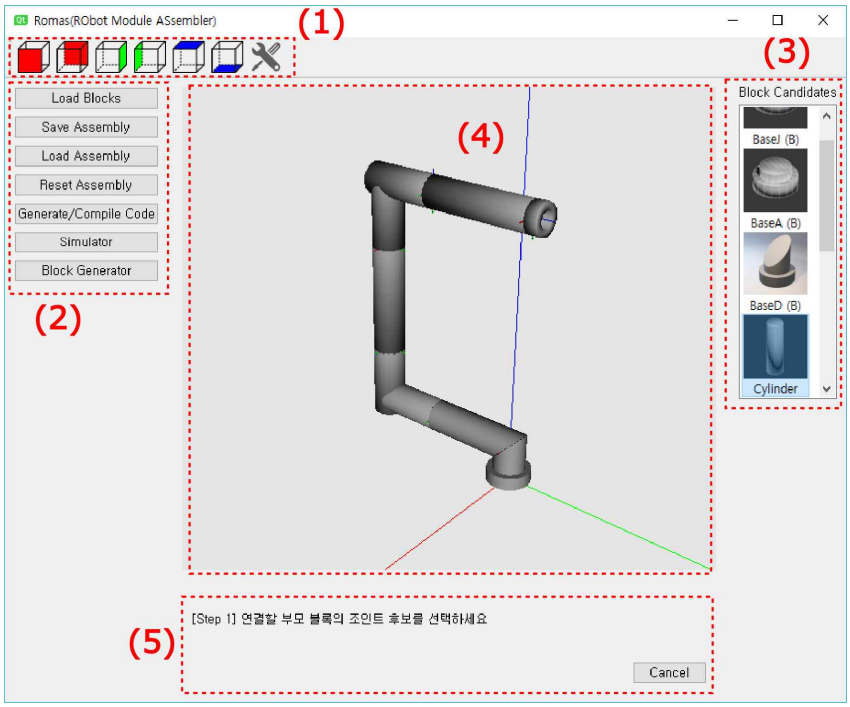
* offset : 회전관절이나 미끄럼 관절의 경우 원점을 변경하려는 목적으로 offset을 설정할 수 있다.

5. Assembly의 예.

아래는 조립이 완료된 6축 Assembly의 모습니다.



