박 정 민
포트폴리오

Client Programmer

010-6419-1616 drpdrum@gmail.com
https://github.com/drPDrum/UnrealPortfolio

포트폴리오 개요

목표
유니티 클라이언트 프로그래머로 일하면서 해왔던 일들을 언리얼 엔진을 이용해서 구현하기

Data Load 모듈, Data Tool
- Data Tool for Unreal
- DataLoadLib 모듈
- DataLoadEditor 모듈
- DataTool, Data Load 모듈 연동 시연 연상
- Unreal Automation Test를 이용한 모듈 Unit Test
- Data Load 모듈 Automation Test 시연 영상

기본 프레임워크 설계 : 공동작업
- Managers 설계
- DataFileManager, LocalizationManager
- UIManager
- FXManager
- SFXManager
- WorldManager, AFrameworkGameModeBase
- AFrameworkGameInst
- AFrameworkCamera
- InputController

게임 컨텐츠 설계 : 공동작업
- ASpawnPoint, AGameMode_Playable
- UserData
- AGameCharacter
- UAnimController, UAnimInst
- ABrainController
- UActionData, UActionController
- 콘텐츠 시연 영상

관련문서 모음

Data Load 모듈,
Data Tool

관련 문서 →

- 1_DataLoad Module - DataTool 사용법.pdf 기본 사용법 가이드 문서
- 2_DataLoad Module - Auto Generated Source Code.pdf 자동 생성되는 소스코드 설명 문서

Data Tool for Unreal

- 엑셀에서 작성된 기본 데이터, Const Data, Localization Data를 바이너리 파일로 변환
- 기본데이터, Enum, Const Data, Localization Data에서 런타임에 사용할 데이터 구조체 소스 코드 자동 생성
- 데이터 구조체의 base class 및 Factory, Containor, 관리 enum파일 소스 코드 자동 생성

관련 문서 →

- BinaryDataSource.h 바이너리 파일 정보를 uasset에 저장하는 UObject.
- DataFileManagerBase.h 데이터 매니저 Base class
- LZManagerBase.h Localization 매니저 Base class
- HowToUseInRuntimeModule.PNG 런타임 모듈에서 사용하는 코드 스샷

DataLoadLib 모듈

- 에디터에서 바이너리 데이터를 읽어 uasset에 저장한 후, 런타임에 해당 uasset을 불러 필요한 구조체에 저장 후 사용하는 모듈
- UBinaryDataSource - 바이너리 파일 정보를 uasset에 저장하는 UObject
  DataLoadEditor 모듈에서 바이트 파일을 읽어 해당 파일 UObject를 생성한다.
- DataFileManagerBase
  - → UBinaryDataSource를 런타임에 읽어, DataTool에서 생성한 CDataFileFactory를 이용하여 데이터를 세팅한다.
  - → 필요한 데이터를 제네릭 함수로 런타임 모듈에서 편하게 사용할 수 있게 설계
- LZManagerBase
  - → UBinaryDataSource를 런타임에 읽어, DataTool에서 생성한 CDataFileFactory를 이용하여 데이터를 세팅한다.
  - → [] 연산자 오버라이딩으로 런타임 모듈에서 편하게 사용할 수 있게 설계

