Carla Simulator 사용 설명서(Manual)

1. 개요
2. Carla 빌드(Linux)
    A. 요구 사양
    B. 의존성
    C. Github
    D. Unreal Engine
    E. Carla 빌드
3. Carla 업데이트
    A. 최신 바이너리 릴리즈 다운로드
    B. Linux와 Windows 빌드 업데이트
4. 시스템 빌드
    A. 설치
    B. LibCarla
    C. CarlaUE4와 Carla 플러그인
    D. PythonAPI
5. 핵심 개념
    A. 첫 번째 단계
        i. World와 Client
        ii. Actors와 blueprints
        iii. Maps와 Navigation
        iv. Sensors와 data
    B. 심화 단계
        i. OpenDrive 독립 실행형 모드
        ii. PTV-Vissim 공동 시뮬레이션
        iii. 녹화기
        iv. 렌더링 설정

v. RSS

vi. SUMO 공동 시뮬레이션

vii. 동기화 및 time-step

viii. 교통 매니저

1. 개요

자율주행은 4차산업혁명 시대를 이끌어가는 주요한 아이디어이다. 하지만 자율주행을 실제로 배포하고 실용화하기 위해서는 많은 도로상황에 대비해야 한다. Carla는 이를 위한 연구로 오픈소스 시뮬레이터의 형태로 개발되었다. Carla는 자율주행시스템의 개발, 훈련, 검사를 지원하기 위해 제작된 Unreal Engine 기반의 무료 오픈 소스 시뮬레이터이다.

Carla는 기본적으로 자율주행과 관련된 세 가지 성능을 연구한다.

- 모듈형 Classic pipeline
- 모방 학습을 통해 훈련된 End-to-End 모델
- 강화 학습을 통해 훈련된 End-to-End 모델

Carla는 다양한 시뮬레이션과 여러가지의 상황을 반복해서 학습할 수 있다. 또한 날씨, 조명, 도로 조건 등을 변경할 수 있다.

주요 기능

- **서버 다중 클라이언트 아키텍처를 통한 확장성**: 동일하거나 다른 노드에 있는 여러 클라이언트가 서로 다른 행위자를 제어할 수 있다.
- **유연한 API**: CARLA는 사용자가 트래픽 생성, 보행자 행동, 날씨, 센서 등 시뮬레이션과 관련된 모든 측면을 제어할 수 있는 강력한 API를 공개한다.
- **자율주행 센서 세트**: 사용자는 LIDAR, 다중 카메라, 깊이 센서, GPS 등 다양한 센서 제품군을 구성할 수 있다.
- **계획 및 제어를 위한 빠른 시뮬레이션**: 이 모드는 렌더링을 비활성화하여 그래픽이 필요하지 않은 트래픽 시뮬레이션과 도로 주행 동작을 신속하게 실행할 수 있다.
- **지도 생성**: 사용자는 RoadRunner와 같은 도구를 통해 OpenDrive 표준에 따라 자신의 지도를 쉽게 만들 수 있다.
- **트래픽 시나리오 시뮬레이션**: Carla의 엔진 ScenarioRunner는 사용자가 모듈러 동작을 기반으로 다른 트래픽 상황을 정의하고 실행할 수 있도록 한다.
- **ROS 통합**: Carla는 ROS-Bridge를 통해 ROS와 통합되어 제공된다.
- **자율 주행 기준선**: AutoWare agent와 조건부 모방 학습 에이전트를 포함한 CARLA의 실행 가능한 에이전트로 자율 주행 기준선을 제공한다.

2. Carla 빌드(Linux)

본 문서는 Linux 기준으로 빌드가 이루어진다. Linux로 빌드하는 과정은 꽤 길기 때문에 문제가 발생한다면 Carla 문서를 살펴보거나 검색해보는 식으로 해결할 수 있을 것이다.

