

AI 축구

이 설명서는 AI 축구 플랫폼(Platform)을 통해 AI와 S/W 프로그래밍을 배우고자 하는 분들을 위해 만들어졌다.

목차:

설명서 사용방법

1. 개요 : AI 와 AI 축구	4
1-1 AI 개요.....	6
1-2 AI 의 심층 학습.....	7
1-3 AI 축구.....	11
2. AI 축구 플랫폼 설치.....	17
2-1 시스템 요구사항	17
2-2 AI 축구 플랫폼 설치 방법.....	18
2-3 Windows OS 에 설치하기.....	19
2-3-1 Webots 시뮬레이터 설치.....	19
2-3-2 Python 설치.....	22
2-3-3 AI 축구 시뮬레이터 설치.....	27
2-3-4 심층학습 라이브러리 설치.....	28
2-4 Linux(Ubuntu) OS 에 설치	39
2-4-1 Webots 시뮬레이터 설치.....	39
2-4-2 Python 설치.....	43
2-4-3 AI 축구 시뮬레이터 설치.....	44
2-4-4 심층학습 라이브러리 설치.....	45
2-5 설치 확인.....	53
2-6 CONFIG.JSON 파일 정보.....	55
3. AI 축구 기본 정보.....	59
3-1 축구 로봇 기본 사양.....	61
3-2 AI 축구의 기본 정보.....	69

4. AI 축구 예제.....	85
4-1 규칙 기반 예제 코드.....	87
4-2 AI와 인간사이의 AI 축구.....	108

설명서 사용 방법

AI 축구 게이머 여러분에게.

이 설명서는 크게 4 개의 절로 이루어져 있다.

1 절에서는 인공지능 개념을 소개한다.

2 절에서는 AI 축구 플랫폼(Platform)을 설치하는 방법이 소개된다. 심층 학습(Deep Learning)에 대해 알고 있는 고급 사용자들을 위한 심층학습 라이브러리와 NVIDIA® driver 를 활용하여 컴퓨터의 연산속도를 향상시키는 방법이 소개된다. 또한 플랫폼 시험과 AI 축구의 기본이 소개된다.

3 절에서는 AI 축구용 코드 작성에 필요한 정보들이 설명된다. 4-1 절에 있는 정보들과 같이 이해하면 수준높은 AI 축구 코딩에 필요한 지식을 갖게 된다.

4 절에서는 AI 축구 예제 코드와 <컴퓨터 AI 와 인간간의 AI 축구>에 대해서 설명한다. 이러한 방식의 AI 축구 게임은 게이머의 코드 수준을 높이는데 효과적이며 유용하다. 컴퓨터 AI 팀을 게이머의 현재 코드로 대체할 경우, 키보드로 조작되는 로봇들을 특정한 전략에 따라 움직이게 함으로써 게이머의 현재 코드를 잘 보완하면 키보드로 조작되는 팀과 유사한 전략을 구사하는 팀을 이길 수 있는 가능성을 높여준다. 학습 기반의 예제 코드와 블록 코딩은 별도의 부록으로 포스팅한다.

만약 전체 설명서를 읽을 시간이 부족하다면, 1 절을 건너뛰고 2 절로 바로가서 플랫폼 설치 부분부터 읽는다. 플랫폼 설치가 이미 되었다면 바로 3 절부터 읽는다.

1 개요: AI와 AI 축구

인공지능(AI)은 학습과 문제 해결 등 인간의 능력과 관련된 기능을 모방하여 실행하는 인공적인 인간의 지능을 가리킨다..



AI에서 기계 학습(Machine Learning)은 중요한 분야이다. AI는 그림 1과 같이 기계 학습을 포함하고, 기계 학습은 심층 학습을 포함한다고 할 수 있다.

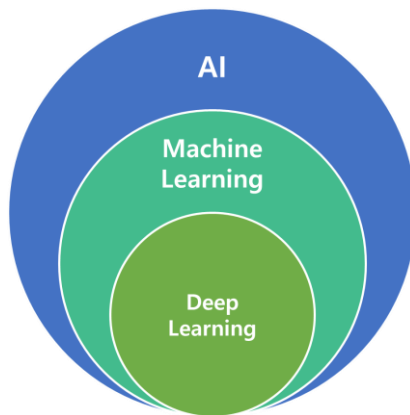


그림 1 인공지능, 기계학습, 심층 학습 상관관계

기계 학습은 주어진 데이터를 분석하여 데이터의 규칙과 특징을 찾을 수 있도록 스스로 학습(훈련)하는 기법이다. 사람이 학습하듯 컴퓨터에 데이터를 제공하여 학습하게 함으로써 새로운 지식을 얻어내게 하는 분야이다. 기계 학습 알고리즘의 개발은 학습(learning = training), 시험(testing = validation), 추론(inference) 세 단계로 구성된다. 학습 단계에서는 주어진 입력 데이터를 바탕으로 원하는 작업(task)을 위한 매핑(mapping) 함수를 얻는다. 시험 단계에서는 아직 살펴보지 않은 데이터에 앞서 학습된 매핑 함수를 적용할 수 있는지 시험해본다. 추론 단계에서는 학습된 매핑 함수를 원하는 작업의 임의의 데이터에 적용해본다.

현재 기계 학습의 여러 가지 알고리즘들이 개발되어 있다. 기계학습의 대표적인

기법으로 지도학습, 비지도 학습, 강화학습이 있다. 지도학습은 주어진 입력에 대해 이미 정해진 출력을 학습 단계에서 준다. 지도학습에 의해 훈련된 신경망은 매핑 함수의 역할과 같이 입력에 대한 출력값을 제공한다. 그러나 비지도 학습의 경우에는 입력만 주어진다. 비지도 학습은 다양한 입력 데이터를 그룹으로 분류하는 클러스터링(Clustering)에 주로 사용된다. 강화학습은 누적 보상(Cumulative Reward)을 극대화하기 위해 에이전트(Agent)가 환경(Environment)에서 어떤 행동(Action)을 취해야 하는지에 대해 다룬다. 예를 들어, 체스를 두는 에이전트는 체스판 위의 말을 움직이는 행동을 취하고 나면 보상을 받게 된다. 이 경우 체스판과 말은 에이전트와 상호 작용할 수 있는 환경(environment)으로 간주된다. 상태(State)는 체스판에 있는 말들의 현재 위치로 정의할 수 있다. 누적 보상은 상대편 말을 제거하거나 경기에서 승리하는 등의 기준으로 설계할 수 있다. 이 경우, 상대편 말이 제거되거나 승리를 거두면 양수(positive number)의 보상을 받는다. 누적되고 있는 보상(미래에 받게 되는 보상)을 최대화하기 위해, 에이전트의 목표는 상대편 말을 제거하면서 승리 가능성을 최대화하는 움직임을 취하게 될 것이다. 강화 학습 알고리즘의 성능은 기본적으로 특정 환경에 대한 상태, 행동 및 보상을 어떻게 설계하는가에 따라 달라진다.

심층학습은 다중 계층(Multiple-layer)을 가진 인공신경망¹을 이용해 데이터의 특징을 추출해 스스로 다른 상황에 대처할 수 있게 하는 기계 학습의 한 종류이다. 심층 학습을 활용한 응용 분야는 컴퓨터 비전(Computer Vision), 음성 처리(Speech Processing), 자연어 처리(Natural Language Processing) 등이 있다.

최근 심층 학습은 구글의 Youtube, 페이스북의 News feed 및 일반적인 이미지 분석 등에 사용되며 주목받고 있다. 심층학습이 획기적인 성과를 거두게 된 이유에는 빅데이터 수집이 가능해진 환경, GPU를 활용한 컴퓨터 계산 능력 향상, 새로운 알고리즘 개발 등이 꼽힌다. 효과적인 심층학습 알고리즘을 얻기 위해서는 대량의 데이터가 필요하다. 빅데이터가 있어도 컴퓨터의 연산 속도가 느릴 때는 이를 처리하기 어렵다. 최근에는 GPU를 사용하여 컴퓨터 연산의 병렬처리가 가능하게 되면서 단기간에 많은 데이터를 처리할 수 있게 되었다.

그렇다면 왜 심층 학습은 다른 기법들보다 우수한가?

¹ 인공신경망(Artificial Neural Network) : 인간이나 동물 뇌의 신경망에 초점을 맞추어 구현되는 시스템의 총칭이다.

인간의 두뇌를 모방한 인공신경망

심층학습이 적용되는 인공신경망은 보통 여러개의 은닉층(hidden layer)을 갖고 있다. 이러한 계층 구조는 데이터의 특징을 단계별로 학습할 수 있도록 한다. 이러한 특징(Feature)들은 인간의 뇌에 있는 신경세포들 사이의 연결 강도(weight)와 유사한 과정을 바탕으로 학습된다. 신경세포(neuron)의 구조는 그림 2 와 같다. 신경접합부의 연결 강도(weight)는 신경세포의 축삭돌기(Axon)와 신경세포의 수지상 조직(dentrite) 사이의 화학 반응작용에 의해 결정된다. 인공 신경망의 매핑함수 역할은 이러한 연결강도값의 분포로 설명할 수 있고, 학습과정을 통해 연결강도값은 조정되게 된다.

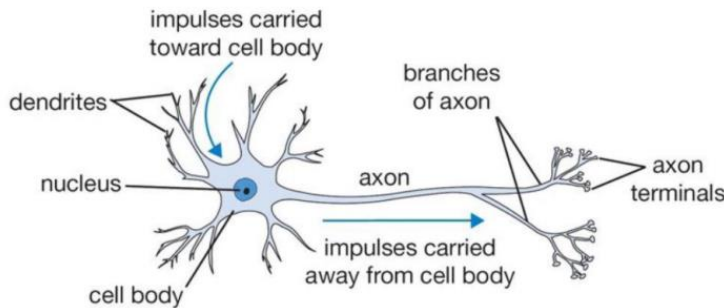


그림 2 신경세포의 구조

계층 구조의 장점은 영상(Image) 분류와 같은 컴퓨터 비전 작업에서 볼 수 있다. 영상 분류 작업에서 각 이미지는 특정 범주(Class)로 분류되어야 한다. 예를 들어 고양이의 영상은 범주 "Cat"과 연관되어야 하고, 개의 영상은 범주 "Dog"와 연관되어야 한다. 영상 분류에서 다중 계층이 있는 인공신경망을 사용하면 하위계층은 영상의 모서리나 선 등 저차원적인 특징들을 학습한다. 반면, 고위 계층은 얼굴의 모양, 물체의 모양 등 고차원적인 특징들을 학습한다. 모든 계층에서 학습된 내용을 종합하여 최종 출력 범주를 정하고 출력계층에서 최종 판단을 내린다. 심층 학습을 이용한 복잡한 비선형 함수들로 특징들을 조합해 최종 출력이 만들어진다. 심층 학습은 정보의 처리를 여러 개의 층으로 나누어서 복잡한 비선형

함수의 특징들을 조합해 최종 사고를 함으로써 심층 학습은 다른 인공지능의 기법들보다 월등한 성능을 보이는 것이다.

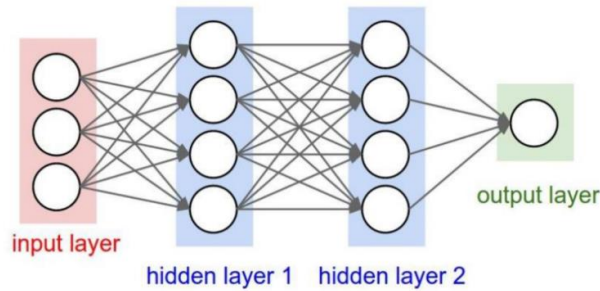


그림 3 인공지능망의 구조 예제

그림 3은 인공지능망의 구조를 예시하여 보여준다. 각 계층은 신경세포로 이루어져 있다. 각 계층의 원들은 개별 뉴런으로 해석될 수 있다. 예를 들어, 그림 3의 각 은닉층은 4개의 신경세포를 포함하고 있다. 인간의 시각과 관련된 업무를 실행하는 방법을 연구하는 분야인 컴퓨터 비전의 경우, 시스템에 대한 입력은 영상으로 되어 있다. 영상은 공간적 특징으로 설명된다. 영상 분류, 의미론적 영상 세분화(Semantic Image Segmentation), 물체 인식(Object Detection) 등의 컴퓨터 비전 문제를 해결하기 위해 영상 특화 컴퓨터 시스템과 CNN(Convolutional Neural Networks)을 사용한다. 그림 4는 고양이를 보여주는 영상에 다양한 컴퓨터 비전 방식이 적용된 결과를 보여준다.

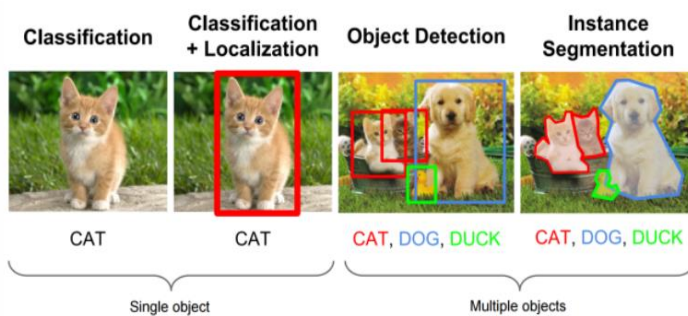


그림 4 다양한 컴퓨터 비전 방식의 최종 결과

인간의 언어와 같이 자연어로 표현된 언어를 컴퓨터가 이해할 수 있는 형태로 만드는 과정을 자연어 처리(NLP : Natural Language Processing)라고 한다. 문장 분류, 영상 제목작성, 기계 번역, 챗봇은 자연어 처리의 대표적인 예이다. NLP 응용 프로그램에서 주로 사용되는 인공신경망을 RNN(Recurrent Neural Networks)이라 한다. 순환신경망은 구글 번역과 같이 일상적으로 사용되는 응용프로그램에서 자연어 처리에 자주 사용된다.

RNN 을 사용하는 또 다른 응용 분야는 음성처리, 특히 음성 합성 및 음성 인식이 있다. 이 분야에서 많이 사용되고 있는 응용 프로그램으로는 애플 시리(Siri)와 구글 알렉사(Alexa)가 있다. 음성인식은 음성을 문자로 변환하고, 음성 합성은 문자를 음성으로 변환하는 것이다. 음성처리는 두 손을 자유롭게 움직일 수 있는 동시에 컴퓨터에 명령을 내릴 수 있기 때문에, 차세대 인간-컴퓨터 인터페이스로 주목받고 있다. 가정용 및 자율 주행 자동차용 AI 스피커와 음성 보조기는 음성 처리를 사용하는 응용 프로그램이다.

AI 연구 초기부터 게임은 AI 성능을 측정하는 데 널리 이용되었다. 1996 년 개발된 AI 체스 프로그램인 딥블루는 세계 챔피언 카스파로프를 제치고 인간 수준 이상의 플레이를 선보였다. 체스와 Backgammon 게임에서 좋은 결과를 얻은 후, 연구자들은 AI 의 바둑 게임 적용에 대해 생각하기 시작하였다. 바둑은 시작할 때 $361(19 \times 19)$ 개의 착점이 있고 계속 진행됨에 따라 한쪽이 대략 $[361 - 2(n-1)]!$, $n=1, \dots, 181$, !=팩토리얼, 의 경우의 수를 모든 착점이 채워질 때까지 진행할 수 있기 때문에, 단순한 AI 로는 프로 기사를 이기는 것이 불가능하다고 여겨졌다. 2016 년 딥마인드가 개발한 알파고는 최고의 바둑기사 이세돌을 4:1 로 꺾고 AI 의 위력을 입증했다. 알파고는 서로 다른 게임 전략을 가지고 자신과 경쟁함으로써 품질을 향상하는 것을 배우는 강화 학습 알고리즘을 사용한다.

알파고는 자기 자신과 계속해서 다른 전략을 시도하는 강화학습(RL: Reinforcement Learning) 알고리즘을 사용하였다. 강화학습 프레임워크의 장점은 유연성으로 보드게임과 비디오게임 등의 환경에도 효과적으로 적용된다. 최근에는 강화학습과 심층 학습을 결합한 방법인 DQN(Deep Q Network)가 Atari 게임에 적용되어 성공을 거두어 많은 관심이 집중되고 있다. 강화학습은 신경망을 이용해 심층학습(Deep Learning)과 결합하여 상태와 행동사이의 매핑을 나타내는 정책(Policy)을 설계하는 것을 목표로 한다. DQN 의 출력은 특정 상태에서 수행할 수 있는 각 행동의 Q-값(Q-value)이다. Q-값은 에이전트가 해당 행동을 취할 경우 받게 될 미래의 누적 보상을 나타낸다. 에이전트는 향후 최대의 누적 보상을 받을 행동을 취하기를 원하므로 Q-값의 최대치를 가지는 행동을 하려고 한다.

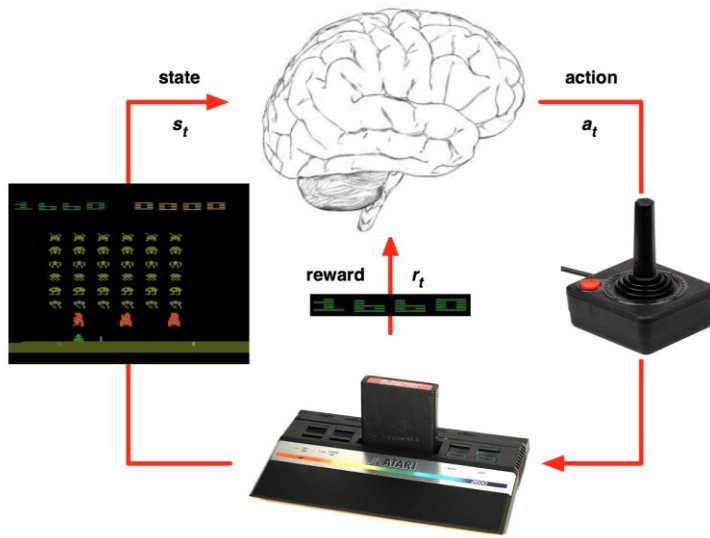


그림 5 상태, 행동, 보상을 포함한 Atari 게임 프레임워크

강화학습은 상태(State), 행동(Action) 보상(Reward)의 함수로 이루어져 있다. Atari 게임을 예로 들면, 상태(State)는 게임 화면이 되고, 보상(Reward)함수는 현재 점수로 대체 가능하며, 행동(Action)은 위의 그림 5 에서와 같이 비디오 게임 컨트롤러에서 가능한 동작이 될 것이다. 사람이 게임을 하는 방식을 심층 신경망을 이용해 구성하면 그림 6 과 같다. 그림 6 에서 “Convolution” 계층들은 영상의 특징들을 추출하고 “Fully connected” 계층은 이를 기반으로 행동을 결정한다.

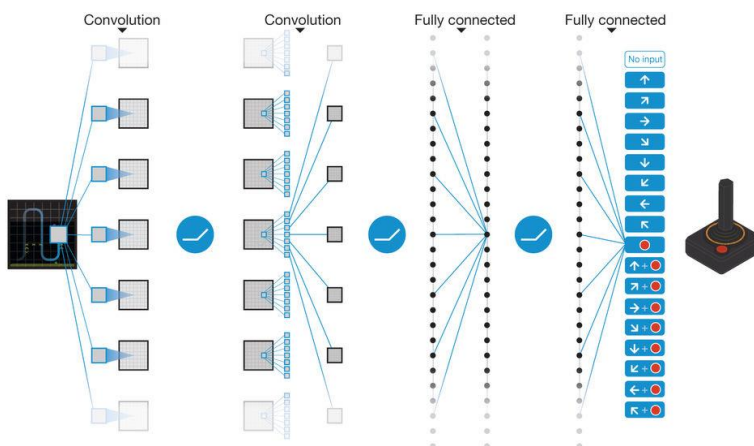


그림 6 화면에 표시된 Atari 게임의 상태에 따라 행동을 결정하는 심층 신경망

따라서, 사람이 하던 게임을 아래의 그림 7 과 같이 인공지능망이 대신하게 된다.

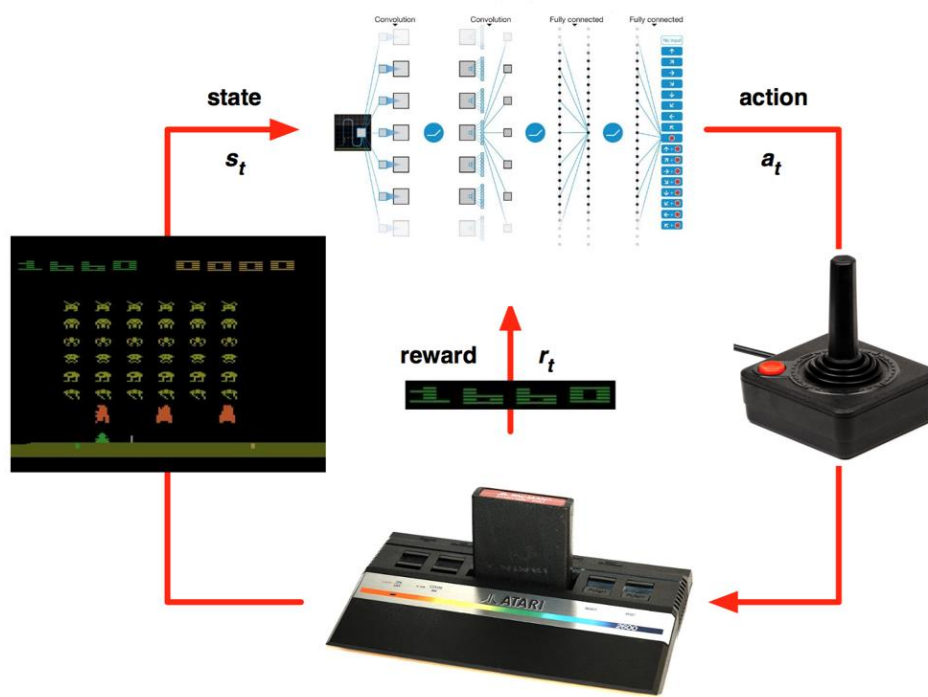


그림 7 게임 화면의 상태를 기준으로 Atari 게임의 행동을 선택하는 심층 신경망으로 대체된 Atari 게임의 구조

AI Soccer 알고리즘을 개발하기 위해서 심층 학습과 결합한 강화학습 프레임워크를 사용할 수 있다. Atari, 바둑, 체스 게임과 AI 축구의 차이점은 축구에는 역동적으로 상호작용하는 여러 로봇이 존재한다는 점이다. Atari 게임과 비슷한 AI 축구는 그림 8 과 같이 해석할 수 있다.

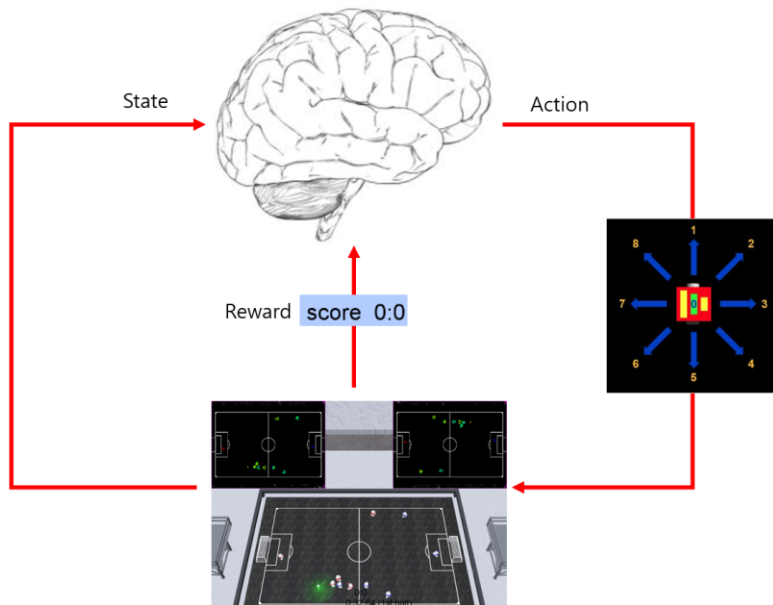


그림 8 AI 축구의 구조는 상태, 행동, 보상으로 구성된다.

그림 8에서 로봇들과 공의 좌표와 방향을 기반으로 게이머가 각 로봇의 동작을 결정해야 한다. 로봇이 어디로 어떻게 움직여야 하는지 등의 행동에 관련된 부분은 3 절에서 자세히 설명될 것이다. Atari 게임의 예처럼 신경망은 각 시간 간격(time step)마다 상태정보를 동작으로 매핑하는 용도로 사용될 수 있다.