6. Robot Module Assembler 화면 구성

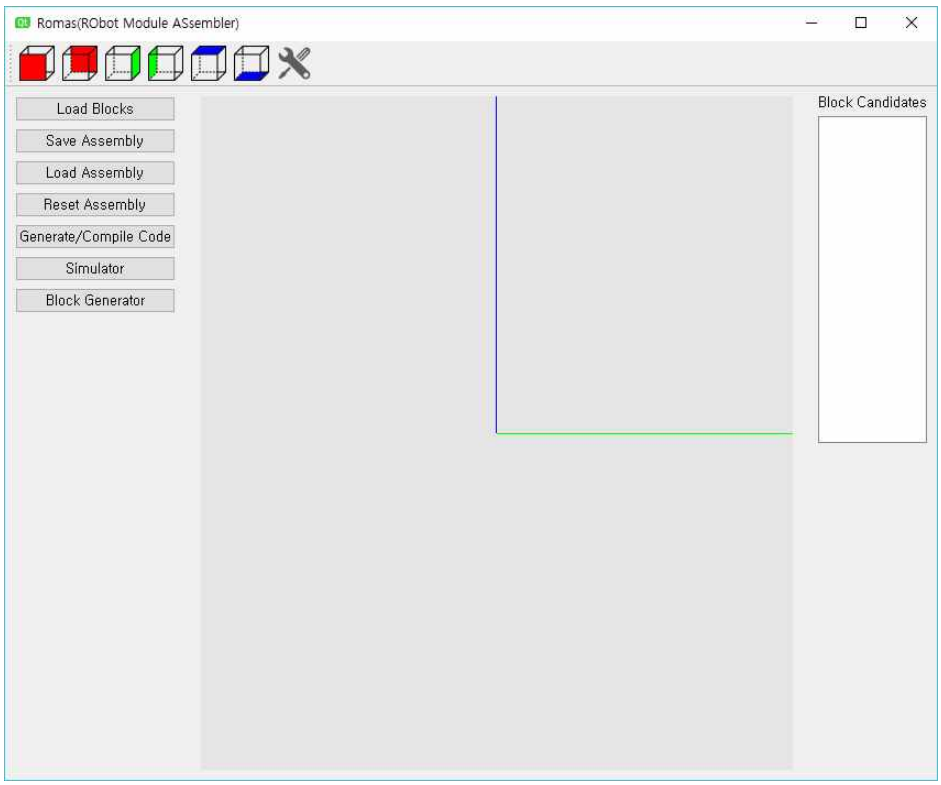


(1) ToolBar : 화면조작을 위한 버튼과 설정을 위한 버튼이 있다. 화면조작버튼의 경우 (4)에 표시된 가상의 삼차원공간을 X축, Y축, Z축 각방향에서 볼수 있도록 카메라 시선을 변경하는

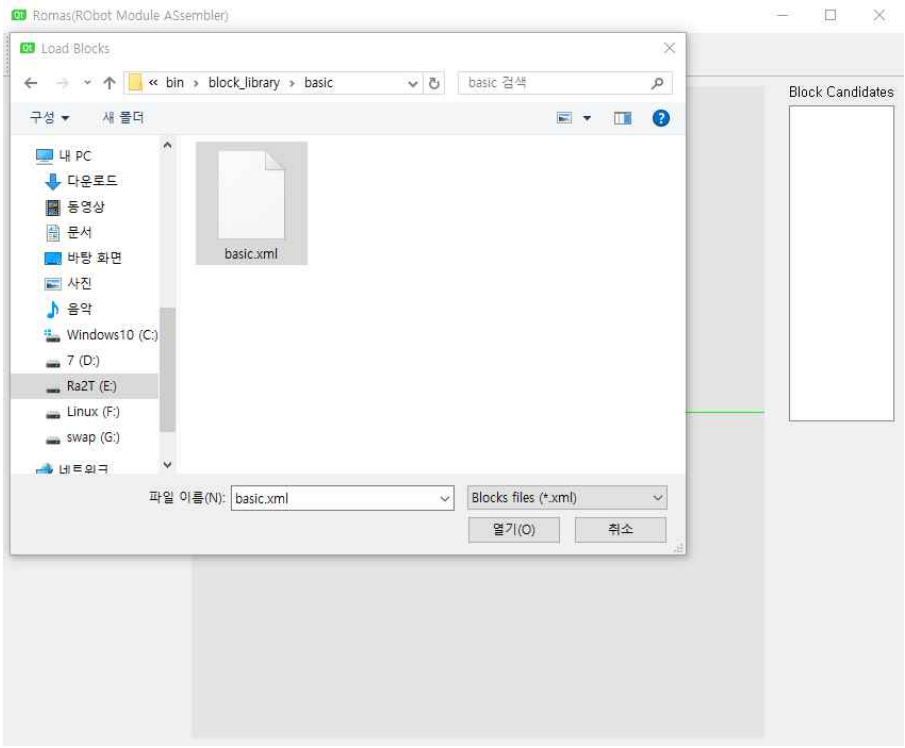
기능을 가지고 있다.
(2) Main Menu : Block을 불러오고 Assembly을 저장하고 소스코드를 생성 및 컴파일하는 주요 기능을 담은 버튼이 있다.
(3) Sub Menu : Block 표현 파일(XML)을 불러온 경우, 파일에 담긴 블록들이 도시된다. 이를 Block Candidates라고 부르며, 여기에 도시된 블록들을 더블클릭하여 (4)의 삼차원 공간으로 가져올 수 있다.
(4) Assemble Space : 사용자와 상호작용하에 Block을 조립할 수 있는 가상의 삼차원공간이다.
(5) Assemble Menu : 조립절차가 진행중일때만 화면에 표시되는 메뉴이다. 조립절차에 따라서 Step1부터 Step4까지 순차적으로 메뉴가 나타나며, 순서를 되돌리거나 중간에 취소하는 것도 가능하다.