A. 요구 사양

시스템 스펙

- **Ubuntu 18.04**: UE가 제대로 작동하려면 적절한 컴파일러가 필요하다. 18.04버전으로 설치하면 이전 버전에 사용되었던 것들을 다 가져오기 때문에 굳이 이전 버전을 설치할 이유가 없다.
- **30GB 디스크 공간**: 필요한 모든 소프트웨어 + Carla + Unreal Engine으로 인해 디스크 공간이 많이 필요하다.
- **적당한 GPU**: Carla는 실제적인 시뮬레이션을 목표로 하기 때문에 서버는 최소한 4GB GPU를 장착해야 한다. Machine Learning을 할 시에는 전용 GPU를 사용하는 것이 권장된다.
- **두 개의 TCP 포트와 양호한 인터넷 연결**: 2000과 2001이 default 값이다. 방화벽이나 다른 응용 프로그램이 이 요소를 차단하지 않도록 해야 한다.

B. 의존성

Carla는 많은 의존성을 요구한다. 이들 중 일부는 Boost.Python과 같이 과정 중 자동으로 빌드된다. 다른 바이너리는 빌드(cmake, clang, 다른 버전의 파이썬 등)를 시작하기 전에 설치해야 하는 바이너리이다. 설치하기 위해서 다음 코드를 터미널에 작성하면 된다.

```
sudo apt-get update &&
sudo apt-get install wget software-properties-common &&
sudo add-apt-repository ppa:ubuntu-toolchain-r/test &&
wget -O - https://apt.llvm.org/llvm-snapshot.gpg.key|sudo apt-key add - &&
sudo apt-add-repository "deb http://apt.llvm.org/xenial/ llvm-toolchain-xenial-8 main" &&
sudo apt-get update
```

* 중요: 다음 명령은 우분투 버전에 따라 다르다. 차이점은 `libpng16-dev`부분과 `libpng-dev`이다.

Ubuntu 18.04~

```
sudo apt-get install build-essential clang-8 lld-8 g++-7 cmake ninja-build
libvulkan1 python python-pip python-dev python3-dev python3-pip libpng-dev
libtiff5-dev libjpeg-dev tzdata sed curl unzip autoconf libtool rsync libxml2-dev
libxerces-c-dev &&
pip2 install --user setuptools &&
pip3 install --user -Iv setuptools==47.3.1 &&
pip2 install --user distro &&
pip3 install --user distro
```

Unreal Engine과 Carla 종속성 사이의 호환성 문제를 피하기 위해 동일한 컴파일러 버전과 C++ 런타임 라이

브러리를 사용해 내용들을 컴파일 하는 것이 좋다.

C. Github

Carla의 내용들이 Github 저장소에 정리되어 있기 때문에 Github 계정이 필요할 것이다. 또한, Git은 터미널에서 명령을 쉽게 실행할 때 빌드 설명에서 사용될 것이다.

비공개로 설정된 Unreal Engine 저장소에 엑세스하려면 Unreal Engine 계정을 만들어 Github 계정에 연결하라. 이를 위해 Unreal Engine의 프로필 설정에 Connected Accounts라는 이름으로 섹션이 있다.

* 주의
새로운 Unreal Engine 계정은 활성화되어야 한다. 계정을 만든 후 확인 메일이 발송된다. 체크아웃하지 않으면 다음 단계에서 UE 저장소가 존재하지 않는 것으로 표시된다.

D. Unreal Engine

현재 Carla 버전에서는 Unreal Engine 4.24에서만 실행된다. 설치 경로는 상관없지만 Carla Simulator 자습서의 내용을 따라가기 위해 `~/UnrealEngine_4.24`에 설치가 이루어진다. 만약 설치 경로가 다르다면 터미널에서 명령을 실행할 때마다 그에 맞게 변경하면 된다.

로컬 컴퓨터에서 Unreal Engine 4.24의 내용들을 복사하는 명령어이다.

```
git clone --depth=1 -b 4.24 https://github.com/EpicGames/UnrealEngine.git ~/UnrealEngine_4.24
```

Unreal Engine이 설치된 폴더에 들어가는 명령어이다. 이 코드가 Unreal Engine 4.24가 복제된 경로로 이동하는 것임을 기억해야 한다.

```
cd ~/UnrealEngine_4.24
```

Unreal Engine 패치를 진행하는 명령이다. 이 패치는 지도를 변경할 때 발생할 수 있는 일부 Vulkan 시각화 문제를 수정해준다. 다음 명령을 실행하면 자동으로 설치된다.

```
wget https://carla-releases.s3.eu-west-3.amazonaws.com/Linux/UE_Patch/430667-13636743-patch.txt ~/430667-13636743-patch.txt
patch --strip=4 < ~/430667-13636743-patch.txt
```

다음은 빌드하는 명령이다.

```
./Setup.sh && ./GenerateProjectFiles.sh && make
```

* 주의 Unreal Engine이 이미 빌드가 된 경우에는 패치를 설치하고 다시 빌드해야 한다.

