KOSSA 한국공개소프트웨어협회 Korea Open Source Software Association

Open jig ware

새로운 로봇제어 표준 모델의 제시

팀명 : 오픈지그웨어

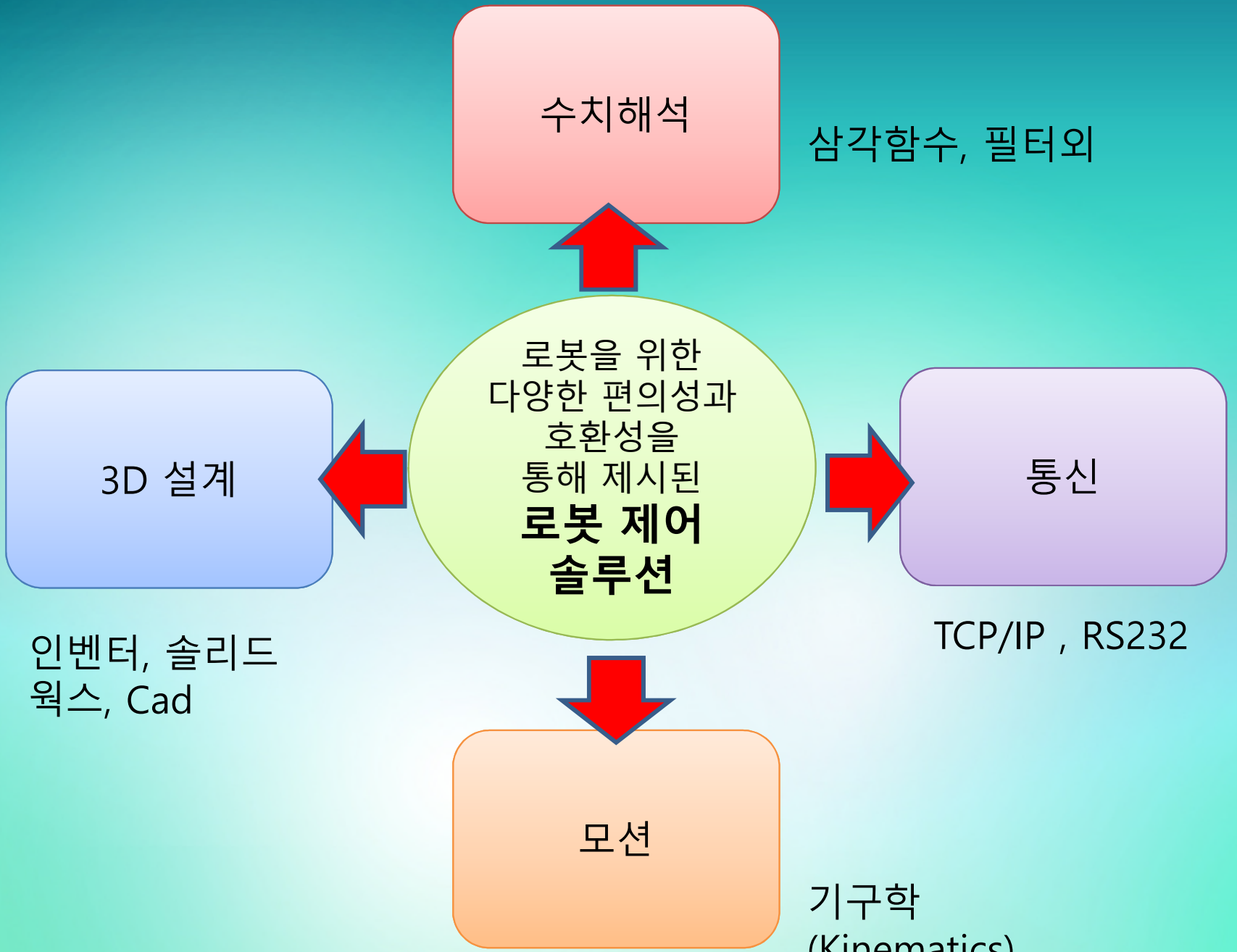
팀원 : 이현종,김주훈,이정환,박진수,장동준,김예승,이상혁,김세훈,온진욱

2016. 11. 11

Contents

OpenJigWare란?

I. OpenJigWare 란?



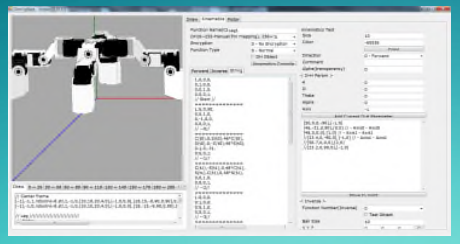
I. OpenJigWare 란?

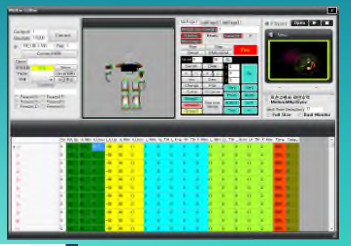
단지 DLL 일뿐... (OpenJigWare.dll)

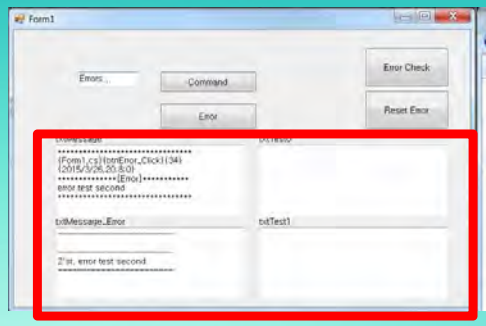
설치가 필요 없다. (DLL 참조만 필요)

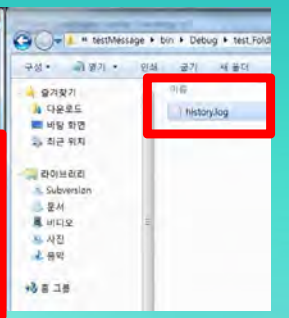
Windows 기반 C# 개발 환경
(부분적으로 라즈베리파이(리눅스)에서 부분적으로 포팅이 되어 있다.)

I. OpenJigWare 란?

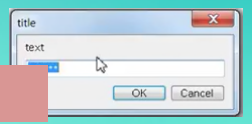








- COjw_00_Main.cs
- COjw_01_Struct.cs
- COjw_02_InputBox.cs
- COjw_03_Message.cs
- COjw_04_Convert.cs
- COjw_05_Math.cs
- COjw_06_Encrypt.cs
- COjw_07_File.cs
- COjw_08_Timer.cs
- COjw_09_Register.cs
- COjw_10_Kinematics.cs
- COjw_11_2D.cs
- COjw_12_3D.cs
- COjw_13_Serial.cs
- COjw_14_Socket.cs
- COjw_15_Mx28.cs
- COjw_16_EmguCv.cs
- COjw_17_Myo.cs
- COjw_18_Mouse.cs
- COjw_19_Herculex.cs
- COjw_20_Grid.cs
- COjw_21_Graph.cs
- COjw_22_DC2408.cs
- COjw_23_Joystic.cs
- COjw_24_System.cs
- COjw_25_Genibo.cs
- COjw_26_Streaming.cs
- COjw_xx_User.cs



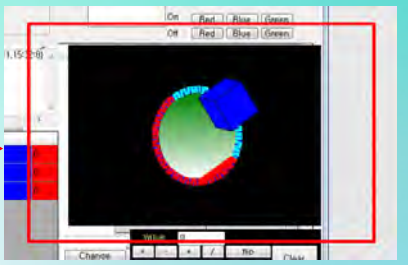
InputBox/Messge 기능

각종 변환 함수 모음

모든 수학/알고리즘 함수 모음

Forward / Inverse 관련(컴파일러 포함)
Matlab 과 유사한 구조를 갖는다.

2D 를 그려주는 함수, 3D도 가능



Socket/Serial 구현
(Simple): 유사한 구현방식

3D 를 그려주는 함수(OpenGL)

마우스 제어 관련

모터 제어 관련

간단한 타임 그래프(최소코드 3줄)

조이스틱 제어 관련

시스템 제어 관련(Shutdown, 중복실행제어 등)

영상 스트리밍(Jpeg, Mjpeg) 구현- 2줄

개발자 본인이 추가해서 사용할 Class

I. OpenJigWare 란?

- Education (교육)
  - 선생님과 학생도 사용할 수 있는 시스템

선생님

- 3D로 로봇 만들기, 수식 입력, 모터 정의 및 상관관계입력
- 다양한 로봇을 학생들에게 접하게 할 수 있고 로봇의 수식을 바로 적용가능
- 클릭하는 로봇의 부품에 따라 회전, 직교, Forward/Inverse 수식적용여부 선택가능
- Forward Kinematics 를 직접 눈으로 예를 들어 보이면서 설명이 가능
- D-H Notation 을 파라미터만으로 3D 모델링가능

학생

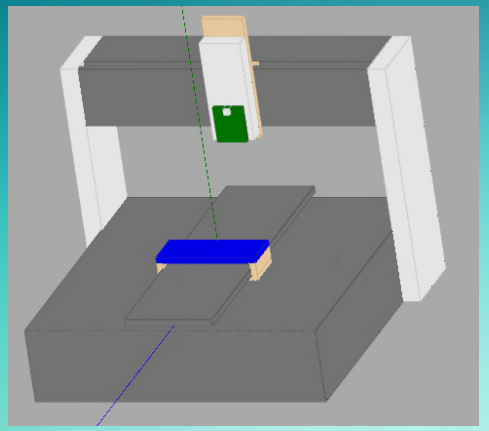
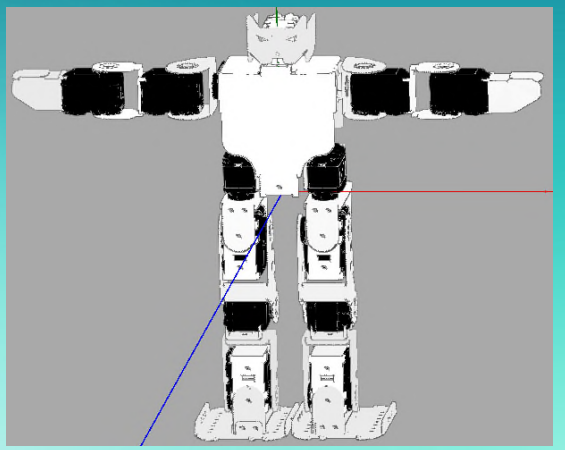
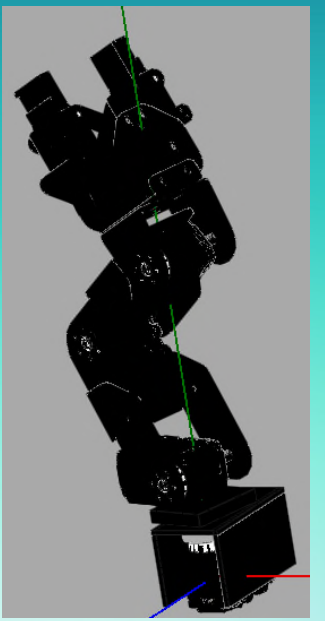
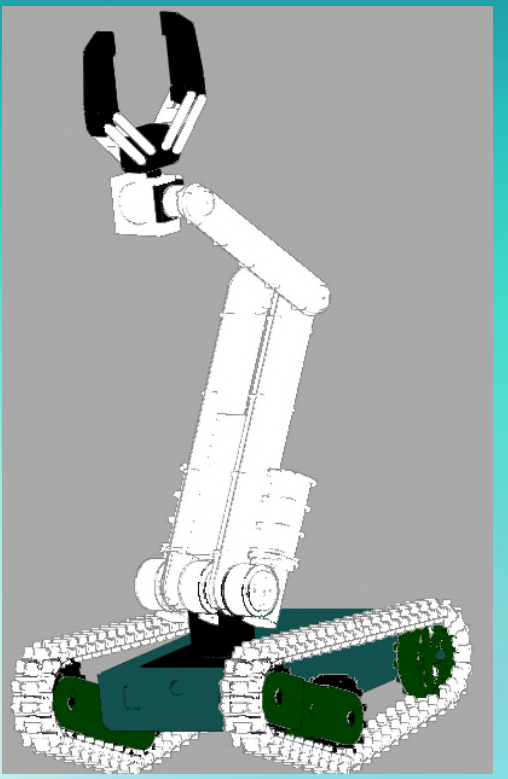
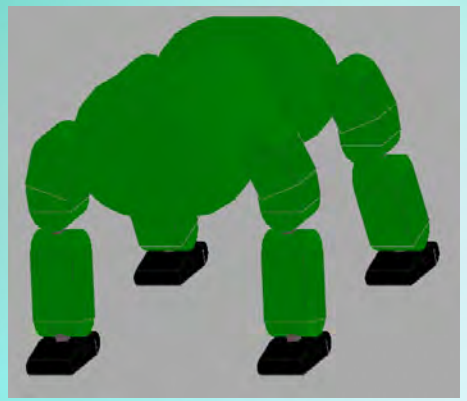
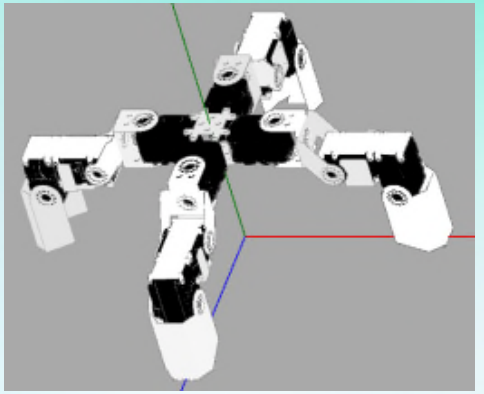
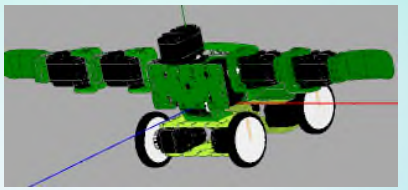
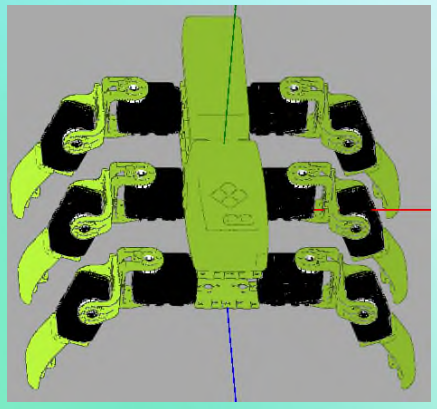
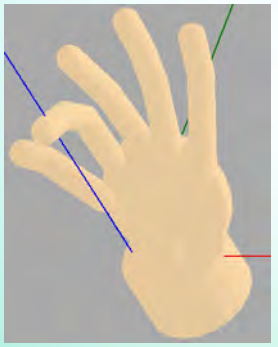
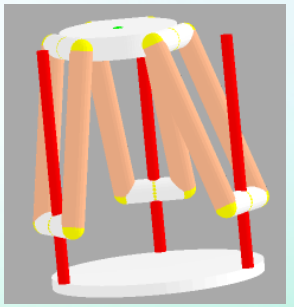
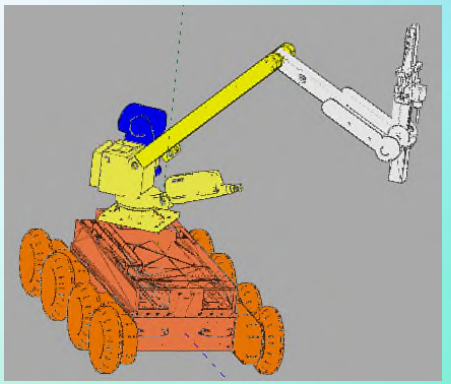
- C# 을 이용해서 간략히 5줄로 3D 로봇 시뮬레이터를 개발 프로그램에 넣을 수 있다 (실제 로봇과 연동 가능)
- 학생들이 직접, 구동방식에 상관없이 어떠한 종류의 로봇이라도 손쉽게 제어할 수 있는 SW 체계.

