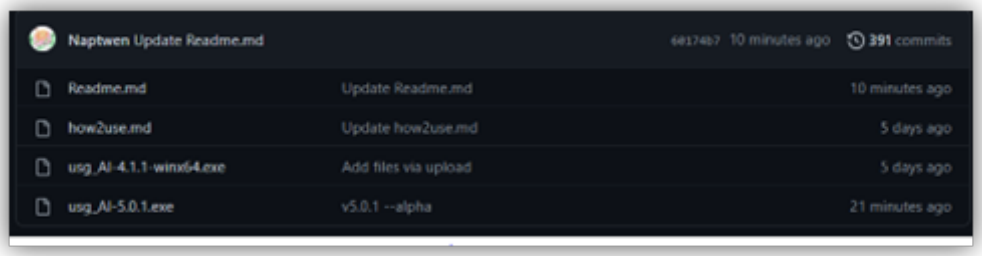
USG AI 엔진 프로그램 사용설명서

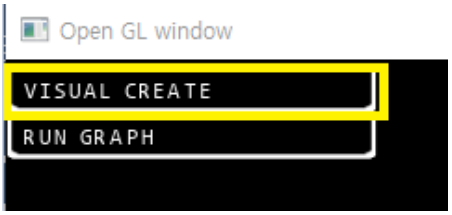
본 프로그램의 소스코드는 모두 본인이 제작하였으며, GNU AFFERO LICENSE V3 를 통해 보호됨과 동시에 특별조항을 명시합니다. 또한 특정 저작권에 속하는 OpenGL, OpenCL, OpenCV, base64, FreeType코드는 프로그램에 첨부된 저작권을 확인 하여주시기 바랍니다.

프로그램 설치법

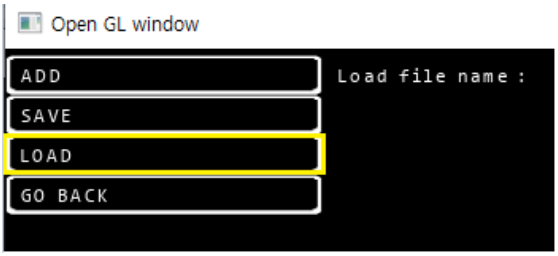
1. 프로그램의 원활한 실행을 위한 사용 소프트웨어 OpenGL, OpenCV, OpenCL을 설치합니다.
2. https://github.com/Naptwen/usgAI 에 접속하여 가장 최신버전의 프로그램 설치파일 받습니다.

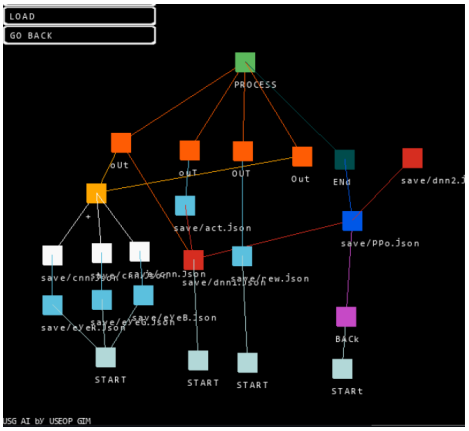


3. 설치된 폴더의 usg_AI 실행파일을 두번 클릭하여 실행합니다.
4. 프로그램의 테스트를 위하여 VISUAL CREATE 버튼을 클릭합니다.



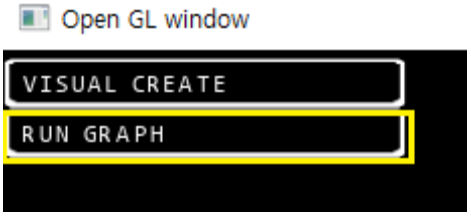
5. LOAD 버튼을 클릭후 “save/test.txt’ 를 입력한후 엔터키를 누릅니다.