7. Robot Module Assembler 사용 방법

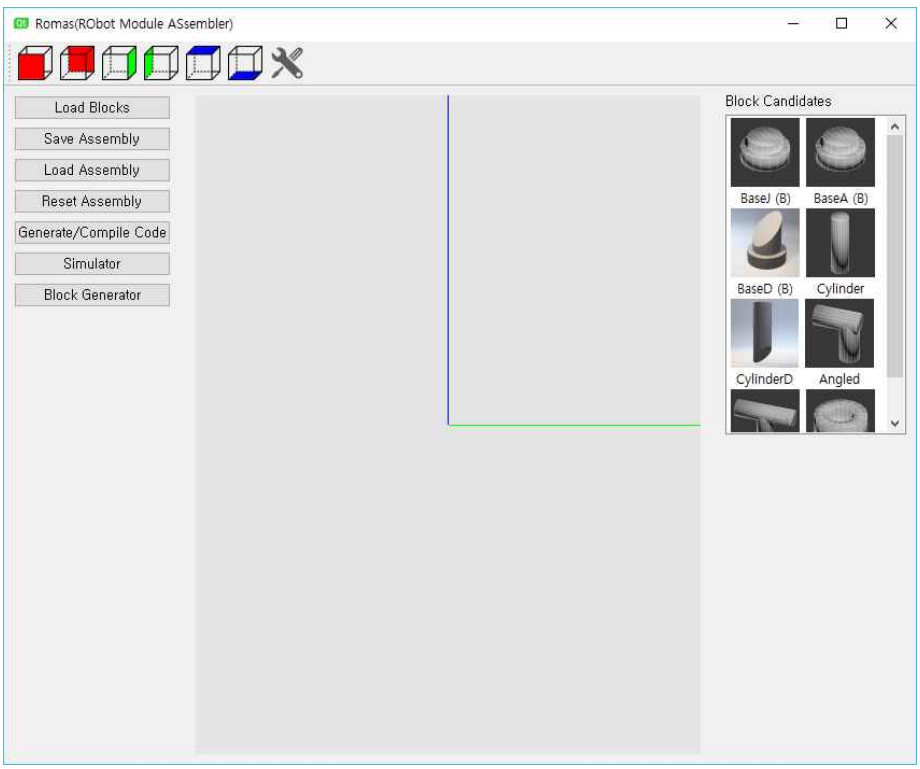
아래에서는 Block을 조립하는 과정을 화면 순서대로 설명한다.
(1) Robot Module Assembler를 실행한다.



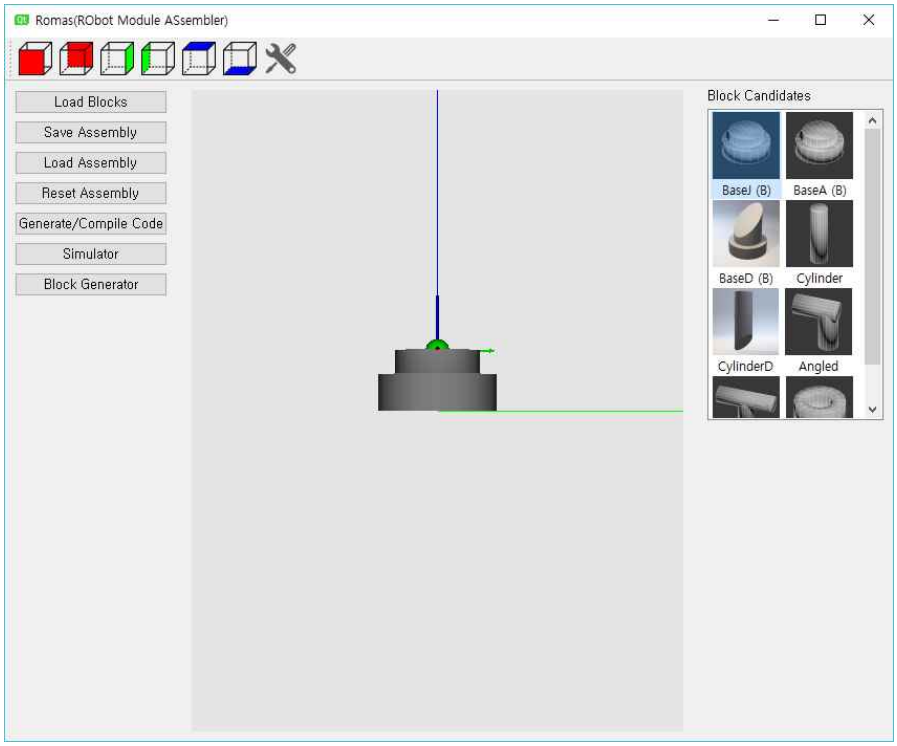
(2) “Load Blocks” 버튼을 눌러서 Block 표현 파일을 불러온다.