- Development (개발)
  - 개발자를 위한 최적화 시스템

- 다양한 로봇을 하나의 DLL 을 사용해 재사용이 가능
- 다양한 수식을 넣기 쉽고 하나의 로봇에 255개까지의 고속 수식 연산을 각 파트별로 넣는 것이 가능.
- Message History, Timer, Joystick, Graph, 암호화, File 저장/불러오기, 레지스터 등 실제 현업에서 잘 사용되는 기능이 정의 되어 있어 사용이 간편.
-3D 모델링의 사용이 간편하고 실제 모터 파라미터를 정의할 수 있어 실제 제어에 사용이 가능
- 프로그램의 수정을 모델링의 수정 만으로 가능하기 때문에 프로그램 유지보수에 훨씬 유리하다.

I. OpenJigWare 란?

< 실제 OpenJigWare 에서 모델링한 로봇들 >


개발 배경

개발동기

개발목표

개발환경

II. 개발 동기

로봇에 사용되는 수식은 너무 어렵다...
하지만 그래도 배워야 한다.

모션등을 만들기가 너무 어렵다.
모션을 쉽게 만들었으면 좋겠다.

로봇은 종류가 너무 많다.
로봇의 종류(바퀴, 관절, 직교, 델타형 등)에 상관없이 세상의 모든 로봇을 다 대응할 수있는 것이 없을까?

II. 개발 동기

다기능 로봇시뮬레이션

- 교육하기 어렵고 적용하기 어려운 Kinematics
- 복잡한 수식에 대한 SW 적용성
- 휴머노이드 / 바퀴형 / 직교좌표형태에 따른 SW

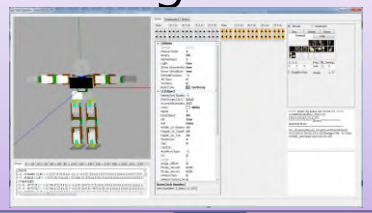
II . OpenJigWare 작성 흐름도

Using OpenJigware.dll

Education

Teacher

Modeling Tool 실행



Drawing(3D)

Motor 정의
(Limit, 기어비 클릭 이벤트 등)

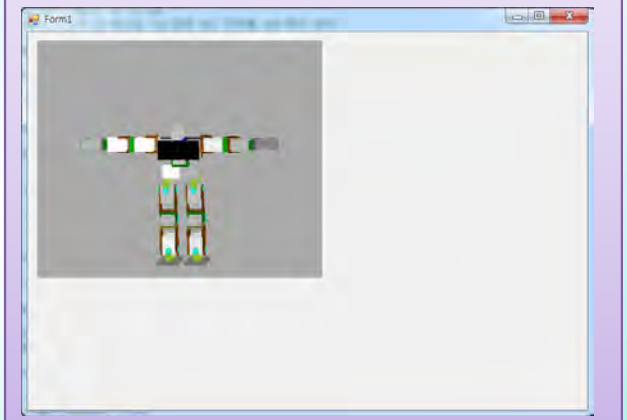
교육 하고자 하는 수식 입력

만들어진 파일 학생에게 전달

Student

Visual Studio C# 실행

프로그래밍 따라하기
(최소 코딩라인 5줄)



Development

Engineer

Modeling Tool 실행

제어할 타입(델타, 직교, 다관절, 바퀴형 외) 결정 후 Drawing(3D)

Motor 정의
(Limit, 기어비 클릭 이벤트 등)

사용하고자 하는 수식 입력

프로그램 제작

Like Iron man System

- Making the suit
  - https://youtu.be/xnE9ErFz3wc

II. 개발 동기

필요성

- Jig 개발에 최적화 되어 있다.
- 모션 제작 및 튜닝에 최적화
- 3D 모델링의 빠른 구현
- 제어기능의 빠른 구현
- 교육적 내용 포함
- 쉬운 사용 & 다른 Tool 과 접목 가능

ROS 가 있는데 이건 무슨 필요?

ROS

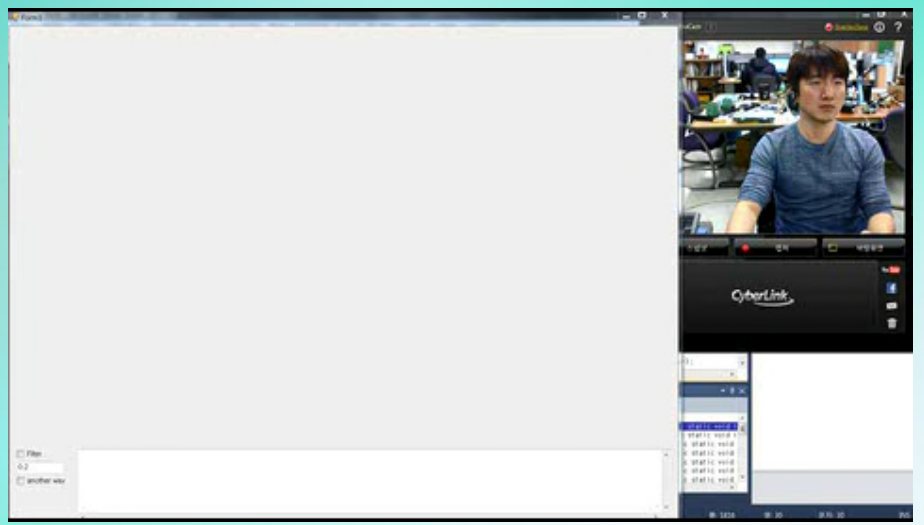
- 사용난이도 상
- 사용자 제한
- 사용환경 복잡

Open Jig

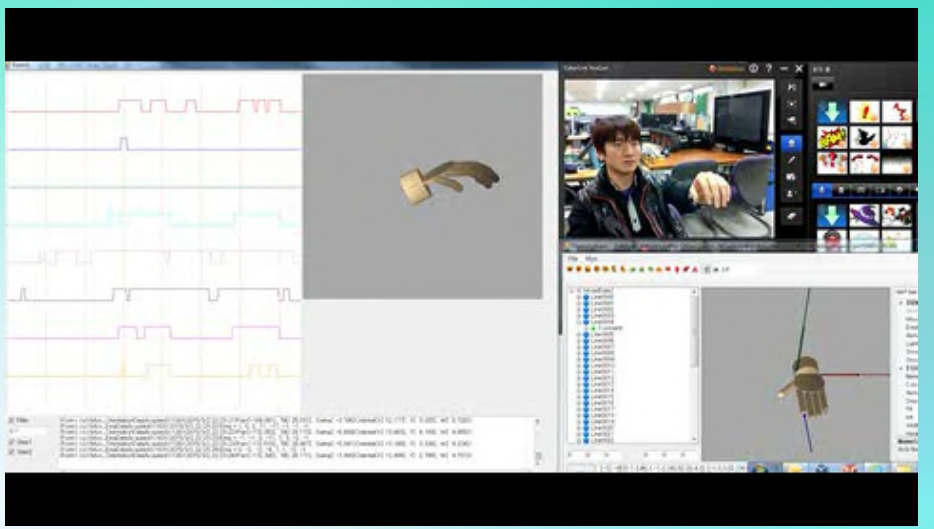
- 개발,교육용등 활용성
- 다양한 호환성
- GUI 기반 솔루션

II. 개발 환경

- Microsoft Visual Studio 2010
  - C# 개발환경 (with Tao frame work :OpenGL)



https://youtu.be/l_QerQFRaZk



https://youtu.be/GY4MUXbbW20

II. 개발 환경

Open Source

- **Source & DLL**

  [DLL - 사용만 할 거라면 여길 참조]
  https://github.com/ojw5014/Released_DLL
  [모델링 프로그램, 모션 프로그램등(DLL 내에서 실행가능하나 편의를 위해 공개)]
  https://github.com/ojw5014/Tool
  [Source]
  https://github.com/ojw5014/OpenJigWare
  [RaspberryPi - 라즈베리파이용 OpenJigWare]
  https://github.com/ojw5014/RaspberryPi

- **Manual**

  [동영상] www.youtube.com
  OpenJigWare 로 검색 후 [Example] 로 되어있는 동영상 참고Ex)
  https://www.youtube.com/watch?v=tqgSO_PA_cU
  [예제 - 사용방법에 대한 간단한 예제들]
  https://github.com/ojw5014/Examples
  [Document - 매뉴얼 & PPT]
  https://github.com/ojw5014/Document
  [Online]
  오로카(http://cafe.naver.com/openrt)에서 OpenJigWare 로 강의명 개설

Ⅲ 개발 내용

특징

상세내용

III. 특징

Open Source

- Message History

  내가 원하는 메시지를 MessageBox 와 연결하여 History Message를 남길 수 있고 원하는 경우 이를 파일로 에러와 구분해서 저장이 가능

- Convert 가 직관적으로 쉽다.

  문자변환 등 형변환 관련(StrToInt, StrToDouble, StrToBytes...)

- Parameter 파일 생성 및 Loading 이 쉽다.

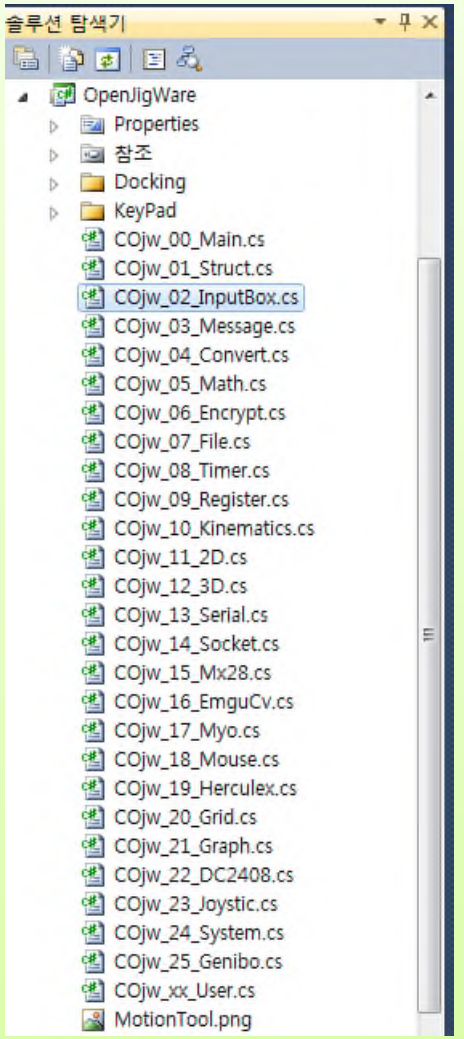
- 모터의 제어를 쉽게 해준다.