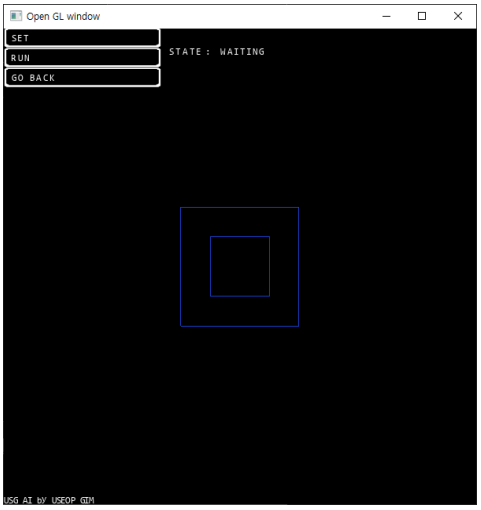


- 위와 같은 화면이 나오지 않는다면, save 폴더안에 예제 파일들이 존재하는지 확인합니다.

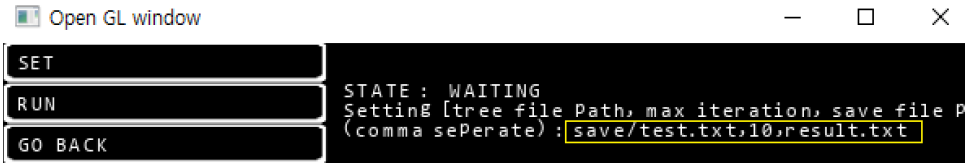
7. RUN GRAPH 버튼을 클릭합니다.



8. 밑의 화면과 같이 중앙에 두사각형이 들어있는 화면이 나오면 SET버튼을 클릭합니다.



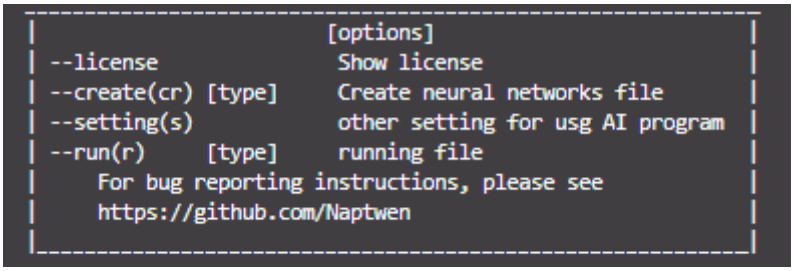
9. 버튼을 클릭한후 “save/test.txt, 10, result.txt” 를 입력합니다.



10. 다음으로 RUN 버튼을 클릭합니다.
11. 중앙의 사각형이 움직이면서 방향키가 자동으로 움직이면 프로그램이 작동하는 것입니다.
12. 중앙의 사격형의 움직임이 멈추면 프로그램을 종료하셔도 됩니다.

기본 콘솔 인터페이스 설명

각모델들은 편집파일을 통해 직접 작성 및 편집 혹은 기본 제공되는 콘솔 인터페이스 프로그램을 통해서 작성 할 수 있습니다.



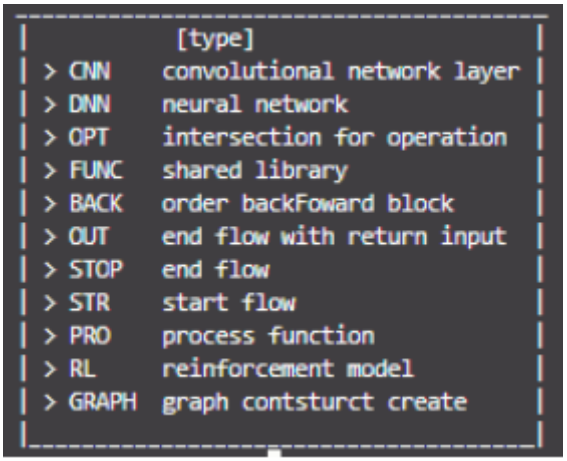
--help : 기본 제공되는 기능을 살펴 볼 수 있습니다.
--license : 라이센스 정보를 표시합니다.
--create "type": 콘솔인터페이스를 통해서 모델들을 작성할 수 있습니다.
--setting : 각종 설정을 조절합니다.

gpgpuon(off) : gpgpu 계산을 on off합니다 (1만개 이상일때 효과가 더 높습니다)
verboseon(off) : 자세한 설명을 추가로 보여줍니다. (터미널은 ASC 설정을 해주세요)
multion(off) : 멀티 쓰레드 환경을 허용합니다.
나머지 설정은 아직 테스트 중입니다.