Data Load 모듈,
Data Tool

DataLoadEditor 모듈

- 바이너리 데이터 파일을 읽어 언리얼에서 사용할 수 있게 uasset파일로 변환하는 모듈
- UBinaryDataFactory
  - → UFactory를 상속받아 UBinaryDataSource를 생성하고, 해당 UObject에 바이너리 파일정보를 전달한다.
  - → 언리얼 에디터 상에서 기본 import처리. (외부에서 파일 드래그 앤 드랍)
- UReimportBinaryDataFactory
  - → FReimportHandler를 상속받아 에디터에서 실시간 바이너리 데이터 변경 대응
  - → 언리얼 에디터 상에서 기본 reimport처리. (외부에서 파일 드래그 앤 드랍)
- CDirectoryStatVisitor
  - → IPlatformFile::FDirectoryStatVisitor를 상속받아 Visit함수 구현
  - → Visit함수 - 특정 폴더에 있는 파일 중 .bytes파일과 .uasset파일 정보를 CBinaryDataRegistry에 전송
- CBinaryDataRegistry
  - → 에디터 off상태 파일 변화 체크 기능
    - ✓ 에디터가 실행될 때 CDirectoryStatVisitor를 이용해서 .byte파일과 .uasset정보를 받아온다.
    - ✓ .uasset파일이 UBinaryDataSource형식의 UObject면 해당 파일과 연결되는 .byte파일과 수정 시간을 체크한다.
    - ✓ 에디터가 꺼져있을 때 신규 바이너리 파일이나, 기존 바이너리 파일이 변경됐는지 체크하여 리스트화 한다.
    - ✓ 엔진 Tick을 돌려 에디터가 완전히 켜지는 것을 기다린 후 해당 파일 변경사항을 노티한 후 UBinaryDataFactory를 이용하여 import처리한다.
  - → 에디터 on상태 파일 변화 체크 기능
    - ✓ "DirectoryWatcher"모듈을 사용하여 컨텐츠 폴더에 변화가 생겼을 시에 콜백을 등록한다.
    - ✓ 해당 콜백 함수가 호출되면 변경사항을 노티한 후 UBinaryDataFactory를 이용하여 import처리한다.

관련 문서 →

- BinaryDataFactory.h
- ReimportBinaryDataFactory.h
- CDirectoryStatVisitor.h
- CBinaryDataRegistry.h

Data Load 모듈,
Data Tool

관련 문서 →

시연 영상 : DataTool, Data Load Module 시연

DataTool, Data Load 모듈 연동 시연 연상

관련 문서 →

DataLoadEditorTest.cpp .byte, .uasset파일 변경 체크 스크립트
DataLoadTest.cpp 데이터로드시간, 유효성 체크 스크립트
BinaryDataTest.bat 외부실행배치파일

Unreal Automation Test를 이용한 모듈 Unit Test

- .byte파일과 .uasset 파일의 변경체크를 해서, 현재 .byte파일이 엔진에 최신으로 적용되어 있는지 체크
- 엔진에 적용되어 있는 모든 UBinaryDataSource의 로드타임 계산
- 모든 UBinaryDataSource 의 키값을 계산하여 유효한 데이터인지 체크
- CI머신이나 에디터 외부에서 테스트를 실행할 수 있게 배치파일 작성

관련 문서 →

시연 영상 : Data Load Module Automation Test 시연

Data Load 모듈 Automation Test 시연 연상

기본 프레임워크 설계

관련 문서 →

- Managers.h, Managers.cpp Managers Class

Managers 설계

- 여러 Manager Class들을 한 곳으로 모아 관리하기 편하게 해주는 매니저 관리 구조체
- Managers라는 하나의 Class에서, 각 매니저들을 Managers의 Static변수로 설정하여 전역접근이 가능하게 설계

관련 문서 →

- DataFileManagerBase.h, DataFileManagerBase.cpp DataFileManager Base Class
- LZManagerBase.h, LZManagerBase.cpp LocalizationManager Base Class
- DataFileManager.h, DataFileManager.cpp Managers에서 사용하는 DataFileManager
- LocalizationManager.h, LocalizationManager.cpp Managers에서 사용하는 Localization Manager

DataFileManager, LocalizationManager

- 바이너리 데이터, 일반 데이터 에셋, Localization Data를 관리하는 매니저
- DataLoadLib 모듈의 DataFileManagerBase, LZManagerBase를 상속받아 구현
- Managers에서 관리 가능하게 IManager.h 이중상속으로 구현
- UBinaryDataSource를 이용한 데이터 구조화 및 관리는 DataFileManagerBase, LZManagerBase에 구현되어 있다.
- 이외에 다른 데이터 에셋들을 사용할 구조 설계