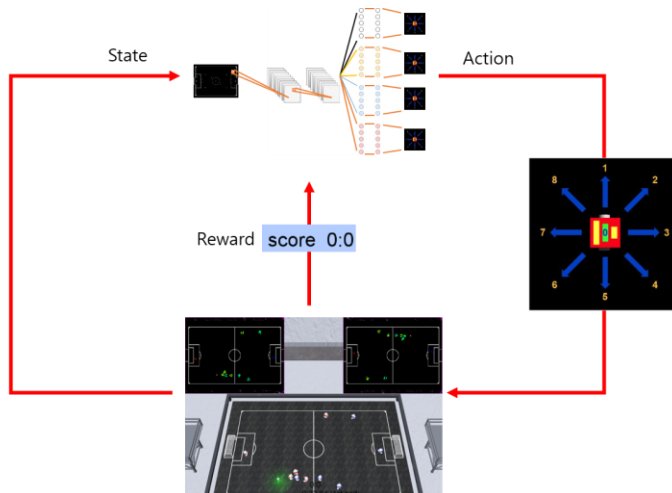


그림 9 게임 상태에 따라 로봇 동작을 선택하는 심층신경망으로 대체한 AI 축구 구조

이 설명서에서 제시될 AI 축구 플랫폼은 국내외 대회에서 2017 년부터 사용되었고, 중국 시안에서 열린 2019 WCG(World Cyber Games)에도 사용되었다. 참가팀들이 복잡하고 역동적인 축구 전략을 만들기 위해 다양한 AI 전략을 개발하였다.

이 설명서는 기본적인 성능을 보이는 예제 프로그램을 포함하여 게이머의 AI 축구 입문을 돕고자 한다.

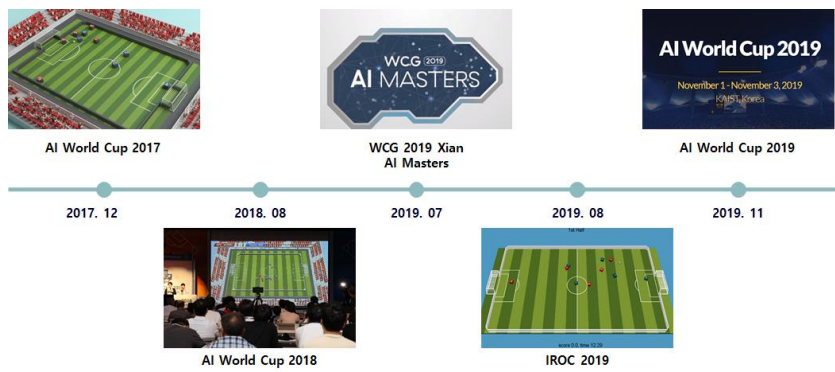


그림 10 AI 축구의 역사

AI 축구

AI 학습 플랫폼

AI 축구는 각 팀에서 5 명의 로봇을 제어하여 상대팀을 이기는 알고리즘을 개발하는 5 : 5 로봇 축구 게임이다. 제공된 프로그램을 사용하여 Python 프로그래밍 언어로 게이머 프로그램을 개발할 수 있다.

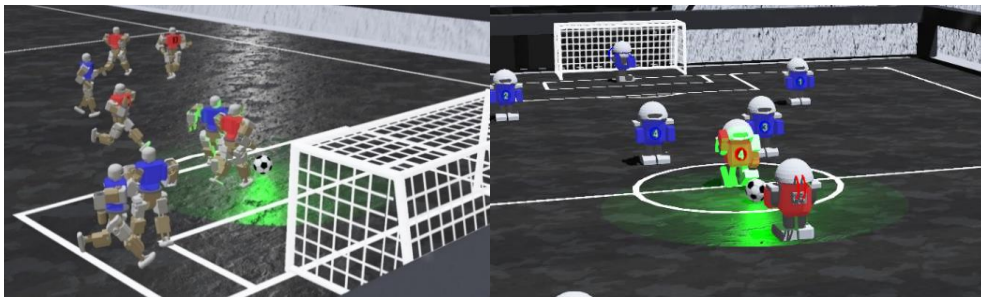


그림 11 로봇 플레이 환경: 왼쪽 -어른 로봇, 오른쪽 - 청소년 로봇

게이머 프로그램은 로봇과 공 좌표를 기반으로 각 로봇의 바퀴 속도를 조절하고, 킥 및 점프 동작을 조절하여 총 5 대의 로봇을 제어할 수 있다. AI 축구 로봇의 이동은 눈에 보이지 않지만 코드에 의해 제어할 수 있는 바퀴의 속도에 의해 결정된다. 자세한 내용은 3 절에 설명되어 있다. Python 에 대한 기초 지식을 갖춘 게이머는 이번 AI Soccer 설명서에 수록된 정보와 예제 프로그램을 바탕으로 자신만의 로봇 축구 전략을 만들 수 있다.

2

AI 추구 플랫폼 설치

이 절에서는 AI 추구 프로그램 개발을 위한 플랫폼 구축 및 실행 방법에 대해 안내하고자 한다.

2-1

시스템 요구사항

메모 [오전1]:

AI 추구 실행을 위해서는 다음과 같은 컴퓨터 하드웨어 사양이 요구된다.

- 최소 클럭 속도가 2GHz 이상의 dual-core CPU 와 2GB 이상의 RAM
- NVIDIA 또는 AMD OpenGL(version 3.3 이상) 지원이 가능한 그래픽 어댑터와 최소 512MB 이상의 그래픽 RAM 을 가진 그래픽카드

GPU 를 사용한 심층학습을 이용하고자 한다면 NVIDIA 그래픽 카드를 사용하고, 제공되는 CUDA, cuDNN library 를 설치하는 것을 추천한다.

GPU 를 사용하지 않더라도 딥러닝 라이브러리 설치가 가능하다.

- 사용 가능한 컴퓨터의 운영체제(OS)는 다음과 같다.

Windows 8.1 과 Window 10

최근 출시된 Ubuntu Linux OS(16.04 이후)

AI Soccer Simulator 를 설치하기 위해서는 다음의 3 단계를 거쳐야 한다.

- 1) Webots simulator 를 설치한다.
- 2) Python interpreter 를 설치한다.
- 3) AI Soccer simulator 파일을 다운로드한다.

만약 Python interpreter 가 이미 설치되어 있다면, 2)를 건너뛰어도 좋다.

게이머가 심층학습 예제 코드를 사용할 계획이라면 딥러닝 라이브러리를 설치하는 추가 단계가 필요하다.

4) 심층학습 라이브러리 설치

심층학습 예제코드를 실행하려면, 2-3-4(WINDOWS) 또는 2-4-4(LINUX)의 "심층학습 라이브러리 설치"에 있는 심층학습 라이브러리의 설치를 따라야 한다. 심층학습 예제 코드에서는 PyTorch 가 주요 딥러닝 라이브러리로 사용된다. 본 설명서에서는 Tensorflow 를 딥러닝 라이브러리의 또 다른 옵션으로 제시한다.

GPU 지원 없이 딥러닝 라이브러리를 설치하려면, 2-3-4(WINDOWS) 또는 2-4-4(LINUX)의 "GPU 를 사용하지 않는 심층학습 라이브러리 설치" 절차를 따른다..

NVIDIA 그래픽 카드가 있고, 심층학습 알고리즘을 교육할 수 있는 기능을 사용하려면 2-3-4(WINDOWS) 또는 2-4-4(LINUX)의 "GPU 를 사용하기 위한 심층학습 라이브러리 설치" 절차를 따른다.

자세한 내용은 유튜브의 "AI 축구 플랫폼 설치 방법" 비디오를 참조하라.

2-3

Windows OS 에 설치하기

2-3-1 Webots 시뮬레이터 설치

Webots Simulator 설치 가이드

AI 축구는 다른 게임과 달리 Webots(Web에서 동작하는 Robot들을 가리킴) 시뮬레이터를 게임 환경의 기반으로 한다. 따라서 Webots 시뮬레이터를 설치하여야 AI 축구를 실행할 수 있다.

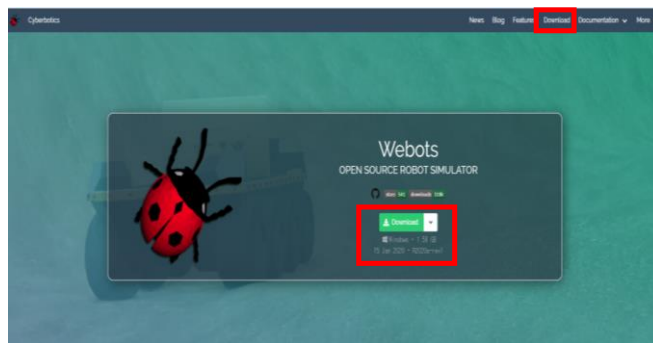


그림 12 Webots 시뮬레이터 다운로드 화면

Webots 시뮬레이터 다운로드 페이지 : <https://cyberbotics.com/#download>

Webots R2020a revision1 Windows installer를 다운로드 후 설치한다.

다운로드 한 뒤, 설치가 완료될 때까지 설치 지침을 따라 한다.

Webots 시뮬레이터 설치 완료 후 Webots 시뮬레이터에 'Tools' → 'Preferences' → 'General' → 'Startup mode' 과정을 따르고 Start mode 는 'Pause'로 설정한다. Python command 는 'python'으로 설정한다. (Python command 는 섹션 2-3-2 의 Python interpreter 설치에 따라 달라진다.)

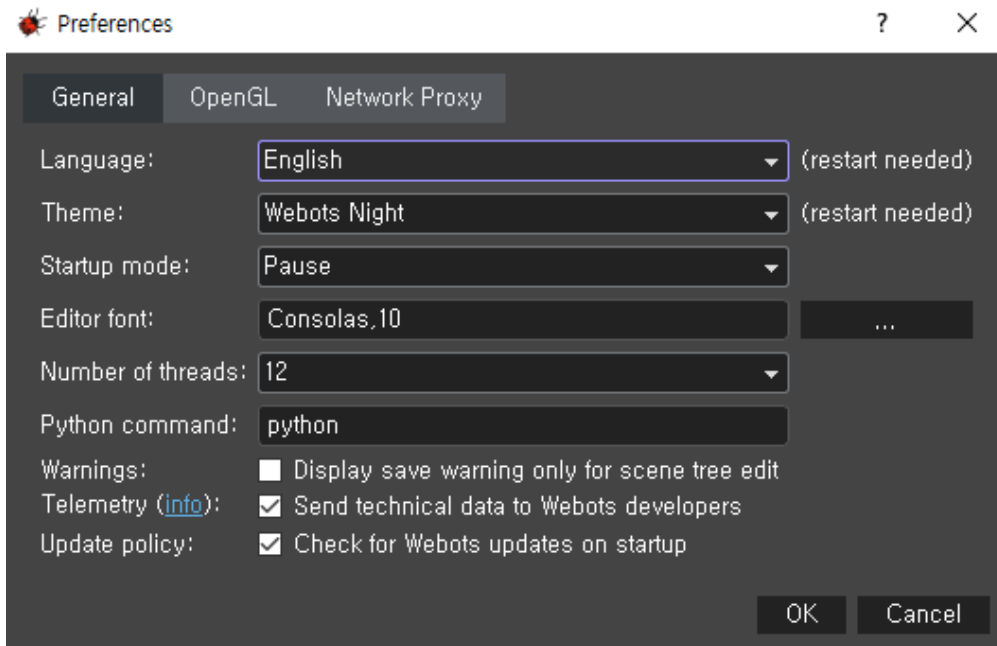


그림 13 Webots 시뮬레이터 Preferences 선택 창

Webots 시뮬레이터에 AI 축구 시뮬레이터를 구동하려면 환경변수(System Properties -> Environment Variables -> System Variables -> New)에 Webots 의 PYTHONPATH 를 추가해야한다. 2-3-3 절에 Python 인터프리터를 설치한 후 컴퓨터에 설치된 Python 버전에 맞춰 PYTHONPATH 를 추가하십시오.

PYTHONPATH = `{WEBOTS_HOME}/lib/controller/python3X`

'python3x'는 설치된 파이썬 버전과 관련되어 있다. 이 과정은 아래의 그림 14 에 설명되어 있다. PYTHONPATH 를 환경변수에 추가하는 방법에 대한 자세한 내용은 <https://cyberbotics.com/doc/guide/running-extern-robot-controllers> 를 참조하면 된다.

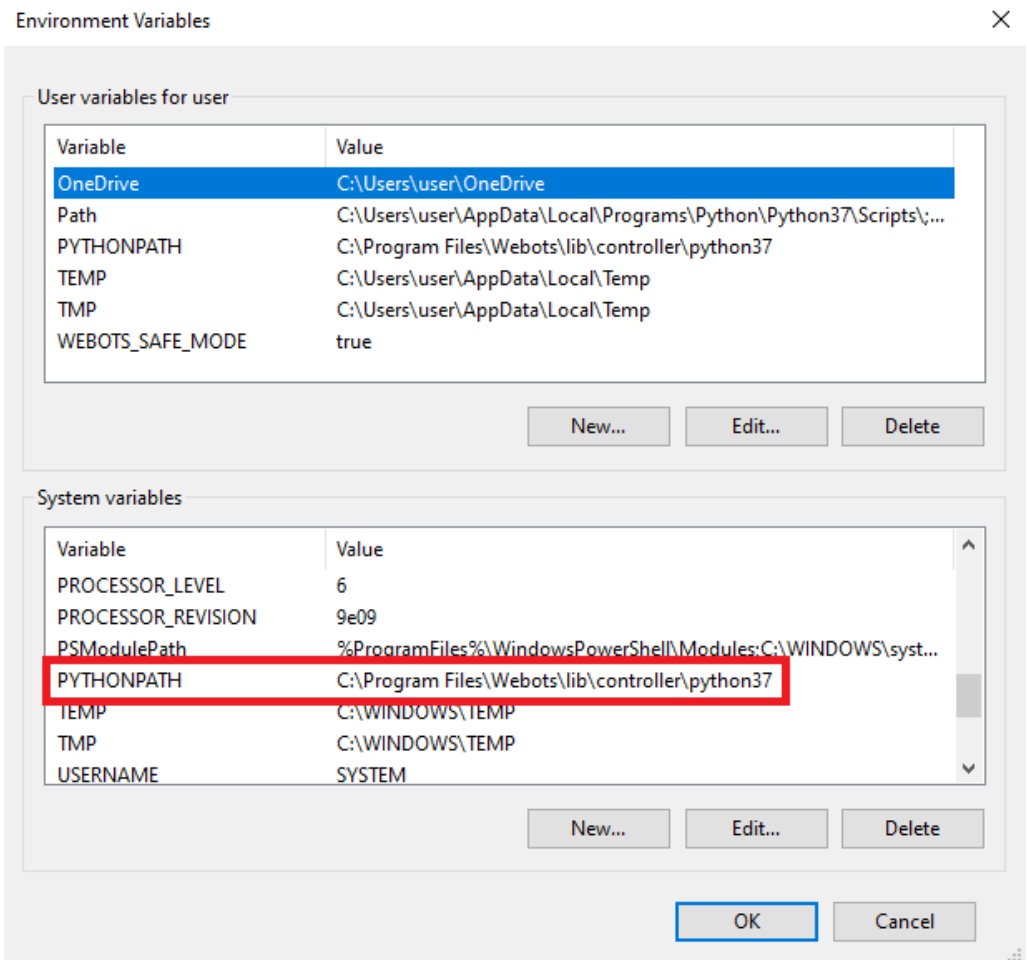


그림 14 시스템 환경변수에 PYTHONPATH 추가하기

2-3-2 Python 설치

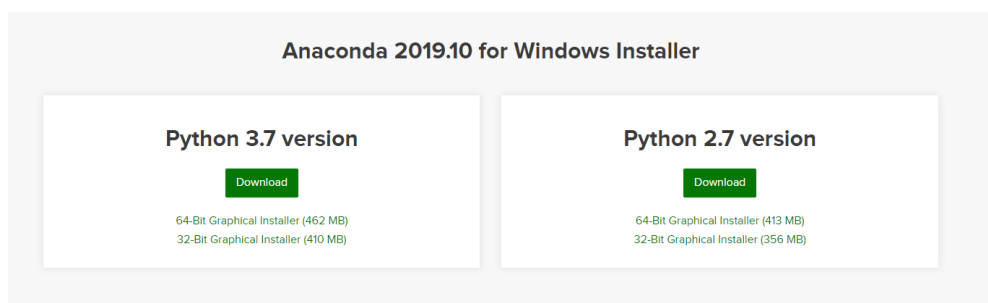
Windows 상에서 AI 추구를 실행시키기 위해선 Python (interpreter) 이 필요하다. 추천 버전은 Python 3.7 이다. Python 을 수동으로 설치하거나, Anaconda(기계 및 심층학습 프로그램 개발을 위한 numpy 등 필수 라이브러리를 포함함)를 이용하여 설치하는 것이 가능하다. Python 수동 설치에 익숙하지 않은 경우, Anaconda 를 이용하여 Python 을 설치하길 권장한다.

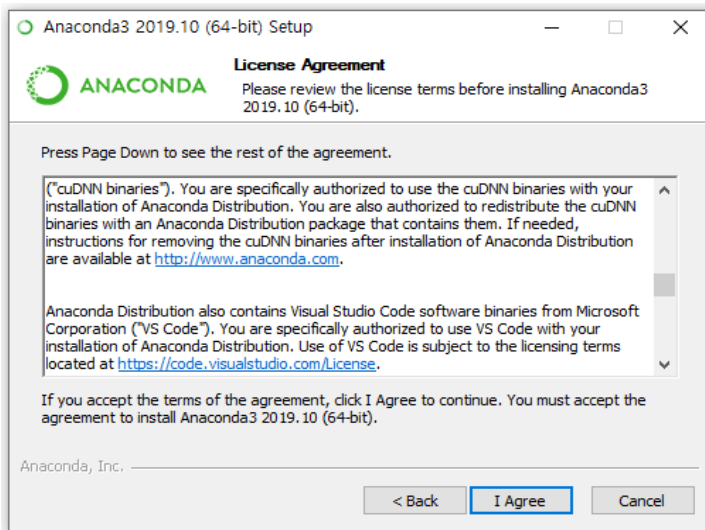
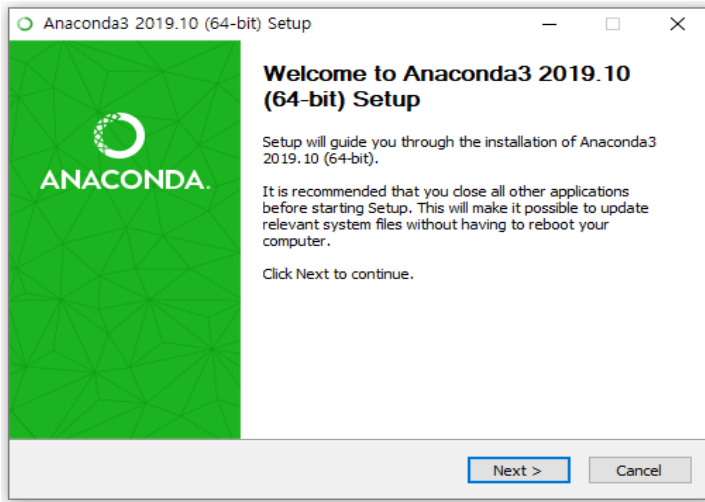
1. Python 수동설치 방법

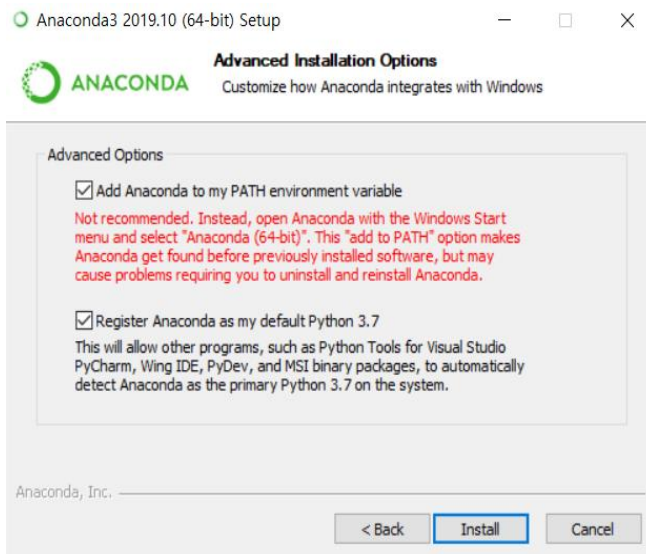
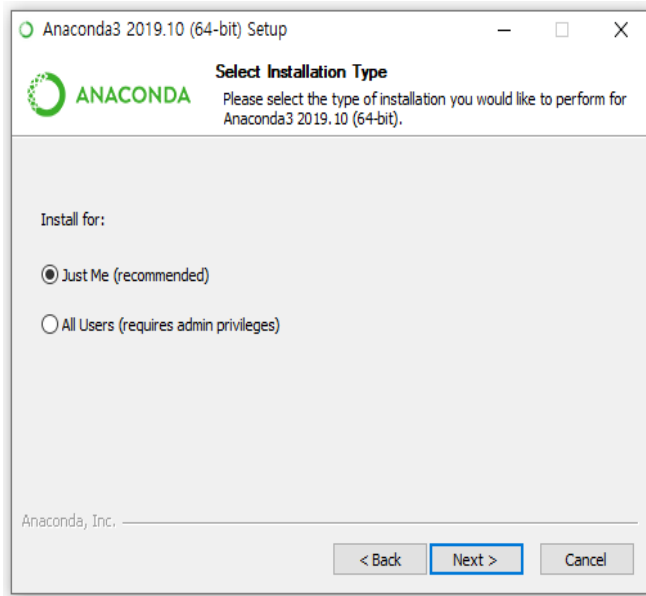
- a. Python 3.7 다운로드 : <https://www.python.org/>
- b. PIP package manager 을 사용하여 필수 라이브러리 설치
 - i. numpy
 - ii. opencv-python

2. Anaconda 를 이용하여 설치 (아래에 설명)

<https://www.anaconda.com/distribution/> 을 통해 Anaconda Python 3.7 버전을 다운로드한다. 실행 중인 컴퓨터 시스템의 Window 버전에 따라 Python 3.7 을 다운로드 한다. (32bit 64bit)







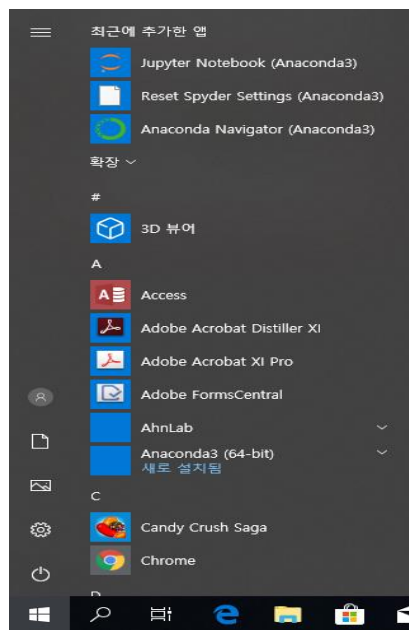
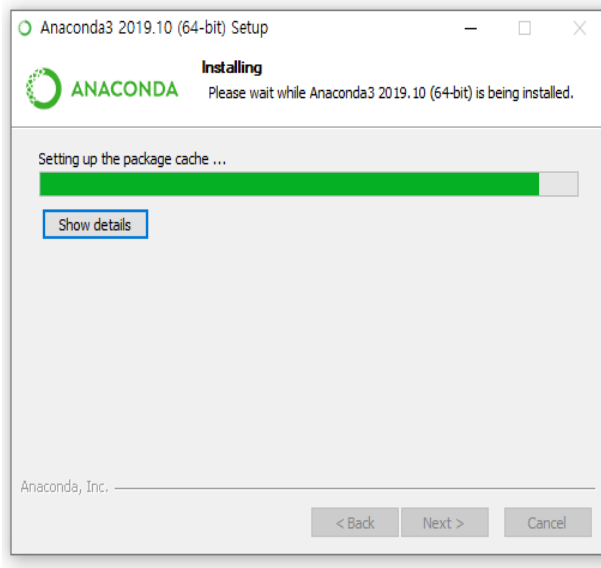


그림 15 Anaconda 설치 과정 스크린샷

다운로드한 .exe 파일을 실행하여 위의 그림 15 와 같이 Anaconda 설치를 진행하며 “Add Anaconda to my PATH environment variable” 옵션을 선택하고 설치를 진행하여 윈도우 터미널에서 바로 Python 을 사용할 수 있도록 한다.