--**run "**불러올 파일 이름**" "**최대 실행 횟수**" "**저장 파일이름**"** : 프로그램을 실행합니다.

불러올 파일 이름 : 반드시 그래프 흐름 파일이어야합니다.
최대 실행 횟수 : 프로그램의 최대 실행 횟수입니다.
저장 파일 이름 : 저장할 파일 이름을 의미합니다.

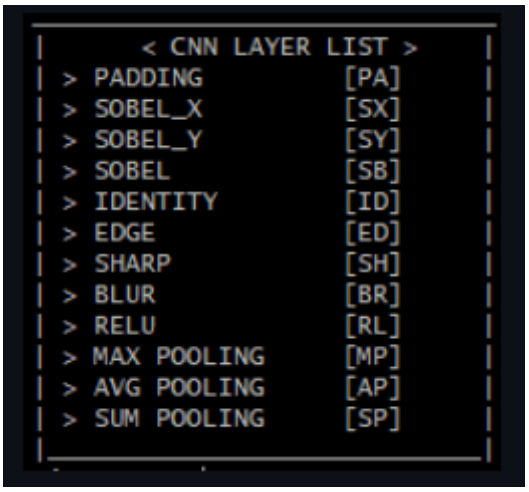
TYPE



CNN : Convolutional Neural Network 블록을 작성합니다.
NER : Deep Neural Network 블록을 작성합니다.
OPT : 사칙연산 블록을 작성합니다. (version 5 는 합과 곱만을 제공합니다.)
FUNC : 동적 라이브러리 블럭을 의미합니다 (런타임에 작동)
BACK: 연결된 부모블럭의 역전파를 실행합니다.
OUT: 그래프 흐름의 끝을 의미하는 블록이며 동시에 받은값을 넘겨줍니다.
STOP: 그래프 흐름의 끝을 의미하는 블록을 작성합니다.
STAT: 그래프 흐름의 시작을 의미하는 블록을 작성합니다.
PRO: 그래프 흐름을 제어할 블록을 작성합니다.
RL: 강화학습 모델 블록을 작성합니다.
RNN : 순환 학습 모델 블록을 작성합니다.
GRAPH: 그래프를 작성합니다 (비추천 프로그램을 실행하여 직접 GUI로 작성해주세요).

프로그램실행전 실제 사용되는 것은 CNN, NEW, OBS,FUNC 들입니다.
GRAPH, OPT, OUT, BACK, STR 은 GUI 프로그램을 통해서 작성가능합니다.

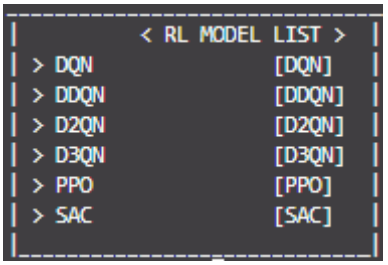
CNN 함수 옵션



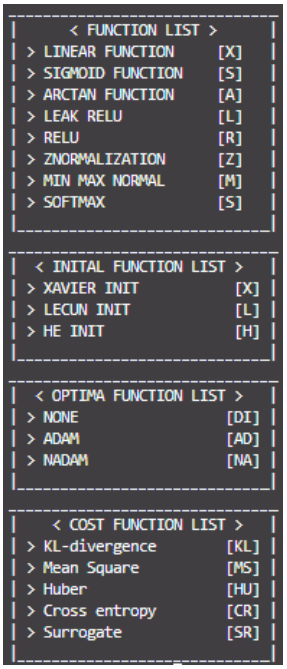
- 모든 *Padding*크기는 *1*입니다.
  PADDING : 0으로 페딩합니다.
- 모든 커널은 *3x3* 크기이며 *Stride* 는 *1*입니다.
  SOBEL_X : Sobel 마스크 X축방향입니다.
  SOBEL_Y : Sobel 마스크 Y축 방향입니다.
  IDENTITY : Identity 마스크
  EDGE : Edge 마스크
  SHARP : Sharp 마스크
  BLUR : Blur 마스크
- 모든 *Pooling*은 *2 x2* 크기 *Stride* 는 *2*입니다.
  MAX POOLING : max 풀링
  AVG POOLING: average 풀링
  SUM POOLING : summation 풀링

*이미지크기가 부족할 경우 이는 버려집니다.
예*)4x4* 이미지 *3x3* 커널은 오른쪽 과 아래 행렬은 무시됩니다.

