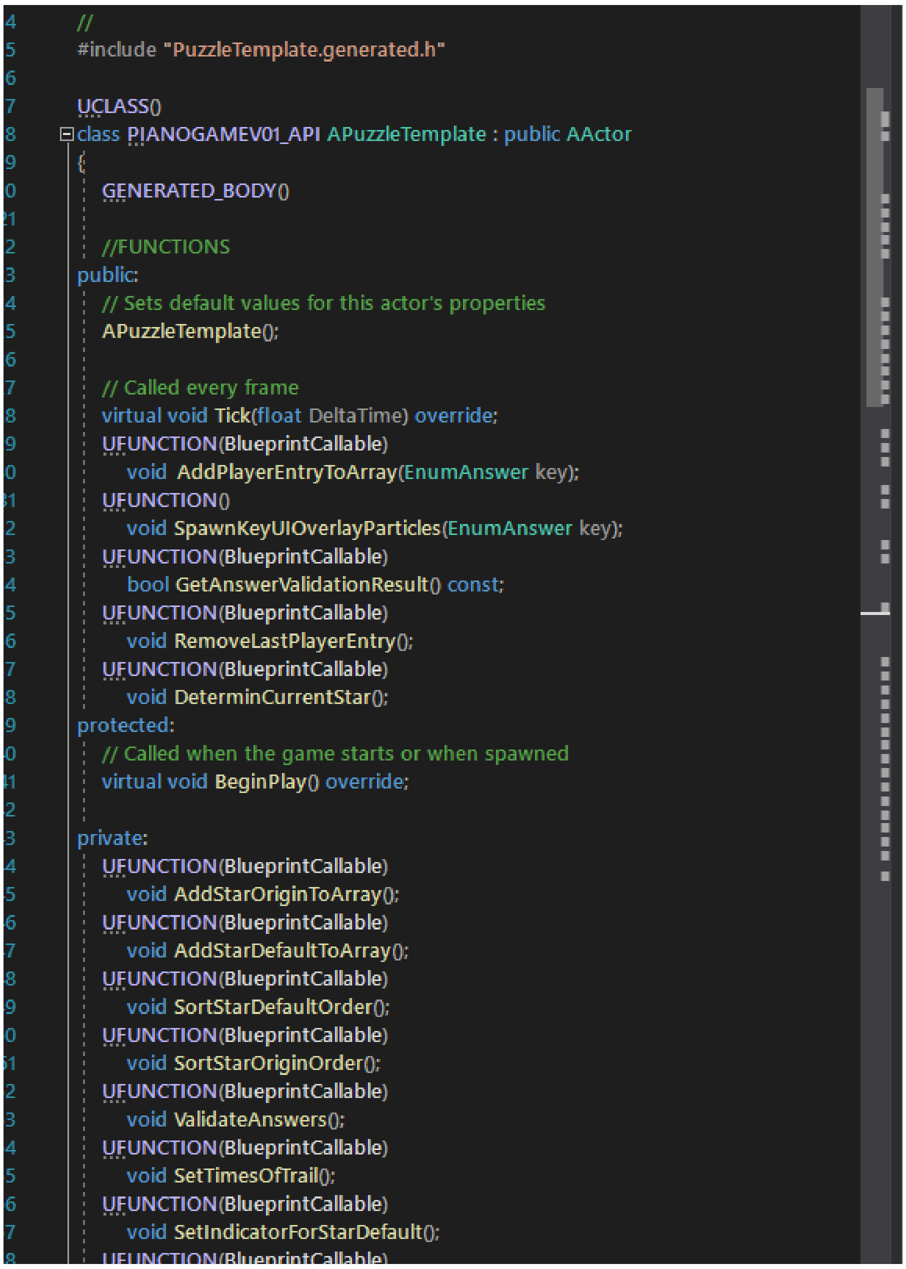
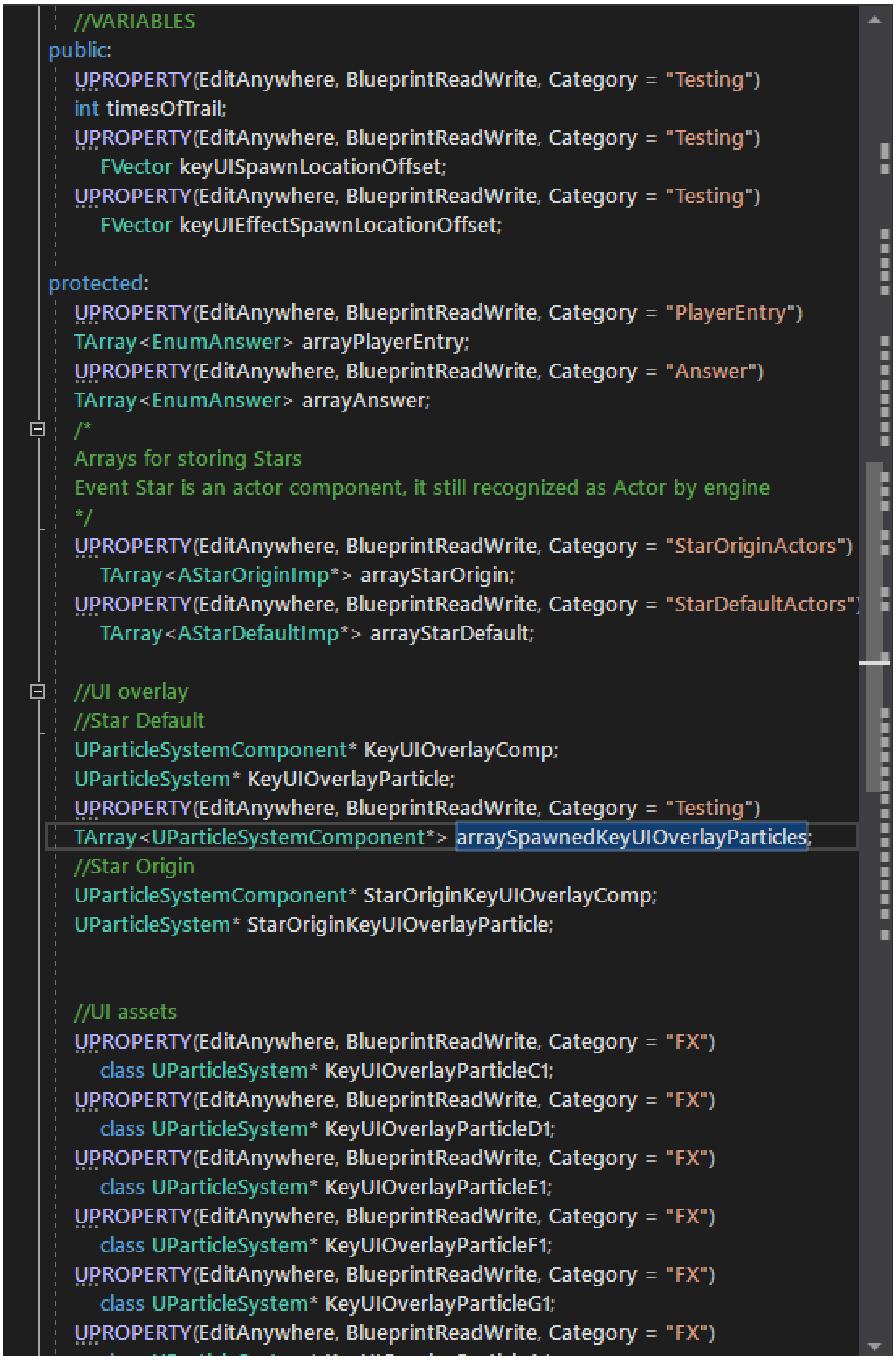
This week I focused on programming a puzzle template that can hold all stars and determine the game logic. This template should be able to produce endless of puzzles with the same game mechanics as I designed in week6. It is intended to be code base and initialized as blueprint in Unreal engine for the level designer to build the puzzle. The template has these main functions: recognize all the stars that the designer put in for a puzzle; let the designer assign the corresponding order and answer of each star;, sort those stars based on the assigned order; record player entry from input; check player entry against answer or each star and produce a result; do game logics based on that result; and some other clean-up functions.