설치 완료 후 Windows 버튼을 누르거나 Windows command 창을 열어 마지막 그림 16 과 같이 Python 이 설치되었는지 확인한다.

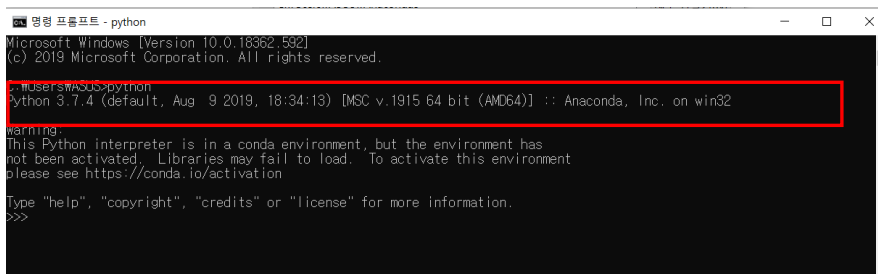


그림 16 Python 설치 확인

명령 프롬프트(cmd)에서 명령어(python --version, conda --version) 실행 후 그림 17 과 같이 버전 정보가 출력되면 설치가 완료된 것이다.



그림 17 Python 버전과 Anaconda 버전 확인

2-3-3 AI 축구 시뮬레이터 설치

AI 축구 시뮬레이터는 AI 축구 게임 구현과 예제 코드가 담긴 패키지이다. AI 축구 시뮬레이터는 Webots 시뮬레이터에 World(환경)형태로 탑재된다. AI 축구 시뮬레이터를 다운로드하려면 : <https://github.com/aisoccer/aisoccer-3d/releases> 에 들어가 zip 파일을 다운로드해야 한다. 원하는 폴더에서 zip 파일을 압축해제 한다.

Windows 에서 AI 축구 시뮬레이터를 사용하려면 "v0.1 Release Windows" 에 포함된 'aisoccer-3d.zip' 파일을 다운로드하면 된다.

2-3-1, 2-3-2 및 2-3-3 에 설명된 단계를 통해 설치의 첫 번째 부분이 완료되었다. AI 축구 시뮬레이터는 이미 규칙 기반 예제 코드에 대한 실행이 가능하다.

규칙 기반 예제 코드를 사용하여 설치를 테스트하려면 2-5 설치 확인으로 이동하십시오.

그러나 심층 학습 사례를 실행하려면 2-3-4 에 있는 심층 학습 라이브러리를 설치해야 한다. 심층 학습 라이브러리와 호환되는 NVIDIA 그래픽 카드가 있는 경우 다음과 같은 두 가지 옵션이 있다.

1. GPU 사용 없이 심층 학습 라이브러리 사용
2. GPU 를 사용하여 심층 학습 라이브러리

2-3-4 심층 학습 라이브러리 설치

NVIDIA 그래픽 카드가 있는 경우 그래픽 카드를 사용하여 심층 학습 라이브러리를 사용하여 컴퓨터 성능을 향상할 수 있다. NVIDIA 드라이버와 소프트웨어인 CUDA 및 cuDNN 을 설치해야 한다.

만약 GPU 지원 라이브러리를 사용하지 않으려면 심층 학습 라이브러리를 CPU 만을 이용하여 심층 학습을 실행할 수 있다. 그것에 대한 설명도 다음에 제시될 것이다.

다음과 같은 순서대로, 필요한 프로그램들을 설치할 것이다.

- NVIDIA DRIVER
- CUDA
- cuDNN
- 심층 학습 라이브러리: Tensorflow, PyTorch

이러한 라이브러리 중 하나가 컴퓨터에 이미 설치되어 있으면 해당 설치를 건너뛸 수 있다.

GPU 를 사용하지 않는 심층 학습 라이브러리 설치

Windows AI 추구 게임 환경 구축

터미널에서 'pip'패키지 관리자를 통해 다음 명령어를 실행시켜 Tensorflow 와 PyTorch 를 설치하십시오. 이 경우 PyTorch 만 설치해도 무방하다.

- `pip install --upgrade tensorflow`
- `pip install torch==1.5.0+cpu torchvision==0.6.0+cpu -f https://download.pytorch.org/whl/torch_stable.html`

GPU 를 사용하기 위한 심층 학습 라이브러리 설치

1. NVIDIA 드라이버 설치

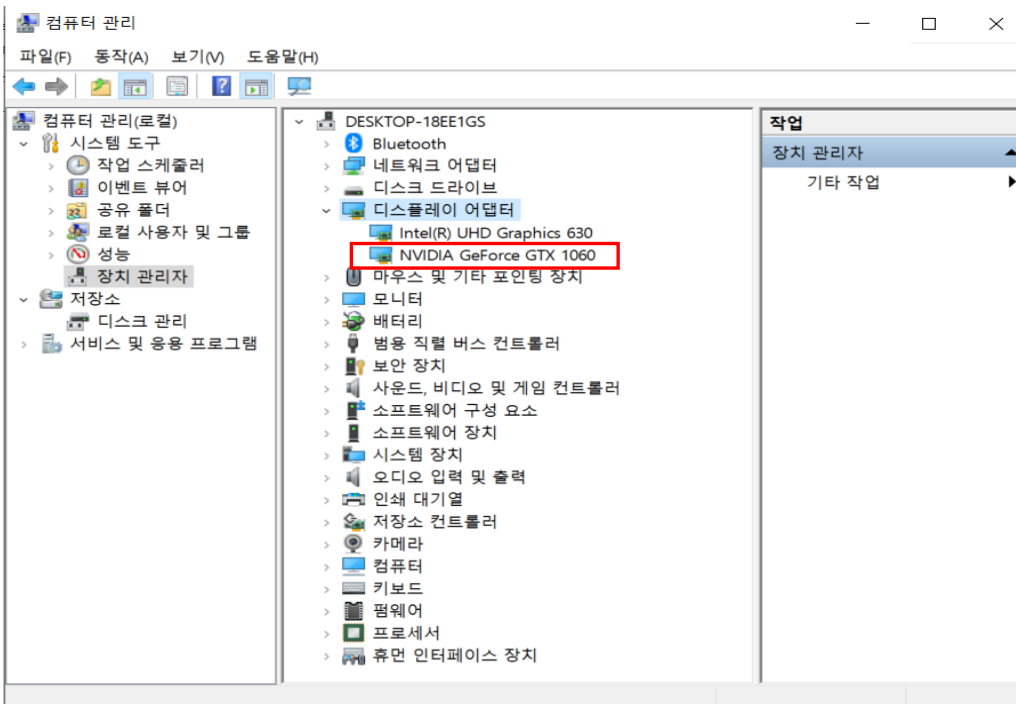


그림 18 장치관리자를 이용한 NVIDIA GPU 정보 찾기

AI 추론 실행시 NVIDIA 드라이버가 설치되어 있어야 원활한 속도로 게임 시뮬레이터를 사용할 수 있다. 자신의 컴퓨터에 NVIDIA GPU 가 없다면 드라이버를 설치할 수 없으며 NVIDIA 드라이버가 자신의 컴퓨터에 없어도 시뮬레이터를 사용할 수 있으나 낮은 속도로 추론이 실행된다. 그림 18 과 같이 컴퓨터 장치 관리자에서 디스플레이 어댑터에 들어가면 자신의 GPU 정보를 확인할 수 있다.

NVIDIA 드라이버 다운로드

옵션 1: 사용자가 사용하는 NVIDIA 제품용 드라이버를 수동으로 검색합니다.

[상세 가이드 보기](#)

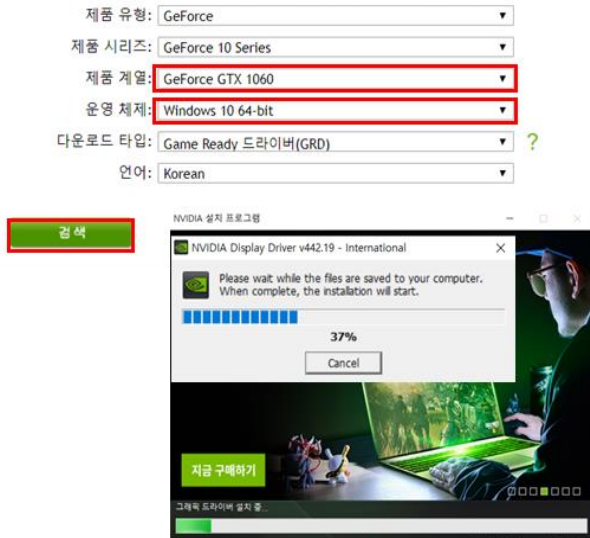


그림 19 NVIDIA Driver download

NVIDIA 드라이버는 <https://www.nvidia.co.kr/Download/index.aspx?lang=kr>

의 페이지에서 **자신의 GPU 모델에 맞는 버전의 드라이버**를 다운로드하여 설치한다.

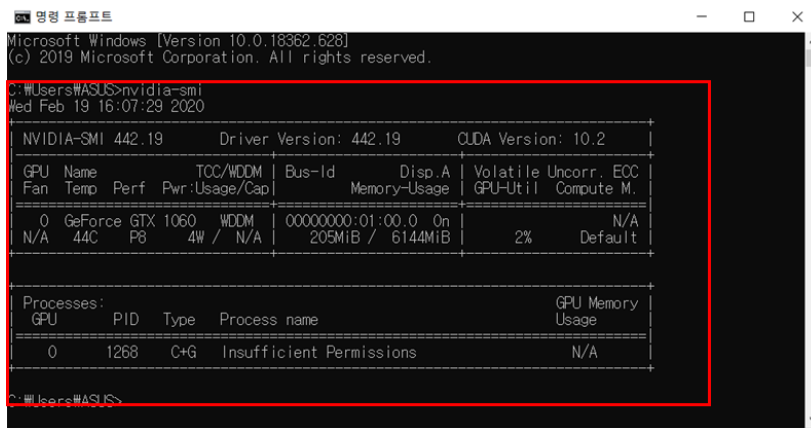


그림 20 명령어 nvidia-smi 실행화면

설치 완료 후 명령 프롬프트(cmd)에서 명령어: nvidia-smi 를 통해 드라이버 설치 여부를 확인할 수 있으며 그림 20 과 같은 출력이 되어야 설치가 완료된 것이다.

2. CUDA, cuDNN 설치(학습 기반 코드 사용 시)

Webots Simulator 는 GPU 보드가 설치되었을 경우 보다 빨리 작동한다. 그렇지만 CUDA, cuDNN 의 설치가 상대적으로 까다롭기 때문에 초보자는 PyTorch 만 설치해도 무방하다. PyTorch 만으로도 학습 기반 코드의 작성이 가능하다. 이 경우 3 으로 건너뛰어도 된다.

Tensorflow, PyTorch 등의 딥 러닝 라이브러리를 사용하기 위하여 NVIDIA 드라이버 설치와 함께 CUDA, cuDNN 를 설치하여야 한다.

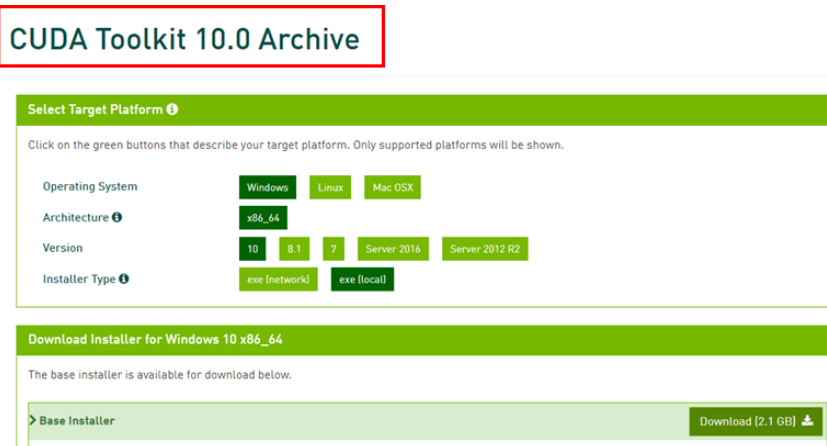


그림 21 CUDA 툴킷 설치 페이지

CUDA 는 <https://developer.nvidia.com/cuda-toolkit-archive> 의 홈페이지에 접속해 자신의 컴퓨터 GPU 버전에 알맞은 CUDA 버전을 다운로드한다.



그림 22 cuDNN 설치 페이지

CUDA 다운로드가 완료되면, <https://developer.nvidia.com/rdp/cudnn-download> 의 홈페이지에서 CUDA Toolkit 과 동일한 버전의 cuDNN 버전을 다운로드한다.

(회원가입 후 다운로드 가능)

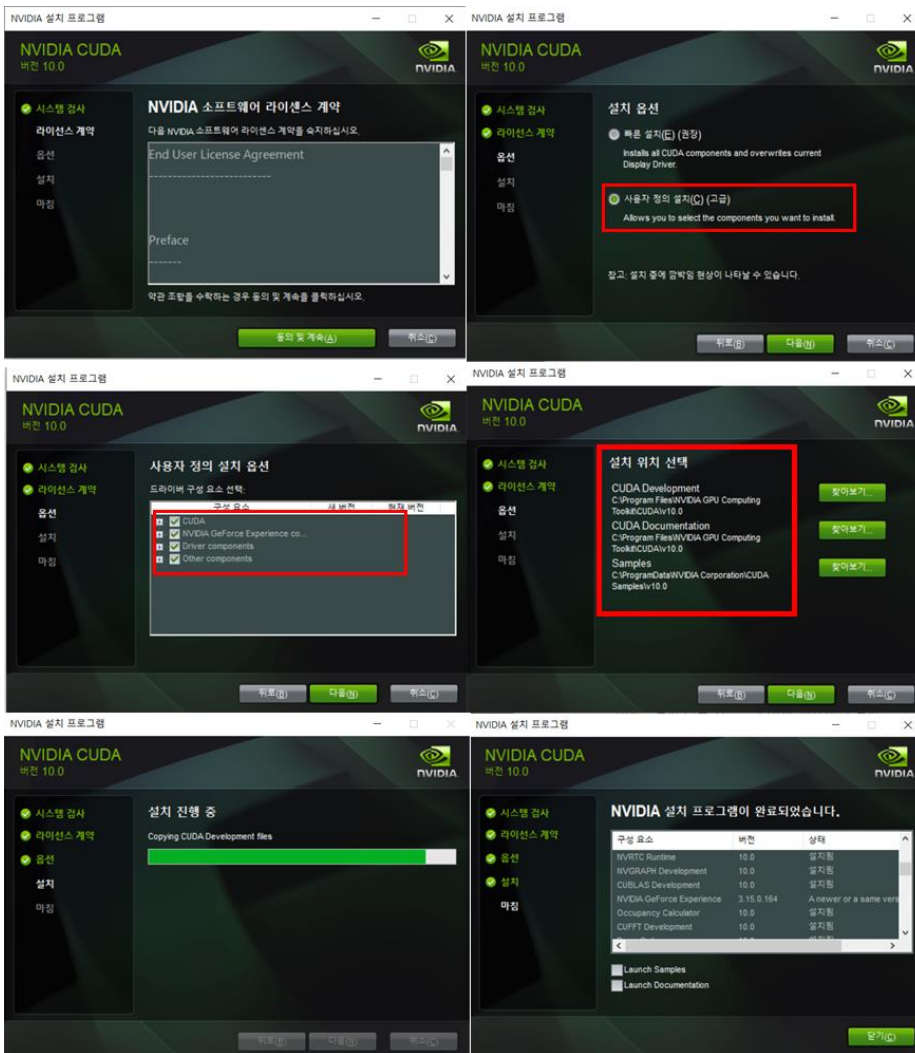
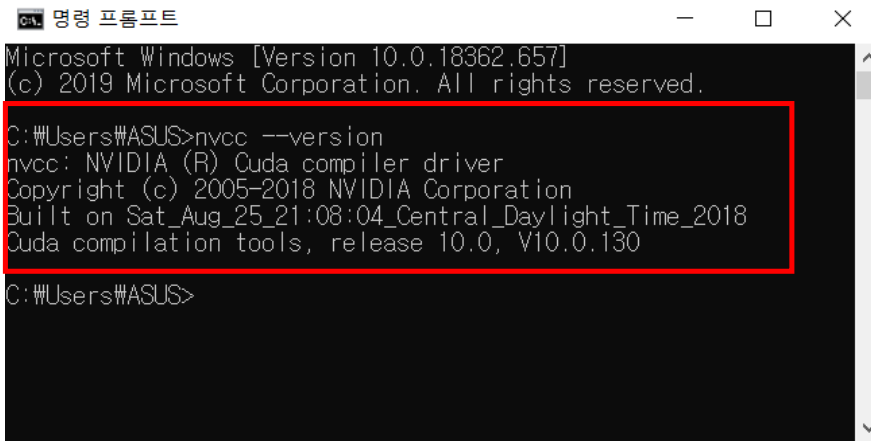


그림 23 CUDA 설치 진행 과정

그림 23 과 같이 '사용자 정의 설치' 옵션으로 CUDA 설치를 진행하며 CUDA 설치 위치를 기억해 둔다.



```
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\#ASUS>nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Sat_Aug_25_21:08:04_Central_Daylight_Time_2018
Cuda compilation tools, release 10.0, V10.0.130

C:\Users\#ASUS>
```

그림 24 CUDA 설치 확인 명령어: nvcc --version 실행 화면

CUDA 설치 완료 후 그림 24 와 같이 명령 프롬프트(cmd)를 실행하여 *명령어: nvcc --version* 실행 시 위와 같이 버전 정보가 출력되는 것을 확인한다.



그림 25 cuDNN SDK 폴더

CUDA 설치 완료 후 위 그림 25 와 같이 cuDNN SDK 압축 풀기를 진행한 후 CUDA 설치 경로로 들어가 cuDNN 의 bin, include, lib 폴더를 복사하여 그림 26 의 CUDA 폴더에 덮어 씌운다.

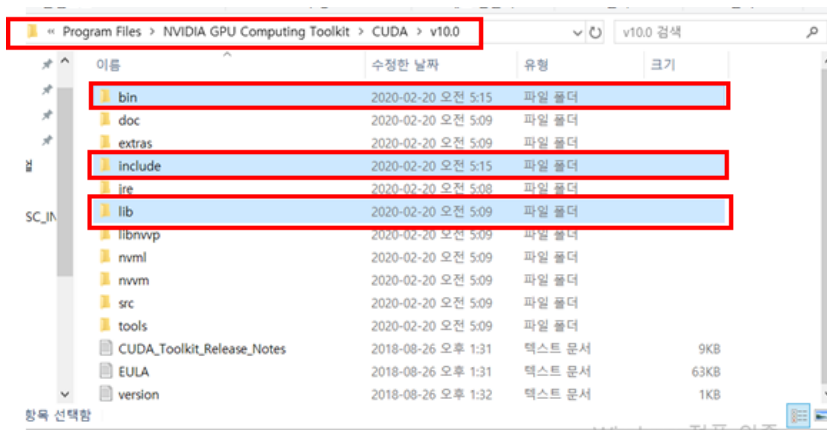


그림 26 CUDA 폴더

3. Tensorflow , PyTorch 설치 (심층 학습 라이브러리 설치)

학습 기반 코드를 사용하기 위하여 NVIDIA 드라이버, CUDA, cuDNN 을 설치한 후에 명령 프롬프트(cmd) 창이나 Anaconda 프롬프트를 사용하여 자신이 사용할 Tensorflow 나 PyTorch 를 명령어로 설치한다.

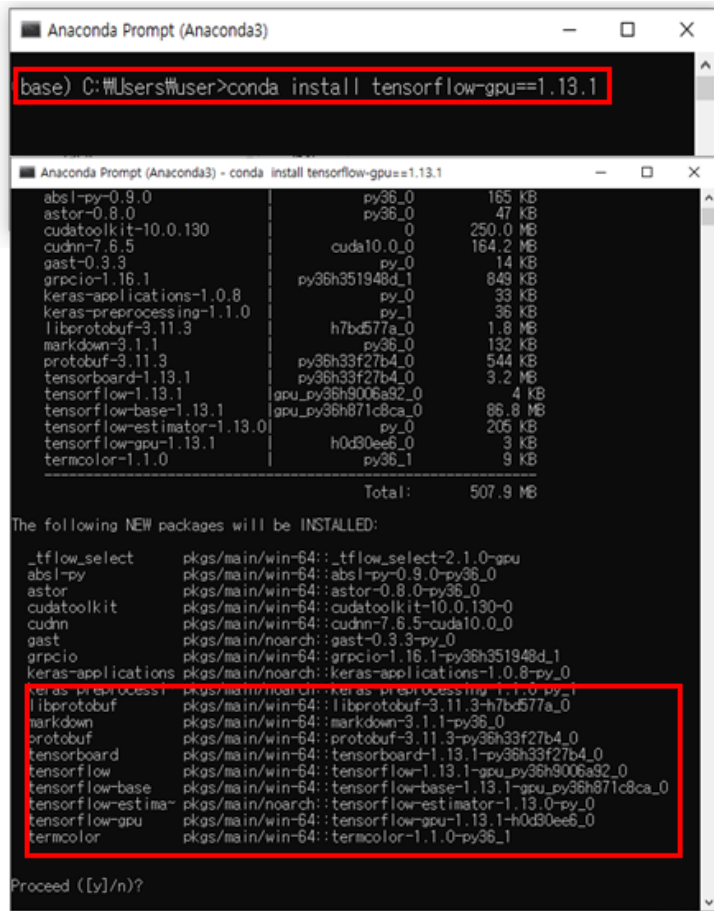


그림 27 Tensorflow 설치 명령어: conda install tensorflow-gpu==1.13.1

Tensorflow 설치를 위한 cmd 는 아래와 같다.

명령어: conda install tensorflow-gpu==1.13.1

```
win_inet_pton 1.1.0 py37_0
win_unicode_console 0.5 py37_0
windertstore 0.2 py37_0
winpty 0.4.3 4
wrapit 1.11.2 py37he774522_0
wrt 1.2.0 py37_0
xlswriter 1.2.1 py_0
xlwings 0.15.10 py37_0
xvnt 1.3.0 py37_0
xz 5.2.4 h2fa13f4_4
yaml 0.1.7 hc54c50b_2
zeromq 4.3.1 h33f27b4_3
zict 1.0.0 py_0
zipp 0.6.0 py_0
zlib 1.2.11 h62ddc97_3
zstd 1.3.7 h508b16e_0

C:\Users\ASUS>nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Sat_Aug_25_21:08:04_Central_Daylight_Time_2018
Cuda compilation tools, release 10.0, V10.0.130

C:\Users\ASUS>conda install pytorch torchvision cudatoolkit=10.0 -c pytorch

The following NEW packages will be INSTALLED:
  cudatoolkit pkgs/main/win-64::cudatoolkit-10.0.130-0
  ninja pkgs/main/win-64::ninja-1.9.0-py37h74a9793_0
  pytorch pytorch/win-64::pytorch-1.2.0-py3.7_cuda100_cudnn7_1
  torchvision pytorch/win-64::torchvision-0.4.0-py37_cu100

The following packages will be UPDATED:
  conda 4.7.12-py37_0 -> 4.8.2-py37_0

Proceed ([y]/n)?
```

그림 28 PyTorch 설치 명령어: conda install pytorch torchvision cudatoolkit=10.0 -c pytorch

PyTorch 설치의 확인을 위한 cmd 는 아래와 같다.

```
명령어: conda install pytorch torchvision cudatoolkit=10.0 -c pytorch
```

2-4-1 Webots 시뮬레이터 설치

Ubuntu 18.04 는 Windows 10 처럼 컴퓨터 환경을 조성하는 운영 체제(OS) 중 하나이며 로봇 시뮬레이터 및 프로그래밍 개발에 최적화된 운영체제이다. Ubuntu 18.04 는 안정적 장기 지원(LTS) 버전이기 때문에, 이 튜토리얼의 Linux OS 로 선택되었다.

Windows 와 달리 운영체제 자체가 Python 언어로 되어 있기 때문에 별도로 Python 을 설치하지 않아도 되며 대부분의 설치가 터미널 창에서 명령어를 통해 가능하다.