강화학습 모델 옵션



뉴럴 네트워크 옵션



활성화 함수 및 정규화 함수

LINEAR FUNCTION : f(x) = x

SIGOMID FUNCTION : sigomid (x)

ARCTAN FUNCTION : arctan(x)

LEAK RELU : f(x) = max(x,0.3)

RELU : f(x) = max(x,0)

ZNORMALIZATION : z-norm function

MIN MAX NORMAL : min max norm function if min == max it return 0

SOFTMAX : softmax(x)

초기화 옵션

XAVIER INIT : xavier uniformal initialization

LECUN INIT : lecun uniformal initialization

HE INIT : He uniformal intialization

역전파 최적화 옵션

NONE : 옵션을 설정하지 않습니다.

ADAM : ADAM 옵션을 설정합니다.

NADAM : Nestrov adam 옵션을 설정합니다.

비용함수 옵션

KL-divergence : KL divergence 두 확률분포의 모양 차이를 계산합니다.

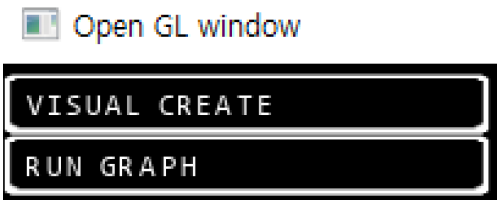
Mean square : mean square 이며 모집단의 수대신 1/2을 사용

Surrogate : 모델의 자체적인 비용함수를 사용합니다.

HUBER : Huber 절대값 1기준으로 L1,L2를 계산합니다.

Cross Entropy : 두 확률분포도의 차이를 계산합니다.

그래픽 인터페이스 설명



VISUAL CREATE : 제어흐름도를 작성하기 위한 프로그램을 실행합니다.
RUN GRAPH : 그래프 파일을 동작하기 위한 프로그램을 실행합니다.

VISUAL CREATE

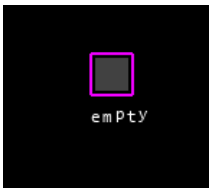
먼저 블록이란 각 알고리즘을 실행할 단위입니다.
본 머신러닝 엔진프로그램은 이러한 블록들의 관계를 통해 제어흐름을 조절하고 보다 쉽게 인공지능의 다양한 알고리즘 구현에 있어서 관계도를 통하여 작성하기 쉽도록 구성되어있습니다.

ADD : 블록을 추가합니다.
SAVE : 현제 선택된 블록과 그 블록의 자식블록들을 모두 저장합니다.
LOAD : 그래프 파일을 불러옵니다.

사용법

1. ADD를 클릭하면 화면에 empty라는 블록이 생성됩니다.

2. 블록을 왼쪽 클릭하면 블록주변에 테두리가 나타나며 현제 선택된 블록을 나타냅니다.

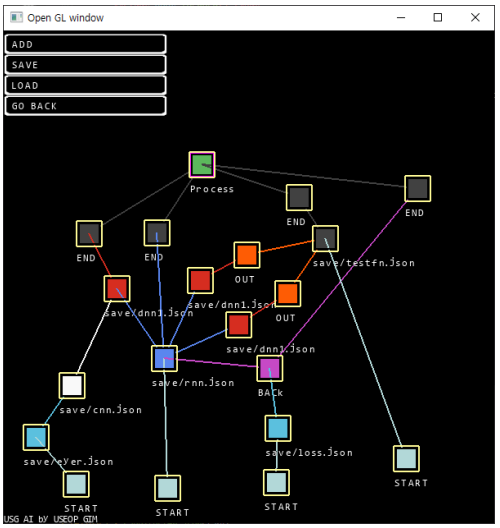


3. 블록 오른쪽 클릭하면 파일이름을 입력할 수 있습니다.