  Dst robot(전 동부로봇) 의 허큘렉스 시리즈, 로보티즈 모터 지원(이 경우 Rpm 제어를 Time 제어로 변환 가능)

- 3D 모델링의 속도 저하가 우려되면 2D 함수를 이용해 3D 모델링을 그릴 수 있다.

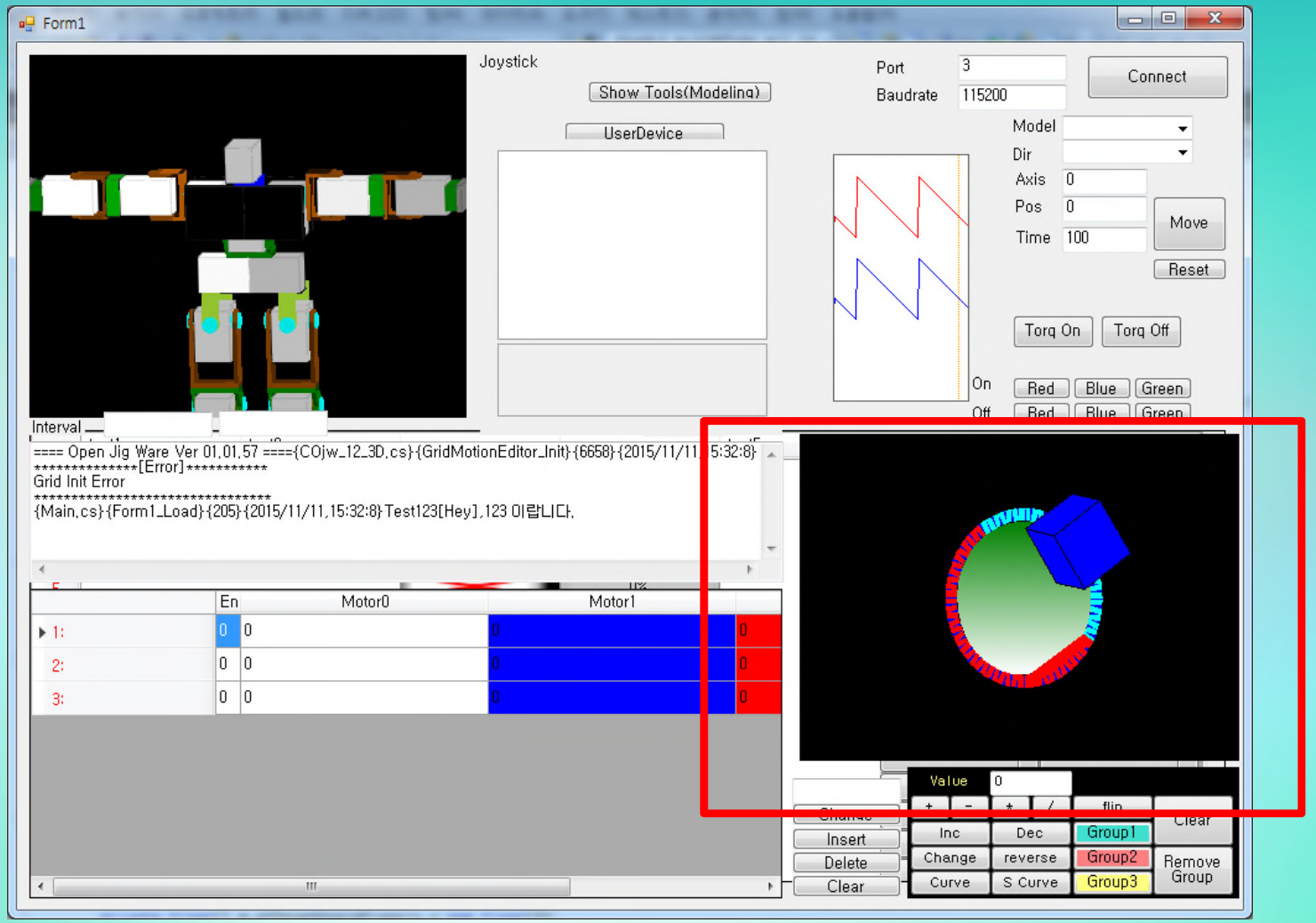
- Forward / Inverse Kinematics 지원

  - Forward : 파라미터만 넣으면 자동으로 3D 모델링이 그려지고 논문에 넣기 쉽게 행렬 수식함수가 자동으로 만들어져서 출력.
  - **Inverse** : Matlab 과 유사한 사용법으로 제작, 총 255개의 수식 함수를 자신이 원하는 부위에 골라서 넣을 수 있다.
  -> **Python** 지원 : 교육적 효과를 위해 수식부를 Python 으로 설정해 사용할 수 있다.



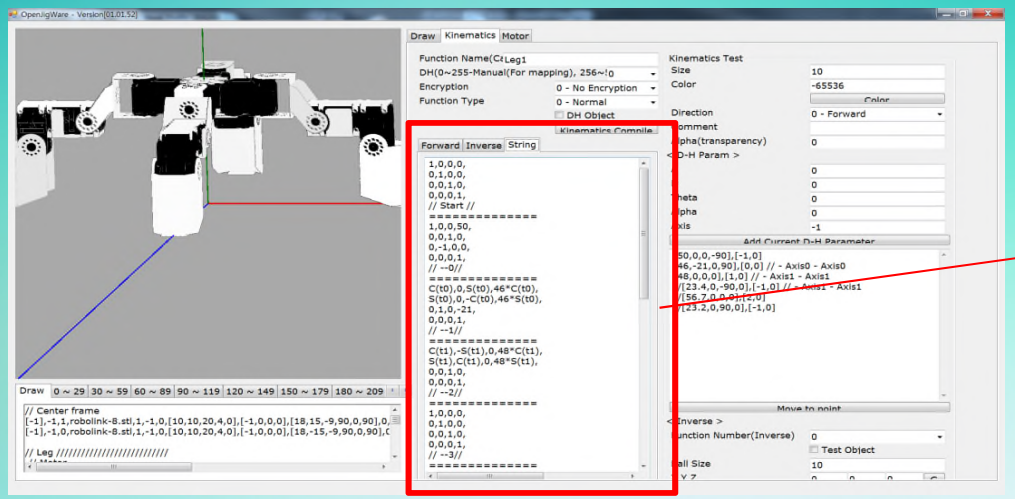
III. 특징

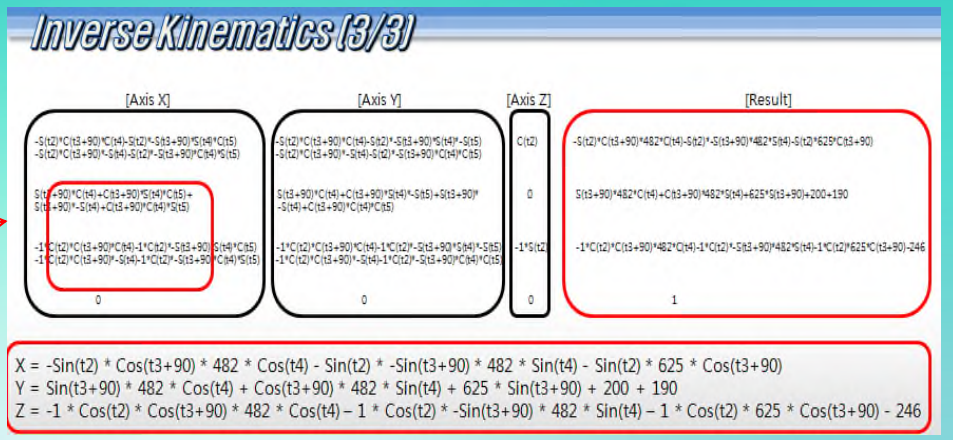
- 2D 그래픽을 이용해 3D 그래픽을 구현한 모습



III. 특징

- 초보자에게 쉬워야 한다.
  - 코딩라인 5줄 이내로 3D 모델링이 가능(참조구문, 변수선언 포함)
- 전문가에 의한 코딩의 다양화가 가능해야 한다.
  - 내부 OpenGL 을 활용한 세부적 접근 가능
- 교육적 컨텐츠가 가능해야 한다.
  - Forward / Inverse Kinematics 의 활용가능
  - 수식의 결과 행렬을 얻어서 눈으로 확인이 가능해야 한다.

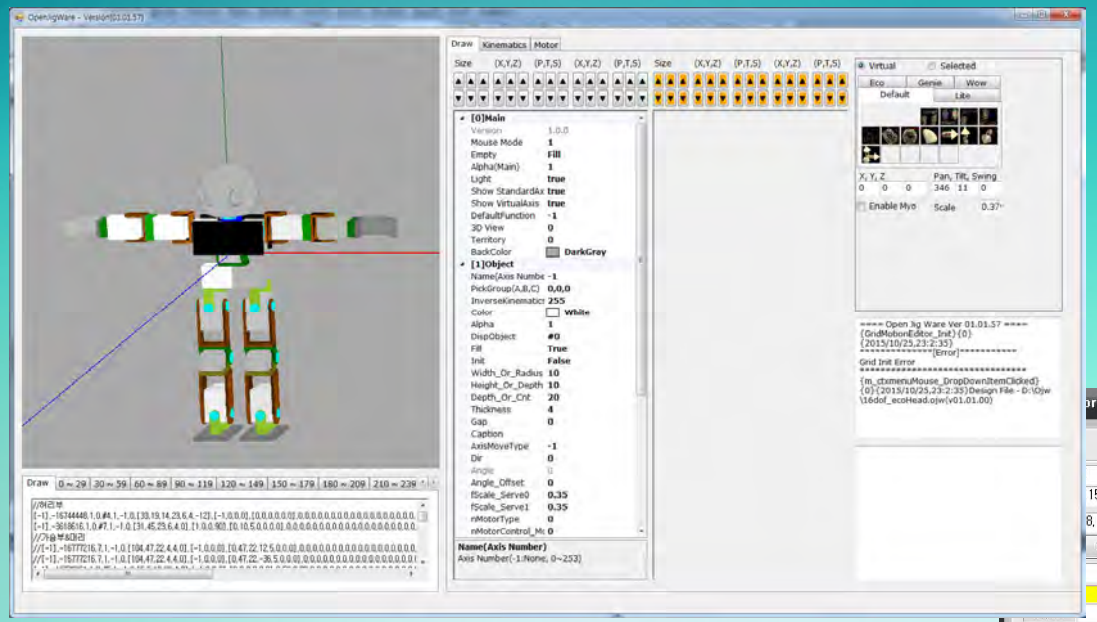




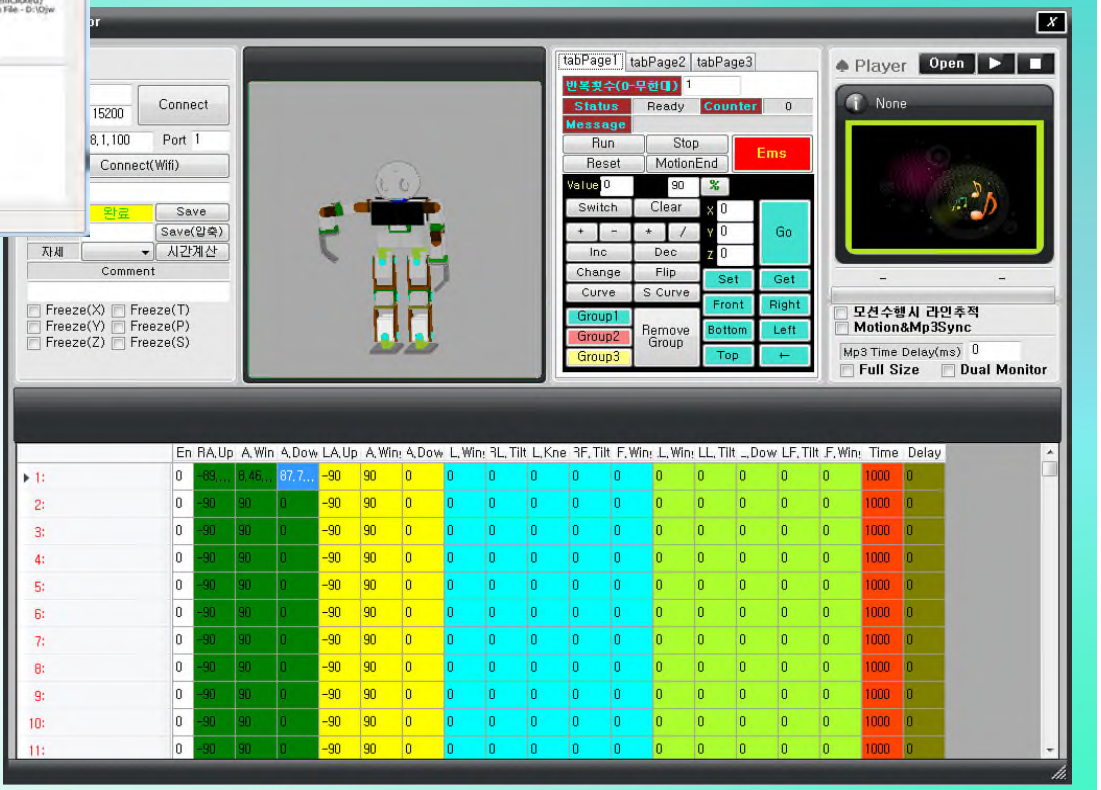
- 속도가 느려선 안된다.
  - 모델링 파일을 불러들이는 단계에서 컴파일 과정을 거쳐 실 속도에 맞게 운용 가능
- Cad 데이터의 모델을 가져올 수 있어야 한다.
  - ASE, OBJ, STL(text/binary) 파일 이용 가능
- 모터의 모든 정보가 들어가야 한다.
  - 기어비, 회전방향, Mirror 모터, 모터의 별칭, Limit, 모터의 타입(속도제어/포지션제어 등), 그룹정보, 실제 ID, Flip 동작시 동작 규정(역회전, 동작없음, Mirror 모터 싱크)
- 모델링, 구현, 모션제작 툴의 구분
  - 다른 구분으로는 교육자/교육대상자
- 모든 종류의 로봇에 대응이 가능해야 한다.
  - 바퀴형, 다관절 형, 델타 형, 직교로봇, 가변 트랙 등.