Code snips:

The template class:

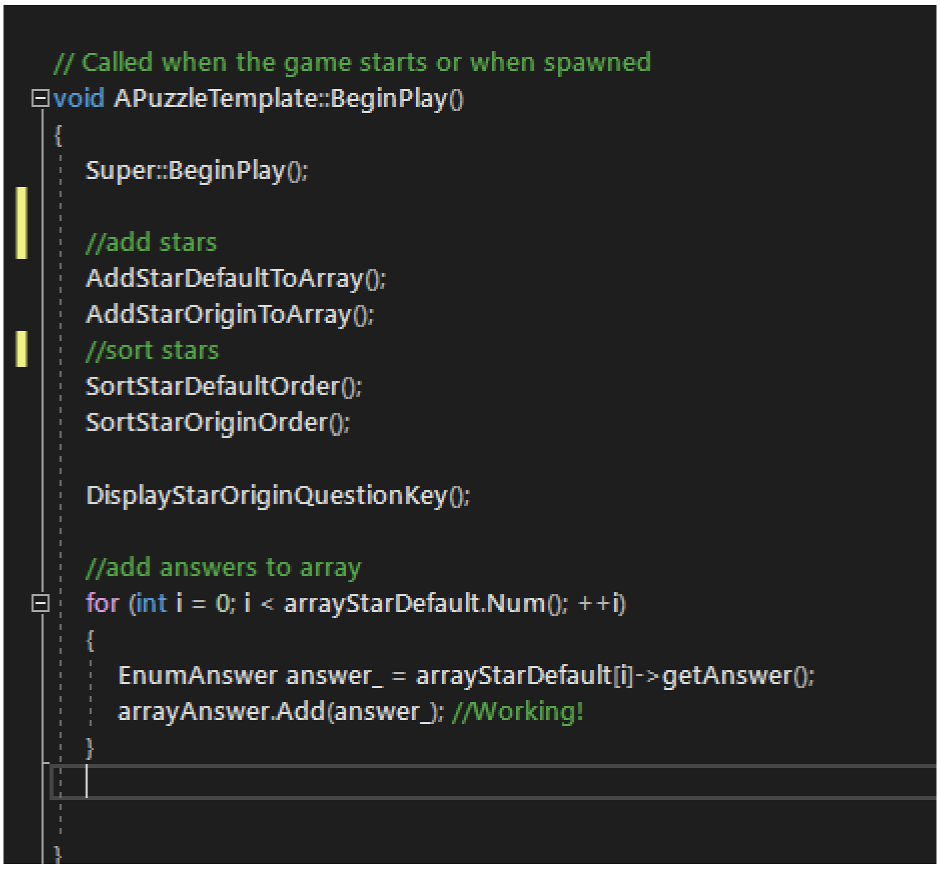


```
//
#include "PuzzleTemplate.generated.h"

UCLASS()
class PIANOGAMEV01_API APuzzleTemplate : public AActor
{
	GENERATED_BODY()

	//FUNCTIONS
public:
	// Sets default values for this actor's properties
	APuzzleTemplate();

	// Called every frame
	virtual void Tick(float DeltaTime) override;
	UFUNCTION(BlueprintCallable)
		void AddPlayerEntryToArray(EnumAnswer key);
	UFUNCTION()
		void SpawnKeyUIOverlayParticles(EnumAnswer key);
	UFUNCTION(BlueprintCallable)
		bool GetAnswerValidationResult() const;
	UFUNCTION(BlueprintCallable)
		void RemoveLastPlayerEntry();
	UFUNCTION(BlueprintCallable)
		void DeterminCurrentStar();
protected:
	// Called when the game starts or when spawned
	virtual void BeginPlay() override;

private:
	UFUNCTION(BlueprintCallable)
		void AddStarOriginToArray();
	UFUNCTION(BlueprintCallable)
		void AddStarDefaultToArray();
	UFUNCTION(BlueprintCallable)
		void SortStarDefaultOrder();
	UFUNCTION(BlueprintCallable)
		void SortStarOriginOrder();
	UFUNCTION(BlueprintCallable)
		void ValidateAnswers();