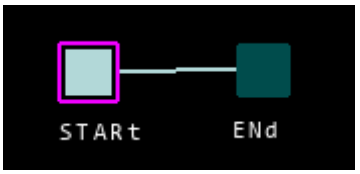
파일명 입력시 “+”,”x”, “OUT”,” START”, “STOP”, “PROCESS”, “BACK” 들은 파일이 없어도 각각의 블록들로 자동으로 변하게됩니다.

4. 왼쪽 클릭 후 다른 블록을 클릭하게 되면 각 블록끼리 연결되며 선택했던 블록이 자식 블록이 되고 마지막에 클릭한 블록이 부모 블록이 됩니다.

5. 부모 블록을 클릭시 연결된 모든 블록을 표시해줍니다

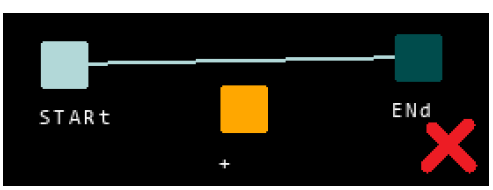


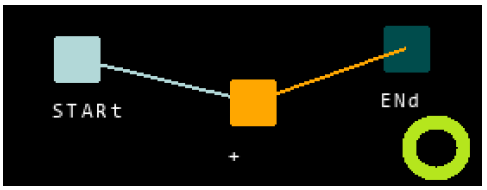
6. 같은 블록을 두번 왼쪽 클릭시 해당 블록에 연결된 모든 관계가 제거됩니다.



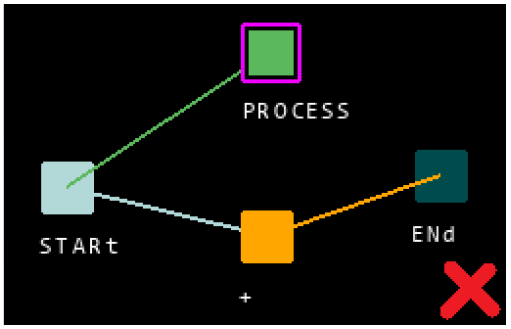
관계도 설명

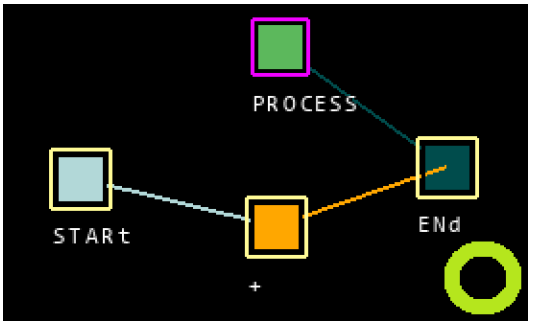
1. 프로그램의 실행과 끝을 의미하는 START 와 END 블록은 반드시 각 블록관계의 처음과 끝에 존재하며, 쌍을 이루어여합니다.



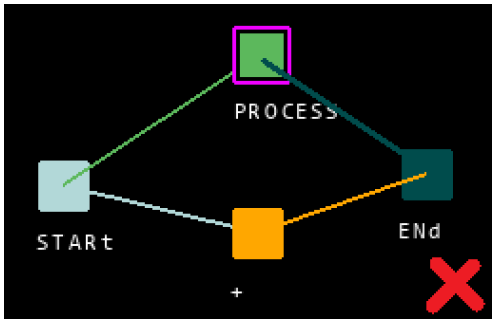


2. 프로그램의 실행순서를 제어할 PROCESS블록이 END블록의 부모블록으로 존재해야합니다.