기본 프레임워크 설계

UIManager

- 런타임에서 사용하는 UMG관련 관리 매니저
- 각 UI Page를 Window라 정의하고, Window의 기본 스택 관리와 Window 내부 UIModule들의 스택 관리 시스템 구현
- 레벨 변경 시 Widget이 자동 해제되지 않게 ImmotalWidget을 설계하고, Window, UIModule들이 ImmotalWidget을 상속받아 구현
- UI 스택은 필요할 때 정리할 수 있게 설계
- 공용 Popup, 로딩 페이지, Window 뒷배경 및 Block 처리 등 UI 기능 구현
  Window마다 개별 KeyMapping을 할 수 있게 하여, 유저 입력 시스템 제어

관련 문서 →

- UIManager.h 기본 매니저 header
- UIManager.cpp Window Stack 관리, 뒷배경 Block 처리
- UIManager+Loading.cpp 로딩 페이지 관리
- UIManager+Module.cpp Window 내부 UIModule 스택관리
- UIManager+Popup.cpp 공용 팝업 관리
- ImmortalWidget.h, ImmotalWidget.cpp 레벨 변경 시에 사라지지 않는 Widget base class
- LoadingBase.h, LoadingBase.cpp 로딩 페이지 base class
- WindowBase.h, WindowBase.cpp Window base class, 런타임에 스택 관리될 때 처리되는 가상 함수 제공
- WindowBase+KeyMapping.cpp Window마다 개별 KeyMapping 처리
- WindowBase+Module.cpp Window내부 UIModule 스택관리
- UIModuleBase.h, UIModuleBase.cpp UIModule base class
- Window_SystemPopup.h, Window_SystemPopup.cpp 공용 팝업

기본 프레임워크 설계

관련 문서 →

- FXManager.h, FXManager.cpp 기본 매니저. FXActor별 FXContainer class를 이용해서 Pooling 관리
- FXActor.h, FXActor.cpp 개별 Effect Object class. 내부적으로 ParticleSystem, NiagaraComponent, GhostComponent 지원

FXManager

- Effect관련 관리 매니저
- 각 Effect별 Object Pool을 관리, Effect 필요하기 전에 예약 및 등록하고 씬 전환 시 풀 정리
- 개별 Effect에서 재생 종료 후 자동으로 Pool로 리턴
- ParticleSystem, NiagaraComponent, GhostComponent 지원

관련 문서 →

- SFXManager.h, SFXManager.cpp 기본 매니저. 하나의 Pool로 관리
- SFXActor.h, SFXActor.cpp 개별 Sound Effect Object class.