	UFUNCTION(BlueprintCallable)
		void SetTimesOfTrail();
	UFUNCTION(BlueprintCallable)
		void SetIndicatorForStarDefault();
	UFUNCTION(BlueprintCallable)
```



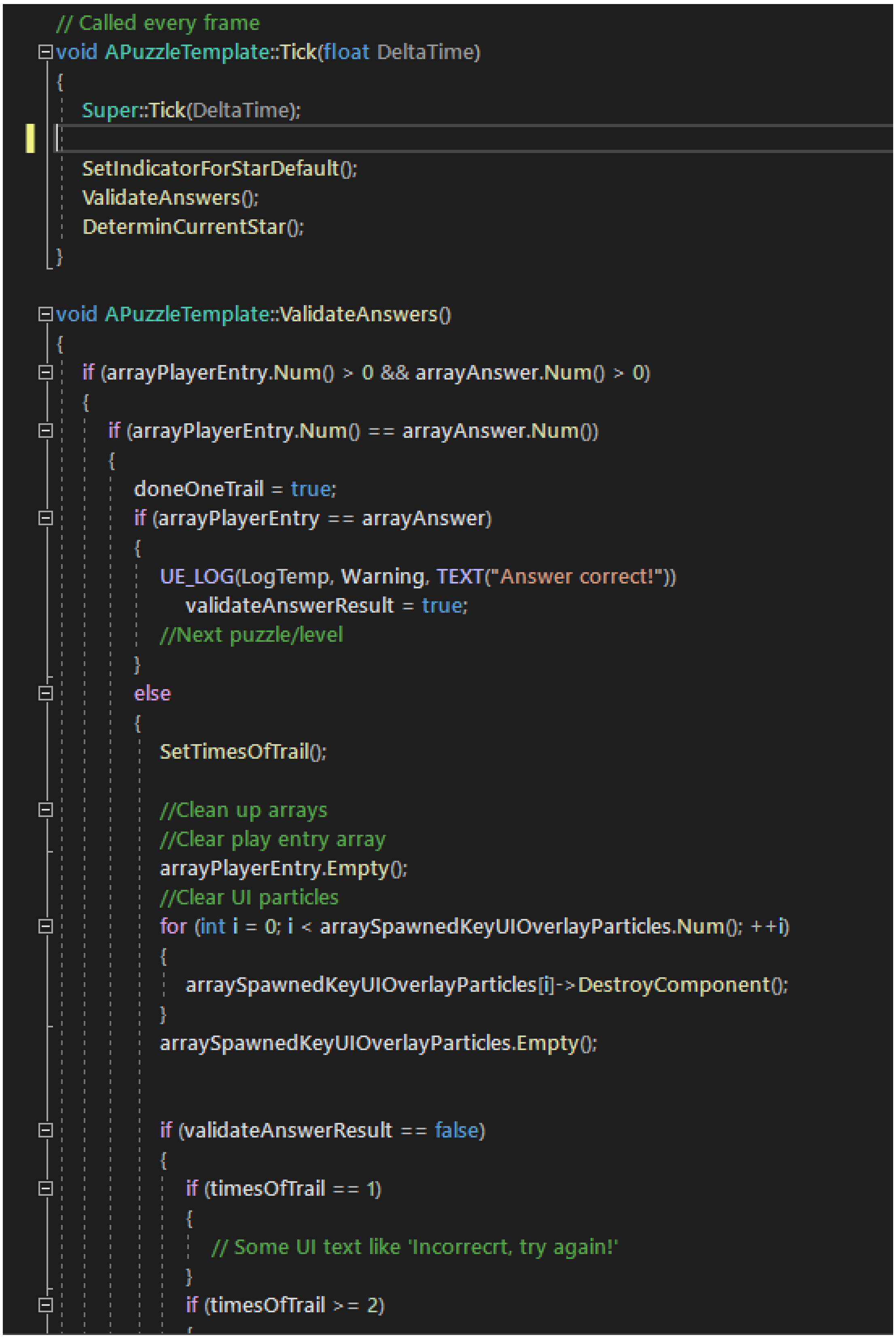
```
    //VARIABLES
public:
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Testing")
    int timesOfTrail;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Testing")
        FVector keyUISpawnLocationOffset;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Testing")
        FVector keyUIEffectSpawnLocationOffset;

protected:
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "PlayerEntry")
    TArray<EnumAnswer> arrayPlayerEntry;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Answer")
    TArray<EnumAnswer> arrayAnswer;
    /*
    Arrays for storing Stars
    Event Star is an actor component, it still recognized as Actor by engine
    */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "StarOriginActors")
        TArray<AStarOriginImp*> arrayStarOrigin;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "StarDefaultActors"
        TArray<AStarDefaultImp*> arrayStarDefault;