Unreal Engine을 시스템에 설치해야 한다.

Editor를 열고 확인하기 위해 `Engine/Binaries/Linux/UE4Editor.sh`을 실행한다.

```
cd ~/UnrealEngine_4.24/Engine/Binaries/Linux && ./UE4Editor
```

만약 뭔가 잘못되었을 경우, 이는 Unreal Engine과 관련이 있을 가능성이 있다. 이럴 경우에는 Unreal Engine에서 제공하는 빌드 설명서가 도움이 될 수 있다.

E. Carla 빌드

시스템은 Carla 구축을 시작할 준비가 되어 있어야 한다. 다음 내용들의 이해를 돕기 위해 지금까지 한 내용들을 간단히 요약해보겠다.

- 최소한의 시스템 스펙은 만족해야 한다.
- 종속된 프로그램들이 올바르게 설치되어야 한다.
- Github 계정이 준비되어야 한다.
- Unreal Engine 4.24가 부드럽게 작동되어야 한다.

* Note aria2를 다음 명령어`sudo apt-get install aria2` 로 다운로드 하면 곧 나오는 명령어들을 수행하는 속도가 빨라진다.

Carla Clone repository(Github): https://github.com/carla-simulator/carla

Carla의 공식 Github repository이다. 다운로드하여 추출하거나 다음 명령어로 복제해도 된다.

```
git clone https://github.com/carla-simulator/carla
```

이제 repository의 `master` branch 프로젝트의 최신 내용들이 Local로 복사되었다!

`master` branch에는 최신 수정 사항과 기능이 포함되어 있다. Stable code는 `stable`이고 이전 버전의 Carla에서는 자체 branch가 있다. `git branch` 명령어로 Git에서 현재 branch를 항상 체크해야 한다.

Asset 받기

Asset Package는 아직 로드되지 않은 상태이다. 이 Package들은 repository를 조금 더 가볍게 만들기 위해 따로 저장된다. Carla는 Asset 없이는 빌드될 수 없다.

최신 버전을 다운로드하여 추출할 수 있는 스크립트가 있다. Package가 3GB를 넘기 때문에 시간이 조금 걸릴 수 있다.

Carla의 루트 폴더로 들어간다. 경로는 방금 복제한 repository와 일치해야 한다.

```
cd ~/carla
```

명령어를 실행하여 Asset을 다운로드 받는다.

```
./Update.sh
```

* 중요
현재 개발중인 Asset을 받기 위해선, Carla 사이트 내 Update Carla를 방문하여 Get development assets를 참고하면 된다.

환경변수 설정

이 부분은 Carla가 Unreal Engine 4.24 설치 폴더를 찾기 위해 필요하다.

```
export UE4_ROOT=~/UnrealEngine_4.24
```

이 변수는 세션 전체에서 지속적으로 설정되도록 `~/.bashrc` 혹은 `~/.profile` 에 추가해야 한다. 그렇지 않으면 현재의 쉘에서만 접근할 수 있다.

Carla 만들기

마지막 단계는 Carla를 빌드하는 것이다. 다른 모듈들을 만들기 위한 다른 `make`명령들이 있다. 모두 Carla의 root 폴더에서 실행된다.

* 주의
서버를 준비하기 위해 `make launch`를 실행하고 클라이언트를 위해 `make PythonAPI`를 실행해라. 아니면 `make LibCarla`명령어가 Carla 라이브러리를 아무 곳에서나 import할 수 있도록 준비할 것이다.

- `make launch`는 서버 시뮬레이터를 컴파일하고 Unreal Engine을 launch한다. Spectator view를 시작하려면 Play를 누르고 종료하려면 편집기 창을 닫으면 된다. 카메라를 WASD 키로 이동하고 마우스를 이동하면서 장면을 클릭하여 회전할 수 있다.

