# 게임 행동 제어

AI를 포함한 세부 논리 제어 자체를 데이터화 할 필요가 있다. 빠른 개발을 위해 필요하다. 특히, C++를 서버의 주 언어로 할 때 더욱 필요하다. 견고하고 빠른 기반을 갖추고 유연성은 데이터로 갖춘다.

일반적인 접근은 스크립트 언어를 사용하는 것이다. 성능만 해결되면 나머지는 쉽다.

## 방향

- 빠른 스크립트 언어
- 쉬운 스크립트 언어

스크립트 실행과 c++ / 스크립트 간 호출 속도가 빨라야 한다. 스크립트 언어의 문법이 명확하고 C++ 연동이 쉬워야 한다.

### 빠름

최종적인 실행이 빠르면 된다. traspiler를 통해 C++로 변환될 수 있으면 좋다. 그렇지 않을 경우 jit나 컴파일을 통해 빠른 코드로 변환되면 된다. lua의 사용 경험으로 보면 주기적인 호출이 많을 경우 C++ 만큼 빠르지 않으며 스크립트와 C++, 또는 C++에서 스크립트 함수 호출이 느린 것으로 보인다.

실제 측정을 통해 확인해야 한다. 알고리즘이나 사용 형태의 조절로 빠르게 만들 수 있는 지 확인한다.

### 쉬움

lua의 경우 쉬운 연동 방법들이 MPL 기법을 사용하여 마련되어 있다. 이 보다 더 쉬운 방법이 있을 수 있는가? 그냥 C++ 오브젝트와 함수 호출이 가능한 스크립트를 만든다면 더 쉬울 수 있다.

언어 개발은 생각보다 어려운 작업인데 이미 확립된 방법을 사용하는 게 나을 수 있다.

# 언어의 개발

cling 스크립트는 c++ 언어이므로 바로 컴파일 가능하다. 인터프리트 될 때 C++ 오브젝트를 사용 가능하면 된다. Visual C++ 과 얼마나 부드럽게 연동 가능한 지가 중요하다. 실제 실행은 빌드에 포함되므로 빠르게 처리 가능하다. 인터프리터 디버깅이 쉬워야 한다.

## 기능

- 연동
  - C struct와 함수의 직접 사용
    - 또는 미리 정의된 C++ 오브젝트와 함수 사용
  - 스크립트 struct와 함수의 직접 호출

        - 어떻게??
- 빠름
    - 머신 코드 생성
    - 링크 또는 DLL 로딩
- 단순함
    - 최대한 단순한 의미를 갖는 코드
    - 함수 단위
    - if, 반복, 호출, 결과, 지역 변수

C++의 부분 언어와 호환 가능한 컴파일 되고 DLL로 로딩 되는 스크립트 언어. 중간에 C++로 변환 후 빌드 될 수 있으며 원래 코드의 의미를 유지한 형태로 C++ 코드가 생성된다.

## 연동

스크립트 코드를 어셈블리 수준에서 생각한다. 여기서 함수 호출과 결과 값 처리 인터페이스를 생각한다. C++로 전환되어 실행되는 흐름은 단순하다. dll 로딩으로 처리한다. Task.Function( 오브젝트, 아규먼트들 ) 로 처리한다.

스크립트로 실행될 때 흐름이 어렵다. C++ 스크립트이다. 따라서, C++ 스크립트 언어로 시작해야 한다.

# cling 테스트

생각하는 언어가 cling에 가깝다. cling을 사용하면서 언어에 제약을 준 것과 동일하다.

사용방법을 정리하고 레퍼런스 위주로 하면 빠르게 개발하면서 확인할 수 있다. C++ 프로그래머에게 날개를 달아줄 수 있다.