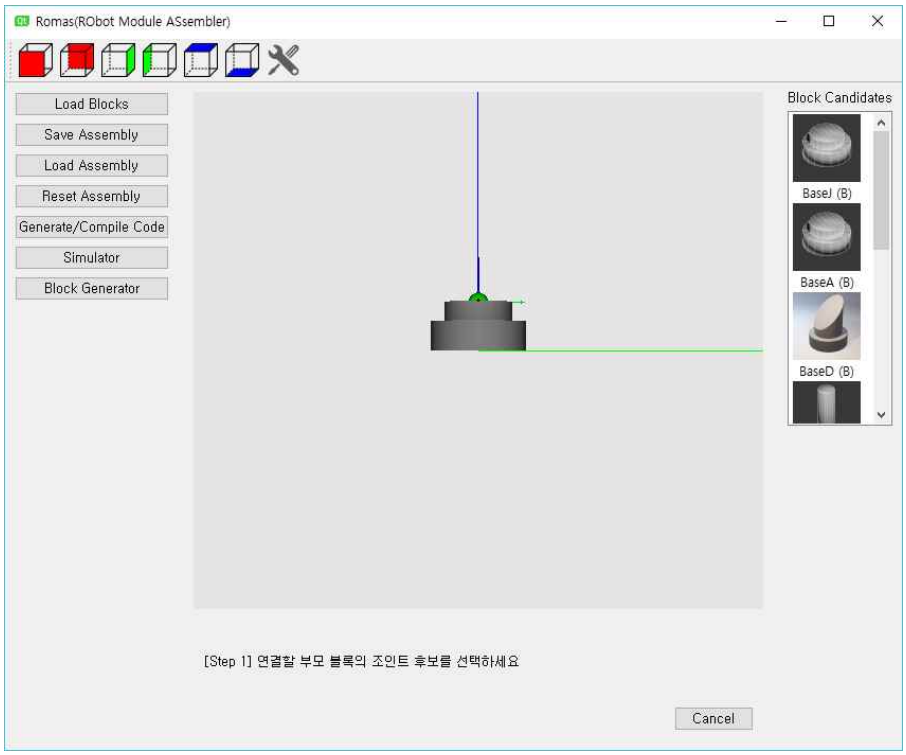
(3) 오른쪽 Sub Menu에 Block Candidates가 도시된다.



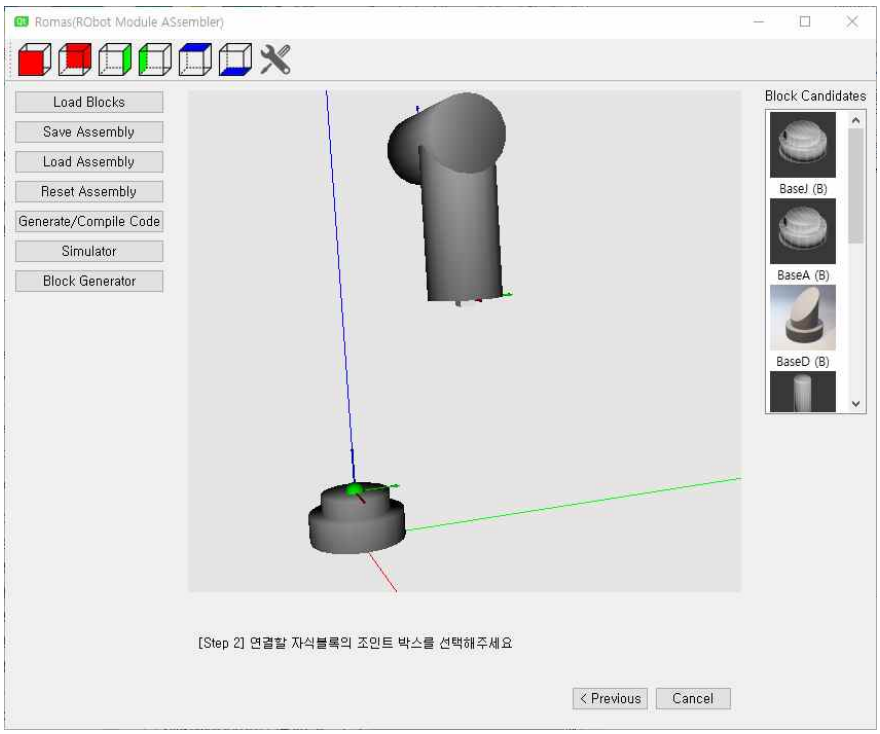
(4) Base Block중에서 하나를 선택하여 더블클릭하면 Assemble Space에 Block이 등장한다.