이 프로젝트는 `UE4Editor-Carla.dll`과 같은 다른 인스턴스를 처음에 요청할 수 있다. 프로젝트를 열려면 동의해야 한다. 첫 launch에서 편집기는 shader와 mech 거리 필드에 대한 경고를 표시할 수 있다. 이것들은 로드되기까지 시간이 걸리고 그때까지는 마을이 제대로 나타나지 않을 것이다.

- `make PythonAPI`는 시뮬레이션을 제어하는데 필요한 API 클라이언트를 컴파일한다. 그것은 처음에만 필요하다. 그렇지만 Carla를 업데이트 했다면 다시 실행해야 한다. 이 명령어가 실행된 후에는 Script가 실행될 수 있다. 다음 명령어를 실행하면 마을에 Npc가 생성된다: `make PythonAPI && cd PythonAPI/examples && python3 spawn_npc.py`

* 중요 만약 시뮬레이션이 낮은 프레임으로 작동한다면, Unreal Engine Editor의 `Edit/Editor preferences/Performance`로 이동해서 `Use less CPU when in background`를 비활성화하면 된다.

이제 Carla를 사용하기 위한 모든 준비가 되었다! 지금까지 사용했던 `make`명령어들을 정리해보자.

명령어	내용
`make help`	가능한 명령어들을 출력해준다.
`make launch`	Editor Window에서 Carla 서버를 실행한다.
`make PythonAPI`	Carla client를 빌드한다.
`make package`	Carla를 빌드하고 배포를 위한 패키지 버전을 만든다.
`make clean`	빌드 시스템에서 생성된 바이너리 및 임시 저장소들을 모두 삭제한다.
`make rebuild`	`make clean` 과 `make launch`를 같이 실행하는 명령어이다.

3. Carla 업데이트

A. 최신 바이너리 릴리즈 다운로드

바이너리 릴리즈는 사전에 패키징되어 특정 버전의 Carla에 연결된다. 최신 버전을 얻기 위해선 이전 버전을 지우고 Quick start installation에 따라 설치하면 된다.

릴리즈는 Carla 저장소의 Development 목록에 나열되어 있다. 또한 Carla의 현재 상태를 담은 매우 실험적인 Nightly build도 있다.

B. Linux와 Windows 빌드 업데이트

업데이트하기 전에 로컬 마스터 분기에 있는지 확인해야 한다. 그런 다음 다른 분기에 대한 변경사항을 병합 또는 재생성해서 발생할 수 있는 충돌들을 해결한다.

```
git checkout master
```

Clean Build

Carla의 메인 폴더로 가서 바이너리와 임시 파일을 삭제하는 명령어이다.

```
make clean
```

Origin에서 Pull

Carla repository에서 최신 버전을 받는 명령어이다.

```
git pull origin master
```

Asset 다운로드

```
./Update.sh
```

서버 실행하기

모든 것이 제대로 작동하는지 확인하려면 서버를 Spectator mode로 실행하면 된다.

```
make launch
```

개발 Asset 가져오기

Carla 팀은 지금까지도 계속 Asset을 개발하고 있다. Asset의 model들과 map들은 Carla 팀이 Git repository 에 정기적으로 업데이트를 하고 있다. Asset이 미완성이기 때문에 개발자만 사용하길 권하고 있다.

이 repository를 다루기 위해 Git-lfs를 설치하는 것이 권장한다. repository는 정기적으로 수정되고, 특히 Git-lfs는 큰 바이너리 파일일 때 더 빨리 동작한다.

repository를 clone 하려면, 기본 Carla 디렉토리로 가서 다음 명령어를 실행하면 된다.