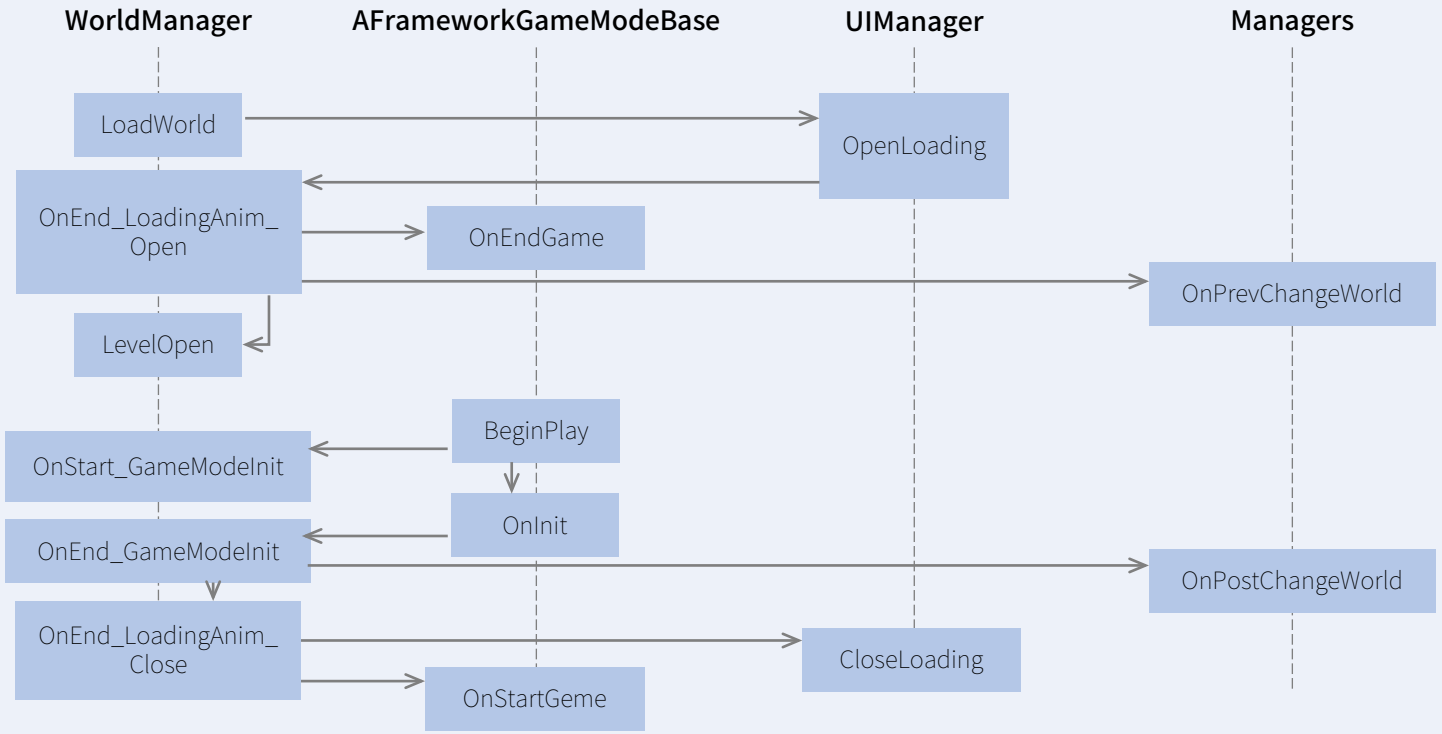
SFXManager

- Sound Effect관련 관리 매니저
- 하나의 Object Pool에서 32개의 Sound Effect를 미리 생성해두고 관리

기본 프레임워크 설계

WorldManager, AFrameworkGameModeBase

- WorldManager - Level 전환, LevelStreaming, 캐릭터 스폰 관리 매니저
- AFrameworkGameModeBase - 중심 Level에 설정된 GameMode base class
- Level 전환 플로우



- 레벨로딩 과정에서 Steaming Level 관련데이터 처리 및 실제 로드 담당

관련 문서 →

- WorldManager.h, WorldManager.cpp 기본 매니저. 레벨 기본 로딩 처리 적용
- WorldManager+LevelStreaming.cpp Level Streaming
- FrameworkGameModeBase.h, FrameworkGameModeBase.cpp 레벨 로딩처리 연동되어 있는 기본 GameMode base class

기본 프레임워크 설계

관련 문서 →

FrameworkGameInst.h, FrameworkGameInst.cpp

AFrameworkGameInst

- 기본 Game Instance
- 인게임 내에서 전역처리 해야 할 UI Widget, Material 등을 관리

관련 문서 →

FrameworkCamera.h, FrameworkCamera.cpp

AFrameworkCamera

- AFrameworkInputController에 빙의되는 Character 전용 카메라
- Body, Neck, Head 3단 구조로 자연스러운 카메라 무빙 적용

관련 문서 →

FrameworkInputController.h, FrameworkInputController.cpp
base class
GameCharInputController.h, GameCharInputController.cpp

InputController

- AFrameworkInputController
  - → 유저 입력을 다양하게 컨트롤 하기 위한 PlayerController
  - → WindowBase+KeyMapping.cpp과 연계하여 UI 입력 처리도 동시 진행
- AGameCharInputController
  - → AFrameworkInputController를 상속받아 키 이벤트 처리
  - → ABrainController에 키 이벤트 입력 전송
  - → 빙의, 빙의 해제 처리