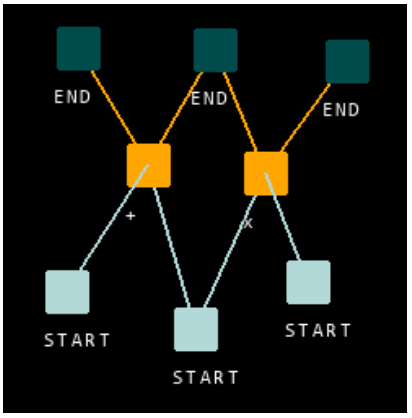




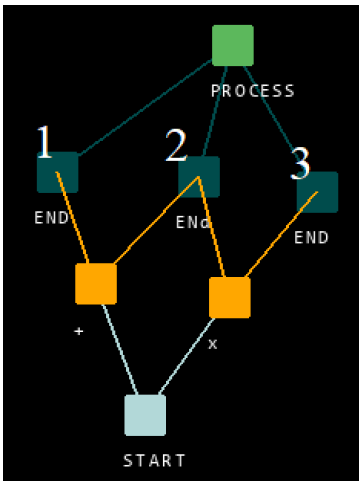
주의 : 부모가 다시 자식으로 연결되는 순환구조는 프로그램을 강제종료시킵니다.



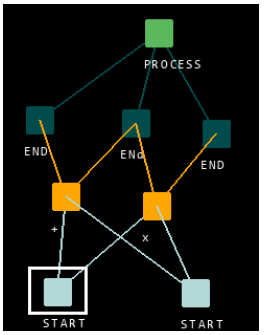
3. 한블록이 가질 수 있는 부모와 자식의 수는 제한이 없습니다.



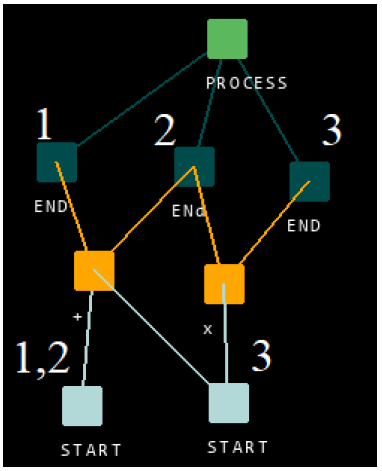
4. 프로그램의 실행 순서는 PROCESS에 연결된 END블록중 가장 왼쪽부터 실행됩니다.



5. 마찬가지로 END블록은 자식 블록중 가장 왼쪽의 START블록으로부터 시작합니다

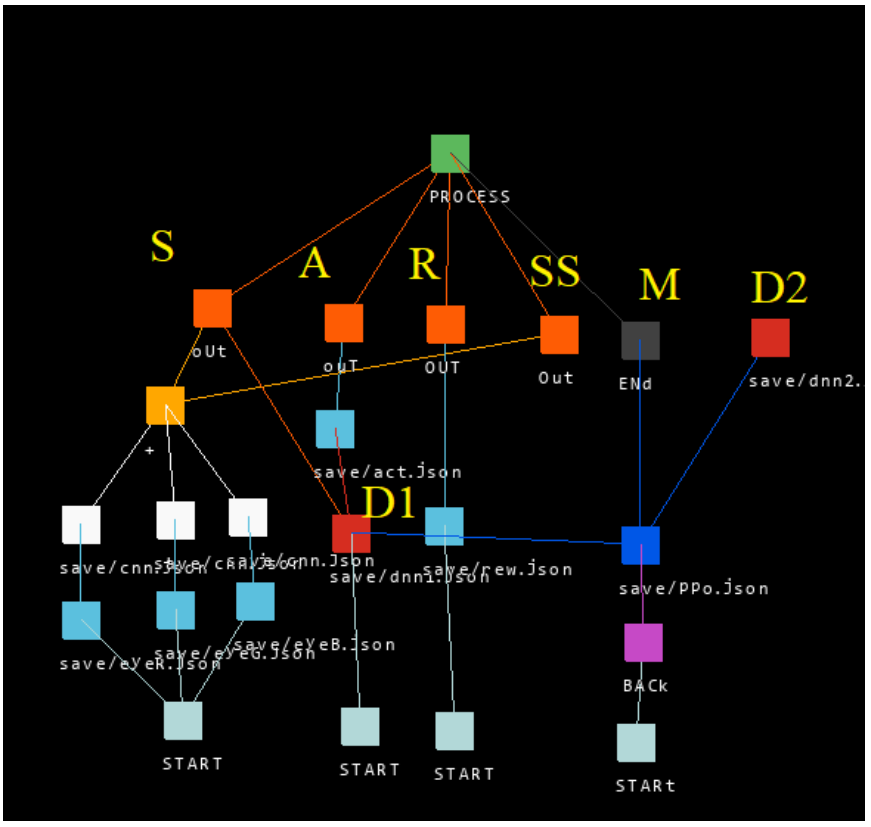


위와 같은경우는 가장 왼쪽의 START만 작동합니다.