III. 특징

Modeling tool



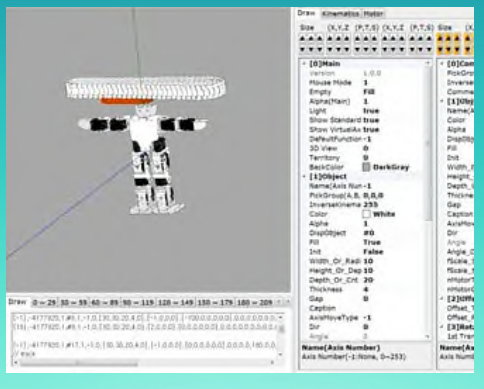
Motion tool



모델링 툴, 수식구현

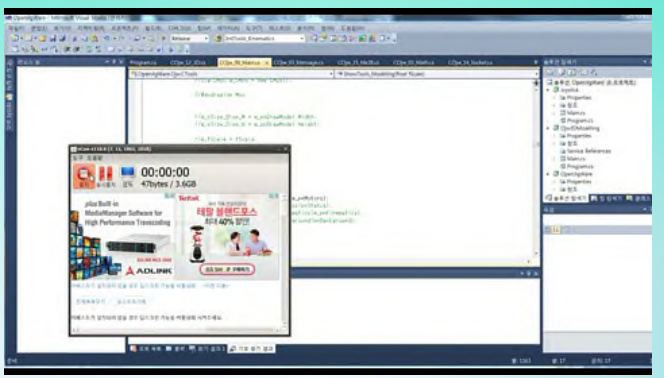
- 모델링 툴/ 수식구현
  - 모델링

https://youtu.be/srUZRiMfd8k



  - 수식구현

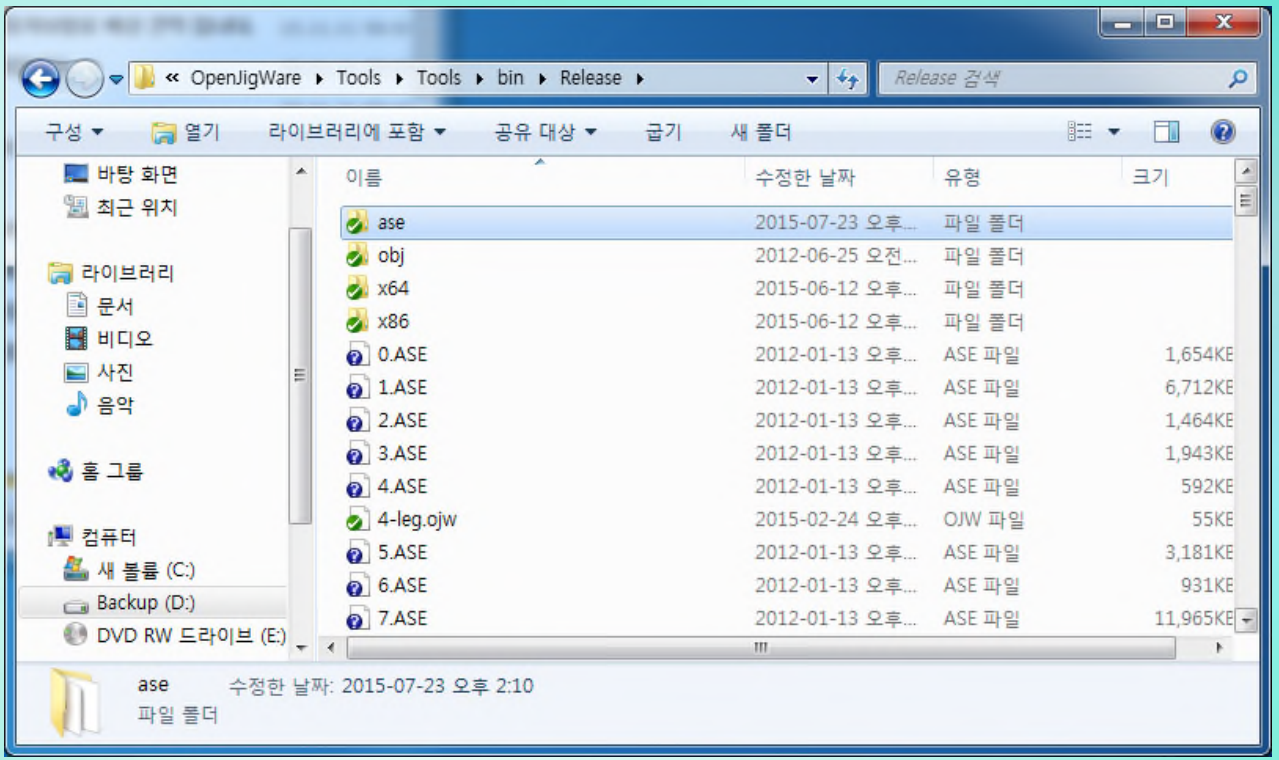
https://youtu.be/BEr_5CZqk-w



- 교육대상자의 프로그래밍 실습
  - 혹은 모델링 파일(*.ojw) 을 가지고 있는 사람의 프로그래밍 실습
  - https://www.youtube.com/watch?v=ol29At-4OWc

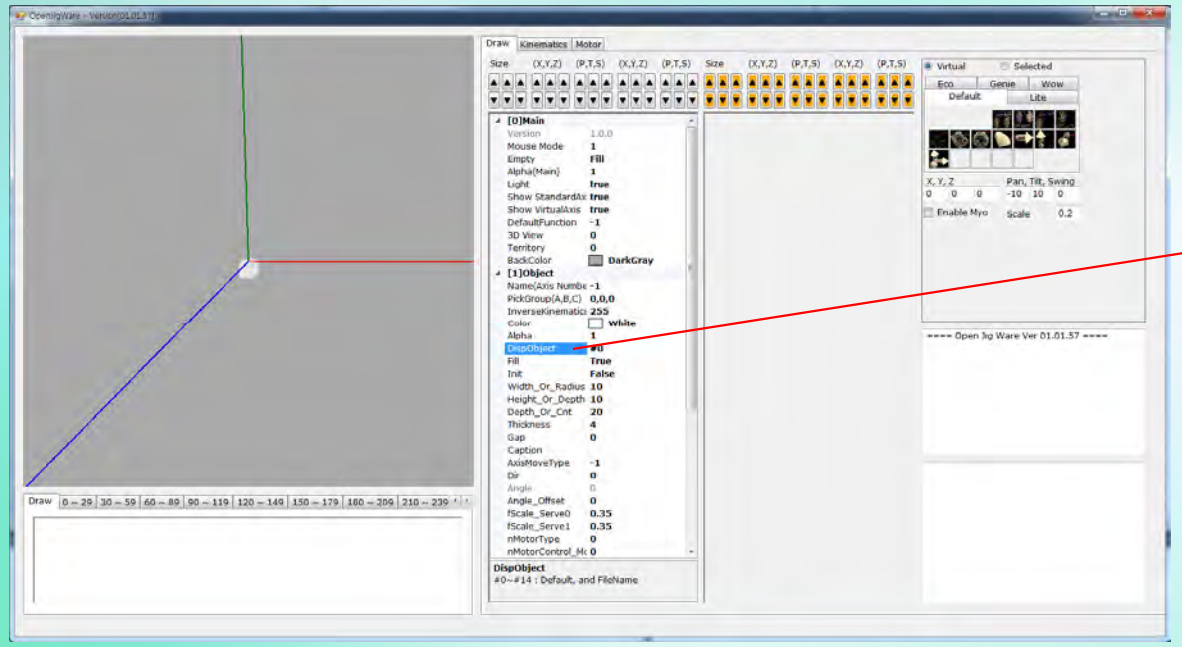
모델링 툴예제

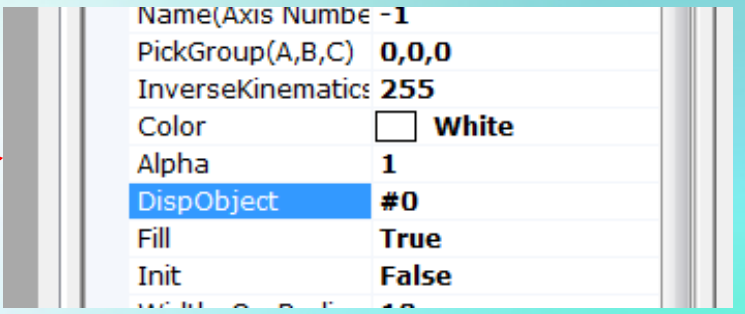
- Cad 를 넣어보자. – Step 1
  - 가지고 있는 Cad 파일(obj, ase, stl 파일을 실행폴더에 넣어준다. – 구 버전에서는 ase 폴더에 넣어준다.)



모델링 툴예제

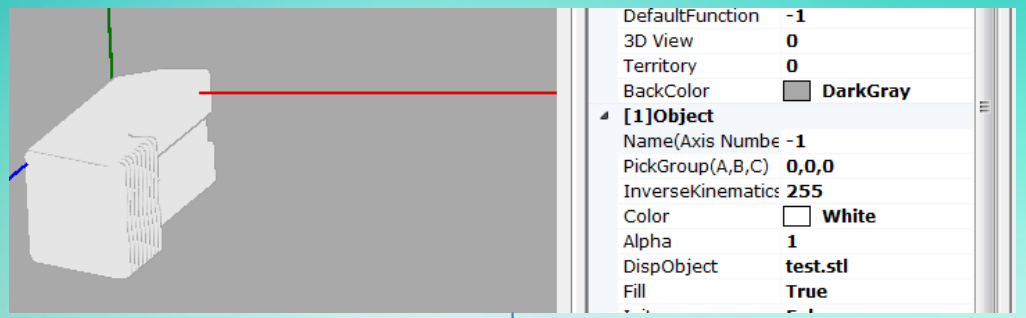
- Cad 를 넣어보자. – Step 2
  - 모델링 툴 안에서DispObject 항목의 이름을 가지고 있는 파일의 이름으로 바꿔준다.
    - # 으로 시작하는 이름은 내부에 정의된 17가지 패턴을 의미
    - Ase 파일은 확장자 없이 적어도 상관없다. 다른 파일들은 확장자를 반드시 기록.



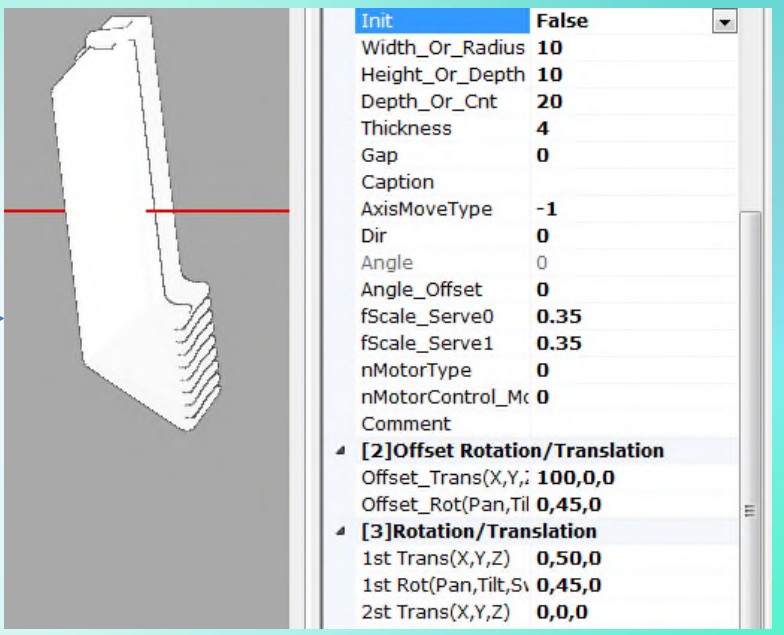


모델링 툴예제

- Cad 를 넣어보자. – Step 3
  - 들어간 모델을 확인 후 원하는 위치로 이동 및 회전 후 Add 한다.