이미 Ubuntu 가 설치되어 있다면, 41 페이지로 바로 넘어가도 좋다.

다운로드 링크:

<https://ubuntu.com/download/desktop/thankyou?country=KR&version=18.04.4&architecture=amd64>

-설치 가이드:

<https://jeongwookie.github.io/2019/05/21/190521-ubuntu-install-usb/>

을 참고하여 Ubuntu 18.04 의 설치가 가능하다.

Ubuntu 18.04 을 설치하기 전에 왼쪽 상단에 있는 'Activities' 클릭 'Activities'에서 '터미널(Terminal)'을 실행한다.


```
test@test: ~
File Edit View Search Terminal Help
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

test@test:~$ sudo apt-get update
[sudo] password for test:
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://kr.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://kr.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://kr.archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
test@test:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
 libdrm-amdgpu1 libdrm-common libdrm-intel1 libdrm-nouveau2 libdrm-radeon1
 libdrm2 libegl-mesa0 libegl1-mesa libexif12 libexiv2-14 libgbm1
 libgl1-mesa-dri libgl1-mesa-glx libglapi-mesa libglx-mesa0 libllvm9
 libnss-myhostname libnss-systemd libpam-systemd libsystemd0 libudev1
 libwayland-client0 libwayland-cursor0 libwayland-egl1 libwayland-egl1-mesa
 libwayland-server0 libxatracker2 libxml2 python3-pil python3-renderpm
 python3-reportlab python3-reportlab-accel systemd systemd-sysv udev
35 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

그림 29 Ubuntu 커널 최신 상태 유지

그림 29 와 같이 명령어(*sudo apt-get update*, *sudo apt-get upgrade*)를 터미널 창에서 사용하여 Ubuntu 커널을 최신 상태로 유지한다. 이후 모든 설치는 터미널(Terminal)을 통해 진행한다.

이 설명서에서는 Webots 시뮬레이터를 Ubuntu 18.04 버전으로 설치하였다. Ubuntu 18.04 버전에서는 터미널에서 'wget' 및 'dpkg' 명령어를 사용하여 다른 프로그램과 마찬가지로 Webots 를 설치할 수 있다.

아래의 명령어들을 사용해 Webots 를 설치한다. (Windows 와 같은 방법으로 사이트에 방문해서 다운로드받아도 무방하다)

- `wget https://github.com/cyberbotics/webots/releases/download/R2020a-rev1/webots_2020a-rev1_amd64.deb`
- `sudo dpkg -i webots_2020a-rev1_amd64.deb`
- `sudo apt --fix-broken install`

Ubuntu 에서는 터미널에 명령어: `webots` 을 통하여 프로그램 실행이 가능하다.

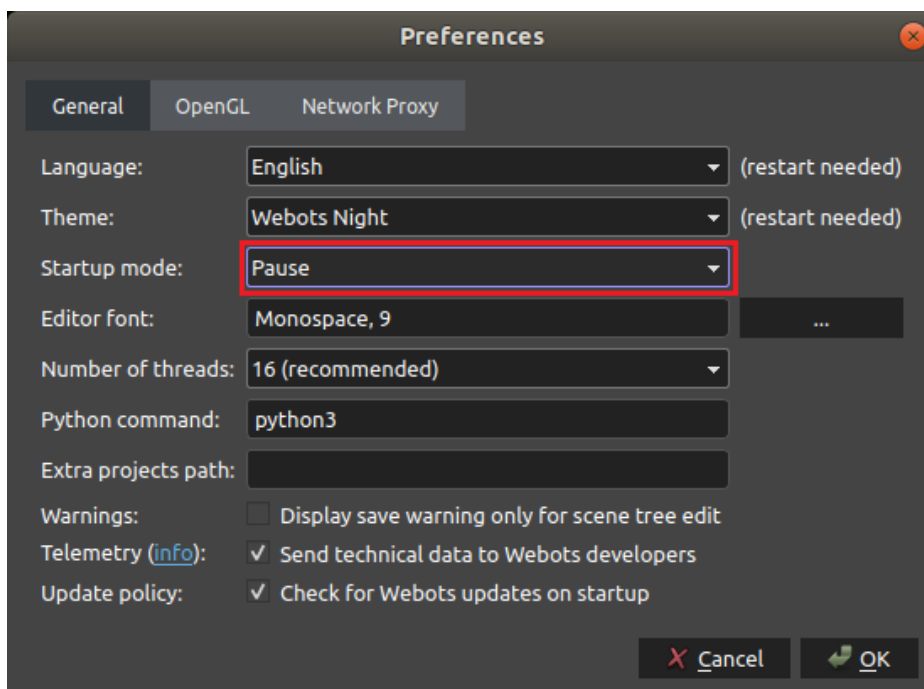


그림 30 Webots 시뮬레이터 Preferences 선택 창(Ubuntu 18.04)

Webots 시뮬레이터에서 AI 축구 시뮬레이터를 사용하려면 환경변수에 Webots의 PYTHONPATH 를 추가해야 한다. Python 을 2-4-2 에 따라 설치후 Python 버전('python3x')에 맞춰서 PYTHONPATH 를 설정한다.

```
PYTHONPATH = ${WEBOTS_HOME}/lib/controller/python3X
```

Ubuntu 에서는 Webots 시뮬레이터 실행 전 매번 PYTHONPATH 를 불러와야 한다. 터미널을 이용하여 별도의 페이지를 열지 않아도 아래의 명령어를 사용하여 불러오는 작업을 수행할 수 있다.

```
export PYTHONPATH = /usr/local/webots/lib/controller/python37
```

만약 Webots 를 실행할 때마다 PYTHONPATH 를 매번 추가하지 않으려면, '~/.bash_profile' 파일에 PYTHONPATH 변수를 추가하면 된다. PYTHONPATH 를 환경변수에 추가하는 방법에 대한 자세한 내용은

<https://cyberbotics.com/doc/guide/running-extern-robot-controllers>.을

참조한다.

2-4-2 Python 설치

일반적으로 Ubuntu OS 에는 기본적으로 Python interpreter 가 설치되어 있다.

Python 3.5 이상의 버전을 사용하기를 권장하며 3.7 을 사용하길 추천한다.

Python 버전을 확인하고 싶다면, 터미널을 열어서

```
python --version
```

을 입력하면 확인할 수 있다.

PIP 패키지 관리자를 통해 필요한 라이브러리를 설치한다.

i. numpy

ii. opencv-python

설치한 Python 버전이 제대로 출력된다면, AI Soccer Python 예제를 실행하기 위한 모든 준비가 끝났다.

2-4-3 AI 축구 시뮬레이터 설치

AI 축구 시뮬레이터는 AI 축구 게임 구현과 예제 코드가 담긴 패키지이다. AI 축구 시뮬레이터는 Webots 시뮬레이터에 World(환경)형태로 탑재된다. AI 사커 시뮬레이터를 다운로드하려면 : <https://github.com/aisoccer/aisoccer-3d/releases> 에 들어가 zip 파일을 다운로드해야 한다. 원하는 폴더에서 zip 파일을 압축해제 한다.

2-4-1, 2-4-2 및 2-4-3 에 설명된 단계를 통해 설치의 첫 번째 부분이 완료되면. AI 축구 시뮬레이터를 이용한 규칙 기반 예제 코드에 대한 실행이 가능하다.

규칙 기반 예제 코드를 사용하여 설치를 테스트하려면 2-5 로 이동하십시오.

그러나 심층 학습 사례를 실행하려면 2-4-4 에 있는 심층 학습 라이브러리를 설치해야 한다. 심층 학습 라이브러리와 호환되는 NVIDIA 그래픽 카드가 있는 경우 다음과 같은 두 가지 옵션이 있다.

1. GPU 사용 없이 심층 학습 라이브러리 사용
2. GPU 를 사용하여 심층 학습 라이브러리

2-4-4 심층 학습 라이브러리 설치

Linux AI 추론 게임 환경 구축

NVIDIA 그래픽 카드가 있는 경우 그래픽 카드를 사용하여 심층 학습 라이브러리를 사용하여 컴퓨터 성능을 향상할 수 있다. NVIDIA 드라이버와 소프트웨어인 CUDA 및 cuDNN을 설치해야 한다.

만약 GPU 지원 라이브러리를 사용하지 않으려면 심층 학습 라이브러리를 CPU만을 이용하여 심층 학습을 실행할 수 있다. 그것에 대한 설명도 다음에 제시될 것이다.

다음과 같은 순서대로, 필요한 프로그램들을 설치할 것이다.

- NVIDIA DRIVER
- CUDA
- cuDNN
- 심층 학습 라이브러리: Tensorflow, PyTorch

GPU 를 사용하지 않는 심층 학습 라이브러리 설치

Linux AI Soccer Game Environment Setup

터미널에서 'pip' 패키지 관리자를 통해 다음 명령어를 실행시켜 Tensorflow 와 PyTorch 를 설치하십시오. 이 경우 PyTorch 만 설치해도 무방하다.

- `pip install --upgrade tensorflow`
- `pip install torch==1.5.0+cpu torchvision==0.6.0+cpu -f https://download.pytorch.org/whl/torch_stable.html`

GPU 를 사용하기 위한 심층 학습 라이브러리 설치

Linux AI Soccer Game Environment Setup

1. NVIDIA 드라이버 설치

AI 축구 실행시 NVIDIA 드라이버가 설치되어 있어야 원활한 속도로 게임 시뮬레이터를 사용할 수 있다. 자신의 컴퓨터에 NVIDIA GPU 가 없다면 드라이버를 설치할 수 없으며 NVIDIA 드라이버가 자신의 컴퓨터에 없어도 시뮬레이터를 사용할 수 있으나 낮은 속도로 축구 경기가 실행된다.

터미널에 아래의 명령어들을 차례로 사용하여 NVIDIA Driver 를 설치한다.

- `sudo add-apt-repository ppa:graphics-drivers/ppa`
- `sudo apt update`
- `sudo ubuntu-drivers autoinstall`

Driver 설치가 완료된 후 터미널에 '명령어: `sudo reboot`'를 통하여 컴퓨터를 재부팅한다.


```
test@test: ~
File Edit View Search Terminal Help
test@test:~$ nvidia-smi
Thu Feb 13 16:00:21 2020

+-----+
| NVIDIA-SMI 440.59          Driver Version: 440.59          CUDA Version: 10.2   |
+-----+-----+-----+-----+-----+-----+
| GPU  Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
| 0   GeForce GTX 108...    Off          | 00000000:09:00.0 On  |          N/A         |
| 0%   55C    P8     12W / 250W | 192MiB / 11175MiB |           2%    Default |
+-----+-----+-----+-----+-----+-----+
| 1   GeForce GTX 108...    Off          | 00000000:0A:00.0 Off |          N/A         |
| 0%   53C    P8     10W / 250W |  2MiB / 11178MiB |           0%    Default |
+-----+-----+-----+-----+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type   Process name                               Usage      |
+-----+-----+-----+-----+-----+-----+
|    0         1291    G     /usr/lib/xorg/Xorg                           14MiB      |
|    0         1339    G     /usr/bin/gnome-shell                         49MiB      |
|    0         1618    G     /usr/lib/xorg/Xorg                           63MiB      |
|    0         1762    G     /usr/bin/gnome-shell                         61MiB      |
+-----+-----+-----+-----+-----+-----+
```

그림 31 nvidia-smi 실행 화면

설치 완료 후 명령어: nvidia-smi 를 사용하여 드라이버 설치 여부를 확인할 수 있으며 그림 31 과 같이 출력 되면 설치가 완료된 것이다.

2. CUDA, cuDNN 설치(학습 기반 코드 사용시)

Tensorflow, PyTorch 등의 심층 학습 라이브러리를 사용하기 위하여 NVIDIA Driver 설치와 함께 CUDA, cuDNN 를 설치하여야 한다. **그렇지만 CUDA, cuDNN 의 설치가 상대적으로 까다롭기 때문에 초보자는 PyTorch 만 설치해도 무방하다. PyTorch 만으로도 학습 기반 코드의 작성이 가능하다. 이 경우 3 으로 건너뛰어도 된다.**

터미널에서 아래의 명령어들을 차례로 사용하여 자신의 컴퓨터 GPU 버전에 알맞은 CUDA 버전을 설치한다. (CUDA 9.0, CUDA 10.0, CUDA 10.1, CUDA 10.2)

터미널에서 아래의 명령어들을 차례로 사용해서 CUDA Toolkit 를 설치한다

- `wget`
`https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin`
- `sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600`
- `wget`
`http://developer.download.nvidia.com/compute/cuda/10.2/Prod/local_installers/cuda-repo-ubuntu1804-10-2-local-10.2.89-440.33.01_1.0-1_amd64.deb`
- `sudo dpkg -i cuda-repo-ubuntu1804-10-2-local-10.2.89-440.33.01_1.0-1_amd64.deb`

- `sudo apt-key add /var/cuda-repo-10-2-local-10.2.89-440.33.01/7fa2af80.pub`
- `sudo apt-get update`
- `sudo apt-get -y install cuda`

CUDA 설치가 완료된 후 터미널에서 명령어: `getit ~/.bashrc` 을 실행한다.

```

# colored GCC warnings and errors
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'

# some more ls aliases
alias ll='ls -lF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "${?} = 0" && echo terminal || echo error)' "$
(history|tail -ni|sed -e '\s/\s*[0-9]\+\s*//;s/;/&]\s*alert$//\`)'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
if [ -f /usr/share/bash-completion/bash_completion ]; then
. /usr/share/bash-completion/bash_completion
elif [ -f /etc/bash_completion ]; then
. /etc/bash_completion
fi
fi

export PATH=$PATH:/usr/local/cuda/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/lib64
export CUDA_HOME=/usr/local/cuda

```

그림 32 명령어: `getit ~/.bashrc` 실행 화면

열린 파일의 맨 밑에 그림 32의 빨간색 상자안의 3 줄을 추가한 후 텍스트 편집기를 종료하고 명령어: `source ~/.bashrc` 를 실행한다.

cuDNN 을 설치하기 위하여 아래의 명령어들을 터미널에 차례로 사용한다.

- `wget`

http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64/nvidia-machine-learning-repo-ubuntu1804_1.0.0-1_amd64.deb

- `sudo apt install ./nvidia-machine-learning-repo-ubuntu1804_1.0.0-1_amd64.deb`
- `sudo apt update`
- `sudo apt install libcudnn7 libcudnn7-dev`

```
test@test:/usr/local/cuda/bin$ gedit ~/.bashrc
test@test:/usr/local/cuda/bin$ source ~/.bashrc
test@test:/usr/local/cuda/bin$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Wed Oct 23 19:24:38 PDT 2019
Cuda compilation tools, release 10.2, V10.2.89
test@test:/usr/local/cuda/bin$
```

그림 33 명령어: `nvcc --version` 실행 화면

명령어 `nvcc --version` 을 통해 CUDA가 설치되었는지 확인한다 .

3. Python 필수 라이브러리 설치 및 심층학습 라이브러리 설치

Windows의 Anaconda Python 환경과 달리 Ubuntu 에서는 터미널에 명령어를 사용하여 pip, Python 필수라이브러리들을 설치하여야 한다.

아래의 명령어들을 사용하여 Python 3 pip을 설치 후 업데이트한다.

- `sudo apt install python3-pip`
- `sudo pip3 install -U pip`
- `sudo pip3 install Xlib` (AI 축구에서 필요로 하는 라이브러리 설치)

학습 기반 코드를 사용하기 위하여 NVIDIA 드라이버, CUDA, cuDNN을 설치한 후에 터미널에 명령어를 사용하여 Tensorflow 와 PyTorch를 설치한다.

- `sudo pip3 install tensorflow-gpu==1.13.1`
- `sudo pip3 install torch==1.4.0`

2-5 설치 확인

이 부분은 사용중인 운영체제(Windows 또는 Ubuntu)에 관계없이 Webots 시뮬레이터에서 AI 축구 world 파일을 로드하고 실행하는 방법을 설명한다.

2-3-3 과 2-4-3 절에서 다운로드하여 압축을 푼 AI 축구 시뮬레이터는 그림 34 와 같은 구조를 가지고 있다.

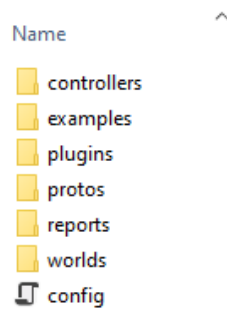


그림 34 AI 축구 시뮬레이터 구조

'controllers' 폴더는 supervisor 의 기능이 들어가 있고, 게이머 코드의 작성과는 무관하다. 'pluggins' 폴더는 게임 진행상화에서 발생하는 로봇간의 충돌의 감지에 관련된 코드들이 있다. 'protos' 폴더는 게임의 로봇 및 경기장등의 상세 내역이 있다. 'reports' 폴더는 게임 기사작성과 관련된 것이다. 게이머에게는 'worlds'와 'examples' 폴더들이 중요하다. 'worlds' 폴더에는 Webots 시뮬레이터에서 열어야하는 청소년 및 어른 로봇의 경기를 위한 환경이 모두 포함되어 있다. 'examples' 폴더에는 자신의 전략을 개발하기 위해 참고하여 사용하는 예제 코드가 포함되어 있다.

'worlds' 폴더에는 다음 그림 35 와 같이 두개의 world 파일이 있다.

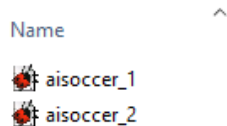


그림 35 AI 축구 시뮬레이터 'Worlds' 폴더

world 파일 'aisoccer_1.wbt'는 청소년 로봇들을 사용하여 경기를 하는 환경이고, world 파일 'aisoccer_2.wbt'는 어른 로봇들을 사용하여 경기를 하는 환경이다. 설치를 확인하려면 Webot 시뮬레이터를 열고 'Ctrl + O'를 눌러 월드 파일을 열 수 있다. 'Worlds' 폴더에서 실행하려는 World 파일을 선택하면 그림 36 과 같이 시뮬레이터가 실행된다.

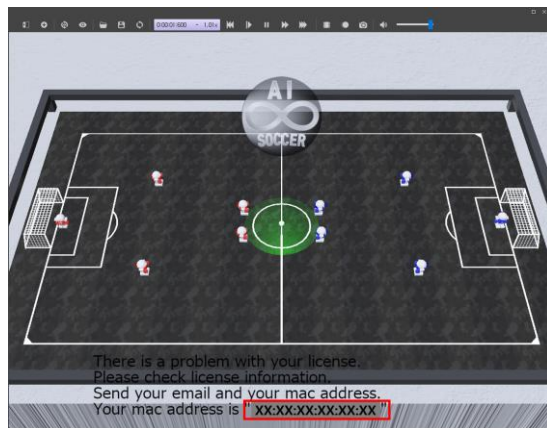


그림 36 Webots 시뮬레이터 실행 창


AI 축구 시뮬레이터를 실행하려면 라이선스가 필요하다. 따라서 Webots 시뮬레이터에서 Play 버튼을 누르면 라이선스와 관련된 메시지가 화면에 출력된다. 라이선스를 받으려면 게이머의 e-mail 주소와 AI Soccer 플랫폼이 설치된 컴퓨터의 MAC 주소(시뮬레이터를 처음 동작시킬 때 화면에 표시됨)를 aisoccer3d@gmail.com으로 보낸다.

AI 축구 시뮬레이터의 root 폴더에 있는 'config.json' 파일을 열고 e-mail 및 라이선스를 입력한다. 'config.json' 파일은 2-6 절에 설명되어 있다. 이후 규칙 기반 두 팀 간의 경기가 진행된다.

2-6

CONFIG.JSON 파일

Webots 시뮬레이터 편집기를 사용하여 'config.json' 파일을 열어 게임 옵션을 구성하고 라이선스(e-mail 주소 및 시그니처 키)를 삽입하고 게이머 strategy codes 를 설정하면 된다. 기본 구성만을 사용하면 로봇을 임의로 이동하는 팀이 A 팀, 간단한 규칙 기반에 따라 로봇을 작동시키는 팀이 B 팀으로 경기가 진행될 것이다. 기본 'config.json' 파일은 그림 37 에 나와있다. 게이머가 작성한 코드는 path 를 정해서 그림 37 과 같이 사용할 수 있다. AI Soccer 게이머와 관련된 부분은 rule, team_a, team_b, tool 이다. rule 의 game_time 은 전후반 각각의 게임시간(second=초)으로 게이머가 원하는대로 바꿀 수 있다.