위와 같은경우는 1번째 2번째 END 블록은 가장 왼쪽의 START와 연결되며 3번째 가장 오른쪽 START와 연결됩니다.

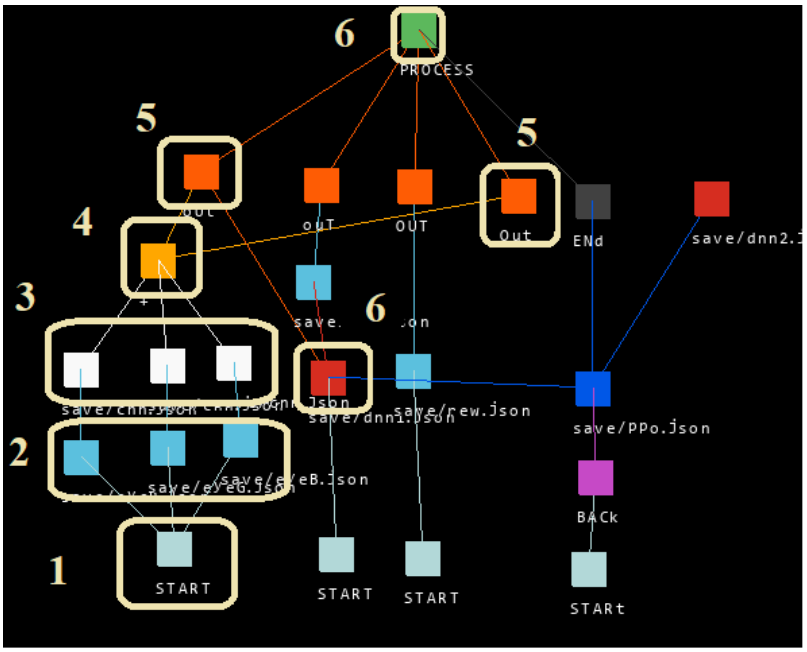
강화학습 모델 예시



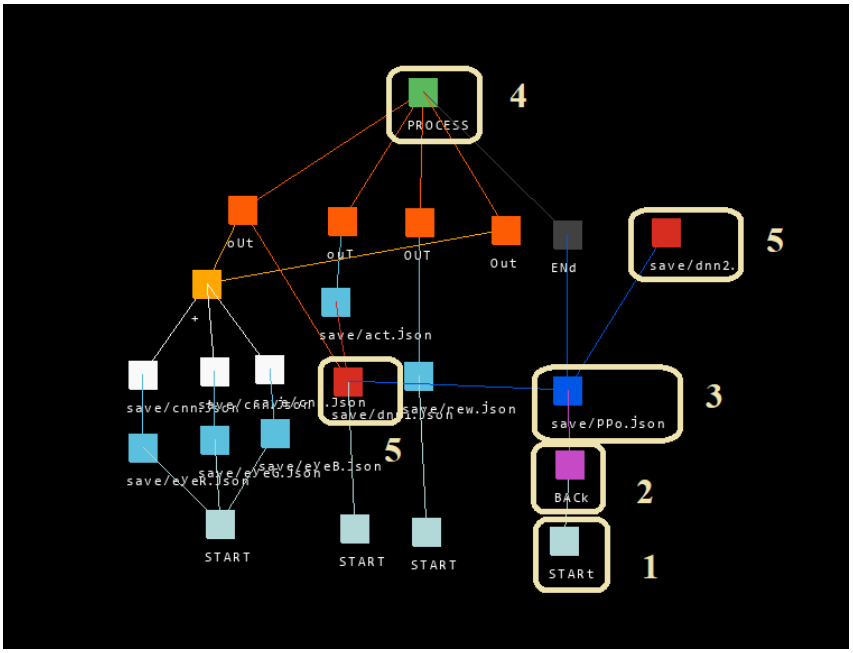
- S : 상태 값을 가져옵니다.
- A : 행동 값을 가져옵니다.
- R: 보상 값을 가져옵니다.
- SS': 변화된 상태값을 가져옵니다.
- M : 모델에게 명령을 내립니다.
- 강화학습 블럭의 뉴럴네트워크 중 가장 왼쪽의 뉴럴 네트워크블럭(D1)이 메인모델이됩니다. (실제 행동을 결정하는 네트워크)