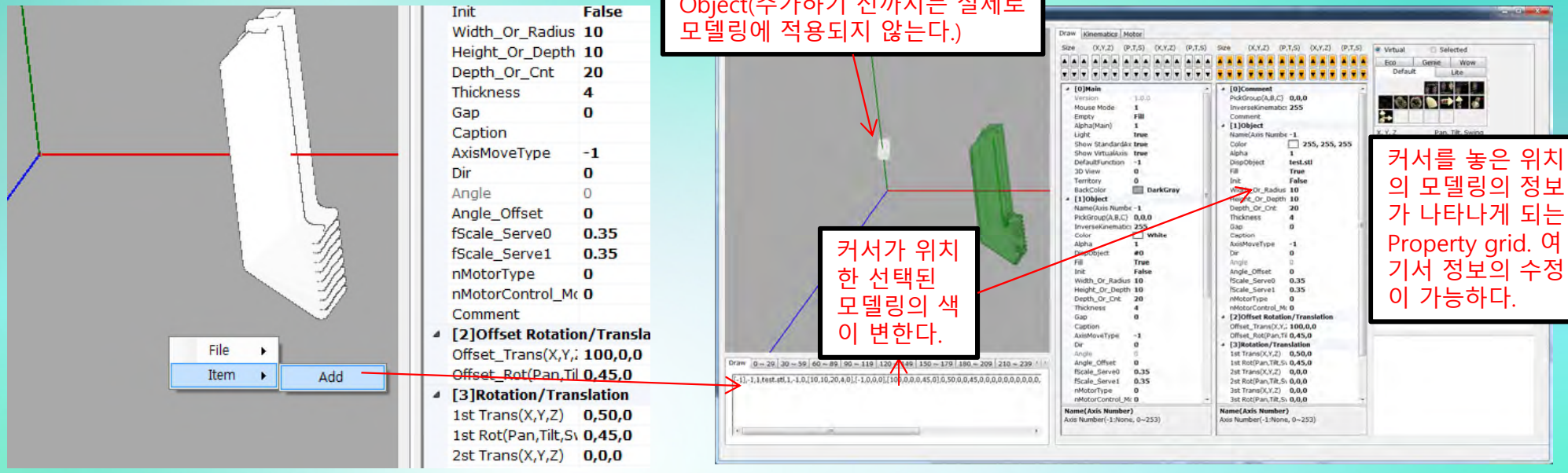
1. Offset 으로 100 이동 후 45도 회전
2. Y축으로 50 이동, Y축 45도 회전



1. Offset 은 실 회전과 상관없이 현재의 자세를 결정
2. Trans / Rotation 은 실제적이 이동 및 회전을 의미(일반적인 OpenGL 3D 모델링에서의 회전 및 이동의 이미와 동일)

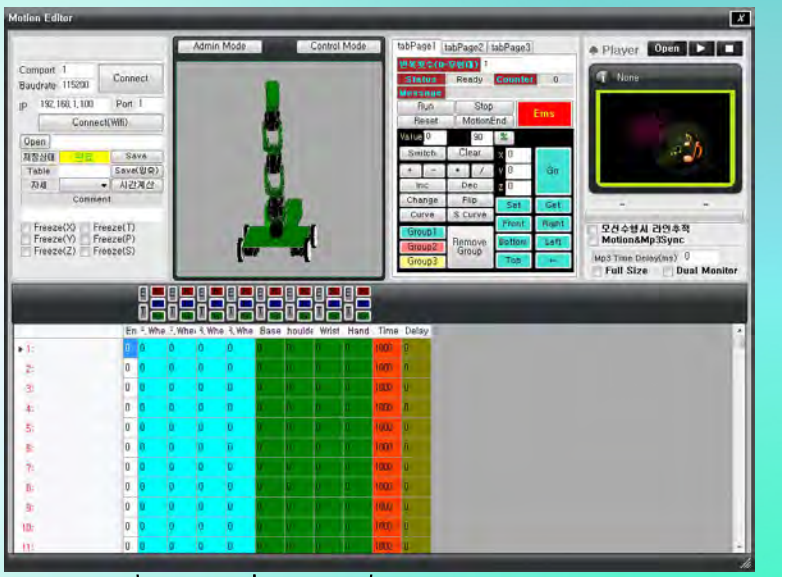
모델링 툴예제

- Cad 를 넣어보자. – Step 4
  – 장치를 Add 한다.(마우스 우클릭)



모션제작 툴

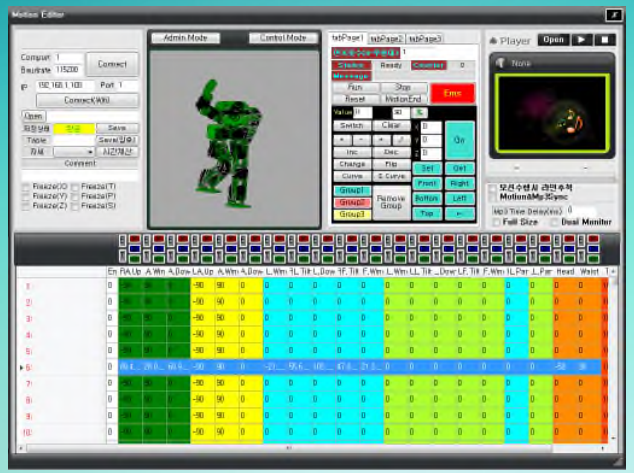
- 불러들이는 모델의 종류에 따라 모션툴의 기능이 다양하게 변화한다.
- 관절의 역할에 따라 색을 달리 정하는 것이 가능하다.
- 클릭된 위치에 따라 어떤식의 동작을 할 것인지 정의하는 것이 가능



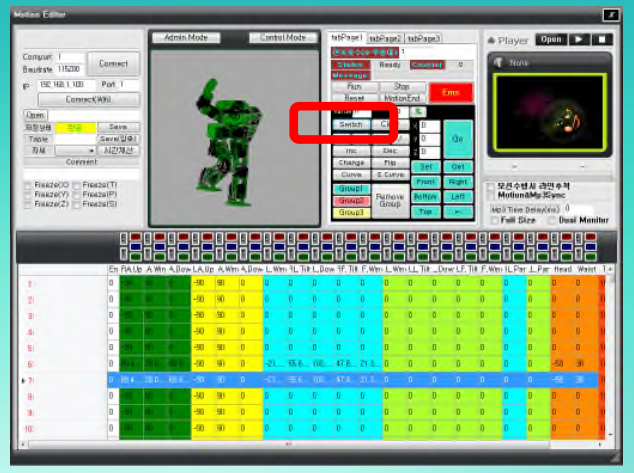
III. 상세내용

모션제작 툴(휴머노이드 모델링의 경우)

모델별 변화되는 기능 예 : Switch

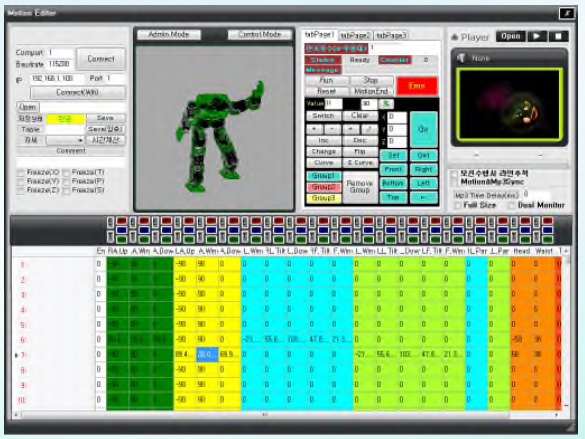


프레임 복사



테스트를 위해 모션 한 프레임을 만들어 본다.
보는 바와 같이 우측 팔과 우측 다리를 들고 머리를 우측으로 돌리며 허리를 숙인것을 볼 수 있다.

비교를 위해 아래에 Ctrl^C, Ctrl^V 하여 프레임 복사를 한다. 이후 Switch 버튼 클릭



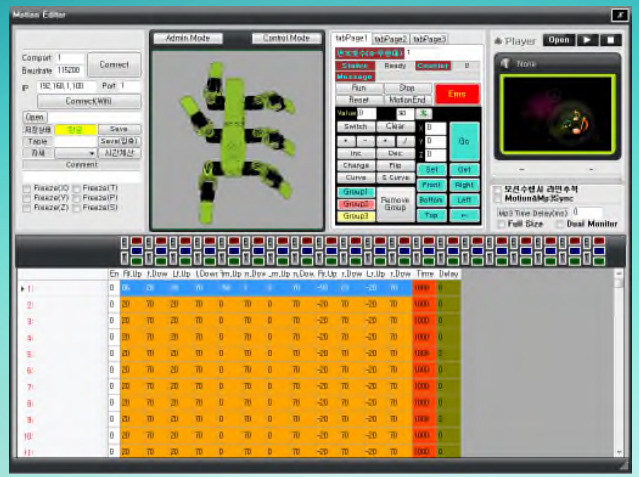
Switch 클릭

우측 팔/다리의 모션이 좌측 팔/다리로 바뀌었고, 머리는 반대방향으로, 허리는 변화 없는 것을 볼 수 있다.

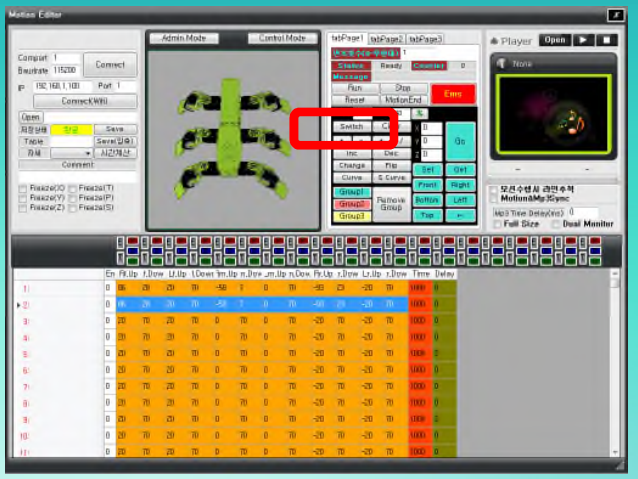
III. 상세내용

모션제작 툴(HexaPod 모델링의 경우)

모델별 변화되는 기능 예 : Switch

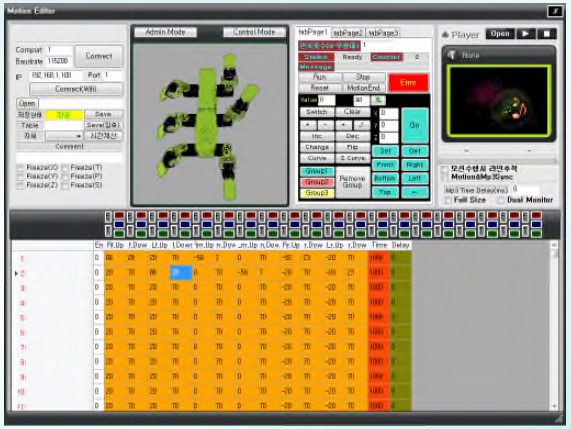


프레임 복사



테스트를 위해 모션 한 프레임을 만들어 본다.
보는 바와 같이 우측 관절은 정해진 초기위치가
아닌 사용자가 입력한 모션을 취하고 있다.

비교를 위해 아래에 Ctrl^C, Ctrl^V 하여 프레임
복사를 한다. 이후 Switch 버튼 클릭



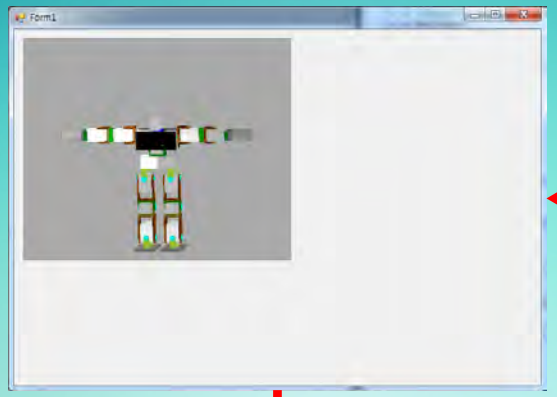
Switch 클릭

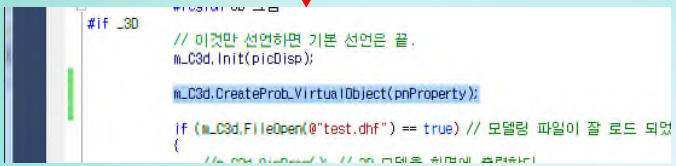
우측 관절 전체의 모션이 좌측관절로 바뀐것을 알 수 있다.