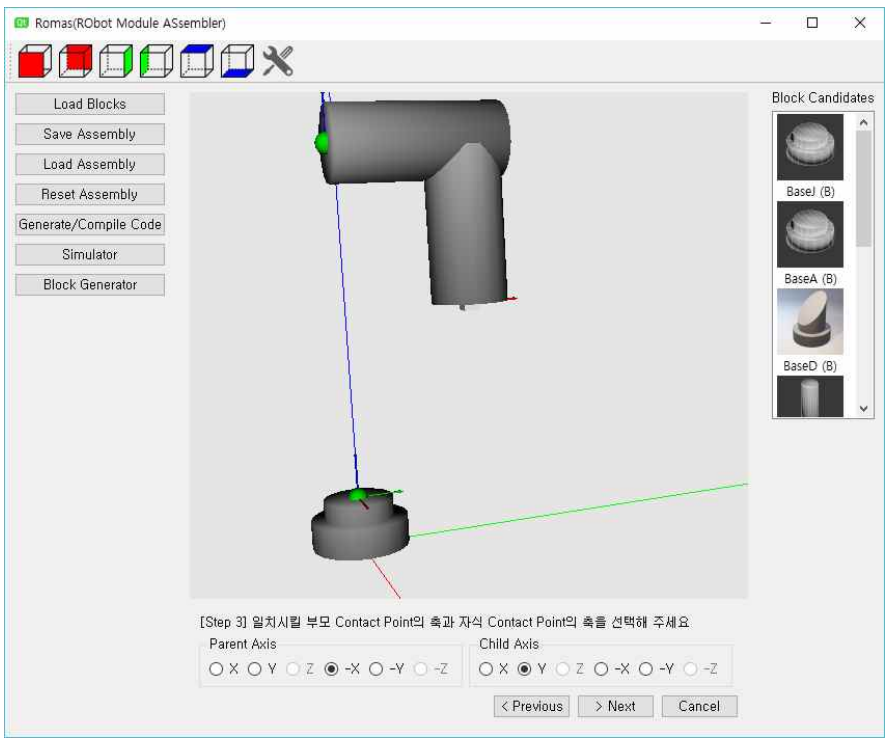
(5) Block Candidates중에서 Base Block에 결합할 Block을 선택하면 조립절차가 진행된다. Assemble Menu에서 Step1 메뉴가 등장한다. 연결할 Contact Point를 클릭한다.



