Windows Unreal Engine 개발자의 Neovim

이효승

Windows Unreal Engine 에서 왜 Neovim?

- Vim이 좋아요!
- 아는 것만 쓰고 싶어요!
- 코드만 보고 싶어요!

Neovim으로 UE에서 무엇을 할 수 있을까요?

->Editting Yes! Debug No!

- 편집 환경은 Visual Studio보다 neovim이 더 넓고 생산성 plugin이 많이 지원됨
- 디버깅 지원은 Unreal Engine에서 visual studio를 최우선 지원하고 있음
- Multithreading 환경의 디버깅은 visual studio가 압도적으로 편리함

목표 - UE 에서 Neovim LSP 사용하기

Windows에서 Neovim 설치 & Setting

- `winget` 이용

```
winget install neovim
```

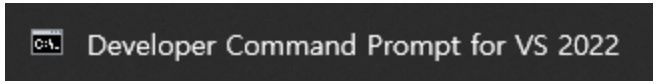
- config: `%AppData%\Local\nvim\init.lua`

Windows에서는 GUI로: Neovim-QT

- https://github.com/equalsraf/neovim-qt
- Neovim GUI frontend
- Neovim 설치 시 기본 설치됨
- 주요 기능
  - drag & drop
  - 한-영 전환 (neovide는 아직.....)
  - scrolling
  - 창 투명도 조절, 폰트 조절

Windows 환경 내 Neovim Plugin 설정

- 일부 source build가 필요한 plugin이 있어 환경 설정을 해야 함
- Visual Studio 설치
- cmake 설치
- 이후 plugin 설치 작업은 Developer Command Prompt for VS 2022 환경에서 진행



Windows 환경 내 Plugin 설정

Telescope

- Telescope의 dependency인 fzf 를 build해야 함
- 기본 neovim config내의 build script는 UNIX make 사용을 전제로 작성되어 있음 -> cmake로 변경

Treesitter

- Treesitter의 parser를 build 해야 함
- Developer Command Prompt 환경 내에서 Neovim 실행 시 자동 build 진행

LSP - Clangd

- Clang project의 LSP
- UE도 linux build를 clang으로 하기 때문에 호환된다.
    - windows build도 clang으로 할 수 있다
- LSP option

```
cmd = { 'clangd',
        '-j=32', # thread number
        '--pch-storage=disk', #PCH storage option (disk, memory)
        '--completion-style=detailed' #completion style
}
```

수동으로는 불가능한 UE Project Setting

- 모듈 별 dependency
  - 모듈 간 dependency 정의가 모듈마다 따로 있음
- 자동 생성 source
  - Garbage collection, Reflection 등 때문에 build system에서 자동으로 생성하는 소스 존재
- 수많은 Macro
  - 최적화, build target 별 build mode 변경을 위한 macro가 상당히 많음

Unreal Build Tool (UBT)

- UE의 개발 환경 설정에 필요한 전반적인 기능을 다루고 있음
- Build 작업 뿐 아니라 project file 자동 생성 역시 담당하고 있음

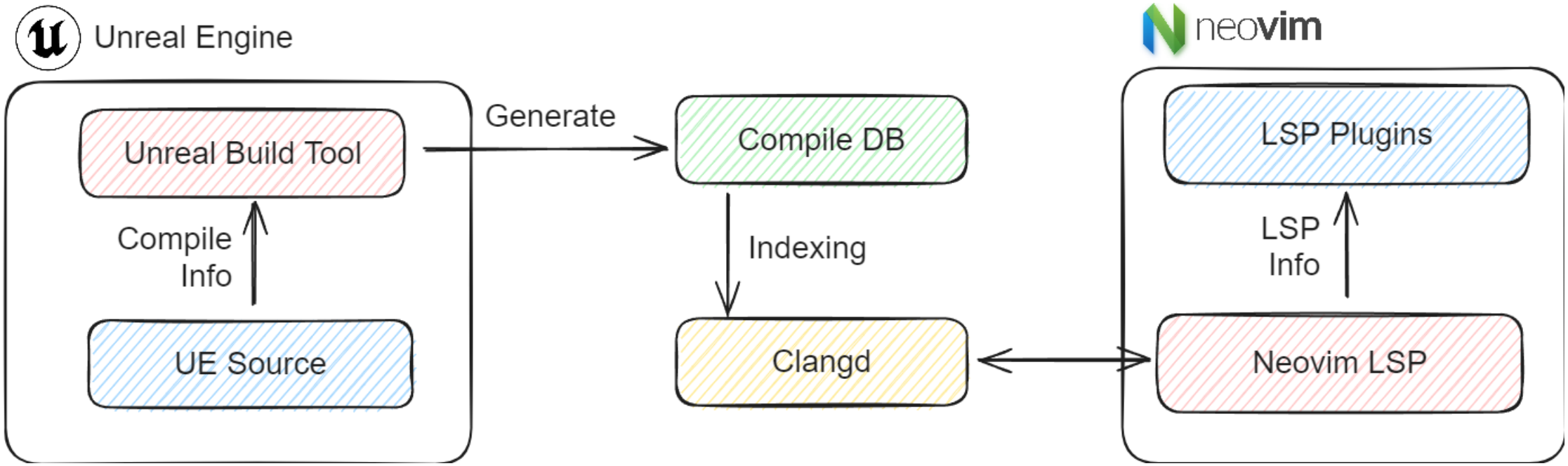
Compile Database

- compile시 명령어 정보를 담고 있는 json 형태 DB
- 일반적으로 `compile_commands.json` 파일명을 가지고 있음
- compile시 source, working directory, compile command 정보를 담고 있음
- compile database가 있을 경우 clangd에서 이를 인식하고 indexing을 진행함

```
[
  { "directory": "/home/user/llvm/build",
    "arguments": ["/usr/bin/clang++", "-Irelative", "-DSOMEDEF=With spaces,
    quotes and \-es.", "-c", "-o", "file.o", "file.cc"],
    "file": "file.cc" },

  { "directory": "/home/user/llvm/build",
    "command": "/usr/bin/clang++ -Irelative -DSOMEDEF=\"With spaces,
    quotes and \-es.\" -c -o file.o file.cc",
    "file": "file2.cc" },
  ...
]
```

UE LSP 환경 설정 구조



Compile Database 생성

- UBT에서 build시 사용되는 명령어를 기준으로 compile database를 생성함
- 에디터 용 (e.g. Project name: TestGame -> TestGameEditor)

```
$ Engine\Binaries\DotNet\UnrealBuildTool\UnrealBuildTool.exe

{GameProject 이름}Editor Win64 DebugGame -project={.uproject 파일 경로} -game
-engine -mode=GenerateClangDatabase
```

- Development build 용 (e.g. Project name: TestGame -> TestGame)

```
$ Engine\Binaries\DotNet\UnrealBuildTool\UnrealBuildTool.exe

{GameProject 이름} Win64 DebugGame -project={.uproject 파일 경로} -game -engine
-mode=GenerateClangDatabase
```

UE + Neovim + Clangd 좋은가요?

-> 쓸만 하다, 하지만 완벽하진 않다

- Neovim의 생산성 tool은 매우 유용
- VS 대비 code에 몰입할 수 있는 환경
- 느린 최초 indexing
  - 최초 indexing 중 memory 사용량이 10GB 이상
  - 3시간 이상 소요 (Ryzen 9 5950X 기준)
- 일부 symbol indexing이 이루어지지 않음
- 현재는 main을 visual studio 로 사용하고 neovim을 side로 사용 중