참고 : 이 모델링 파일은 그룹 구분이 되어 있지 않아 색 구분이 없이 주황색으로 통일 된 상태

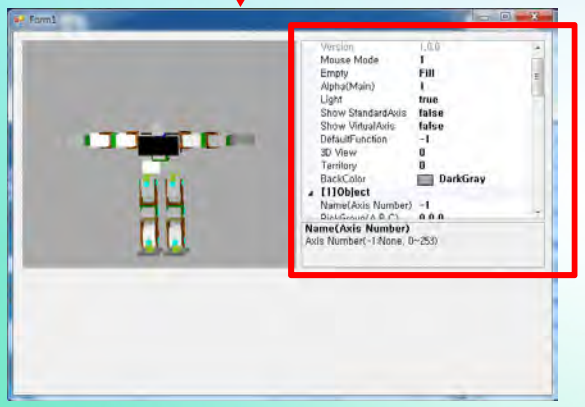
III. 사용방법

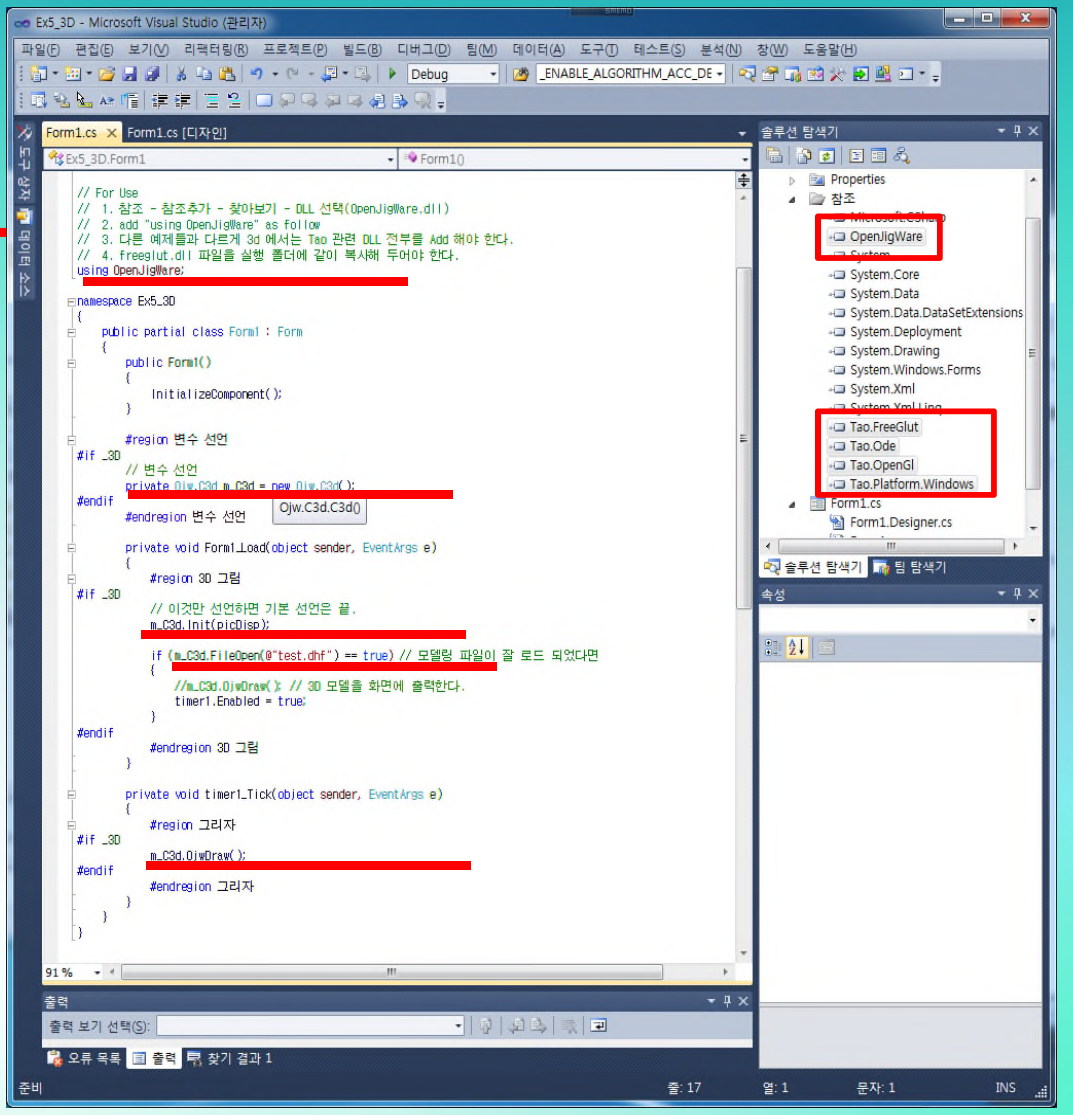
- 실행프로그램





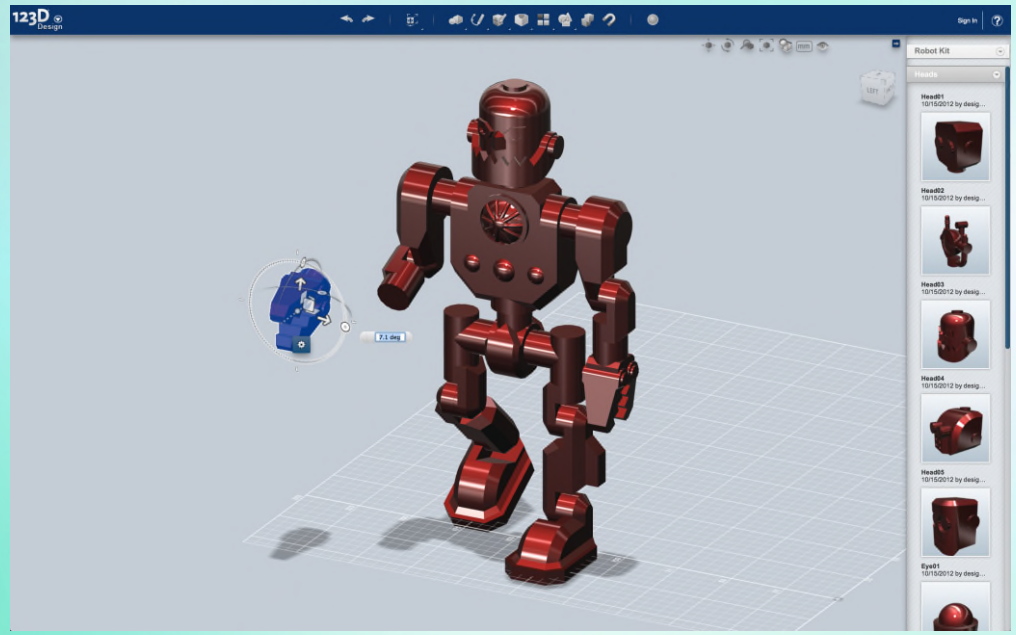
한줄 더 추가하면...





모델링 데이터는 어디서 구할까? [1/2]

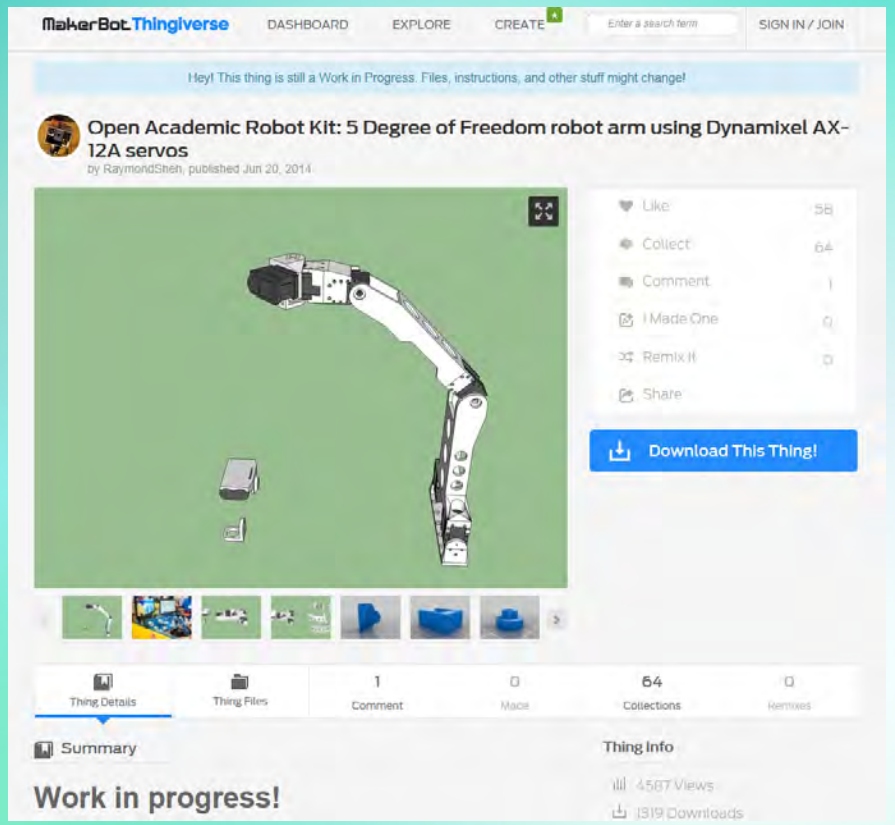
- 직접 그린다.
  - 솔리드 웍스, Maya, 3D Max 및 기타 3D 모델링 툴 이용(파일 저장은 **obj, ase, stl[Binary or Text]** 로 한다.)
  - Autodesk 123Design 같은 툴을 사용한다.(무료)
  - 주의사항
    - 정점이 많으면 Drawing 하는데 시간이 걸리므로 프로그램이 느려지는 효과가 나타난다.



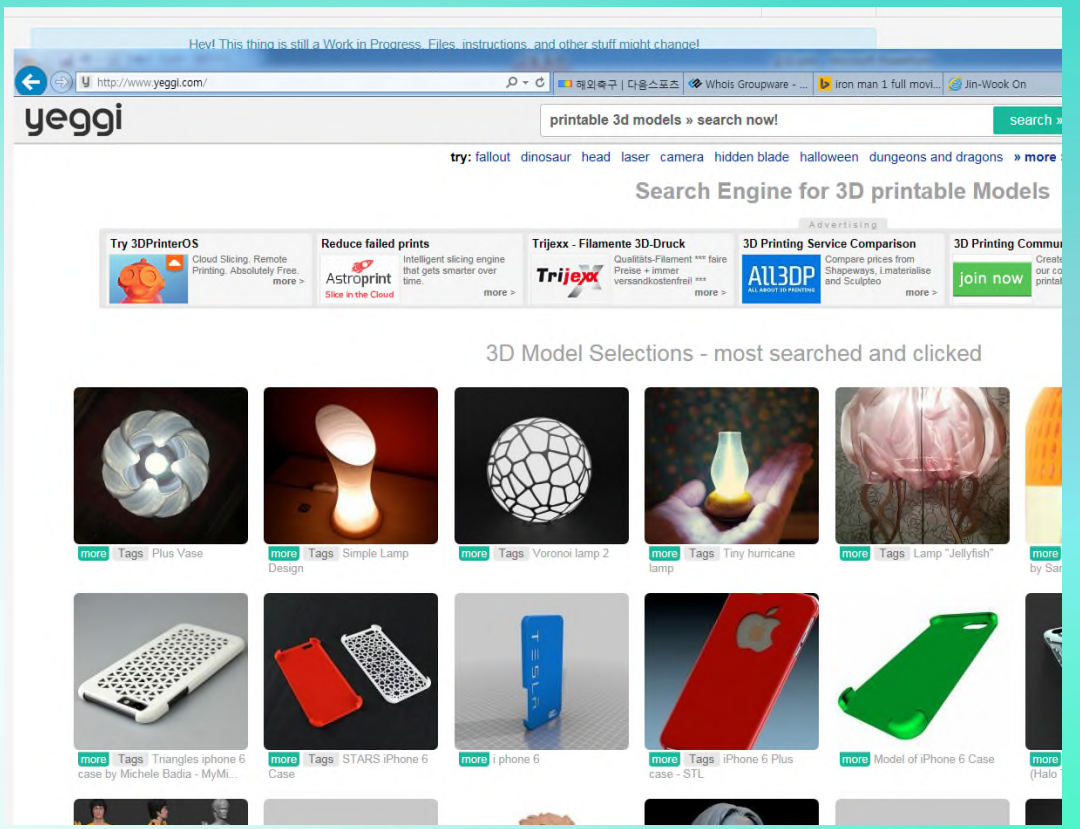
III. 사용방법

모델링 데이터는 어디서 구할까? [2/2]

- 난 그릴줄 몰라!!!
  - 다운로드한다.
    - 3D 프린터와 동일, Thingiverse, Yeggi 등의 사이트 이용



http://www.thingiverse.com



http://www.yeggi.com/

Kinematics

- 왜 Kinematics 를 풀까?

Kinematics

- Forward kinematics
  - Joint Angle ➔ Cartesian space(x, y, z)
  - Simple and Unique solution
- Inverse kinematics
  - Cartesian space(x, y, z) ➔ Joint Angle
  - It is not a unique solution, difficult(but delta system)
  - Sometimes we have singular position or some position we cannot go

Kinematics

Forward Kinematics(1/2)

- Translation

$$\begin{aligned} X &= x + dx \\ Y &= y + dy \\ Z &= z + dz \end{aligned}$$

- to Matrix

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{vmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & dx \end{vmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Kinematics

Forward Kinematics (D-H Notation)

- T(Transformation Matrix)
  - It has rotation and translation in their 4 * 4 matrix

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} & & & dx \\ & R & & dy \\ & & & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  - Parameter

a_i : 길이(*length*)　　d_i : 오프셋(*offset*)

α_i : 비틀림(*twist*)　　θ_i : 각도(*angle*)

  - Define
    - X(i+1) and Z(i) is orthogonal
    - X(i+1) and Z(i) has a matching point

(*DH*1) x_{i+1}축은 z_i 축과 수직이다.

(*DH*2) x_{i+1}축은 z_i 축과 만난다.