ASpawnPoint, AGameMode_Playable

- ASpawnPoint
  - → 레벨에 배치하는 액터로 SceneComponent는 배치할 캐릭터의 Location, Rotation 정보를 갖는다.
  - → PROPERTY 설정으로 이전 월드에 따라 배치 변경, 스트리밍 레벨 처리 시 캐릭터 Active/Inactive 기능 등을 한다.
  - → SplineComponent를 이용하여 몬스터 캐릭터의 패트롤을 설정할 수 있다.
- AGameMode_Playable
  - → AFrameworkGameModeBase를 상속받아 구현
  - → 초기화 단계에서 ASpawnPoint를 이용하여 캐릭터 스폰, 몬스터 스폰 데이터를 생성한다.
  - → 스폰 데이터를 기반으로 WorldManager를 이용하여 초기 캐릭터 위치에 따른 스트리밍 레벨처리를 진행한다.
  - → 스트리밍 레벨 처리가 종료되면 팀 세팅, 캐릭터 스폰, 카메라, Input설정, 스트리밍 레벨에 따른 몬스터 스폰을 진행하고 초기화 단계를 종료한다.
  - → OnStartGame overriding 함수를 이용해 캐릭터 Active, 몬스터 AI시작 등을 진행한다.

관련 문서 →

- SpawnPoint.h, SpawnPoint.cpp
- SpawnPoint+Character.cpp 캐릭터 스폰, 활성화 비활성화 처리
- GameMode_Playable.h, GameMode_Playable.cpp

게임 콘텐츠

관련 문서 →

- UserInfo.h, UserInfo.cpp 관리 Info class
- UserCharacterInfo.h, UserCharacterInfo.cpp Character 관련 UserData 관리 Info class
- CUserCharData.h, CUserCharData.cpp User Character Data
- CUserItemData.h, CUserItemData.cpp User Item Data

UserData

- User Play Data 관리 클래스
- UserInfo라는 하나의 Class에서, 각 UserData를 관리하는 Info Class들을 Static변수로 설정하여 전역접근이 가능하게 설계
- 서버가 붙으면, 서버 정보와 결합하여 사용

관련 문서 →

- GameCharacter.h, GameCharacter.cpp 기본 캐릭터 클래스. 기본 Getter 함수들, Hit Navi Location 처리 등 담당
- GameCharacter+Init.cpp UserData를 기반으로 초기화 처리
- GameCharacter+Team.cpp 팀 처리 및 컬리전 프로파일 처리

AGameCharacter

- 게임 캐릭터 클래스
- CUserCharData를 기반으로, CCharacterData를 이용해서 캐릭터 기본 설정, CStatData를 이용해서 스텟 설정
- ABrainController를 이용하여 AAIController, APlayerController간 자유롭게 빙의, 빙의해제 할 수있게 설계
- UActionData와 이를 관리하는 UActionController를 이용해서 캐릭터의 모든 액션 처리 설계
- UAnimController를 이용하여 UAnimInst를 상속받은 Blueprint Animation class 제어
- GameMode에서 설정한 팀 정보를 기반으로 Collision Profile 설정

게임 콘텐츠

관련 문서 →

- AnimController.h, AnimController.cpp
- AnimInst.h, AnimInst.cpp
- AnimInst_Blueprint.PNG AnimInst를 상속받은 애니메이션 블루프린트

UAnimController, UAnimInst

- UAnimController - UAnimInst 컨트롤러, 기본 Locomotion을 제외하고는 Animation Montage로 재생하게끔 설계
- UAnimInst - AGameCharacter로부터 Tick을 받아 Locomotion 관련 처리, Montage 재생

관련 문서 →

- BrainController.h, BrainController.cpp 기본 브레인 클래스, 기본 Getter 함수들, 기본 캐릭터 방향설정 및 타겟 설정
- BrainController+AI.cpp AI 행동 제어 클래스
- BrainController+Input.cpp 유저 입력 제어 클래스
- AIBehaviorTree.PNG AIBehaviorTree
- BTTask_Patrol.h, BTTask_Patrol.cpp 패트롤 관련 처리 base class. 이동 중 탐색 처리
- BTTask_PatrolLoop.h, BTTask_PatrolLoop.cpp 패트롤 중 한방향 반복 이동 처리