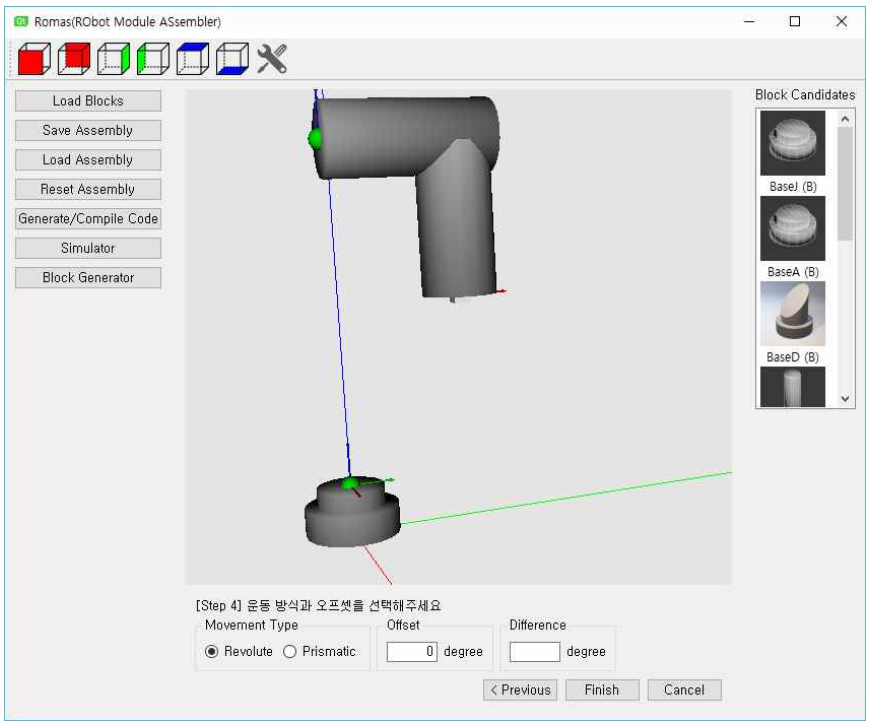
(6) Child Block에서 연결할 Contact point를 클릭한다.



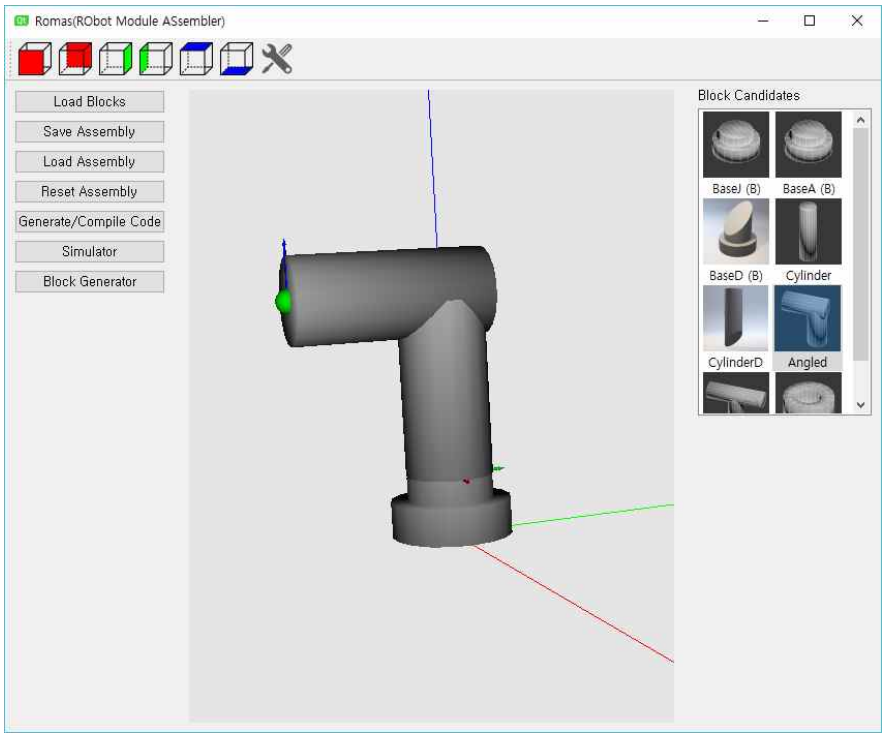
(7) Parent Block측과 Child Block측에서 서로 일치시킬 Matched Axis를 선택한다.