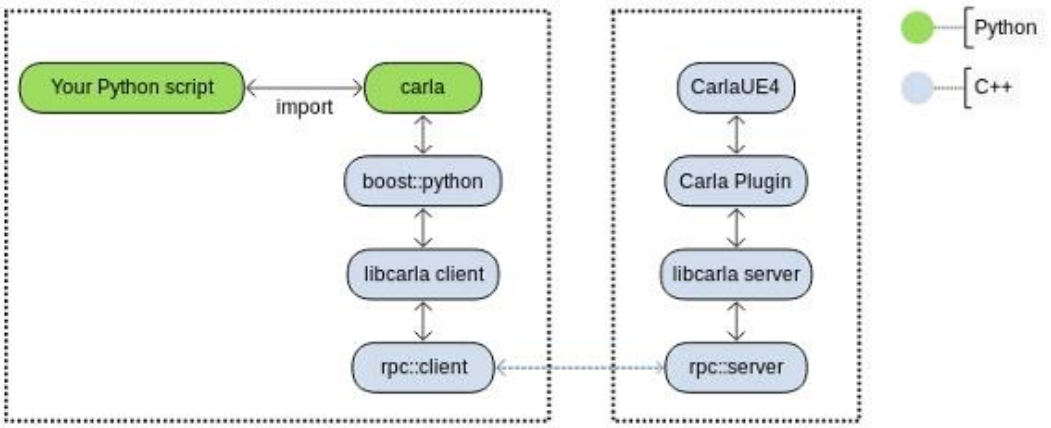
```
git clone https://bitbucket.org/carla-simulator/carla-content Unreal/CarlaUE4/Content/Carla
```

* 주의 repository에 clone하기 전에 Asset이 들어있는 `/Carla` 폴더를 삭제해야 한다. 그렇지 않으면 에러가 발생할 수도 있다.

4. 시스템 빌드

설정에서 가장 어려운 부분은 a) 서버에서는 Unreal Engine, b) 클라이언트에서는 파이썬과 호환되는 모든 종속성 및 모듈을 컴파일하는 것이다.

아래 그림에서 왼쪽으로 분리된 파이썬 과정에서 Unreal Engine의 기능을 호출할 수 있는 것이 우리의 목표이다.



리눅스에선 clang-8.0과 C++14 표준과 관련된 모든 종속성을 컴파일한다. 하지만 Unreal Engine과 연결될 모든 코드는 `libc++`를 사용하여 컴파일해야 하기 때문에, 코드를 사용할 위치에 따라 다른 Runtime C++ Library에 대해 링크한다.

A. 설치

설치 명령어는 다음과 같다.

```
make setup
```

종속성을 가져오고 컴파일하는 목록은 아래와 같다.

- llvm-8(libc++과 libc++abi)
- rpblic-2.2.1(libstdc++과 libc++)
- boost-1.72.0(헤더들과 libstdc++를 위한 boost_python)
- googletest-1.8.1(libc++를 위한)

B. LibCarla

cmake로 컴파일되었다(최소한 cmake 버전이 3.9이상이어야 한다.).

명령어는 다음과 같다.

```
make LibCarla
```

C. CarlaUE4와 Carla 플러그인

명령어는 다음과 같다.

```
make CarlaUE4Editor
```

Unreal Engine의 편집기 실행을 시작하려면 다음 명령어를 실행하면 된다.

```
make launch
```

D. PythonAPI

파이썬의 `setuptools`("setup.py")를 사용하여 컴파일한 것이다. 그래서 파이썬, libpython-dev, 파이썬 2.7과 3.5용 libboost-python-dev가 설치돼 있어야 한다.

명령어는 다음과 같다.

```
make PythonAPI
```

이 명령어는 두개의 "egg" 패키지를 생성한다.

```
PythonAPI/dist/carla-X.X.X-py2.7-linux-x86_64.egg
```

```
PythonAPI/dist/carla-X.X.X-py3.7-linux-x86_64.egg
```

이 패키지들을 시스템 경로에 추가하여 파이썬 스크립트로 직접 가져올 수 있다.

```
#!/usr/bin/env python
import sys
sys.path.append(
    'PythonAPI/dist/carla-X.X.X-py%d.%d-linux-x86_64.egg' % (sys.version_info.major,
                                                             sys.version_info.minor))
import carla
# ...
```

다른 방법으로는 `easy_install` 명령어로 설치할 수 있다.

```
easy_install2 --user --no-deps PythonAPI/dist/carla-X.X.X-py2.7-linux-x86_64.egg
```

```
easy_install3 --user --no-deps PythonAPI/dist/carla-X.X.X-py3.7-linux-x86_64.egg
```