ABrainController

- 팀정보를 이용해서 아군과 적군을 판단하여, 주변 탐색 및 타겟 설정 등 어떤 액션을 하기 위한 근거를 판단하는 두뇌 클래스
- AITree, 입력 등으로부터 판단의 근거를 만들어 ActionController에서 액션을 행하게 한다.
- AAIController를 상속받았으며, APlayerController에 빙의된 캐릭터면 유동적으로 AAIController를 빙의해제하게 설계
- Spawn 데이터를 이용한 AI 행동 제어

게임 콘텐츠

관련 문서 →

- ActionData.h, ActionData.cpp 에디터 설정 데이터 에셋 구조체
- ActionRuntime.h, ActionRuntime.cpp ActionData 구조체를 이용해서 제어하는 런타임 클래스
- ActionEventDataBase.h, ActionEventRuntimeBase.h 액션 이벤트 데이터 구조체와 런타임 클래스
- ActionEventRuntime_MoveNavU.h, ActionEventRuntime_MoveNavU.cpp 네비메쉬를 따라 실제 움직임을 담당하는 액션 이벤트 런타임 클래스
- ActionController.h, ActionController.cpp 액션 데이터를 이용해서 런타임 데이터 생성 및 동작
- ActionData_NormalATK_00.PNG 일반 공격 Action Data 에디터 설정

UActionData, UActionController

- 캐릭터별 액션을 에디터에서 정의하고, 데이타화 한 후 브레인의 판단과 순차 진행 여부에 따라 액션을 실행한다.
- ActionData
  - → UPrimaryDataAsset을 상속받아, 각 액션 이벤트 데이터를 에디터에서 조합할 수 있는 데이터 에셋인 UActionData와 런타임에서 해당 데이터를 실행해주는 UActionRuntime으로 구성되어있다.
  - → 각 액션 Event들도 에디터에서 정보를 저장할 FActionEventDataBase과 런타임에서 동작할 ActionEventRuntimeBase로 구성되어 있다.
- UActionController
  - → 해당 캐릭터가 가진 모든 ActionData를 Brain의 판단에 맞게 Event를 실행시킨다.

관련 문서 →

- 시연 영상 : Game Contents

게임 콘텐츠 시연 영상

관련문서 모음

Data Load 모듈, Data Tool
- 1_DataLoad Module - DataTool 사용법.pdf 기본 사용법 가이드 문서
- 2_DataLoad Module - Auto Generated Source Code.pdf 자동 생성되는 소스코드 설명 문서
- BinaryDataSource.h 바이너리 파일 정보를 uasset에 저장하는 UObject.
- DataFileManagerBase.h 데이터 매니저 Base class
- LZManagerBase.h Localization 매니저 Base class
- HowToUseInRuntimeModule.PNG 런타임 모듈에서 사용하는 코드 스샷
- BinaryDataFactory.h
- ReimportBinaryDataFactory.h
- CDirectoryStatVisitor.h
- CBinaryDataRegistry.h
- 시연 영상 : DataTool, Data Load Module 시연
- DataLoadEditorTest.cpp .byte, .uasset파일 변경 체크 스크립트
- DataLoadTest.cpp 데이터로드시간, 유효성 체크 스크립트
- BinaryDataTest.bat 외부실행배치파일
- 시연 영상 : Data Load Module Automation Test 시연