    //UI overlay
    //Star Default
    UParticleSystemComponent* KeyUIOverlayComp;
    UParticleSystem* KeyUIOverlayParticle;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Testing")
    TArray<UParticleSystemComponent*> arraySpawnedKeyUIOverlayParticles;
    //Star Origin
    UParticleSystemComponent* StarOriginKeyUIOverlayComp;
    UParticleSystem* StarOriginKeyUIOverlayParticle;


    //UI assets
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "FX")
        class UParticleSystem* KeyUIOverlayParticleC1;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "FX")
        class UParticleSystem* KeyUIOverlayParticleD1;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "FX")
        class UParticleSystem* KeyUIOverlayParticleE1;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "FX")
        class UParticleSystem* KeyUIOverlayParticleF1;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "FX")
        class UParticleSystem* KeyUIOverlayParticleG1;
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "FX")
```

Template initialization on game start:



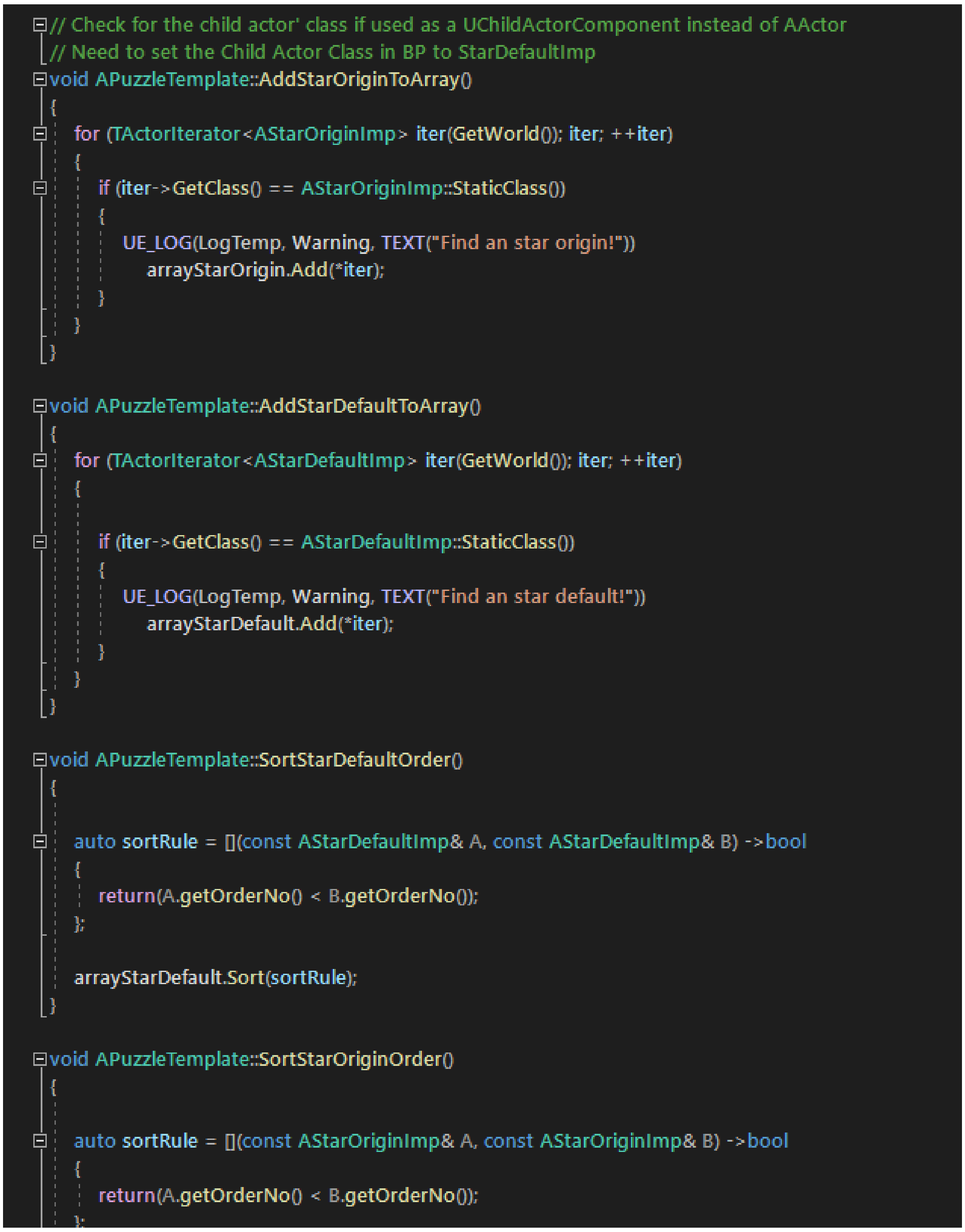