예제) save/test.txt

흐름도 예시

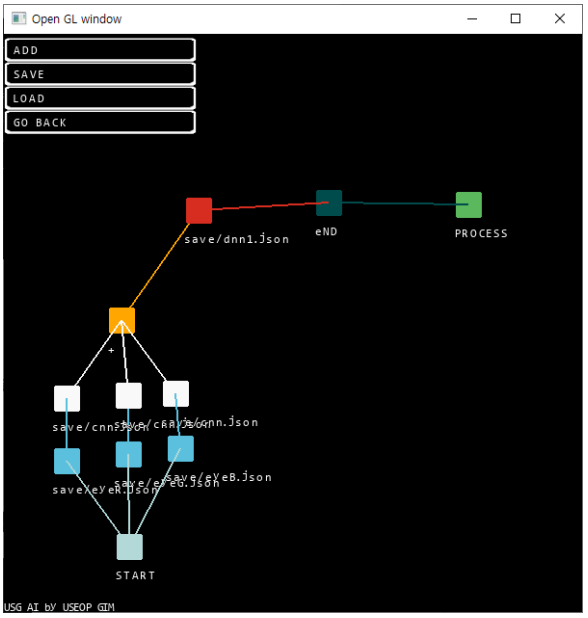


1. PROCESS 블록에서 START 부터 시작하라는 명령을 받습니다.
2. START 블록에서 EYE 블록인 DLL 명령어를 실행하여 화면값을 읽어옵니다.
3. EYE블록에서 각각의 CNN 블록으로 값을 전달해줍니다.
4. CNN블록에서 OPEARION + 블록을 통해 모든 값들을 합쳐 더해줍니다.
5. OPERATION + 에서 OUT 블록으로 값을 전달합니다.
6. OUT 블록은 뉴럴 네트워크 Dnn1 과 PROCESS로 값을 전달합니다.



1. PROCESS에서 START부터 시작하라는 명령을 받습니다.
2. START에서 BACK블록을 실행합니다.
3. BACK에서 RL 블록인 PPO 블록에게 BACKWARD명령어를 전달합니다.
4. RL블록인 PPO는 PROCESS에 저장된값들을 가져와 리플레이에 저장합니다.
5. RL블록은 리플레이에 저장된 값들이 MINI 배치 크기 이상일경우 DNN1, DNN2를 업데이트합니다.

실제 시뮬레이션시 작성 모양



위와 같이 학습후 저장된 뉴럴네트워크 블럭을 연결하여 process를 직접 실행하시면 됩니다.
예제) save/simul.txt

공유 라이브러리 함수 모양

```
extern "C" __declspec(dllexport) void rewardfn(std::vector<float> &src, std::vector<float> &dist)
```

공유함수의 모양은 위와 같아야하며.

```
{
"FUNC":{
        "lib":["extrafn.dll"],
        "func":["statefn"],
        "pre_Input":[0, 0, 400, 400, 96, 96, 1]
    }
}
```

실행 파일 포맷은

1. FUNC : 태그 이름
2. LIB : 공유라이브러리 위치 이름 (Linux 는 확장자명 a)
3. FUNC : 공유 라이브러리 속 함수이름
4. PRE_INPUT : 초기설정값(만약 입력이 없으면 변하지 않는 초기 값)

위와 같은 형식의 동적 라이브러리 함수는 무엇이는 실행이 됩니다.
실제 함수 작성시 예제) src/extrafn.cpp 를 참고해주십시오