```
1 {
2   "rule": {
3     "game_time": 300,
4     "deadlock": true
5   },
6   "team_a": {
7     "name": "teamA",
8     "executable": "examples/player_random-walk_py/player_random-walk.py",
9     "datapath": "examples/team_a_data",
10    "keyboard": false
11  },
12  "team_b": {
13    "name": "teamB",
14    "executable": "examples/player_rulebasedB_py/player_rulebasedB.py",
15    "datapath": "examples/team_b_data",
16    "keyboard": false
17  },
18  "commentator": {
19    "name": "commentator",
20    "executable": "examples/commentator_skeleton_py/commentator_skeleton.py",
21    "datapath": "examples/commentator_data"
22  },
23  "reporter": {
24    "name": "reporter",
25    "executable": "examples/reporter_skeleton_py/reporter_skeleton.py",
26    "datapath": "examples/reporter_data"
27  },
28  "tool": {
29    "repeat": false,
30    "multi_view": false,
31    "record": false,
32    "record_path": "",
33    "replay": false
34  },
35  "license": {
36    "email": "user@email.com",
37    "signature": ""
38  }
39 }
40
```

그림 37 Webots 시뮬레이터 config.json 파일

'Config.json'파일의 경로는 AI 축구 시뮬레이터의 상위 폴더에 상대적이다. 상대 경로에 대한 간단한 설명은 다음에 제시되어 있다.

1. 절대 경로 : 파일이 가지고 있는 고유한 경로를 의미함
예를 들어 바탕화면에 있는 config.json 파일을 예로 들어보자
절대 경로 = C:\Users\UserID\Desktop\config.json
 2. 상대 경로 : 현재 위치한 곳(현재 디렉토리)을 기준으로 파일이 있는 위치를 의미함
- 기준 경로를 기준으로 절대 경로가 구성되며, 상대 경로로 파일의 위치를 찾을 수 있는 이유는 기준 경로가 절대 경로로 변환하여 OS 에게 전달하기 때문임
상대 경로 = Desktop\config.json

config.json 파일 구조	
항목	의미
rule	Game_time(경기시간)과 deadlock(교착상태=공이 4 초동안 로봇간의 접촉으로 인한 교착으로 움직이지 않을 경우 공이 재배치됨)으로 구성됨
Game_time	Game time (default: 300 seconds)
deadlock	False 로 설정하면 교착 상태에 대한 규칙이 무시된다 (default: True).
Team_a Team_b	경기할 축구팀을 정함. name, executable, datapath, keyboard 로 구성됨
name	팀 이름
executable	AI 축구 실행 파일 경로 (AI 축구를 실행하려면 두 팀 모두 올바르게 지정되어야 함)
datapath	AI 가 일부 파일을 출력할 수 있는 경로

keyboard	True 로 설정하면 키보드를 통해 로봇 조작용 가능하다.
commentator reporter	AI 축구 해설자와 리포터의 정보를 정한다 (다른 AI WorldCup 종목에서 사용)
tool	추가적인 정보. repeat, record, record_path, replay 로 구성됨
repeat	True 로 설정하면 게임 종료 후 동일한 게임 옵션으로 게임이 반복된다. True 일 때 'reset_reason'의 'GAME_END'는 'EPISODE_END'로 대체된다 (default: False).
multi_view	True 로 두면 경기를 진행하는 동안 3 차원 카메라가 공을 따라 움직이고 몇몇 상황에서 사용되는 카메라가 바뀐다. <i>코드 개발 및 시험 단계에서는 False 로 두기를 권장한다.</i>
record	True 로 설정하면 경기를 녹화하고 게임이 끝났을 때 "record_path"에 저장된다. "repeat"과 함께 사용할 수 없다. "repeat"이 True 이면 "record"내부적으로 False 가 된다(default: False). 경기 종료 후 녹화된 비디오가 저장되는데 몇 분이 걸린다. Video creation finished.'라는 메시지가 Webots Console (Ctrl + L)에 나올 때까지 기다려야 한다.
record_path	- 녹화된 비디오를 저장할 경로 (default: ""). 비디오 파일 이름이 아니라 경로이다. - 파일이름은 자동으로 <code>{{timestamp}}{team_a_name}_{team_b_name}.mp4</code> 로 설정된다. 기본값으로 사용하면 루트 경로에 비디오를 저장한다. 비디오를 저장할 적절한 경로를 지정해야 한다.
replay	True 로 설정하면 골이 들어갔을 때 골이 들어가기 3 초

	전부터 골이 들어가고 난 후 3 초까지의 영상이 재생된다. "multi_view" 옵션이 True 이면 다른 각도의 리플레이 영상이 재생된다.
license	게이머의 정보를 확인. email, signature 로 구성됨
email	게이머의 email 주소 입력
signature	관리자로부터 받은 시그니처 키 입력 (다른 값을 입력한 경우 게임이 진행되지 않는다.)

메모 [오전2]: 표 없애고 그냥 이렇게..
이 뒤에 그 '저장하고 리셋하고 시작' 이말 넣으면 될것같아요

config.json 파일 수정이 끝난 후, AI 축구를 'Ctrl + 2'(Play Button)키를 눌러 실행시킨다.

3 AI 축구 기본 정보

AI 축구 경기장, AI 로봇 사양

AI 축구 경기장 사양

청소년용 로봇을 위한 AI 축구 경기장은 그림 38 과 같다. 어른용 로봇은 1.3 배 큰 경기장에서 경기한다.

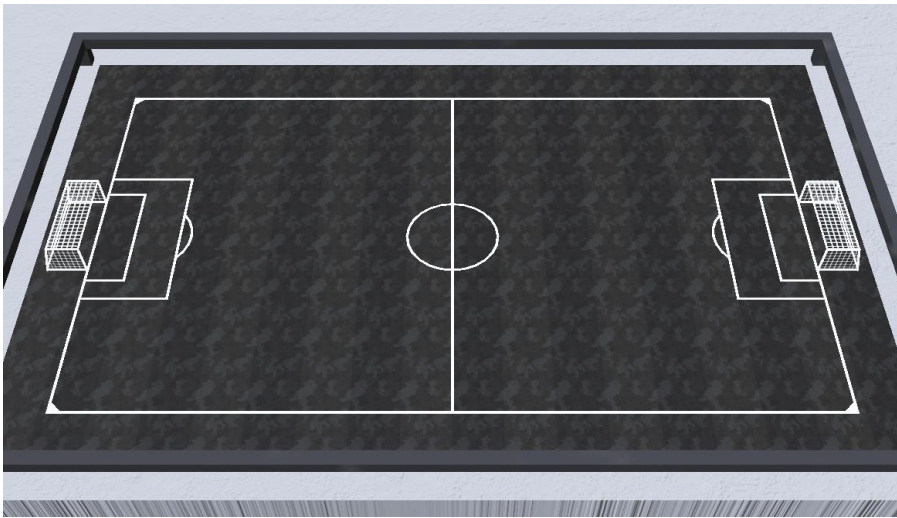


그림 38 AI 축구 경기장

청소년 로봇과 어른 로봇의 경기장 사양은 각각 그림 39 와 40 에 자세히 설명되어 있다.

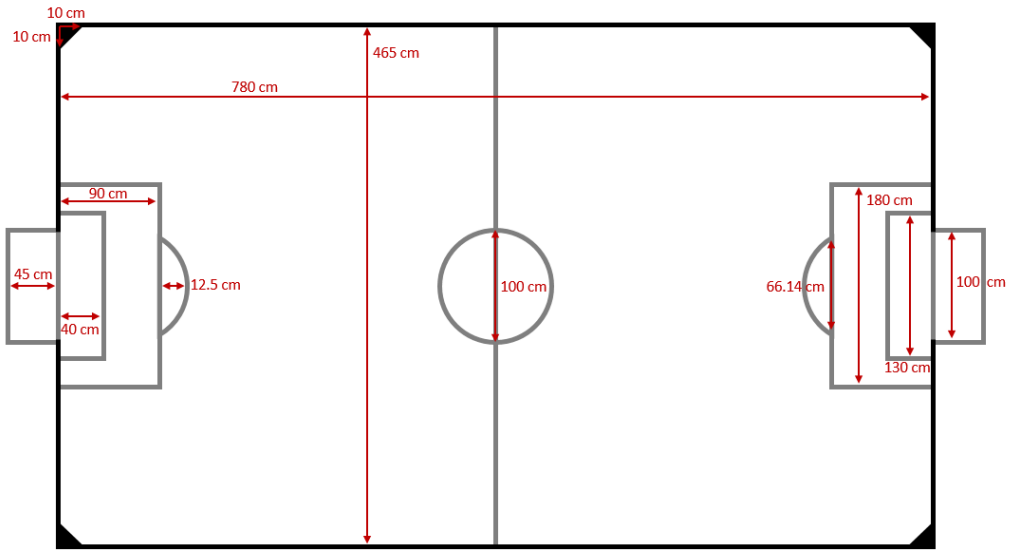


그림 39 축구 로봇 경기장 (청소년용)

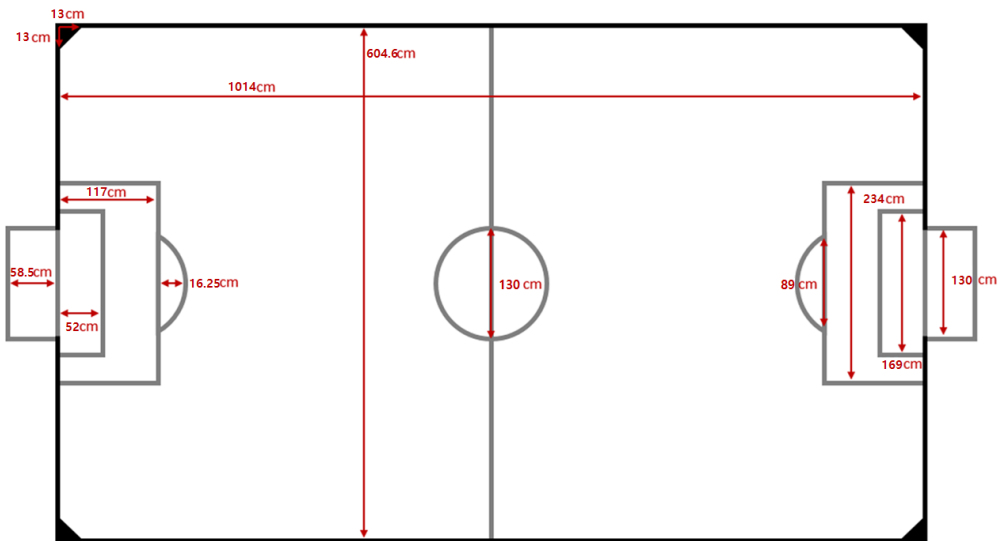


그림 40 축구로봇 경기장 (어른용). 어른 로봇용 AI 축구장의 각 치수는 청소년용의 1.3 배이다. 예를들어 청소년용의 경우 하프라인의 중앙 원은 지름이 100cm 인 반면 어른용의 경우 지름이 130cm 이다



3-1. 축구 로봇 기본 사양

1. 축구 로봇 역할

AI 축구 로봇은 골키퍼(GK), 수비수(D1, D2), 공격수(F1, F2)의 세 종류로 나눌 수 있다. 이러한 기호는 각 로봇을 가리키는 데 사용되며, 이 시점부터 본 설명서에서 일반적으로 사용된다.

	골키퍼 (GK)	수비수 (D1, D2)	공격수 (F1, F2)
A 팀 번호	GK	D1 D2	F1 F2
B 팀 번호	GK	D1 D2	F1 F2

2. 축구 로봇 크기

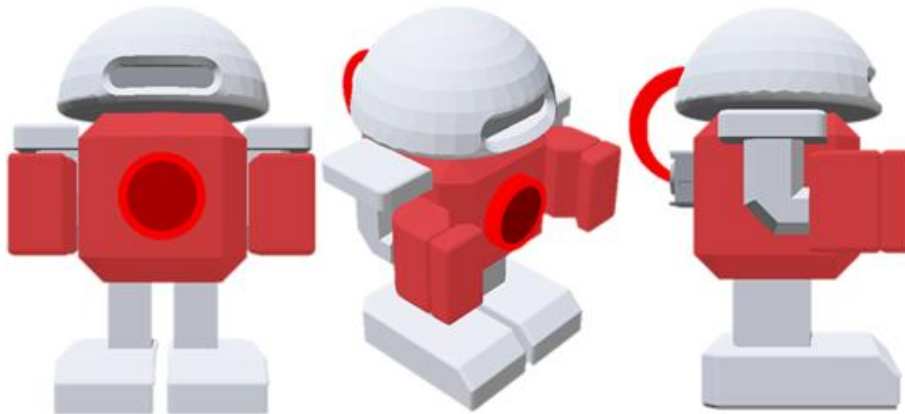


그림 41 청소년용 로봇

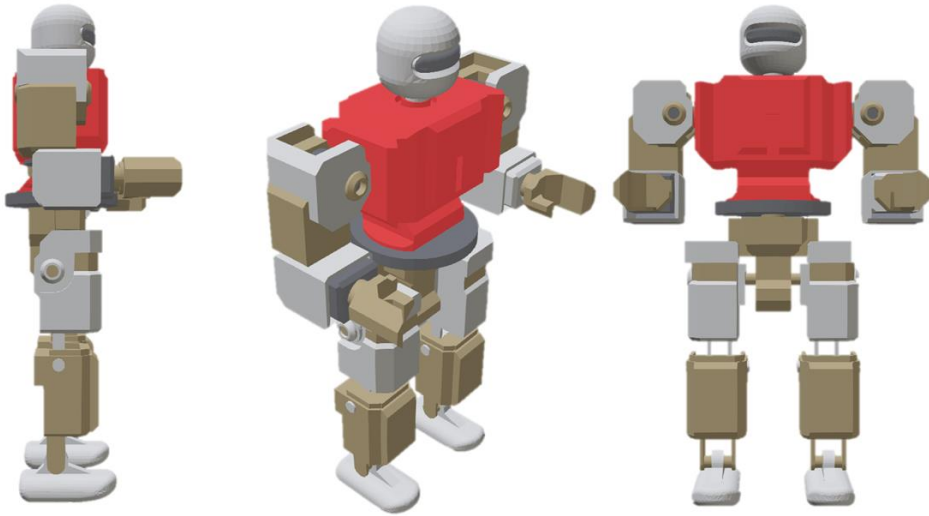


그림 42 어른용 로봇

AI 축구에 사용되는 로봇은 청소년용 로봇과 어른용 로봇 두 종류가 있다. 그림 41 에 제시된 로봇은 청소년용이고 그림 42 에 제시된 로봇은 어른용이다. 이들의 차이점은 시각적 형태와 크기뿐이다.

그림 41 과 42 의 시각적 모양은 로봇이 Webots 시뮬레이터 화면에서 보이는 것과 일치한다. 시각적 형태는 팔, 다리, 머리를 포함한다. Webots 시뮬레이터에 의해 내부적으로 수행되는 물리적인 모양은 시각적 모양을 단순화한 형태인 그림 43 과 44 와 같다. 그림 43 과 44 에 표시된 로봇의 키와 발 크기의 물리적인 크기는 그림 41 과 42 에 제시된 시각적 형상의 키와 발의 크기와 같다. 그외의 부분들은 게임 실행의 편의성을 위해 시뮬레이터 내부적으로 단순화된다. 단순화는 로봇이 어떻게 공, 벽, 그리고 다른 로봇과 접촉하게 되는지와 관련이 있다. 로봇의 신체 사양은 머리와 몸통, 다리를 대신하는 상자 모양의 하체로 구성된다. 팔은 골키퍼의 경우에만 물리적인 모양에 포함된다. Webots 시뮬레이터에서 로봇을 클릭하여 로봇의 물리적인 사양을 확인할 수 있다.

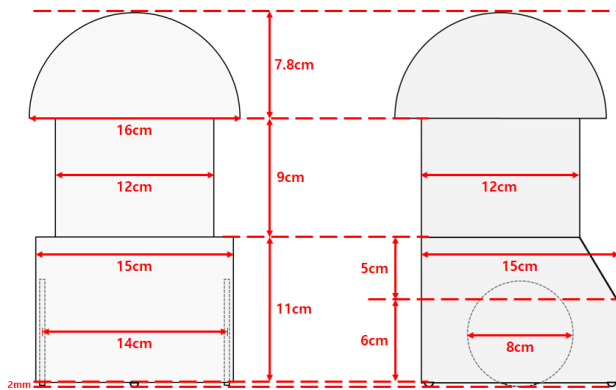


그림 43 청소용 로봇 사양 : 왼쪽 그림은 로봇을 전면에서 본 모양이며 오른쪽 그림은 로봇의 측면에서 본 모양에 해당된다. 로봇의 아래쪽에 있는 원은 로봇의 움직임을 위한 바퀴를 나타낸다.

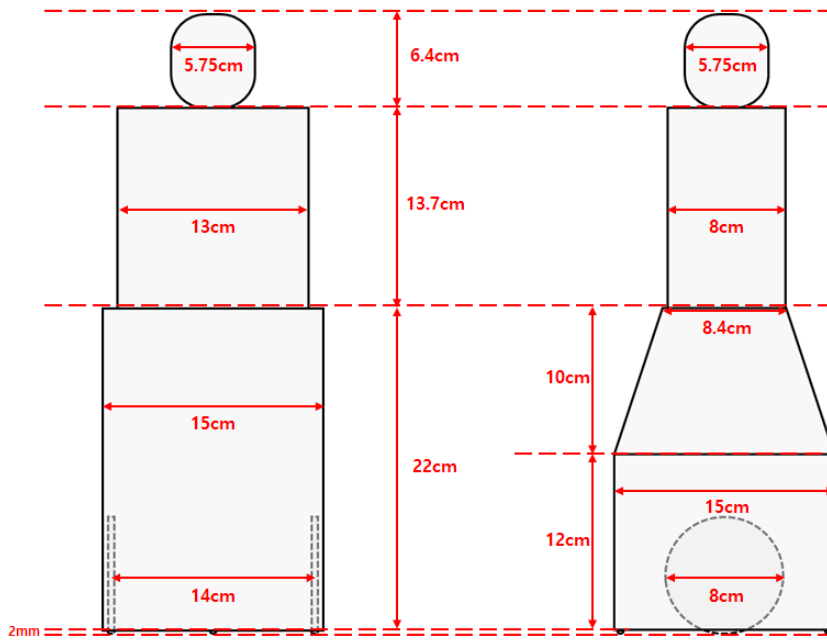


그림 44 어른 축구 로봇 사양 : 왼쪽 그림은 로봇을 전면에서 본 모양이며 오른쪽 그림은 로봇의 측면에서 본 모양에 해당된다. 로봇의 아래쪽에 있는 원은 로봇의 움직임을 위한 바퀴를 나타낸다.

로봇이 기우는 것을 막기 위하여 로봇의 바닥에 두개의 작은 반구 형태의 지지대가 있는데 게이머의 코드작성과 무관한 것으로 간주해도 된다.

3. 축구 로봇 행동

로봇 안에는 그림 45 와 같은 2 개의 바퀴와 추가적으로 2 개의 슬라이더가 있다. 바퀴는 왼쪽과 오른쪽 바퀴가 각각 움직이며(차동) 로봇의 움직임에 사용된다. 차동 바퀴의 사용에 대한 자세한 내용은 이 절의 뒷부분에 설명된다. 2 개의 슬라이더는 로봇의 전면과 하단에 위치한다. 전면 슬라이더는 shoot(cross, kick, quickpass)에 사용된다. 크로스(cross) 또는 높게 공을 찰 때 각도 조정은 전면 (앞쪽) 슬라이더의 높이를 조정하여 수행할 수 있다. 하단 (아래쪽) 슬라이더는 헤딩을 할 경우 점프를 위해 사용된다.



그림 45 축구 로봇 슬라이더 : 발 아래의 빨간색 부분은 하단 슬라이더에 해당한다. 발 앞쪽의 빨간 부분은 전면 슬라이더에 해당한다.

골키퍼의 경우 슬라이더를 다르게 설계해 세이브 (Save) 기회를 만들어 낸다. 슬라이더는 골키퍼 하단에 위치하고 앞쪽, 왼쪽, 오른쪽의 세이브를 시도하기 위해 세가지 방향으로 슬라이딩 할 때 사용된다.

축구 로봇은 바퀴와 슬라이더의 속도를 변화시킴으로써 다양한 행동 특성을 달성할 수 있다. 그 특징의 예는 그림 46 에 나와 있다. 그림 46 에서 로봇은 이동 (move), 점프 (jump), 상대방 골대로 슈트 (shoot)이 가능함을 보여준다.

* 축구 로봇의 행동 특징 3 가지

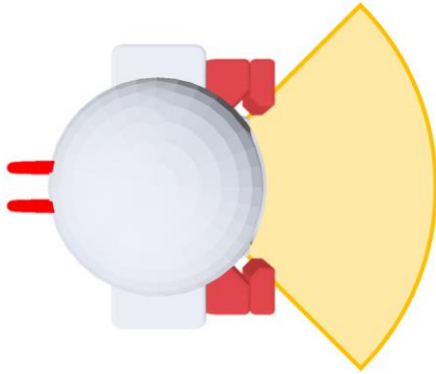
- ✓ 첫 번째는 앞, 뒤, 회전을 포함한 **'move'**
- ✓ 두 번째는 **'jump'**
- ✓ 세 번째는 **'shoot'**



그림 46 로봇의 세가지 행동

로봇의 공 제어 능력을 높이기 위해 드리블 모드도 정의되었다. 드리블 모드는 로봇이 공을 점유하고 있을 때 활성화 할 수 있다. 드리블 모드가 활성화 되고 로봇이 공을 점유할 때 로봇과 함께 공이 움직인다. 드리블 모드가 활성화 되어도 상대팀이 볼 점유상태로 돌입하는 것을 막지 못한다. 드리블 모드의 규칙은 다음과 같다.

- ① 드리블 모드는 로봇 전방의 드리블 영역에 공이 있을 때 활성화 할 수 있다. 드리블 영역은 정면 중앙선을 중심으로 좌우 45 도 도합 90 도의 부채 모양이며, 원으로 보이는 머리의 원주에서 5~20cm 의 길이(로봇의 속도에 따라 다름)로 정의 된다. 로봇이 정지한 상태이면 5cm 이고 최대의 속도(maximum linear velocity)로 움직이면 20cm 이다. 드리블 영역은 다음 그림과 같다.



- ② 다른 로봇의 드리블 지역과 공이 겹칠 경우 드리블 모드가 해제되고 두 로봇 모두 볼 점유를 할수 있기 때문에, 다툼을 벌인다.
- ③ 드리블 중 shoot 또는 jump 를 사용하면 드리블이 종료된다.
- ④ 로봇이 넘어진 경우 드리블이 종료된다.

게이머들의 편의를 위해 규칙 기반 예제 코드의 'action.py' 파일에 kick (shoot)과 jump 동작이 미리 구현되어 있다. 게이머 코드는 각 시간 단계에서 각 로봇에 대해 바퀴, 슬라이더 및 드리블 모드 변수를 설정하고 그것들을 AI 축구 시뮬레이터로 전송해야 한다. 바퀴 및 슬라이더 사양에 대한 자세한 내용은 다음 절인 "AI 축구에서 제공하는 기본 정보"를 참조하면 된다.

4. 축구 로봇 사양

표 1 축구 로봇 사양

역할 \ 사양	골키퍼 (0)	수비수 (1, 2)	공격수 (3, 4)
로봇 무게	2.5 kg	2.0 kg	1.5 kg
로봇 무게중심	지상 1.5 cm		
바퀴 무게	각 0.15 kg		
슬라이더 무게	각 0.5kg		
최대 속도	1.8 m/s	2.1 m/s	2.55 m/s
최대 회전 토크	0.8 N*m	1.2 N*m	0.4 N*m

로봇의 사양은 위의 표와 같으며, 역할에 따라 일부 사양은 다르다. 골키퍼가 다른 로봇보다 무거운 이유는 상대 팀 수비수와 공격수가 골키퍼를 골 지역 밖으로 밀어내는 상황을 피하기 위해서다. 공격수가 수비수보다 가벼운 이유는 공격수가 페널티지역에서 골대 바깥방향으로 수비수들을 밀어 넣어 득점 기회를 만드는 상황을 피하기 위해서다.

축구 공

1. 축구공 크기

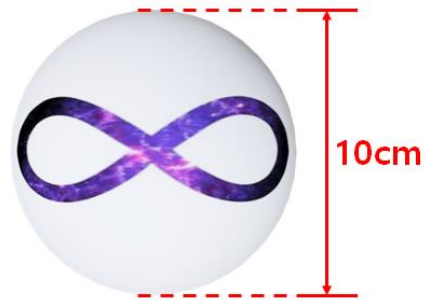


그림 27 축구공 사양

축구 공은 지름 10cm의 단순한 구 형태이고 무게는 18.4g이다. 축구공에 있는 “∞” 마크는 AI 축구에서 개발할수 있는 무한한 종류의 전략을 나타낸다.

3-2 AI 축구의 기본 정보

가상함수, info, frame, 로봇제어

이번 절에서는 게이머의 코드와 AI 축구 시뮬레이터 (이 설명서에서는 Supervisor 이라고도 함) 사이에서 주고 받는 정보에 대한 내용을 설명한다. 그림 48 과 49 에 AI 축구 시뮬레이터와 게이머 코드 간의 통신이 상세하게 설명되어 있다.

AI 축구는 공격과 수비가 반복되면서 이루어진다. 이를 위해 공격과 수비에 해당하는 경기장 영역과 각 영역에 적합한 로봇의 행동들이 정의되어야 한다. 공격의 경우, 각 로봇들이 게임전략에 해당하는 포메이션을 유지하기 위해 정해진 위치를 적절한 속도로 이동해 크로스, 킥, 패스를 적당한 속도와 높이로 실행하는 것이 중요하다. 수비의 경우, 상대팀 공격수의 움직임에 대응한 위치선정과 패스가 중요하다. 이러한 경기진행을 위해 로봇과 경기장의 특징에 대한 정보가 필요하고, 매 시간 단계마다 경신되는 경기 상태 정보가 필요하다. 결국, 게이머의 코드는 이러한 정보를 이용해 로봇의 위치설정, 바퀴를 이용한 이동(move)의 속도 제어, 전면 슬라이더와 하단 슬라이더를 이용한 크로스/킥/패스를 적절한 시간에 효과적으로 실행하기 위한 것이다. 로봇과 경기장에 관한 정보는 게임 시작후 1 회만 수신하는 'info' 딕셔너리 (Dictionary)와 50ms 마다 수신하는 'frame' 딕셔너리로 받고 게임 전략은 update() 함수를 중심으로 구현된다. 참고로 시간 단계와 프레임(frame)은 똑같이 50ms 의 시간 경과후에 바뀐다.

그림 49 에서 AI 축구 시뮬레이터는 게임에 관련된 로봇 및 공의 위치, 게임 상태 (Game State), 스코어와 시간등을 게이머 코드의 호출이 있을 때마다 알려준다.

그림에서 (경기장+로봇+공) image 는 이번에 배급된 version 에서는 로봇과 공의 좌표값들이 AI 축구 시뮬레이터에서 제공되므로 사용되지 않는다. 게이머는 AI 축구 시뮬레이터로부터 게임에 관련된 데이터를 받는 3 가지 가상 함수를 이용하여 게임 상태에 맞게 바퀴속도 (wheel velocity)와 전면 슬라이더등을 이용해서 로봇들을 제어할 수 있다. 세가지 가상 함수는 init(), update(), finish()이다. init()은 게이머가 시뮬레이터와 성공적으로 연결된 후에 호출된다. update()는 매 단위 시간 50ms 마다 호출된다. 경기가 끝날때, finish()가 호출된다.

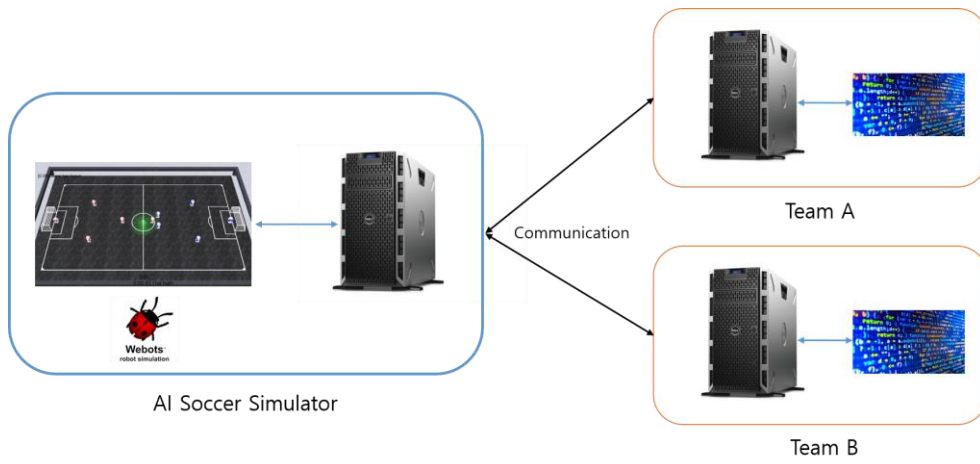


그림 48 AI 축구 시뮬레이터(AI Soccer Simulator)와 게이머 코드 간의 통신(Communication)

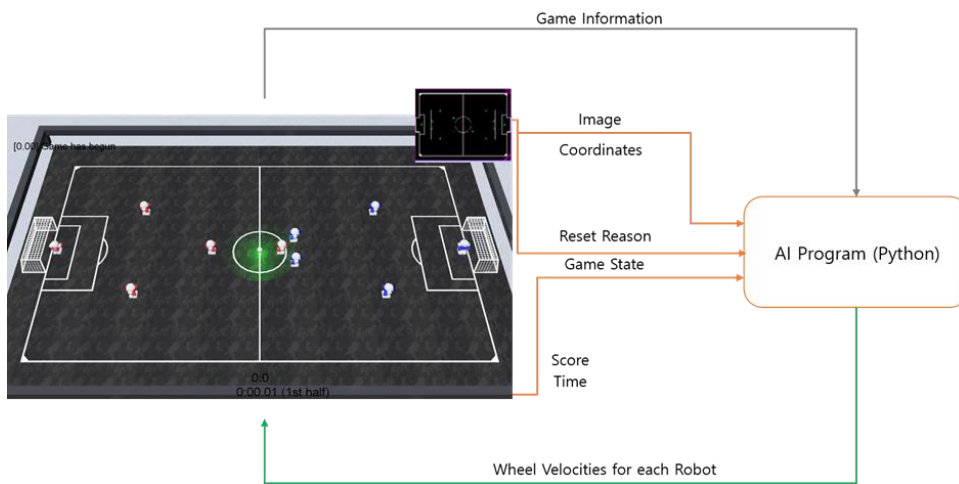


그림 49 AI 축구 시뮬레이터와 게이머 코드가 송수신 하는 정보

✿ 가상함수

AI Soccer 코드에서는 3 개의 가상 함수가 호출된다. 제공된 예제코드를 보면서 읽으면 더욱 쉽게 이해될 것이다.

① 가상 함수 init()

가상 함수 '**init()**'은 게이머 코드가 시뮬레이터와 연결에 성공한 직후 한번만 호출된다. 그림 48 과 같이 supervisor 로부터 경기장 상황과 로봇의 상태에 대한 사전 정보(다음 시간 단계에서 쓰일 정보)를 수신하고 게이머 코드에 의해 게임 전략 구현에 필요한 변수를 초기화 하는데 사용된다. AI 축구 시뮬레이터의 'examples' 폴더에 위치한 예제 코드에서 변수의 초기화 및 데이터 기록이 어떻게 작동하는지에 대한 것을 확인 할 수 있다.

② 가상 함수 'update()'

가상 함수 '**update()**'는 시뮬레이터가 게이머 코드로 새 데이터를 전송할 때마다 호출된다. AI 축구에서는 단위 시간 단계가 50ms 로 설정되어 있어 시뮬레이터도 50ms 마다 새로운 데이터를 전송하고 'update()'도 50ms 마다 호출된다. Supervisor 가 게이머 코드에게 보낸 자료를 frame(프레임)이라고 한다. 게임의 각 프레임에 관한 정보가 수록된 프레임 데이터는 update() 함수의 인자인 'frame'에 들어있다. 'frame'변수에는 그림 49 에 나타낸 것과 같이 게임의 현재 상태에 해당하는 이미지, 좌표, 재설정 이유(reset reason), 게임 상태, 점수 및 시간이 포함되어 있다. 게이머 코드는 수신된 현재 축구 경기 상태의 내용을 담은 프레임 데이터를 확인하고, 생성할 제어 신호(로봇의 동작)를 결정하고, 제어 신호를 다시 시뮬레이터로 전송한다. 따라서 **게이머의 게임 전략이 구현된 함수**이다. 제어 신호 전송 방법은 80 페이지에 설명되어 있다. 제어 신호를

생성하기 위한 규칙 또는 심층학습을 사용하는 예제 코드는 'examples' 폴더에 제공됩니다.

③ 가상 함수 'finish()'

가상 함수 '**finish()**'는 게이머 코드가 종료되기 전에 한번만 호출된다. 여기서 게이머 코드는 게임 내내 기록된 모든 데이터를 저장할 수 있고, 다음 게임에 유용하게 사용 될 수 있다.

'info' Dictionary

'info' 딕셔너리에는 경기장의 크기, 로봇 사양 등 경기 중 변하지 않는 값들과 기본 정보를 담고 있다. 이 딕셔너리는 게이머 코드의 'init()' 함수에서 수신한 'info' 변수를 통해 접근 가능하다(예: 'info['Game_time']'는 경기 시간을 나타낸다). 사용하고 싶은 변수가 있다면 'init()'에서 class 변수로써 저장해야 한다. 이 딕셔너리에 있는 정보들은 AI 축구 기본 사양에 있는 정보와 동일하다.

딕셔너리의 정보는 표 2 와 같이 정리되어 있다. 이러한 정보와 딕셔너리 값의 사용방법은 예제 코드의 'general_check-variables.py'와 'general_image-fetch.py'을 통해서 확인 가능하다.

표 2 'info' Dictionary

'info' Dictionary 의 정보		
Member Variable	Data Type	설명
field	list of floats (length 2)	경기장 크기 [x, y] (단위: m)
goal	list of floats (length 2)	골대 크기 [x, y] (단위: m)
penalty_area	list of floats (length 2)	페널티 구역 크기 [x, y] (단위: m)
goal_area	list of floats (length 2)	골 구역 크기 [x, y] (단위: m) ※ 이 구역과 관련된 규칙은 없음.
ball_radius	float	공 크기 (단위: m)
ball_mass	float	공 무게 (단위: kg)
robot_size	list of floats (length 5)	로봇 크기 [GK, D1, D2, F1, F2] (단위: m) ※ 로봇 하체 기준 (0.15m)
robot_height	list of floats (length 5)	로봇 높이 [GK, D1, D2, F1, F2] (단위: m)

axle_length	list of floats (length 5)	바퀴 사이 거리 [GK, D1, D2, F1, F2] (단위: m)
robot_body_mass	list of floats (length 5)	로봇 무게 [GK, D1, D2, F1, F2] (단위: kg)
wheel_raidus	list of floats (length 5)	바퀴 반지름 [GK, D1, D2, F1, F2] (단위: m)
wheel_mass	list of floats (length 5)	바퀴 무게 [GK, D1, D2, F1, F2] (단위: kg)
max_linear_velocity	list of floats (length 5)	바퀴 최대 선속 [GK, D1, D2, F1, F2] (단위: m/s)
max_torque	list of floats (length 5)	바퀴 최대 토크 [GK, D1, D2, F1, F2] (단위: N*m)
resolution	list of ints (length 2)	이미지 크기 [가로, 세로] (단위: pixel)
number_of_robots	Int	로봇 개수
codewords	list of ints (length 5)	로봇 식별 이미지에 부여된 10 진수의 해밍 코드 [GK, D1, D2, F1, F2] ※ 3-2. 이미지 정보 참고
game_time	Float	경기 시간 (단위: s)
team_info	Dictionary	두 팀의 정보 포함
key	String	랜덤 문자열 key
Keyboard	Bool	키보드 조작 사용 여부

'frame' Dictionary

'frame' 딕셔너리는 게임의 각 프레임의 정보를 가지고 있으며, 현재 게임 상태와 연관되어 있다. 게임이 진행됨에 따라 로봇과 공의 좌표 등 'frame' 딕셔너리에서 수신되는 정보가 매 시간 단계마다 다르다. 'update()'가 호출될 때마다 업데이트된 데이터와 함께 새로운 'frame' 변수가 수신된다.

예 : `FRAME['coordinates'][MY_TEAM][GK][X]`는 우리 팀 골키퍼의 x 좌표이다. frame 에서 x 좌표로 순차적인 top-down 방식의 표현으로 좌표값을 얻는다. MY_TEAM, GK, X 는 미리 정의되어 있어야 한다.

표 3 'frame' Dictionary

'frame' Dictionary 의 정보		
변수	데이터 종류	설명
Time	float	현재 경기 시간 (단위: 초)
Score	list of ints (length 2)	현재 점수 [자신 팀, 상대 팀]
reset_reason	int	현재 프레임 직전 경기가 일시 정지된 이유. 이 값은 다음의 값들 중 하나이다: 0. NONE – 일시 정지되지 않음. 1. GAME_START – 경기 시작 직후, 킥오프로 경기 진행. 2. SCORE_MYTEAM – 자신 팀 득점, 킥오프로 경기 진행. 3. SCORE_OPPONENT – 상대 팀 득점, 킥오프로 경기 진행. 4. GAME_END – 경기 종료.

		<p>5. DEADLOCK – 공 재배치.</p> <p>6. GOALKICK – 골 킥으로 경기 진행.</p> <p>7. CORNERKICK – 코너 킥으로 경기 진행</p> <p>8. PENALTYKICK – 페널티 킥으로 경기 진행</p> <p>9. HALFTIME – 후반전 시작 직후, 킥오프로 경기 진행</p> <p>10. EPISODE_END – 경기 종료 직후('repeat' 옵션이 켜져 있으면 GAME_END 로 대체).</p>
game_state	int	<p>현재 경기 상태</p> <p>이 값은 다음의 값들 중 하나이다:</p> <p>0. STATE_DEFAULT – 기본</p> <p>1. STATE_KICKOFF – 킥오프</p> <p>2. STATE_GOALKICK – 골 킥</p> <p>3. STATE_CORNERKICK – 코너 킥</p> <p>4. STATE_PENALTYKICK – 페널티 킥</p>
ball_ownership	Bool	<p>킥오프, 코너 킥, 페널티 킥, 골 킥과 같은 상태에서 자신의 팀이 공을 소유하는지(True) 아닌지(False)를 나타내는 지표. 이 값은 기본 상태에서는 아무 의미가 없음.</p>
half_passed	Bool	<p>현재 경기의 전반전인지(False) 후반전인지(True)를 나타내는 지표</p>
subimages	list of items	<p>새 프레임을 얻으려면 이미지 파편을 이전 이미지 프레임과 병합해야 한다.</p> <p>로봇과 공의 좌표가 주어지므로 중요하지 않음</p>
coordinates	list of [my team coordinates list, opponent team coordinates list,	<p>nested list. 현재 로봇들과 공의 위치</p> <p>※ 세부 내용은 아래 표에 있음</p>

	ball coordinate list]	
EOF	Bool	프레임의 끝을 의미하는 마커

표 3-1 좌표 정보

'my team coordinate list'/'opponent team coordinate list'의 정보		
번호	데이터 종류	설명
0	robot coordinate list	GK 로봇의 좌표
1	robot coordinate list	D1 로봇의 좌표
2	robot coordinate list	D2 로봇의 좌표
3	robot coordinate list	F1 로봇의 좌표
4	robot coordinate list	F2 로봇의 좌표
'robot coordinate list'의 정보		
번호	데이터 종류	설명
0	float	로봇의 x 좌표 (단위: m)
1	float	로봇의 y 좌표 (단위: m)
2	float	로봇의 z 좌표 (단위: m)
3	float	로봇의 방향 (단위: rad) ※ $[-\pi, \pi]$ 범위, 실제 값을 확인하여 원하는 범위로 변환할 수 있음.
4	Bool	로봇이 현재 움직일 수 있는지 여부를 나타내는 지표. 일부 로봇들은 킥오프, 코너 킥, 페널티 킥, 골 킥과 같은 특별한 상태에서 움직일 수 없음. 또한 퇴장되었을 때 움직일 수 없음.
5	Bool	바로 전 프레임에서 로봇이 공과 접촉했는지 여부를 나타내는 지표. 좌표나 이미지를 통해 공과 로봇이

		접촉하였는지 파악하기 어렵기 때문에 이 값을 제공함.
6	Bool	바로 전 프레임에서 공이 로봇 앞 특정 범위에 들어왔는지 여부를 나타내는 지표. 이 지표를 공을 차는 기준으로 사용할 수 있음
'ball coordinate list'의 정보		
번호	데이터 종류	설명
0	double	공의 x 좌표 (단위: m)
1	double	공의 y 좌표 (단위: m)
2	double	공의 z 좌표 (단위: m)

이미지 정보

로봇의 공 좌표, 방향 외에도 시뮬레이터에서 이미지를 제공할 수 있다. 제공된 이미지는 640 x 480 의 수정된 이미지인데 경기영상과 별개로 제공되어 이미지 기반의 로봇과 공의 인식이 가능하도록 하였다.



그림 50 게이머에게 제공되는 이미지 정보(상단의 좌우 이미지)

그림 50 과 같이, 게이머 코드에 전송된 이미지는 게임 프레임과 함께 표시된다. 검은색 배경은 경기장 바닥을, 주황색은 축구공을, 로봇은 특수 마커로 된 이미지가 제공된다. 왼쪽 상단 모서리의 이미지는 A 팀으로 전송되고 오른쪽 상단 모서리의 이미지는 B 팀으로 전송된다. 즉, 이미지를 사용하여 전략코드를 구현할 수도 있다. 이미지를 사용하여 전략 코드를 만들 수 있음에도 불구하고, 예제 코드는 게이머의 이해를 단순화 하기 위해 로봇과 볼의 좌표와 방향에 초점을 맞춰 진행된다. 이미지를 가져오는 방법에 대한 예제 코드는 'general_image-fetch.py' 에 나와 있다. **AI Soccer 에 필요한 로봇과 공의 위치정보는 좌표값으로 제공되므로 이미지 데이터를 활용하여 알고리즘을 설계할 것이 아니라면, 게이머는 이미지 데이터를 사용할 필요가 없다.**

로봇제어

로봇은 아래 몸통 양 옆에 부착된 두 개의 바퀴로 움직일 수 있다. 그림 43 과 44 를 참조하라. 또한 몸통 아래쪽과 앞쪽에 부착된 슬라이더를 이용하여 점프하거나 공을 찰 수 있다. 마지막으로 드리블 모드는 로봇이 드리블 기회를 가질 경우 활성화 될 수 있다. 게이머 코드에서 바퀴, 슬라이더, 드리블 모드 제어는 각 시간 단계에서 'set_speeds()'를 호출하여 이루어진다. 'set_speeds()' 기능은 원하는 바퀴, 슬라이더, 드리블 속도 값을 시뮬레이터로 전송하기 위해 호출된다. 각 로봇은 6 개의 변수를 갖는다. 첫번째 변수는 로봇의 좌측 바퀴 속도와 관련이 있다. 두번째 변수는 로봇의 우측 바퀴 속도와 관련이 있다. 세번째와 네번째 변수는 각각 전면 슬라이더의 속도 및 높이와 관련이 있다. 다섯번째 변수는 하단 슬라이더 속도와 연관되어 있다. 여섯번째 변수는 드리블 모드와 관련이 있다.

표 4 로봇 제어 변수들. 각 열은 각각의 로봇에 필요한 6 가지 제어 변수를 보여준다.

	[0]	[1]	[2]	[3]	[4]	[5]
데이터 종류	float	float	float	float	float	float
설명	GK, 왼쪽 바퀴 속도	GK, 오른쪽 바퀴 속도	GK, 전면 슬라이더 속도	GK, 전면 슬라이더 높이	GK, 하단 슬라이더 속도	GK, 드리블 모드 전환
	[6]	[7]	[8]	[9]	[10]	[11]
데이터 종류	float	float	float	float	float	float
설명	D1, 왼쪽 바퀴 속도	D1, 오른쪽 바퀴 속도	D1, 전면 슬라이더 속도	D1, 전면 슬라이더 높이	D1, 하단 슬라이더 속도	D1, 드리블 모드 전환
	[12]	[13]	[14]	[15]	[16]	[17]
데이터 종류	float	float	float	float	float	float

설명	D2, 왼쪽 바퀴 속도	D2, 오른쪽 바퀴 속도	D2, 전면 슬라이더 속도	D2, 전면 슬라이더 높이	D2, 하단 슬라이더 속도	D2, 드리블 모드 전환
	[18]	[9]	[20]	[21]	[22]	[23]
데이터 종류	float	float	float	float	float	float
설명	F1, 왼쪽 바퀴 속도	F1, 오른쪽 바퀴 속도	F1, 전면 슬라이더 속도	F1, 전면 슬라이더 높이	F1, 하단 슬라이더 속도	F1, 드리블 모드 전환
	[24]	[25]	[26]	[27]	[28]	[29]
데이터 종류	float	float	float	float	float	float
설명	F2, 왼쪽 바퀴 속도	F2, 오른쪽 바퀴 속도	F2, 전면 슬라이더 속도	F2, 전면 슬라이더 높이	F2, 하단 슬라이더 속도	F2, 드리블 모드 전환

(*) GK: 골키퍼, D1: 수비수 1, D2: 수비수 2, F1: 공격수 1, F2: 공격수 2

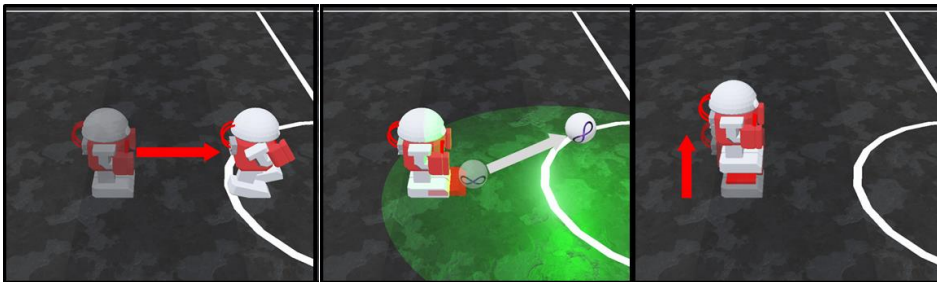


그림 31 로봇의 가능한 행동들

표 4 를 예를 들어 설명하면,

[24]과 [25]의 값을 바꾸면, F2 가 움직이게 만들 수 있다. 이때, [24]와 [25]의 값은 각각 왼쪽 바퀴와 오른쪽 바퀴의 값을 변경한다. 두 값이 양수이고 크기가 같다면 로봇은 일정한 속도로 전진한다. 두 값이 모두 음수이고 크기가 같다면 로봇은 일정한 속도로 후진한다. 같은 절댓값의 숫자를 한쪽에는 음수값을 사용하고 다른

쪽에는 양수값을 사용한다면 로봇의 제자리 회전을 구현하여 각도 조절을 할 수 있다.

[26]과 [27]의 값을 바꾸면, 슛의세기와 높이를 각각 조절할 수 있다. 앞에서 슛은 크로스, 킥, 패스로 구성된다고 설명하였다. 두번째 그림에서 로봇 발쪽에 있는 빨간색이 전면 슬라이더가 위치하는 부분이다.

[28]의 값을 바꾸면 F2 가 하단 슬라이더를 움직여 점프를 한다.

속도값들이 시뮬레이터에 보내지면, 시뮬레이터는 다음 중 하나가 발생할 때까지 동일한 속도들을 유지한다.

- ① 게이머 코드가 새로운 속도들을 전송하였을 때. 이 경우 속도는 그에 따라 경신된다.
- ② 경기가 킥오프, 코너킥, 페널티 킥, 골 킥과 같은 특수 상태로 들어간 경우, 모든 로봇의 속도는 0으로 설정된다. 그다음 상태가 시작되면 그 상태에서 움직일 수 있는 로봇의 속도를 다시 업데이트할 수 있다. 그 상태가 끝나면 그 상태에서 움직일 수 없었던 로봇의 속도를 다시 경신할 수 있다.
- ③ 로봇이 퇴장한 경우. 로봇의 속도는 0으로 설정된다. 로봇이 경기장으로 복귀하면 속도의 경신이 가능하다.

게이머 코드에서 슬라이더의 속도에 0~10의 값을 줄 수 있는데 슬라이더들이 계속 나와있는 상황을 방지하기 위하여 시뮬레이터 프로그램에서 슬라이더의 속도가 한번 0 이상이 되면 일정 시간 후 0으로 되돌리고 1초 동안은 계속 0을 유지한다.

전면 슬라이더의 높이도 0~10의 값을 줄 수 있는데 높이 조절의 효과가 현재 프레임(50ms의 시간 단계)에서 바로 나타나지 않아서 높이 조절은 미리 해 두는 것을 권장한다. 상대 쪽 페널티 지역에서는 전면 슬라이더가 비활성화된다. (골키퍼만 페널티 구역에서 높이 찰 수 있다.)

슬라이더는 경기 장면에 보이지 않지만 시뮬레이터는 슬라이더 위치를 인식하여 슬라이더가 공과 접촉할 경우 물리적 법칙에 의해 공이 움직이게 된다.

AI 축구 로봇 제어에 대한 자세한 내용은 이 설명서의 예제 코드와 부록에 설명되어 있다.

4 AI 축구 예제

이 절에서는 AI 축구 예제에 대한 설명을 한다. 그림 52 는 AI 축구 시뮬레이터에 제공된 예제 코드를 보여준다. 게임에서 정보를 얻는 방법, 로봇 바퀴의 속도 조절 등 시뮬레이터의 기본적인 사용법을 습득할 수 있는 예제 코드가 있다. 또한, 규칙 기반(rule based)과 심층 학습(Deep Learning) 기반의 코드가 제공된다.

규칙 기반 예제의 주된 설명은 'player_rulebasedA_py' 폴더 안에 있는 파일의 설명으로 이루어진다. 3 절에서 언급한 함수와 딕셔너리 정보들이 예제 프로그램에서 어떻게 활용되는지 확인할 수 있다. 심층 학습의 기초를 배우기 위해 특정한 일을 수행하기 위한 단일 로봇 학습(single agent learning), 또한 팀으로 협동시키려는 복수 로봇 학습(multi-agents learning) 등의 예제가 주어진다.

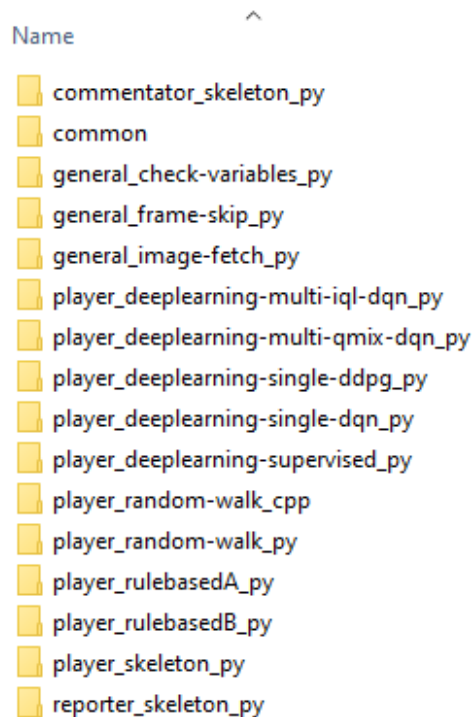


그림 52 AI 축구 시뮬레이터에 있는 examples 폴더

AI Soccer 예제 코드를 실행하기 위해 'config.json' 파일을 수정해야 한다. 'config.json' 파일을 Webots 시뮬레이터에서 열어서 라이선스 번호를 입력하고 예제 코드에 해당하는 팀을 선택한다. 예를 들어 예제 코드가 'examples'폴더내의 'player_random-walk.py'폴더에 player_random-walk.py 으로 저장되어 있으면 그림 53 에 보인바와 같이 teamA 로 설정하고 "executable"을 "examples/player_random-walk.py/player_random-walk.py"와 같이 하면 된다. 'config.json' 파일을 저장하고 게임을 시작하면 된다. 이 때 상대팀(teamB)는 원하는 팀을 같은 방식으로 path 를 잡아서 설정하면 된다.