```
// Called when the game starts or when spawned
void APuzzleTemplate::BeginPlay()
{
    Super::BeginPlay();

    //add stars
    AddStarDefaultToArray();
    AddStarOriginToArray();
    //sort stars
    SortStarDefaultOrder();
    SortStarOriginOrder();

    DisplayStarOriginQuestionKey();

    //add answers to array
    for (int i = 0; i < arrayStarDefault.Num(); ++i)
    {
        EnumAnswer answer_ = arrayStarDefault[i]->getAnswer();
        arrayAnswer.Add(answer_); //Working!
    }


}
```

Game logic:



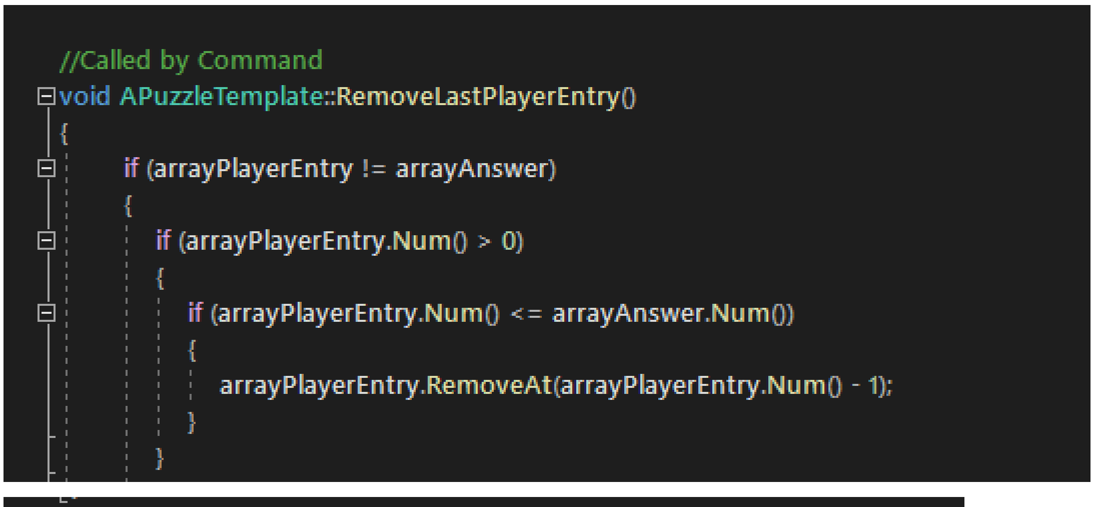
```
// Called every frame
void APuzzleTemplate::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

    SetIndicatorForStarDefault();
    ValidateAnswers();
    DeterminCurrentStar();
}

void APuzzleTemplate::ValidateAnswers()
{
    if (arrayPlayerEntry.Num() > 0 && arrayAnswer.Num() > 0)
    {
        if (arrayPlayerEntry.Num() == arrayAnswer.Num())
        {
            doneOneTrail = true;
            if (arrayPlayerEntry == arrayAnswer)
            {
                UE_LOG(LogTemp, Warning, TEXT("Answer correct!"))