Kinematics

D-H Notation Matrix

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

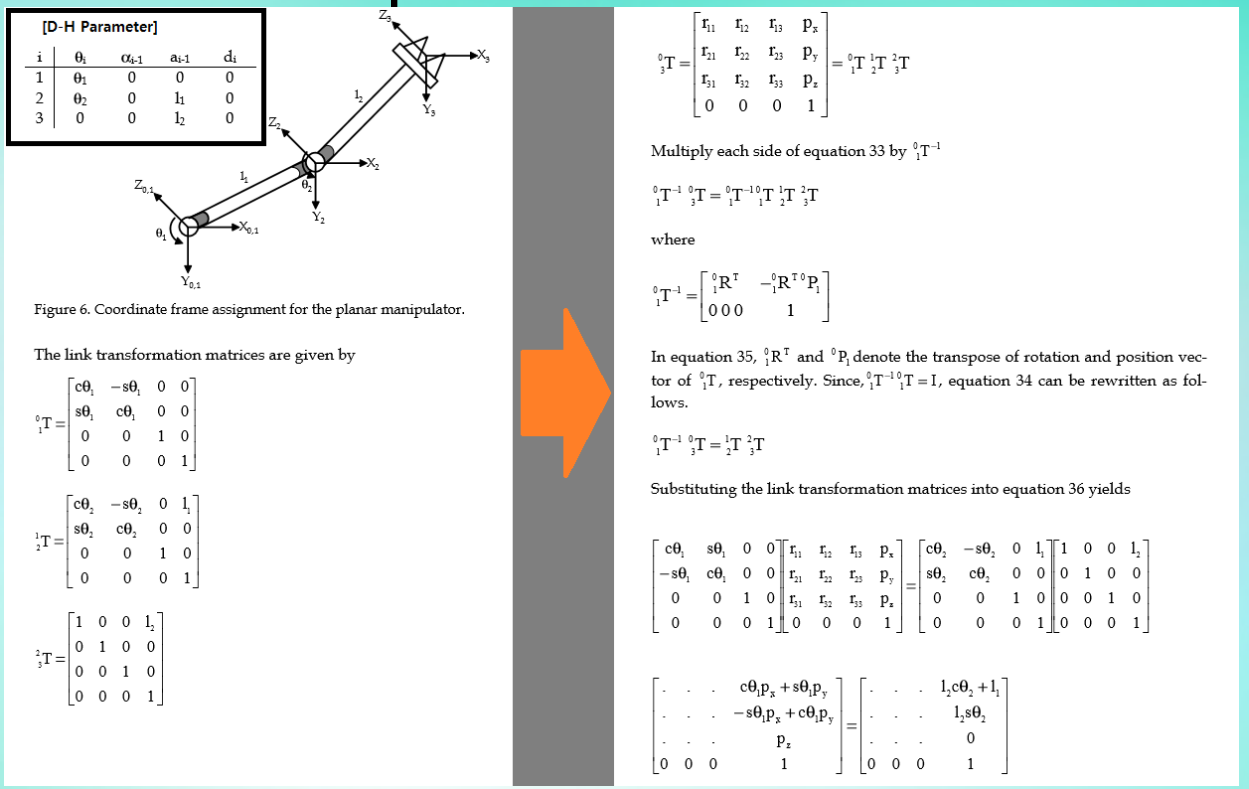
$$= \begin{bmatrix} C_{\theta_i} & -S_{\theta_i} & 0 & 0 \\ S_{\theta_i} & C_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_{\alpha_i} & -S_{\alpha_i} & 0 \\ 0 & S_{\alpha_i} & C_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_{\theta_i} & -S_{\theta_i} C_{\alpha_i} & S_{\theta_i} S_{\alpha_i} & a_i C_{\theta_i} \\ S_{\theta_i} & C_{\theta_i} C_{\alpha_i} & -C_{\theta_i} S_{\alpha_i} & a_i S_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Kinematics

실제 Forward Kinematics 를 풀면...

- http://cdn.intechweb.org/pdfs/379.pdf 문서의 Page 14 – Example 2 를 봐보자.



[D-H Parameter]

i	θ_i	α_{i-1}	a_{i-1}	d_i
1	θ_1	0	0	0
2	θ_2	0	l_1	0
3	0	0	l_2	0

Figure 6. Coordinate frame assignment for the planar manipulator.

The link transformation matrices are given by

$$ {}^0_1T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^1_2T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_1 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^2_3T = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0_3T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0_1T\,{}^1_2T\,{}^2_3T $$

Multiply each side of equation 33 by ${}^0_1T^{-1}$

$$ {}^0_1T^{-1}\,{}^0_3T = {}^0_1T^{-1}\,{}^0_1T\,{}^1_2T\,{}^2_3T $$

where

$$ {}^0_1T^{-1} = \begin{bmatrix} {}^0_1R^T & -{}^0_1R^T\,{}^0P_1 \\ 0\,0\,0 & 1 \end{bmatrix} $$

In equation 35, ${}^0_1R^T$ and ${}^0P_1$ denote the transpose of rotation and position vector of ${}^0_1T$, respectively. Since, ${}^0_1T^{-1}\,{}^0_1T = I$, equation 34 can be rewritten as follows.

$$ {}^0_1T^{-1}\,{}^0_3T = {}^1_2T\,{}^2_3T $$

Substituting the link transformation matrices into equation 36 yields

$$ \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & 0 \\ -s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_1 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ \begin{bmatrix} \cdot & \cdot & \cdot & c\theta_1 p_x + s\theta_1 p_y \\ \cdot & \cdot & \cdot & -s\theta_1 p_x + c\theta_1 p_y \\ \cdot & \cdot & \cdot & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cdot & \cdot & \cdot & l_2 c\theta_2 + l_1 \\ \cdot & \cdot & \cdot & l_2 s\theta_2 \\ \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

단순히 2 Dof 의 모델이지만 3개의 행렬을 정의하고 곱을 해야 한다.
– 헷갈리는 요소가 많다.

Kinematics

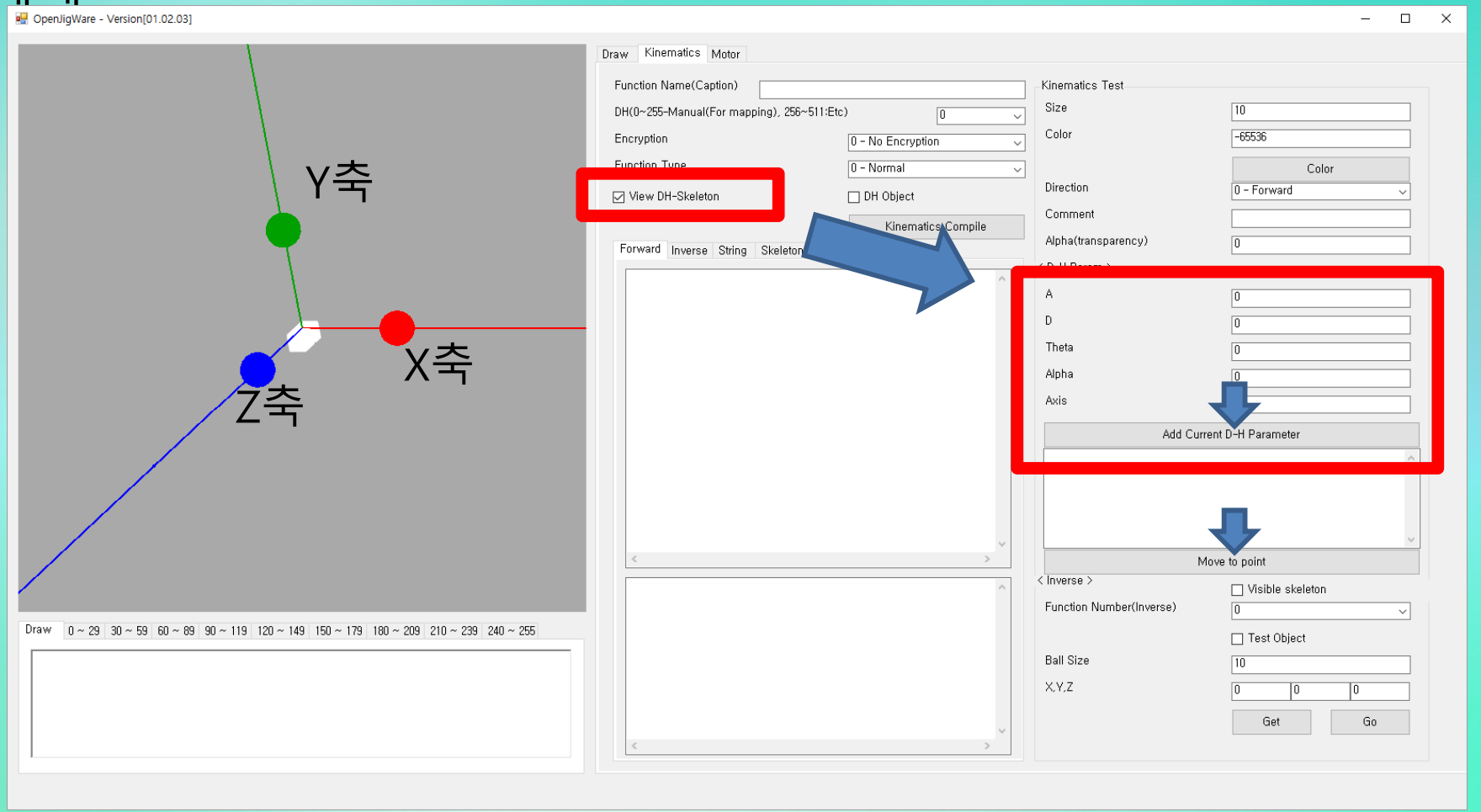
어렵다... 헷갈린다...

Open-Jig 적용한 Kinematics

III. 사용방법 Kinematics

OpenJigWare로 DH 를 그려보자

- Forward-Kinematics 를 눈으로 보면서 직접 값을 넣어보자
  - View DH-Skeleon 을 체크하면 DH 좌표축이 나타나게 된다.(헷갈리지 않는다.)
  - 그린 DH-Skeleton 에서 실제 사용된 수식과 3D 모델링을 자동으로 얻을 수 있다.
- 예제

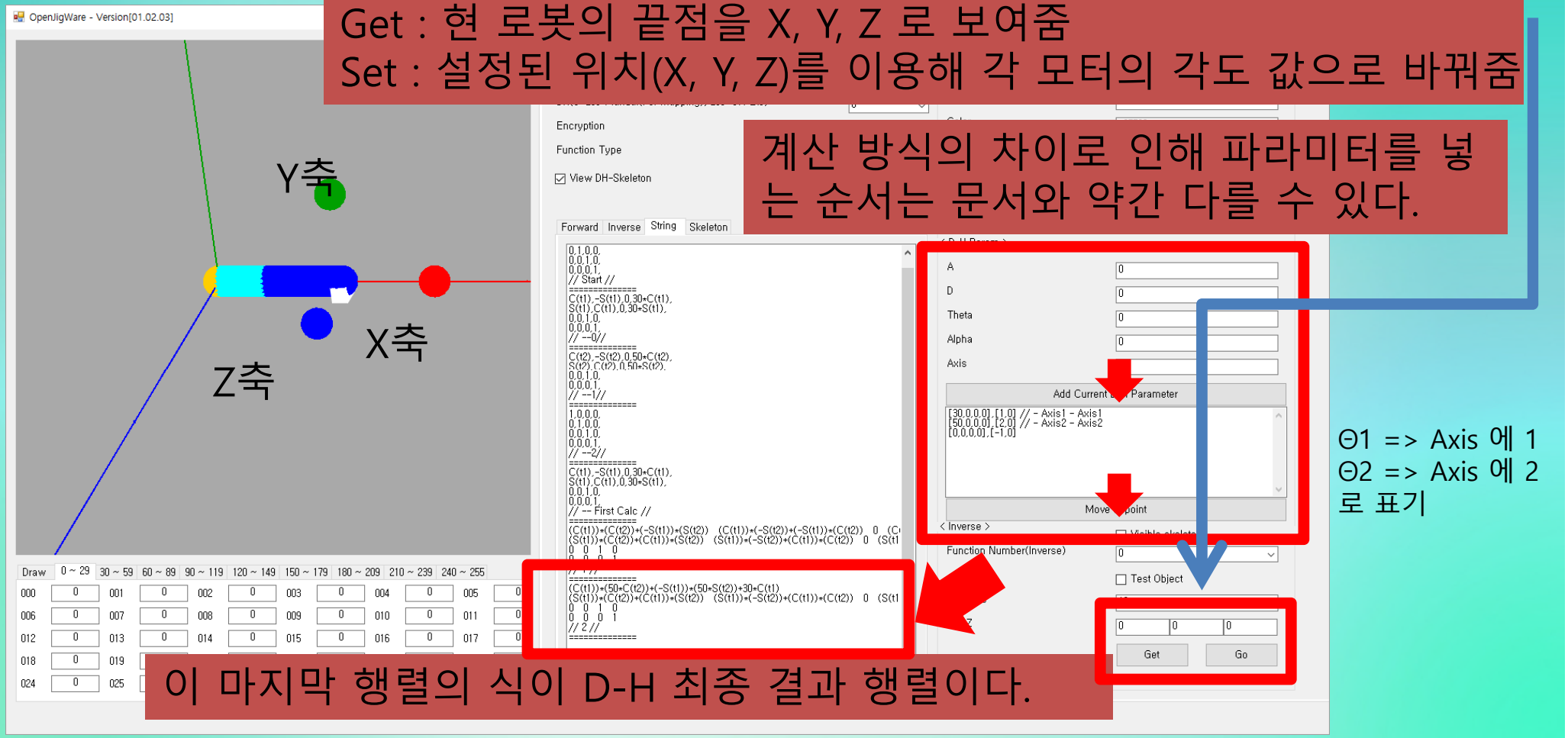


III. 사용방법 Kinematics

OpenJigWare로 DH 를 그려보자

- 파라미터 입력시 자동으로 그림이 그려진다.
  - 임의로 L1을 30, L2 를 50 으로 정의해 보고 그려보자.