```
config.json x
1 {
2   "rule": {
3     "game_time": 300,
4     "deadlock": true
5   },
6   "team_a": {
7     "name": "teamA",
8     "executable": "examples/player_random-walk_py/player_random-walk.py",
9     "datapath": "examples/team_a_data",
10    "keyboard": false
11  },
12  "team_b": {
13    "name": "teamB",
14    "executable": "examples/player_rulebasedB_py/player_rulebasedB.py",
15    "datapath": "examples/team_b_data",
16    "keyboard": false
17  },
18  "commentator": {
19    "name": "commentator",
20    "executable": "examples/commentator_skeleton_py/commentator_skeleton.py",
21    "datapath": "examples/commentator_data"
22  },
23  "reporter": {
24    "name": "reporter",
25    "executable": "examples/reporter_skeleton_py/reporter_skeleton.py",
26    "datapath": "examples/reporter_data"
27  },
28  "tool": {
29    "repeat": false,
30    "multi_view": false,
31    "record": false,
32    "record_path": "",
33    "replay": false
34  },
35  "license": {
36    "email": "user@email.com",
37    "signature": ""
38  }
39 }
40
```

1) Put the path of your algorithm here
2) save it
3) reset and run the simulation

그림 53 Webots 시뮬레이터로 보는 config.json 파일

4-1 규칙 기반 예제 코드

AI 축구 예제 코드

'examples' 폴더 안의 'player_rulebasedA_py'와 'player_rulebasedB_py' 폴더들에는 규칙 기반 예제가 있다. 조금 더 간단한 'player_rulebasedA_py'을 예로 들어 이 장의 내용을 소개하고자 한다. 규칙 기반 알고리즘의 예를 들어보면, 현재 볼의 위치나 플레이어의 위치에 따른 현재 상태를 기반으로 알고리즘을 설정하게 된다. 골키퍼의 역할을 예로 들어 생각해 볼 수 있다. 공이 우리 골대와 멀리 떨어져 있을 때는 상대 팀의 중거리 슈트를 막을 수 있도록 위치만 잡으면 공을 막을 수 있을 것이다. 반대로, 공이 내 진영의 페널티 지역 근처에 있다면, 적극적으로 공 근처에 접근 해야 상대팀의 득점을 막거나 공을 걷어내 위험에서 벗어날 수 있을 것이다. 로봇에 대해 미리 설정된 규칙대로 로봇은 움직이게 될 것이다.

'player_rulebasedA_py' 폴더는 4 개의 파일로 구성된다. 시뮬레이터에서 정보를 받아 다시 보낼 수 있는 파일인 'player_rulebasedA.py', 게임 상태를 바탕으로 팀 전략을 구현한 players.py, 로봇 행동 및 바퀴 및 슬라이더 관련 계산이 가능한 action.py, 사용되는 기능의 구현에 필요한 함수들이 정의된 helper.py 로 구성된다.

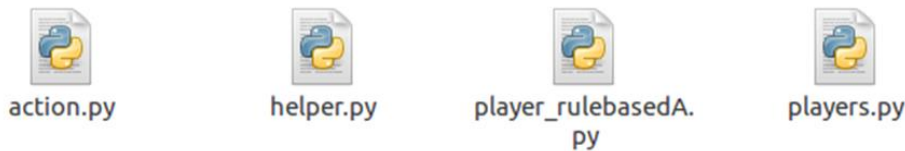


그림 54 'player_rulebasedA_py' 폴더 안에 있는 AI 축구 시뮬레이터

예제 코드들은 크게 상수의 선언부분, init()를 활용한 변수들의 초기화, 로봇들과 공의 좌표를 얻는 get_coord(), 이 좌표값들을 이용해 알고리즘을 구현한 update()로 구성된다. 매 50ms 마다 시뮬레이터에서 보내주는 정보를 update() 부분에서 게임 전략이 구현된 알고리즘을 활용하여 경기를 진행한다.

모든 파일에서 직관적인 이해를 위해 상수(변하지 않는 값)가 전역 변수(함수 밖에서 선언된 변수)로 선언되어 있다. 상수의 선언은 그림 55 에서 확인할 수 있다.