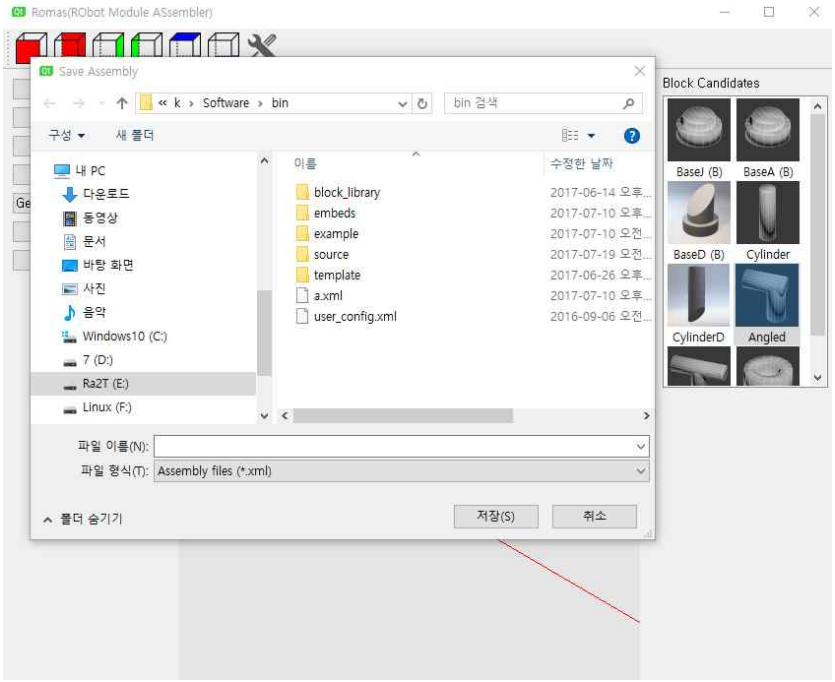
(8) 관절의 종류와 Offset을 지정하고 Finish 버튼을 누른다.



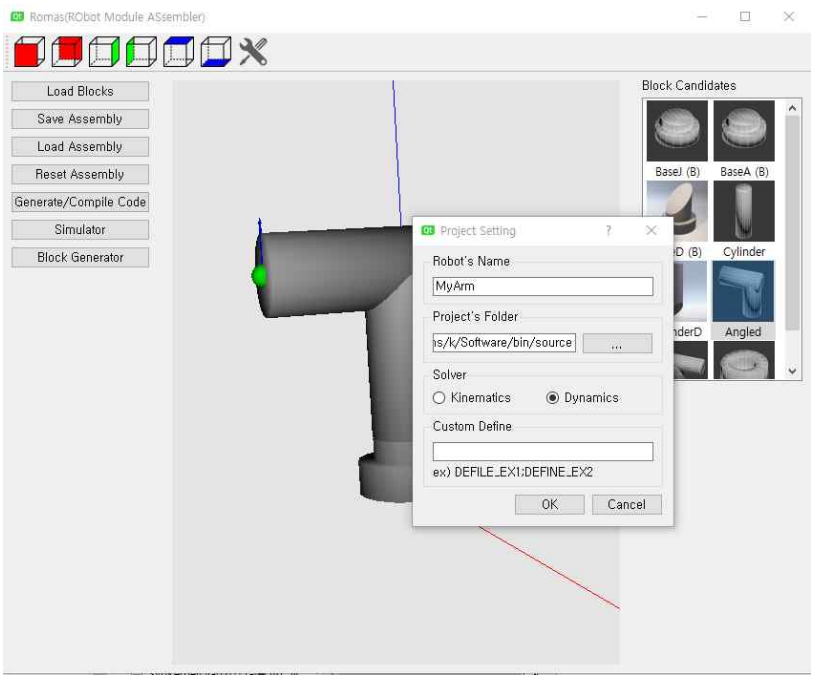
(9) 두 Block이 결합되어 하나의 Assembly가 되었다. 같은 방식을 반복하여 다자유도의 Assembly를 만들 수 있다.



(10) Main Menu에서 Save Assembly 버튼을 클릭하여 Assembly를 파일로 저장할 수 있다.



(11) Main Menu에서 Generate/Compile Code 버튼을 클릭하면 Robot 이름, 프로젝트가 저장될 폴더, 제어방식 등을 선택할 수 있는 대화상자가 나온다.



(12) 대화상자에서 OK를 누르면 소스코드가 생성되고 자동으로 컴파일이 진행된다. 컴파일이

끝나면 실행가능한 바이너리 파일이 생성된다.