기본 프레임워크 설계 : 공동작업
- Managers.h, Managers.cpp Managers Class
- DataFileManagerBase.h, DataFileManagerBase.cpp DataFileManager Base Class
- LZManagerBase.h, LZManagerBase.cpp LocalizationManager Base Class
- DataFileManager.h, DataFileManager.cpp Managers에서 사용하는 DataFileManager
- LocalizationManager.h, LocalizationManager.cpp Managers에서 사용하는 Localization Manager
- UIManager.h 기본 매니저 header
- UIManager.cpp Window Stack 관리, 뒷배경 Block 처리
- UIManager+Loading.cpp 로딩 페이지 관리
- UIManager+Module.cpp Window 내부 UIModule 스택관리
- UIManager+Popup.cpp 공용 팝업 관리
- ImmortalWidget.h, ImmotalWidget.cpp 레벨 변경 시에 사라지지 않는 Widget base class
- LoadingBase.h, LoadingBase.cpp 로딩 페이지 base class
- WindowBase.h, WindowBase.cpp Window base class, 런타임에 스택 관리될 때 처리되는 가상 함수 제공
- WindowBase+KeyMapping.cpp Window마다 개별 KeyMapping 처리
- WindowBase+Module.cpp Window내부 UIModule 스택관리
- UIModuleBase.h, UIModuleBase.cpp UIModule base class
- Window_SystemPopup.h, Window_SystemPopup.cpp 공용 팝업
- FXManager.h, FXManager.cpp 기본 매니저. FXActor별 FXContainer class를 이용해서 Pooling 관리
- FXActor.h, FXActor.cpp 개별 Effect Object class. 내부적으로 ParticleSystem, NiagaraComponent, GhostComponent 지원
- SFXManager.h, SFXManager.cpp 기본 매니저. 하나의 Pool로 관리
- SFXActor.h, SFXActor.cpp 개별 Sound Effect Object class.
- WorldManager.h, WorldManager.cpp 기본 매니저. 레벨 기본 로딩 처리 적용
- WorldManager+LevelStreaming.cpp Level Streaming
- FrameworkGameModeBase.h, FrameworkGameModeBase.cpp 레벨 로딩처리 연동되어 있는 기본 GameMode base class
- FrameworkGameInst.h, FrameworkGameInst.cpp
- FrameworkCamera.h, FrameworkCamera.cpp
- FrameworkInputController.h, FrameworkInputController.cpp base calss
- GameCharInputController.h, GameCharInputController.cpp
- SpawnPoint.h, SpawnPoint.cpp
- SpawnPoint+Character.cpp 캐릭터 스폰, 활성화 비활성화 처리
- GameMode_Playable.h, GameMode_Playable.cpp

게임 콘텐츠 설계 : 공동작업
- UserInfo.h, UserInfo.cpp 관리 Info class
- UserCharacterInfo.h, UserCharacterInfo.cpp Character 관련 UserData 관리 Info class
- CUserCharData.h, CUserCharData.cpp User Character Data
- CUserItemData.h, CUserItemData.cpp User Item Data
- GameCharacter.h, GameCharacter.cpp 기본 캐릭터 클래스. 기본 Getter 함수들, Hit Navi Location 처리 등 담당
- GameCharacter+Init.cpp UserData를 기반으로 초기화 처리
- GameCharacter+Team.cpp 팀 처리 및 컬리전 프로파일 처리
- AnimController.h, AnimController.cpp
- AnimInst.h, AnimInst.cpp
- AnimInst_Blueprint.PNG AnimInst를 상속받은 애니메이션 블루프린트
- BrainController.h, BrainController.cpp 기본 브레인 클래스, 기본 Getter 함수들, 기본 캐릭터 방향설정 및 타겟 설정
- BrainController+AI.cpp AI 행동 제어 클래스
- BrainController+Input.cpp 유저 입력 제어 클래스
- AIBehaviorTree.PNG AIBehaviorTree
- BTTask_Patrol.h, BTTask_Patrol.cpp 패트롤 관련 처리 base class. 이동 중 탐색 처리
- BTTask_PatrolLoop.h, BTTask_PatrolLoop.cpp 패트롤 중 한방향 반복 이동 처리
- ActionData.h, ActionData.cpp 에디터 설정 데이터 에셋 구조체
- ActionRuntime.h, ActionRuntime.cpp ActionData 구조체를 이용해서 제어하는 런타임 클래스
- ActionEventDataBase.h, ActionEventRuntimeBase.h 액션 이벤트 데이터 구조체와 런타임 클래스
- ActionEventRuntime_MoveNavU.h, ActionEventRuntime_MoveNavU.cpp 네비메쉬를 따라 실제 움직임을 담당하는 액션 이벤트 런타임 클래스
- ActionController.h, ActionController.cpp 액션 데이터를 이용해서 런타임 데이터 생성 및 동작
- ActionData_NormalATK_00.PNG 일반 공격 Action Data 에디터 설정
- 시연 영상 : Game Contents

포트폴리오 GitHub 주소