```
#reset_reason
NONE = Game.NONE
GAME_START = Game.GAME_START
SCORE_MYTEAM = Game.SCORE_MYTEAM
SCORE_OPPONENT = Game.SCORE_OPPONENT
GAME_END = Game.GAME_END
DEADLOCK = Game.DEADLOCK
GOALKICK = Game.GOALKICK
CORNERKICK = Game.CORNERKICK
PENALTYKICK = Game.PENALTYKICK
HALFTIME = Game.HALFTIME
EPISODE_END = Game.EPISODE_END

#game_state
STATE_DEFAULT = Game.STATE_DEFAULT
STATE_KICKOFF = Game.STATE_KICKOFF
STATE_GOALKICK = Game.GOALKICK
STATE_CORNERKICK = Game.CORNERKICK
STATE_PENALTYKICK = Game.STATE_PENALTYKICK

#coordinates
MY_TEAM = Frame.MY_TEAM
OP_TEAM = Frame.OP_TEAM
BALL = Frame.BALL
X = Frame.X
Y = Frame.Y
Z = Frame.Z
TH = Frame.TH
ACTIVE = Frame.ACTIVE
TOUCH = Frame.TOUCH
BALL_POSSESSION = Frame.BALL_POSSESSION
```

그림 55 규칙기반 예제 코드에 쓰이는 상수선언

그림 55 에 포함된 상수들은 부록에 제시된 게임 규칙과 관련이 있다. 표 3 와 표 3-1 에 있는 frame 딕셔너리에 관련된 상수들이다.

왼쪽 그림의 상단에 있는 'reset_reason' 상수는 경기중 리셋(reset)이 발생했을 경우 이유를 알려주고, 게임의 로봇들과 공이 특정 위치에 재배치되는 경우에 사용된다. 경기 시작 시, 득점 후, 코너킥, 하프타임, 파울의 발생등의 경우 로봇들과 공의 위치가 재설정된다. 예를 들어, SCORE_MYTEAM 은 내 팀이 득점했을 경우 재설정되는데 이때 Game.SCORE_MYTEAM 이라는 Supervisor 에서 보내는 값으로 상수를 설정한다. 각 줄의 오른쪽 상수들은 이미 Supervisor 에 의해 설정되었기 때문에 게이머에 의한 이름의 변경이 허용되지 않는다.

오른쪽 그림의 상단의 'game_state'는 특정 상태가 수행되고 있는지 감지하는 데 사용된다. 코너킥, 페널티킥 등 게임의 특정 상태에서만 사용되는 전술을 선택하는 경우에 이 상수를 이용하면 된다. STATE_DEFAULT 는 킥오프, 골킥, 코너킥 및 페널티킥이 발생하지 않은 경우의 게임상태를 나타낸다. 각 줄의 오른쪽 상수들은 이미 설정되었기 때문에 이름의 변경이 허용되지 않는다.

오른쪽 그림의 중간의 'coordinates'는 로봇들과 공의 위치 좌표를 x, y, z 3 차원 좌표로 나타낼 수 있다. 예를 들어 FRAME['coordinates'][MY_TEAM][GK][X]는 내 팀 골키퍼의 x 좌표이다. frame 에서 X 좌표로 순차적인 top-down 방식의 표현으로 좌표값을 얻는다. MY_TEAM, GK, X 는 미리 정의되어 있어야 한다. 여기서 표현하지 않은 상수들은 제공된 예제 코드들을 참조하여 이해를 돕기를 바란다.

가장 중요한 상수는 게이머 코드가 수신한 정보와 관련이 있다. 로봇들이 게임 상태를 포함하는 프레임에 접근하고 자신과 상대 팀의 좌표와 방향, 볼 위치, 그리고 공을 건드렸거나 현재 점유하고 있는 상태(예를 들어, 이때 킥을 시도할 수 있다)인 경우와 관련된 정보를 얻을 수 있도록 도와준다.

(1) player_rulebasedA.py 활용방법

```
class Frame(object): ...  
  
class Player(Participant):  
    def init(self, info): ...  
  
    def get_coord(self, received_frame): ...  
  
    def update(self, received_frame): ...  
  
if __name__ == '__main__':  
    player = Player()  
    player.run()
```

그림 56 player_rulebasedA.py 파일

player_rulebased.py 파일에서 게이머들이 변경해야 하는 값들은 없다. 하지만, 전체적인 프로그램의 골격을 담고 있는 파일이기 때문에, 정확히 이해하고 가는 것이 필요하다. player_rulebasedA.py 는 게이머 코드의 기본이 되는 함수들이 정의되어 있는 파일이다. player_rulebasedA.py 는 크게 '**init()**'과 '**update()**' 두 부분으로 나눌 수 있고 'get_coord()'는 이름이 의미하듯이 로봇들과 공의 3차원 좌표를 얻는 함수이다.

player_rulebasedA.py 는 Supervisor로부터 정보를 수신하여 게이머 코드의 기본 변수들과 함수들을 정의한 후 Supervisor에게 제어 신호를 보낸다.

```

def init(self, info):
    self.field = info['field']
    self.max_linear_velocity = info['max_linear_velocity']
    self.robot_size = info['robot_size'][0]
    self.goal = info['goal']
    self.penalty_area = info['penalty_area']
    self.goal_area = info['goal_area']
    self.number_of_robots = info['number_of_robots']
    self.end_of_frame = False
    self._frame = 0
    self.speeds = [0 for _ in range(25)]
    self.cur_posture = []
    self.cur_posture_opp = []
    self.cur_ball = []
    self.previous_ball = []
    self.predicted_ball = []
    self.idx = 0
    self.idx_opp = 0
    self.previous_frame = Frame()
    self.defense_angle = 0
    self.attack_angle = 0

```

```

self.gk_index = 0
self.d1_index = 1
self.d2_index = 2
self.f1_index = 3
self.f2_index = 4
self.GK = Goalkeeper(self.field, self.goal, self.penalty_area,
                      self.goal_area, self.robot_size,
                      self.max_linear_velocity[self.gk_index])
self.D1 = Defender_1(self.field, self.goal, self.penalty_area,
                     self.goal_area, self.robot_size,
                     self.max_linear_velocity[self.d1_index])
self.D2 = Defender_2(self.field, self.goal, self.penalty_area,
                     self.goal_area, self.robot_size,
                     self.max_linear_velocity[self.d2_index])
self.F1 = Forward_1(self.field, self.goal, self.penalty_area,
                    self.goal_area, self.robot_size,
                    self.max_linear_velocity[self.f1_index])
self.F2 = Forward_2(self.field, self.goal, self.penalty_area,
                    self.goal_area, self.robot_size,
                    self.max_linear_velocity[self.f2_index])
helper.printConsole("Initializing variables...")

```

그림 47 init() 함수 안에 있는 변수 초기화

먼저 'init()'에서는 변수들(예: cur_posture, 내팀 로봇들의 좌표와 방향과 공 접촉여부와 현재 시간단계에서 움직임 가능여부를 보여주는 list, 표 4 참조)을 선언 및 초기화할 수 있고, 다른 파일에서 필요한 데이터를 불러올 수 있다. 예를들어 게이머가 학습된 인공지능망을 자신의 전략에 탑재하고 싶은 경우, init()함수에 추가할 수 있다. 앞에서 설명한 대로 player_rulebasedA.py 는 'info'딕셔너리를 통해 게임에 대한 기본적인 정보를 수신한 다음, 지역 또는 전역변수로 저장한다. 또한 게임에 필요한 다른 변수들도 초기화한다. AI 축구의 5 개의 로봇은 'players.py' 파일에 정의된 개체(class)로 선언된다. 로봇 개체는 'self.GK', 'self.D1', 'self.D2', 'self.F1', and 'self.F2' 이고, init()안에 구현되어 있다. 게이머의 필요에 따라 추가적인 변수를 선언 할 수 있다. player_rulebasedA 의 init()함수에 선언된 전역변수들은 아래의 표 4 와 같다.

표 5 init() 함수 안에 선언된 전역 변수들

변수	데이터 종류	설명
self.field	list of floats (length 2)	축구 경기장 크기 [x, y] (단위: m)
self.max_linear_velocity	list of floats (length 5)	바퀴 최대 선속 [GK, D1, D2, F1, F2] (단위: m/s)
self.robot_size	float	로봇 크기 (단위: m) ※ 5 개 로봇 크기 동일
self.goal	list of floats (length 2)	골대 크기 [x, y] (단위: m)
self.penalty_area	list of floats (length 2)	페널티 구역 크기 [x, y] (단위: m)
self.goal_area	list of floats (length 2)	골 구역 크기 [x, y] (단위: m) ※ 이 구역과 관련된 규칙은 없음.
self.number_of_robots	int	로봇 개수
self.end_of_frame	Bool	프레임의 끝을 의미하는 마커
self._frame	int	현재 프레임 번호 ※ 50ms/프레임
self.speeds	list of floats (length 25)	5 개 로봇의 바퀴 및 슬라이더 속도
self.cur_posture	list of floats (size 5*7)	내 팀 5 개 로봇의 좌표, 각도 및 상태
self.cur_posture_opp	list of floats (size 5*7)	상대 팀 5 개 로봇의 좌표, 각도 및 상태
self.cur_ball	list of floats (length 3)	공의 좌표
self.previous_ball	list of floats	이전 프레임 공의 좌표

	(length 3)	
self.predicted_ball	list of floats (length 3)	두 프레임 후 공의 예측 좌표 ※ 'update()'에서 예측 스텝 변경 가능
self.idx	int	공과 가장 가까운 자신 팀 로봇의 번호
self.idx_opp	int	공과 가장 가까운 상대 팀 로봇의 번호
self.previous_frame	object	이전 프레임의 정보를 담고 있는 class
self.defense_angle	float	내 팀 골대 중앙과 공을 이은 직선과 자신 팀 골대 중앙과 경기장 중앙을 이은 직선 사이의 각도 (단위: rad)
self.attack_angle	float	상대 팀 골대 중앙과 공을 이은 직선과 상대 팀 골대 중앙과 경기장 중앙을 이은 직선 사이의 각도 (단위: rad)
self.gk_index self.d1_index self.d2_index self.f1_index self.f2_index	int	포지션 별 로봇 번호
self.GK self.D1 self.D2 self.F1 self.F2	object	포지션 별 로봇이 필요한 정보를 초기화하고 매 업데이트 마다 현재 프레임을 기준으로 다음 행동을 결정하는 class

다음으로 update() 함수를 분석한다. update() 함수는 50ms 마다 게임 프레임을 수신하여 'received_frame' 변수에 할당한다. 'end_of_frame' flag 가 True 로 설정된 후에 전체 프레임이 성공적으로 수신된다. 따라서 update() 함수의 프로세스는 "received_frame.end_of_frame"이 시작할 때 True 이어야 수행된다. 'received_frame'을 기반으로 'get_coord()'함수를 호출하여 로봇과 볼의 좌표 및 방향 등 프로그램의 전역 변수를 갱신(업데이트)한다. 다음 시간 단계의 공 위치 예측, 공에 가장 가까운 로봇, 유용한 수비/공격 각도와 같은 게이머 전략의 다른 중요한 변수들은 보통 update()안에서 호출되는 'helper.py' 파일에 정의된 기능을 사용하여 계산된다.

로봇 바퀴와 슬라이더를 정의하는 전략에 필요한 정보를 가지고 로봇 class 의 'move'기능을 호출하여 Supervisor 에게 보내야 하는 모든 로봇의 변수를 포함하는 self.speeds 라는 list 를 만든다. 'move'기능은 players.py 스크립트에 정의되어 있으며, 현재 게임 상태에 따라 다음 시간 단계에서 로봇이 이동할 위치와 로봇이 슛 동작을 시도할지를 선택한다. 5 개의 로봇 (GK, D1, D2, F1, F2) 모두에 대해 'move' 기능이 호출 된 후 'self.speed' list 가 전송된다. 속도는 'self.set_speed' 함수를 사용하여 Supervisor 에게 전송된다.

update() 함수를 종료하기 위해 현재 게임 상태를 'self.previous_frame'변수에 저장하며, 다음 시간 단계에서 공의 속도 등 중요한 정보를 계산하는데 사용된다. 또한 'print_debug_flag'함수를 호출하여 전략의 어느부분이 수행되었는지를 보여준다. debug_ flag 는 게이머가 자신의 전략이 계획대로 작동하였는지 확인하는데 사용 할 수 있다.

```

def update(self, received_frame):
    if (received_frame.end_of_frame):
        self.frame += 1

    if (self.frame == 1):
        self.previous_frame = received_frame
        self.get_coord(received_frame)
        self.previous_ball = self.cur_ball

    self.get_coord(received_frame)
    self.predicted_ball = helper.predict_ball(self.cur_ball, self.previous_ball)
    self.idx = helper.find_closest_robot(self.cur_ball, self.cur_posture, self.number_of_robots)
    self.idx_opp = helper.find_closest_robot(self.cur_ball, self.cur_posture_opp, self.number_of_robots)
    self.defense_angle = helper.get_defense_kick_angle(self.predicted_ball, self.field, self.cur_ball)
    self.attack_angle = helper.get_attack_kick_angle(self.predicted_ball, self.field)

    #update the robots wheels)
    # Robot Functions
    self.speeds[6 * self.gk_index : 6 * self.gk_index + 6] = self.GK.move(self.gk_index,
                                                                            self.idx, self.idx_opp,
                                                                            self.defense_angle, self.attack_angle,
                                                                            self.cur_posture, self.cur_posture_opp,
                                                                            self.previous_ball, self.cur_ball, self.predicted_ball)

    self.speeds[6 * self.d1_index : 6 * self.d1_index + 6] = self.D1.move(self.d1_index,
                                                                            self.idx, self.idx_opp,
                                                                            self.defense_angle, self.attack_angle,
                                                                            self.cur_posture, self.cur_posture_opp,
                                                                            self.previous_ball, self.cur_ball, self.predicted_ball)

    self.speeds[6 * self.d2_index : 6 * self.d2_index + 6] = self.D2.move(self.d2_index,
                                                                            self.idx, self.idx_opp,
                                                                            self.defense_angle, self.attack_angle,
                                                                            self.cur_posture, self.cur_posture_opp,
                                                                            self.previous_ball, self.cur_ball, self.predicted_ball)

    self.speeds[6 * self.f1_index : 6 * self.f1_index + 6] = self.F1.move(self.f1_index,
                                                                            self.idx, self.idx_opp,
                                                                            self.defense_angle, self.attack_angle,
                                                                            self.cur_posture, self.cur_posture_opp,
                                                                            self.previous_ball, self.cur_ball, self.predicted_ball)

    self.speeds[6 * self.f2_index : 6 * self.f2_index + 6] = self.F2.move(self.f2_index,
                                                                            self.idx, self.idx_opp,
                                                                            self.defense_angle, self.attack_angle,
                                                                            self.cur_posture, self.cur_posture_opp,
                                                                            self.previous_ball, self.cur_ball, self.predicted_ball)

    self.set_speeds(self.speeds)

    self.previous_frame = received_frame
    self.previous_ball = self.cur_ball

    helper.print_debug_flag(self)

```

그림 58 update() 함수

그림 58 에 사용하는 변수들은 게이머가 임의로 명명하여 쓸 수 있다. 앞에 언급했듯이, update() 함수는 게이머 전략 알고리즘을 구현할 수 있는 부분이다. 그림 58 에 있는 다양한 변수들, predicted_ball(다음 프레임에서 예측되는 공의 위치), find_closest_robot(공과 가장 가까운 위치에 있는 로봇)등은 helper.py 에 구현되어 있다.

요약하면, 로봇들과 공의 위치를 이용하여 공격수, 수비수, 골키퍼의 위치 선정과 패스 및 킥과 드리블등을 'helper.py', 'player.py', 'action.py' 을 통해 실행하는 부분이다.

(2) action.py 의 활용방법

'action.py' 파일에는 바퀴, 슬라이더 및 드리블 모드와 관련된 변수를 계산하는 기능이 포함되어 있다. 로봇이 이동해야 할 위치를 기준으로 좌우 바퀴 속도를 계산하는 컨트롤러는 'go_to()'와 'go_fast()' 함수에 정의되어 있다. 'pass_to()'는 특정 방향으로 패스를 할 때 로봇의 회전 운동을 돕기 위한 컨트롤러이다. 컨트롤러를 사용하는 이유는 Webots 시뮬레이터가 물리적 법칙에 의해 동작하기 때문에 적절한 제어를 위해 필요하기 때문이다. 'shoot' 함수는 3 가지 동작 cross, kick, quickpass 를 정의한다. 원하는 경기 동작에 따라 전면 슬라이더 속도와 높이를 조절한다. 'jump()' 함수는 게이머가 헤더골을 얻기 위한 점프동작을 시도하고 싶을 때등에 하단 슬라이더를 조절한다. 마지막으로 'dribble()'기능은 드리블모드를 활성화 및 비활성화 한다.

```
def go_to(id, x, y, cur_posture, cur_ball, max_linear_velocity):
def go_fast(id, cur_posture, cur_ball):
def shoot(cross, kick, quickpass):
def jump(flag):
def dribble(flag):
def pass_to(id, x, y, speed, height, cur_posture, cur_ball, max_linear_velocity):
```

그림 59 action.py 파일

표 6 'action.py'에 정의되어 있는 함수들

함수(매개변수)	설명
go_to(id, x, y, cur_posture, cur_ball, max_linear_velocity)	id 에 해당하는 로봇이 현재 위치(cur_posture)에서 x, y 로 정의된 새로운 위치로 가기 위한 바퀴 속도 값 반환. 현재 공의 위치(cur_ball)는 go_to()에서 호출하는 go_fast()에서 필요함. max_linear_velocity 는 반환하는 바퀴 속도가 최대값을 넘지 않게 하기 위해 필요함. (더 정밀한 위치제어를 위하여 수정 가능)
go_fast(id, cur_posture, cur_ball)	느려지지 않아야 하는 상황을 판단해 True/False 반환. 공쪽으로 갈 경우 현재 공의 위치가 필요함. (‘go_to()’에서 목적지에 가까운 경우 속도가 느려짐)
shoot(cross, kick, quikpass)	각 flag 의 True/False 에 따라 전면 슬라이더 속도와 높이를

	<p>반환.</p> <p>1,0,0 이면 cross 를 의미함. 두개 이상의 flag 가 실수로 참이 될 경우, cross, kick, quickpass 순으로 우선순위를 가지게 되어 1 개만 유효하게 함. 빠른 shoot 동작을 위해, 전면 슬라이더 속도/높이가 cross=10/8, kick=10/0, quickpass=5/0 로 셋팅되어 있음. 슬라이더의 높이는 0 일때 땅볼이고 10 일때 가장 높은 공중볼. kick 과 quickpass 의 차이는 속도. 슬라이더의 높이와 공의 궤적 관계를 주의할것. (필요에 따라 수정 가능)</p>
jump(flag)	<p>flag 의 True/False 에 따라 하단 슬라이더 속도(0~10)를 반환. 속도가 10 일때 청소년용 로봇 GK 는 최대 25cm 의 높이를 가질 수 있음. update()에서 self.speeds[4] = 10 으로 한 뒤 helper.py 의 self.printConsole 을 사용해 cur_posture[0][Z]를 출력하면 확인해 볼 수 있다. 4 는 표 3 에서 GK 의 하단 슬라이더 속도로 정의되고 0 은 GK 의 id 임. (필요에 따라 수정 가능)</p>
dribble(flag)	<p>flag 의 True/False 에 따라 드리블 모드의 전환(on/off)</p>
Pass_to(id, x, y, speed, height, cur_posture, cur_ball, max_linear_velocity)	<p>id 에 해당하는 로봇이 현재 위치에서 x,y 로 공을 보내기 위한 값들을 반환. 반환되는 값들은 'action.py'파일에서 확인.</p> <p>슬라이더의 힘(속도에 해당)과 높이를 조정하여 공을 띄워서 패스할 수 있음.</p> <p>공이 가까이 있지 않다면, 로봇이 가까이 접근하도록 설정되어 있음</p>

따라서, 게이머는 각 상황에서 로봇을 이동시킬 좌표를 결정하고 action.py 의 'go_to()'함수를 사용하여 좌우 바퀴 속도를 계산해야 한다. 동작이 가능할 때 그 기회를 인식하기 위해 'BALL_POSSESSION' flag 를 사용할 수 있다. 로봇이 공을 점유하고 있다면 kick 을 시도하다가 성공할 가능성이 높다. 볼 위치 좌표 중 Z 축 좌표를 이용하여 점프 동작 기회를 예측할 수 있다.

(3) helper.py 의 활용방법

helper.py 파일에는 게이머가 자신의 전략을 실행하고 게임 상태에 관한 중요한 상황 정보를 계산하는 데 도움이 되는 여러가지 보조 함수들이 포함되어 있다. player_rulebasedA.py, players.py, action.py 에서도 'helper.py 에서 선언된 함수들을 사용한다.

