유충 낙원 프로젝트 코드 설명서

목차

1. 매니저 구조

프로젝트 내부 구조는 크게 GameInstance인 InsectHeaven 클래스와 기능들을 보조하기 위한 각종 매니저, 해당 컨텐츠에 필요한 클래스들로 이뤄져 있습니다.

```
UCLASS(config = game)
1 derived blueprint class  More...
class UInsectHeaven : public UGameInstance
{
    GENERATED_BODY()

public:
    UInsectHeaven();
    virtual ~UInsectHeaven() override;

    virtual void Init() override;
    virtual void Shutdown() override;

    virtual void OnWorldChanged(UWorld* OldWorld, UWorld* NewWorld) override;
    virtual void LoadComplete(const float LoadTime, const FString& MapName) override;

    void ExitGame();

    bool Tick(float _fDeltaSeconds);

private:
    void BeginTick();
    void RegistState();

    void PreProcess(float fDeltaSeconds);
```

```
#pragma once

#include ...

DECLARE_DELEGATE_FiveParams(FLoadResourceDelegate, const FString&, const FString&, int32, class UObject*, bool);

UCLASS()
No derived blueprint classes
class UIH_SingletonManager : public UObject
{
    GENERATED_BODY()

public:
    static UIH_SingletonManager* CreateInstance();
    static void DestroyInstance();
    static UIH_SingletonManager* GetInstance() { return Instance;}
    static bool HasInstance();

protected:
    static UIH_SingletonManager* Instance;

public:
    FORCEINLINE void SetGameInstance(class UGameInstance* _GameInstacne) { GameInstance = _GameInstacne; }
    FORCEINLINE class UGameInstance* GetGameInstance() const { return GameInstance; }

    bool IsBuiltInInitialized() const { return bIsBuiltInInitialized; }
    bool IsInitialized() const { return bIsInitialized; }
```

1. 매니저 구조

모든 매니저는 싱글톤 매니저 클래스를 통해 관리합니다. 자세한 구조는 SingletonManger 클래스를 참조해주세요.

1)현재 데이터를 관리하기 위한 TableManager

2) 위젯를 관리하기 위한 WidgetManager

3) 씬 관리를 위한 SceneManager

4) 다이얼로그를 관리하기 위한 DialogueManager

```
void UIH_SingletonManager::RegisterSingletons()
{
    Singletons.Reset();

    Singletons.Emplace(UIH_TableManager::MakeInstance());
    Singletons.Emplace(UIH_SceneManager::MakeInstance());
    Singletons.Emplace(UIH_WidgetManager::MakeInstance());
    Singletons.Emplace(UDialogueManager::MakeInstance());
}

void UIH_SingletonManager::RegisterSingletonsForTick()
```

2. 위젯 구조

위젯은 상기했듯 위젯 매니저를 통해 관리되며 위젯 매니저는 내부에 자체 오브젝트 풀을 통해 위젯들을 관리합니다.

위젯은 오브젝트 풀을 통해 관리하는 지 여부에 따라 두가지로 나뉘며 각각을 생성하기 위한 함수가 다릅니다.

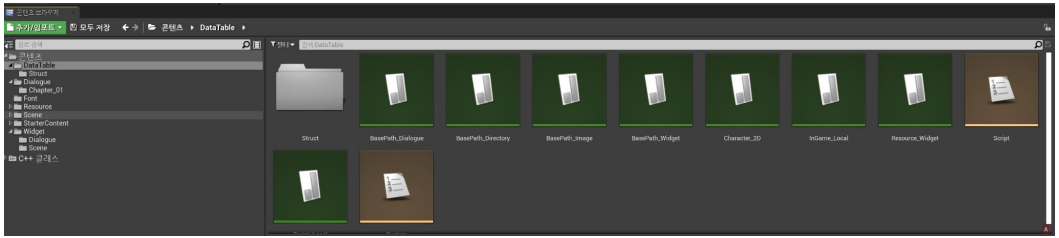
```
//Widget Object Control
UIH_Widget* CreateWidgetByName(const FName& _TypeName);
UIH_Widget* CreateWidgetNotManaging(const FSoftObjectPath _SoftObjectPath);
```

3. 테이블 구조

테이블은 크게 데이터 구조, 데이터 에셋, 맵퍼, 테이블 매니저로 구성되어 있습니다.

1) 데이터 구조: 해당 데이터의 형식을 정의하는 클래스입니다. ex)Resource_Widget.h

2) 데이터 에셋: 데이터 구조에 따라 생성한 데이터가 담겨있는 파일입니다.



3) 맵퍼: 데이터에셋의 정보를 저장하고, 필요에 따라 가공하는 클래스입니다. ex) UIH_Mapper_Resource_Widget

4) 테이블 매니저: 각 맵퍼를 저장해두고, 코드에서 필요에 따라 해당 맵퍼를 제공하는 역할

4. 다이얼로그 툴 시스템

다이얼로그 툴은 사용자에게 캐릭터 대화를 자유롭게 편집 가능하도록 해주는 에디터 유틸리티 위젯입니다. 자세한 내용은 해당 위젯을 통해 확인 가능합니다.

다이얼로그는 다이얼로그 액션이라는 클래스의 서브클래스 집합으로 이뤄져있습니다.


```
UCLASS(BlueprintType)
No derived blueprint classes
class UDialogueAction_Talk : public UDialogueAction
{
    GENERATED_BODY()
public:
    virtual void SaveToJson(TSharedPtr<FJsonObject> _JsonObject) override;
    virtual void LoadFromJson(TSharedPtr<FJsonObject> _JsonObject) override;
    virtual FText Get_Name() override;
    virtual FSlateColor Get_Color() override;
    virtual FText Get_Description() override;

    virtual void Execute() override;
    virtual bool Progress(float _fDelta) override;
    virtual void OnInput() override;
```

4. 다이얼로그 툴 시스템

각각의 다이얼로그 액션의 데이터는 json형식으로 전환할 수 있으며, 해당 다이얼로그에 사용된 각 액션을 모아 하나의 json파일로 저장함으로써 다이얼로그를 저장하고 불러올 수 있게 합니다.

```
UCLASS(BlueprintType)
1 derived blueprint class
class AIH_DialoguePlayer : public AActor
{
    GENERATED_BODY()

public:
    virtual void BeginPlay() override;
    virtual void Tick(float DeltaTime) override;
    virtual void EndPlay(const EEndPlayReason::Type EndPlayReason) override;

public:
    UFUNCTION(BlueprintCallable, Category = "ADialogueActor")
    bool SaveToJason(const FString& _strFilePath);  No blueprint usages

    UFUNCTION(BlueprintCallable, Category = "ADialogueActor")
    bool LoadToJson(const FString& _strFilePath);  No blueprint usages

    UFUNCTION(BlueprintCallable, Category = "ADialogueActor")
```