[버튼 클릭]
Get : 현 로봇의 끝점을 X, Y, Z 로 보여줌
Set : 설정된 위치(X, Y, Z)를 이용해 각 모터의 각도 값으로 바꿔줌



계산 방식의 차이로 인해 파라미터를 넣는 순서는 문서와 약간 다를 수 있다.

Θ1 => Axis 에 1
Θ2 => Axis 에 2
로 표기

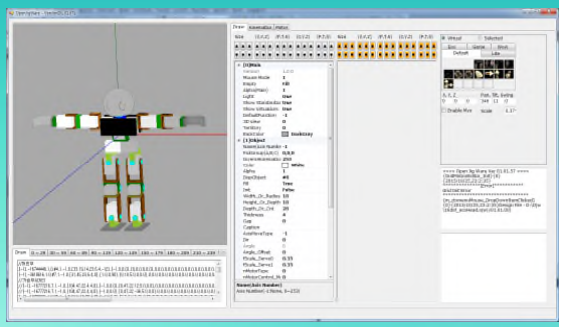
이 마지막 행렬의 식이 D-H 최종 결과 행렬이다.

III. 사용방법 Kinematics

대표적인 프로그램 사용 방법

교사

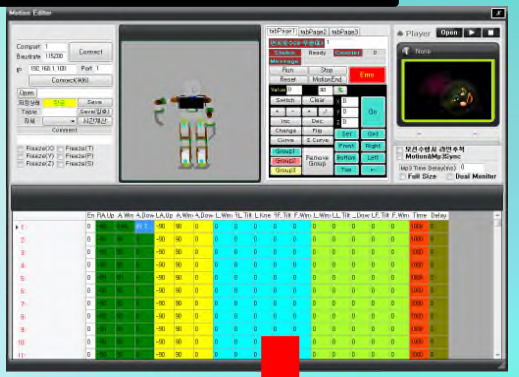
- 내부 DLL 의 함수를 이용해 3D 모델링 툴을 불러와 모델링을 그린다.(직접 그리거나 3D 프린터 공개자료 사이트-thingiverse, yeggi 등-활용)
- 내부에 제어에 필요한 수식을 넣어두고 이 모델링 파일을 학생에게 전달
- 참고
  - 모델링 파일을 불러오는 순간 내부의 수식 및 모터정보, 각 부분명칭 및 모터의 세부 기어비등의 detail 한 로봇 정보가 전체적으로 같이 들어오게 된다.

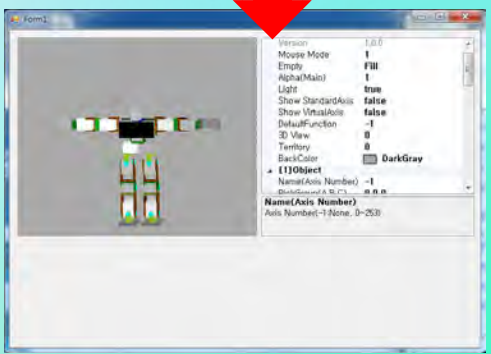


학생

- 내부 DLL 함수를 이용해 모션툴을 불러와 모션을 작성한다.
  : 로봇 종류에 따라 모션툴은 자동으로 변한다.
- 주어진 DLL 과 모델링 파일을 이용해 3D 시뮬레이터를 프로그래밍

- 작성한 모션을 DLL 함수를 이용해 불러 사용하거나 혹은 모터를 직접 제어한다.





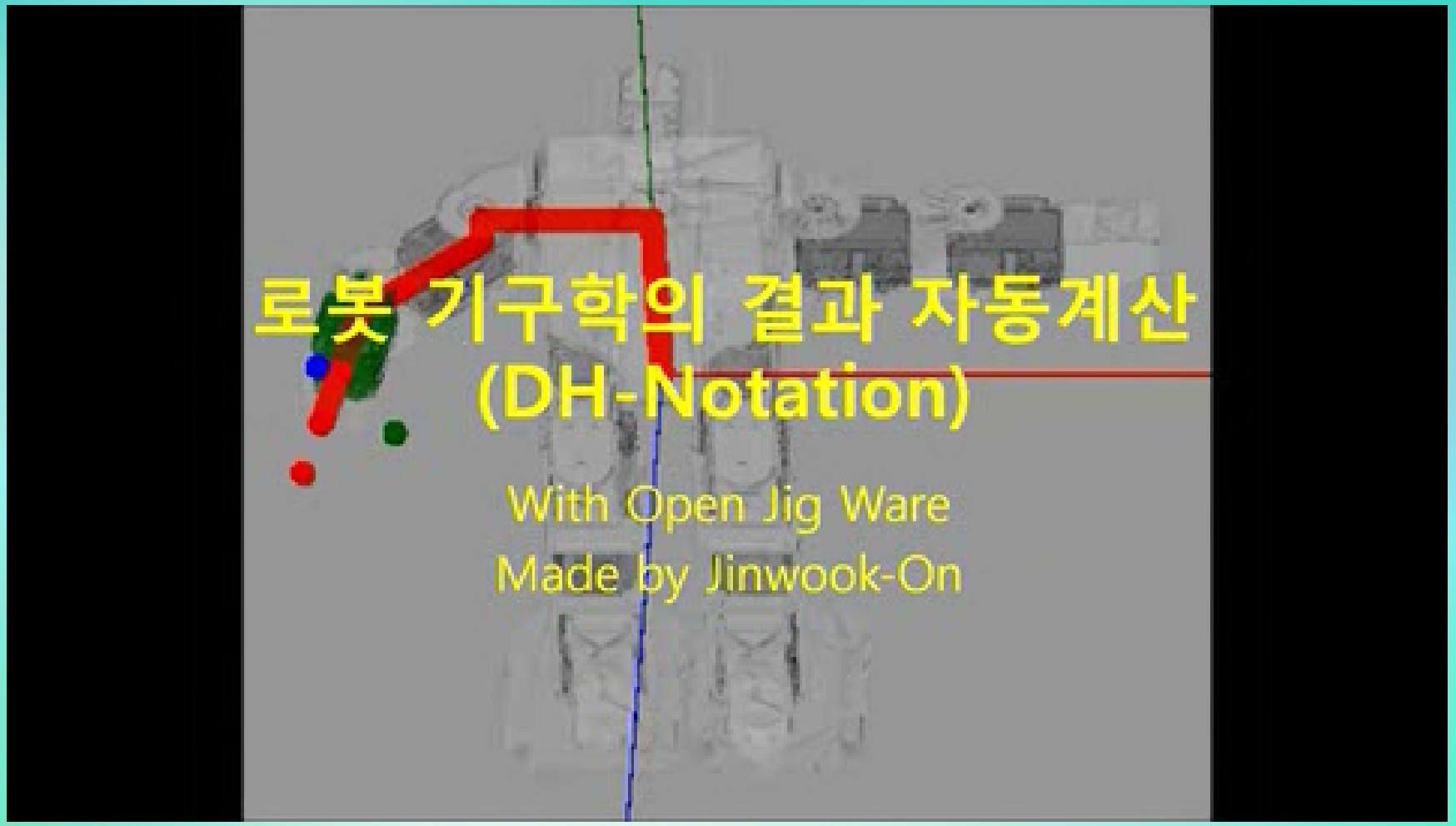
Kinematics 성과...

- kinematics
  - D-H Notation 교육 2시간 만에 상당수 학생들이 D-H 파라미터를 이용한 3D 매니퓰레이터 제작 성공
  - 자신이 스스로 3D 매니퓰레이터를 제작한 것에 흥미를 가짐
  - D-H Notation 에 대한 이해를 쉽게 가짐.
    - 헷갈리면 그려보면 된다...0

III. 사용방법

쉽게 가 보자

- D-H Notation 눈으로 확인해 보기
  - https://youtu.be/7lqSJNfkxe8



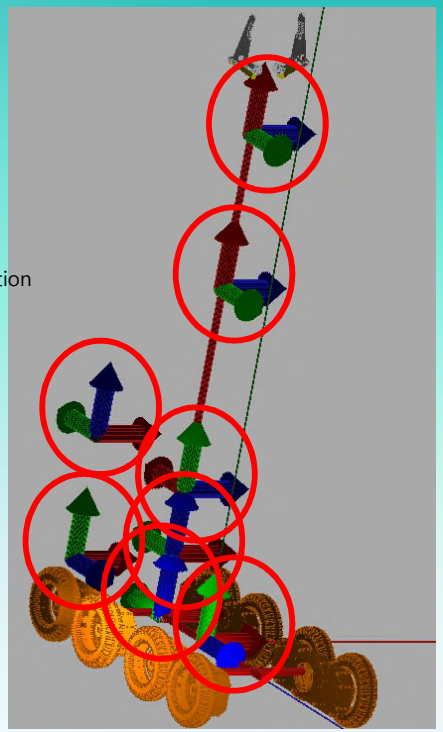
III. 사용방법

Forward Kinematics & Program(1/2)

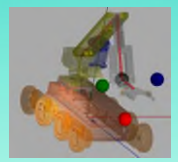
D-H Notation

	a	d	θ	α
0	[0,	-246,	0,	0]
1	[0,	0,	0,	-90]
2	[0,	190,	θ2,	0]
3	[0,	200,	0,	0]
4	[0,	0,	90,	90]
5	[625,	0,	θ3+90,	0]
6	[482,	0,	θ4,	0]
7	[0,	0,	θ5,	0]

// Orientation

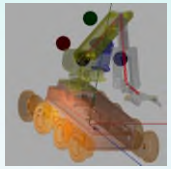


Add Rotation(θ5)

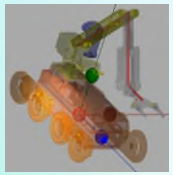


$$\begin{matrix} C(t4) & -S(t4) & 0 & 482*C(t4) \\ S(t4) & C(t4) & 0 & 482*S(t4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

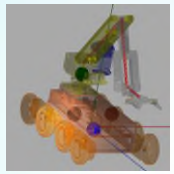
$$\begin{matrix} C(t3+90) & -S(t3+90) & 0 & 625*C(t3+90) \\ S(t3+90) & C(t3+90) & 0 & 625*S(t3+90) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$



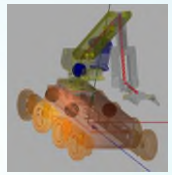
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$



$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -246 \\ 0 & 0 & 0 & 1 \end{matrix}$$

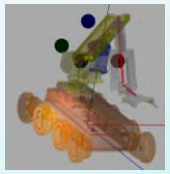


$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$



$$\begin{matrix} C(t2) & -S(t2) & 0 & 0 \\ S(t2) & C(t2) & 0 & 0 \\ 0 & 0 & 1 & 190 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 200 \\ 0 & 0 & 0 & 1 \end{matrix}$$



https://youtu.be/detjVR1dG08

향후계획

IV. 장점 및 현 문제점

- 장점
  - 모델링 파일이 있는 경우 하나의 모델링 파일 안에 모든 로봇의 세부적인 정보가 수식을 포함하여 전부 들어있다.
    - 추후 프로그래밍 수정이 필요 없이 모델링 파일만 수정하는 것으로 프로그램의 유지 보수도 가능하다.
  - 로봇의 종류에 상관없이 모든 로봇에 제어가 가능(바퀴, 회전/직동 관절, 체인, 병렬제어 등)

- 문제점
  - 현재 개발되어 있는 모터 제어 함수가 한정되어 있다.
    - 모터 구매비용이 너무 비싸요~ ㅠ.ㅠ
  - Visual Studio C# 에서만 사용 가능.
    - 현재 라즈베리파이에서 부분 포팅이 되어 있습니다.

IV. 나아가고 싶은 방향

- **3D 모델링을 그리게 되면 이게 STL 파일로 저장되거나 아님 바로 Gcode 로 slice 를 해서 3D 프린팅과 연동이 되었으면...**
- **Myo, Leap Motion, Arduino 등의 디바이스 연계가 되었으면...**
- **리눅스(라즈베리파이), 안드로이드, Unity 에 포팅이 되었으면...**
  - 현재 리눅스는 Herculex2 Class 가 포팅되어 있다.
- **수식 컴파일러가 프로그래밍 스타일도 가능했으면...**
  - 현재 Python 으로 선택시 프로그래밍 스타일로 수식을 넣을 수 있다.

오랫동안 꿈을 그리는 사람은 마침내 그 꿈을 닮아간다.
– 앙드레 말로

A person longing for any dream for a long time resembles that dream at last.
-Andre Georges Malraux

감사합니다.

감사합니다