```
def distance(x1, x2, y1, y2): """
def degree2radian(deg): """
def radian2degree(rad): """
def wrap_to_pi(theta): """
def predict_ball(cur_ball, previous_ball): """
def find_closest_robot(cur_ball, cur_posture, number_of_robots): """
def ball_is_own_goal(predicted_ball, field, goal_area): """
def ball_is_own_penalty(predicted_ball, field, penalty_area): """
def ball_is_own_field(predicted_ball): """
def ball_is_opp_goal(predicted_ball, field, goal_area): """
def ball_is_opp_penalty(predicted_ball, field, penalty_area): """
def ball_is_opp_field(predicted_ball): """
def get_defense_kick_angle(predicted_ball, field, cur_ball): """
def get_attack_kick_angle(predicted_ball, field): """
def set_wheel_velocity(max_linear_velocity, left_wheel, right_wheel): """
def printConsole(message): """
def print_debug_flag(self): """
```

그림 60 helper.py 파일

그림 60 에 있는 각 함수들은 표 7 에 설명되어 있다.

표 7 'helper.py'에 정의된 함수들

Function(parameters)	설명
distance(x1, x2, y1, y2)	두 점 (x1, y1), (x2, y2) 사이의 거리 반환
degree2radian(deg)	각도 단위 변환 (degree->radian)
radian2degree(rad)	각도 단위 변환 (radian->degree)
wrap_to_pi(theta)	각도를 $[-\pi, \pi]$ 범위로 변환
predict_ball(cur_ball, previous_ball)	한개 프레임 (50ms) 경과후 예측된 공 위치 반환
find_closest_robot(cur_ball, cur_posture, number_of_robots)	공과 가장 가까운 로봇의 번호 반환
ball_is_own_goal(predicted_ball, field, goal_area)	공 위치를 예측해 내 진영의 goal area/penalty area/내 진영 에 위치하면 True 아니면 False. 내 진영의 나머지 지역은 내 진영의 goal area 와 내 진영의 penalty area 를 제외한 나머지 지역
ball_is_own_penalty(predicted_ball, field, penalty_area)	
ball_is_own_field(predicted_ball)	
ball_is_opp_goal(predicted_ball, field, goal_area)	공 위치를 예측해 상대팀 진영의 goal area/penalty area/상대팀 진영에 위치하면 True 아니면 False. 상대팀 진영의 나머지 지역은 상대팀 진영의 goal area 와 상대팀 진영의 penalty area 를 제외한 나머지 지역
ball_is_opp_penalty(predicted_ball, field, penalty_area)	
ball_is_opp_field(predicted_ball)	
get_defense_kick_angle(predicted_ball, field, cur_ball)	내 팀 골대 중앙과 공을 이은 직선과 내 팀 골대 중앙과 경기장 중앙을 이은 직선 사이의 각도 반환

Function(parameters)	설명
<code>get_attack_kick_angle(predicted_ball, field)</code>	상대 팀 골대 중앙과 공을 이은 직선과 상대 팀 골대 중앙과 경기장 중앙을 이은 직선 사이의 각도 반환
<code>set_wheel_velocity(max_linear_velocity, left_wheel, right_wheel)</code>	왼쪽이나 오른쪽 바퀴 속도가 최대속도를 넘는 값이면 비율을 유지하며 더 큰 속도를 최대속도에 맞춘 뒤 변경된 바퀴 속도 값 반환
<code>printConsole(message)</code>	message 를 시뮬레이터 Console 창에 표시
<code>print_debug_flag(self)</code>	팀의 각 로봇의 flag 를 시뮬레이터 Console 창에 표시. 알고리즘의 실행되고 있는 부분 확인에 사용 가능

메모 [오전3]: 설명 추가

(4) player.py 활용방법

```
class Goalkeeper: ...
class Defender_1: ...
class Defender_2: ...
class Forward_1: ...
class Forward_2: ...
```

그림 61 player.py 에 있는 로봇 class

players.py 는 팀의 로봇들의 class 를 정의하는 파일이다. 그림 61 과 같이 Goalkeeper(GK), Defender_1(D1), Defender_2(D2), Forward_1(F1), Forward_2(F2)가 정의 되어 있다. 로봇 class 는 ‘_init_()’과 ‘move()’의 두가지 기능으로 나뉜다. 로봇 Forward_2 에 의해 정의된 ‘_init_()’과 ‘move()’ 함수의 예제는 그림 62 에 나와 있다.

```
class Forward_2:
    def __init__(self, field, goal, penalty_area, goal_area, robot_size, max_linear_velocity):
        self.field = field
        self.goal = goal
        self.penalty_area = penalty_area
        self.goal_area = goal_area
        self.robot_size = robot_size
        self.max_linear_velocity = max_linear_velocity

    def move(self, id, idx, idx_opp, defense_angle, attack_angle, cur_posture, cur_posture_opp, previous_ball, cur_ball, predicted_ball):
        kick = False
        jump = False

        if helper.ball_is_own_goal(predicted_ball, self.field, self.goal_area):
            x = -0.5
            y = 1
        elif helper.ball_is_own_penalty(predicted_ball, self.field, self.penalty_area):
            x = -8.5
            y = 1
        elif helper.ball_is_own_field(predicted_ball):
            x = cur_ball[X]
            y = cur_ball[Y]
        elif helper.ball_is_opp_goal(predicted_ball, self.field, self.goal_area):
            x = cur_ball[X]
            y = cur_ball[Y]
            if cur_posture[id][BALL_POSSESSION]:
                kick = True
        elif helper.ball_is_opp_penalty(predicted_ball, self.field, self.penalty_area):
            x = cur_ball[X]
            y = cur_ball[Y]
            if cur_posture[id][BALL_POSSESSION]:
                kick = True
        else:
            x = cur_ball[X]
            y = cur_ball[Y]
            if cur_posture[id][BALL_POSSESSION]:
                kick = True

        left_wheel, right_wheel = action.go_to(id, x, y, cur_posture, cur_ball, self.max_linear_velocity)
        slider_horizontal, slider_vertical = action.kick(kick)
        slider_jump = action.jump(jump)
        return left_wheel, right_wheel, slider_horizontal, slider_vertical, slider_jump
```

그림 62 Forward_2 ‘_init_()’과 ‘move()’ 함수

player_rulebasedA.py 의 'init()' 함수에서 class 개체를 만들 때 '_init_()' 함수가 호출된다. '_init_()' 함수는 5 개의 전역 변수를 선언하여 필드, 골, 페널티 영역, 골 영역, 로봇 크기 및 최대 선형 속도 정보를 나중에 'move' 함수에 사용할 수 있도록 한다.

'move()' 함수는 규칙 기반 예제에서 가장 중요한 함수다. move() 함수는 게임의 주요 전략을 담고 있다. 'move()' 함수는 'player_rulebasedA'의 'update()' 함수쪽에서 현재의 게임 상태를 변수로 전달받아 왼쪽 바퀴 속도, 오른쪽 바퀴 속도, 전면 슬라이더 속도, 전면 슬라이더 높이, 하단 슬라이더 속도, 드리블 모드 flag 을 'update()' 쪽으로 반환한다.

rulebasedA 예제 코드에서 GK, D1, D2, F1 로봇들의 움직임은 게임 상태에 상관 없이 고정되어 있다. 만약 player_rulebasedA.py 를 실행시켜 보면, GK 는 상대 킥으로부터 골대를 보호하려고 움직이고, 축구장에서 D1, D2, F1 은 고정된 위치로 이동하는 것을 볼 수 있다. 여기서 그림 62 에 제시되어 있는 로봇 F2 의 전략 코드를 분석해본다. F2 의 전략은 현재 공의 위치를 기준으로 6 가지 상황으로 나눈다. **좌표값에 따른 로봇의 위치는 그림 63 을 참조하기 바란다.**

1. 다음 시간 단계에 공이 내 진영 골 지역 안에 있을 것으로 예상되면 (-0.5, -1) 위치로 이동한다.
2. 다음 시간 단계에 공이 내 진영 페널티 지역에 있을 것으로 예상되면 (-0.5, -1) 위치로 이동한다.
3. 다음 시간 단계에 공이 내 진영의 나머지 지역에 있을 것으로 예상되면 현재 공 위치로 이동한다.
4. 다음 시간 단계에 공이 상대 진영 골 지역에 있을 것으로 예상되면 현재 공 위치로 이동하여 로봇이 공 점유를 한 경우 킥을 시도한다.
5. 다음 시간 단계에 공이 상대 진영 페널티 지역에 있을 것으로 예상되면 현재 공 위치로 이동하고 로봇이 공 점유를 한 경우 킥을 시도한다.
6. 다음 시간 단계에 공이 상대 진영의 나머지 지역에 있을 것으로 예상되면 현재 공 위치로 이동하고 로봇이 공 점유를 한 경우 킥을 시도한다.

그림 62 에 보인 코드는 항상될 가능성이 많은 기본적인 행동 전략이다. 이러한 행동 전략을 개선하는 방법을 소개하겠다. 로봇을 현재 공 위치로 이동하는 것보다는 공이 갈 것으로 예상되는 위치로 움직이도록 하는것이다. 또다른 방법은 공의 궤적이 상대방 골대의 방향일 경우, 로봇이 킥을 하게 하는 것이다. 다음에 코드를 이용하여 여러 개의 로봇을 사용하여 보다 복잡한 행동을 구현하는 방법을 소개할 것이다.

요약하자면, 게이머 전략을 구현하는 규칙은 'player.py'에 정의되어 있다. 게임 상태에 따라서 게이머는 if-else 문을 사용하여 로봇이 어떻게 움직이고 행동하는지 설계한다.

규칙기반 예제 코드를 사용하면 'self.flag'값을 정의하여 알고리즘의 어느 부분이 현재 시간 단계에서 실행인지 확인함으로써 전략을 디버깅 할 수 있다. 'self.flag' 값들은 'player_ruleBasedA.py'파일 안에 있는 'print_debug_flag' 함수를 이용하여 콘솔의 알림창에 출력해서 확인 할 수 있다.

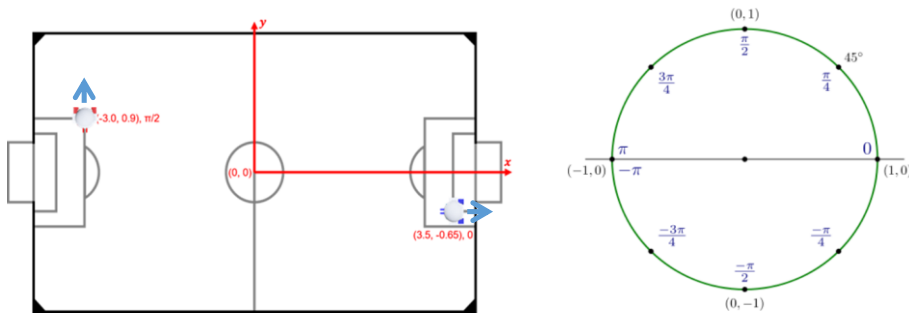


그림 63 경기용 좌표계와 각도 시스템

그림 63 에 x, y 좌표계와 각도 시스템을 나타내었다. 모든 좌표 단위는 m(eter)이고 방향을 나타내는 각도는 $-\pi$ 에서 π 의 라디안으로 표현한다. 그림 63 의 왼쪽 부분에서 보인바와 같이 로봇의 방향은 로봇이 주시하는 화살표 방향으로 정의한다. 왼쪽 로봇은 90° 혹은 $\pi/2$ 라디안 방향을 주시하고 있고, 오른쪽 로봇은 0° 혹은 0 라디안 방향을 주시하고 있다.

규칙 기반 전략에 대한 추가 설명

앞에서 player_rulebasedA.py 에 있는 예제 코드를 살펴보았다. 하지만 여러 로봇으로 더 조직적인 행동을 할 수 있는 방법에는 초점을 맞추지 못하였다. 이제 로봇 F2 가 공을 로봇 F1 에게 패스하려고 하는데, 로봇 F1 이 스스로 위치를 잡아 패스를 기다리며 공을 상대 골대로 차는 게임 상황을 가정하여 전략을 세워보고자 한다.

상대 수비를 상대로 하는 내 팀의 로봇 F1 과 F2 의 초기 위치에서 예상 동작은 다음과 같다. 먼저 공은 상대 필드 하단에 위치하며, 로봇 F2 가 공에 접근해 로봇 F1 에 대한 패스를 준비해야 한다. 한편 F1 은 패스를 받기 위해 위치를 잡아야 한다. 이 전략은 그림 64 에 설명되어 있다.

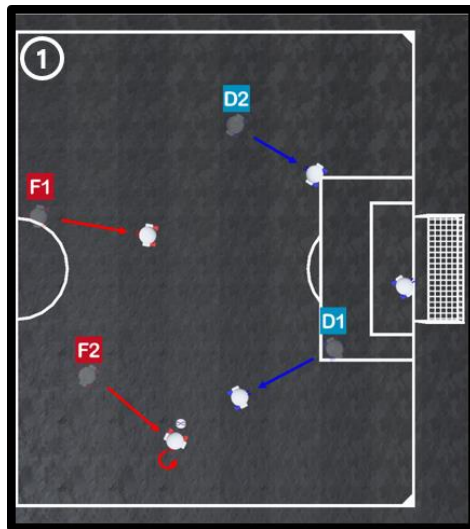


그림 64 F1 과 F2 에게 기대하는 움직임: F2 는 공으로 가까이 가고, F1 은 패스를 받아 킥을 하기 좋은 위치로 움직인다.

그 다음 프레임에서 F1 이 패스를 기다리는 동안 F2 는 패스하고, F1 은 골대로 킥을 차기 좋은 각도로 방향을 바꾼다. 이 행동은 그림 65 에 설명되어 있다.

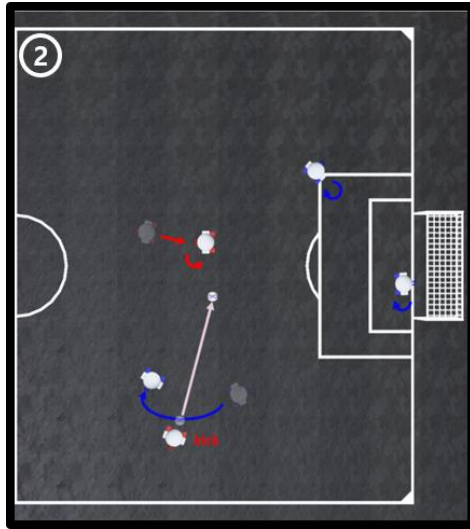


그림 65 F1 과 F2 에게 기대하는 움직임: F2 는 패스를 하고, F1 은 패스를 받아 킥을 하기 좋은 각도로 방향을 바꾼다.

F1 은 상대 골대 방향으로 킥을 시도하고 F2 는 공을 쫓아간다. 이 행동은 그림 66 에 설명되어 있다.

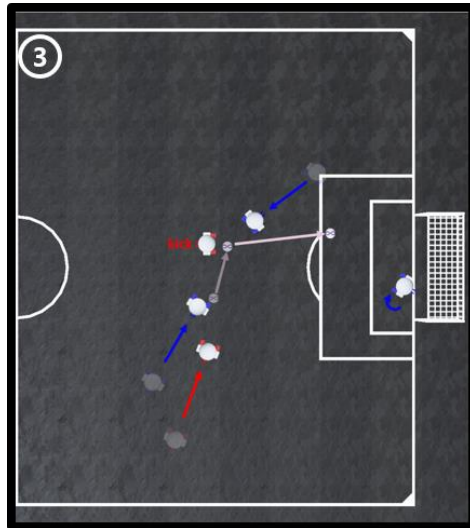


그림 66 F1 과 F2 에게 기대하는 움직임: F1 은 킥을 하고 F2 는 공을 따라서 움직인다.

킥이 끝난 뒤 두 로봇은 모두 상대 골키퍼가 공을 막아 냈을 때를 대비하여 또 다른 킥을 할 수 있도록 공을 쫓아간다. 이 행동은 그림 67 에 설명되어 있다.

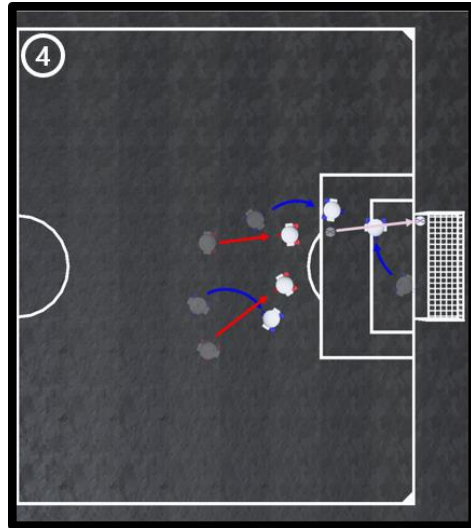


그림 67 F1 과 F2 에게 기대하는 움직임: F1 과 F2 는 모두 공을 따라서 움직이고, 골키퍼가 막아낸 튕겨나온 공을 다시 찰 준비를 한다.

규칙 기반 환경에서 F1 와 F2 가 위에서 보여준 행동들을 Flowchart 로 정리하였다.

Flowchart 는 로봇의 class move() 함수에 위치한 것과 유사한 코드로 변환 해 볼 수 있다. 그림 68 과 69 은 flowchart 에 해당한다.

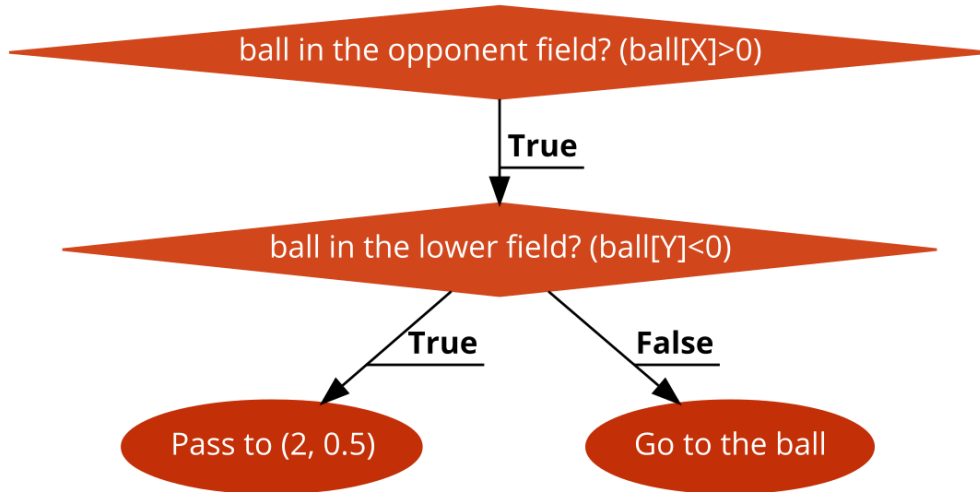


그림 68 Flowchart : F1 에 기대되는 행동

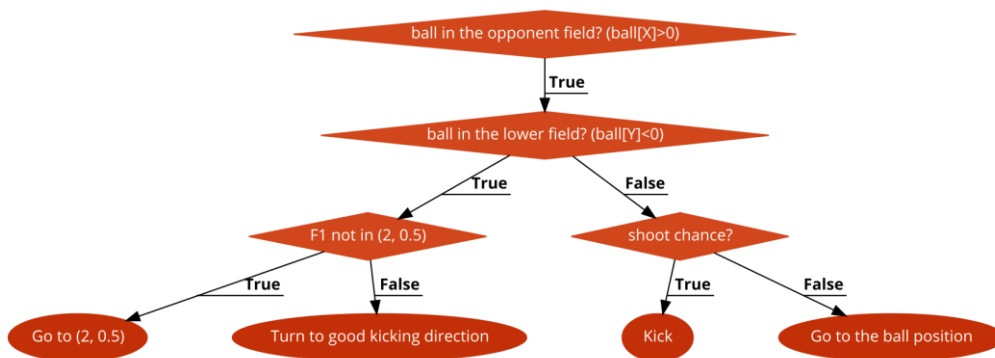


그림 69 Flowchart : F2 에 기대되는 행동

4-2 AI 와 인간사이의 AI 축구

AI Soccer Example Program

AI 축구 플랫폼은 컴퓨터 AI와 게이머간의 경기를 위한 기능이 포함되었다. 한 팀의 로봇 4개는 미리 구현한 AI 알고리즘으로 움직이고 1대의 로봇은 게이머의 키보드 조작에 의해 움직이게 할 수 있다. 이 때 상대팀은 규칙 기반이나 학습 기반으로 만들어진 컴퓨터 AI이다.

이러한 방식의 AI 축구 게임은 게이머의 코드 수준을 높이는데 효과적이며 유용하다. 컴퓨터 AI 팀을 게이머의 현재 코드로 대체할 경우, 게이머가 키보드로 조작되는 로봇을 특정한 전략에 따라 움직이게 함으로써 게이머의 현재 코드를 잘 보완하면 키보드로 조작되는 팀과 유사한 전략을 구사하는 팀을 이길 수 있는 가능성을 높여준다.

게이머가 조작하는 로봇은 노란색 조명으로 표기되고, 조작할 로봇을 선택하기 쉽도록 로봇 머리 위에 마크 GK, D1, D2, F1, F2가 표시된다.



그림 70 키보드 사용 환경 - 노란색 불이 켜진 로봇은 게이머가

직접 키보드로 조작하고 있음.

키보드 조작 시 각 키의 기능이 표 8에 나와있다. 그림 71은 AI 추구를 할 때 사용되는 키(버튼)의 위치를 보여준다.

표 8 키보드 설명

Key	설명
up	두 바퀴 속도를 1로 바꾼다(전진).
down	두 바퀴 속도를 -1로 바꾼다(후진).
left	왼쪽 바퀴 속도는 줄이고 오른쪽 바퀴 속도는 높인다(좌회전).
right	왼쪽 바퀴 속도는 높이고 오른쪽 바퀴 속도는 줄인다(우회전).
q	드리블 모드/일반 모드를 전환한다.
e	두 바퀴의 속도를 높인다. 누르고 있으면 1에서 2.2까지 0.2씩 올라간다. 버튼에서 손을 떼면 1까지 0.2씩 내려간다.
a	전면 슬라이더의 높이를 바꾼다(0: 땅볼, 8: cross와 같은 공중볼). 기본 0, 0 → 8, 8 → 0
s	전면 슬라이더에 속도 5를 준다(킥패스).
d	전면 슬라이더에 속도 10을 준다(킥).
w	하단 슬라이더에 속도 10을 준다(점프).
z	조작되는 로봇이 GK일 때만 작동, 왼쪽 슬라이딩
x	조작되는 로봇이 GK일 때만 작동, 앞쪽 슬라이딩
c	조작되는 로봇이 GK일 때만 작동, 오른쪽 슬라이딩
1 ~ 6	조작하는 로봇의 번호를 바꾼다. 1~5: GK, D1, D2, F1, F2 / 6: 공과 가장 가까운 로봇

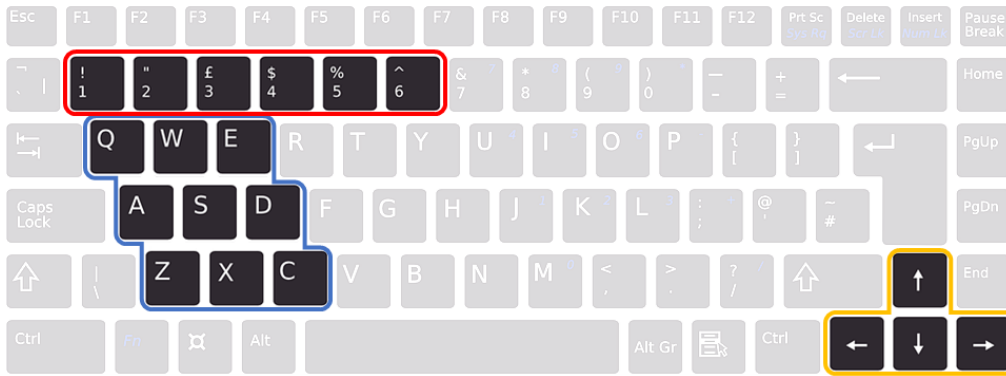


그림 71 게이머가 조작할 수 있는 키

's', 'd', 'w', 'z', 'x', 'c'는 한 프레임 동안만 바뀐다. 두개의 방향키를 동시에 누르고 있으면 원을 그리면서 움직인다.