## 목표

- 전체 기능의 이해
- 적용 가능성
    - 설치가 쉬워야 함
        - 바이너리 배포로 사용 가능하게 만듦
    - 사용이 쉬워야 함
        - 스크립트처럼 동작
        - How?
        - C++ transpiler
- 다른 가능성 검토
    - visual c++과 hot reloading

# 설치

# 테스트

# 연동

# 평가

# 자료

## 윈도우에서 빌드 가능한가?

https://indico.cern.ch/event/543158/contributions/2206091/attachments/1291866/1924498/Cling_on_Windows.pdf

- cling on windows
    - visual studio 2015

https://github.com/root-project/cling/issues/186

- VS 2015
- 완료되었다고 한다
- 확인이 필요

## 다른 접근

https://fungos.github.io/blog/2017/11/20/cr.h-a-simple-c-hot-reload-header-only-library/

- cr.h
- 토이 / 연습용으로 개발
- like C-Toy

    - tcc based c scripting environment

# Ravi 테스트

luajit 만큼 빠르지는 않다고 하나 LLVM을 사용한 Ravi가 있다. 흔히 실수 하는 로컬 변수와 같은 개선을 포함하고 있다. lua와 호환되므로 기존의 바인딩 라이브러리를 사용할 수 있다. 빠른 바인딩이 중요하다.

## 루아 C++ 바인딩

https://github.com/SteveKChiu/lua-intf

https://github.com/jeremyong/Selene

http://oolua.org

http://cppcast.com/2016/07/elias-daler/

https://github.com/ThePhD/sol2

- claims the fastest binding

c 호출보다 빠르다고 하니 Ravi와 함께 사용해 보면 결과를 알 수 있을 듯 하다. 디버깅을 가능하게 하고 미리 컴파일 해 놓고 다른 문제 없는 지 철저하게 테스트 하면 된다.

## 주의할 점들

- 성능
- 메모리 관리
    - 수명 관리

## 목표

- 개선된 lua 기반 스크립팅 기능 확보
- 스크립트 관리
    - 스크립트 빌드 캐싱 (바이너리 로딩)
    - 변경시 빌드 후 캐싱
- 바인딩 안정성
    - 메모리 관리
- 디버깅
    - 리모트 소스 디버깅 기능 확보

- 성능 측정
- 개선 작업

깔끔한 사용법 위주로 정리하고 성능을 미리 측정해 둔다. 작은 게임들은 lua 기반으로 작성할 수도 있다.

# 데이터 기반 접근

조건을 포함한 데이터를 외부 파일로 내보내고 C++로 코드를 구현한다. VM이 아닌 데이터 기반으로 코드를 제어한다. 일반적이지 않지만 대부분의 케이스는 적용 가능하지 않을까?

## 아이디어

- boolean expression에 따른 분기
- 함수 호출과 결과 값 분기
- 메세지 함수

```
// how to configure?
// extend a language?

on MsgGameStageChanged
{
    if ( self.GetStage() == 4 )
    {
        self.ChangeSkill( 10004 );
        self.RestartBattle();
    }
}
```

메세지 함수들이고 데이터는 별 상관 없다. C++ 코드이다. cling처럼 실행 가능하면 되는데 코드를 생성하고 로딩하는 방식을 사용한다. msg와 self가 기본으로 주어진다.

```
stage[4] = {
    skill: 1004,
    restart: true
}
```

데이터에 기반할 경우 항상 전체 동작을 정의해야 한다. 예외를 수용하기가 어려워진다. 이것이 단점이다.

# 자료

https://github.com/bitfunnel/nativejit/

- Bing 프로젝트의 부산물
- EDSL의 Jit 컴파일

https://root.cern.ch/cling

- clang / llvm 기반 C++ interpreter
- jit
- https://github.com/vgvassilev/cling
- https://cdn.rawgit.com/root-project/cling/master/www/index.html
- 결국 컴파일 시간의 문제는 남을 듯 하다
    - 방대해진다.

https://www.youtube.com/watch?v=viVroGmeQDw

- C++ scripting

http://wren.io/

- 또 다른 스크립트 언어
- C++ 연동이 쉽지는 않다