                    validateAnswerResult = true;
                //Next puzzle/level
            }
            else
            {
                SetTimesOfTrail();

                //Clean up arrays
                //Clear play entry array
                arrayPlayerEntry.Empty();
                //Clear UI particles
                for (int i = 0; i < arraySpawnedKeyUIOverlayParticles.Num(); ++i)
                {
                    arraySpawnedKeyUIOverlayParticles[i]->DestroyComponent();
                }
                arraySpawnedKeyUIOverlayParticles.Empty();


                if (validateAnswerResult == false)
                {
                    if (timesOfTrail == 1)
                    {
                        // Some UI text like 'Incorrecrt, try again!'
                    }
                    if (timesOfTrail >= 2)
```

Add and sort stars:

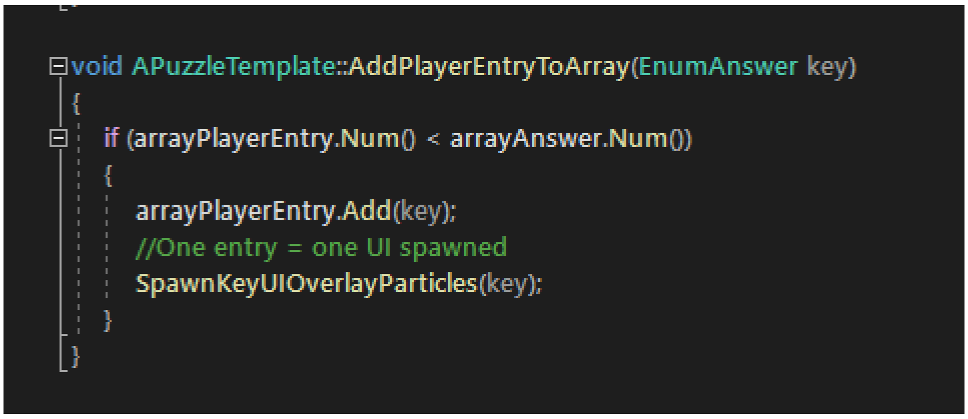


```
// Check for the child actor' class if used as a UChildActorComponent instead of AActor
// Need to set the Child Actor Class in BP to StarDefaultImp
void APuzzleTemplate::AddStarOriginToArray()
{
    for (TActorIterator<AStarOriginImp> iter(GetWorld()); iter; ++iter)
    {
        if (iter->GetClass() == AStarOriginImp::StaticClass())
        {
            UE_LOG(LogTemp, Warning, TEXT("Find an star origin!"))
                arrayStarOrigin.Add(*iter);
        }
    }
}

void APuzzleTemplate::AddStarDefaultToArray()
{
    for (TActorIterator<AStarDefaultImp> iter(GetWorld()); iter; ++iter)
    {

        if (iter->GetClass() == AStarDefaultImp::StaticClass())
        {
            UE_LOG(LogTemp, Warning, TEXT("Find an star default!"))
                arrayStarDefault.Add(*iter);
        }
    }
}

void APuzzleTemplate::SortStarDefaultOrder()
{

    auto sortRule = [](const AStarDefaultImp& A, const AStarDefaultImp& B) ->bool
    {
        return(A.getOrderNo() < B.getOrderNo());
    };

    arrayStarDefault.Sort(sortRule);
}

void APuzzleTemplate::SortStarOriginOrder()
{

    auto sortRule = [](const AStarOriginImp& A, const AStarOriginImp& B) ->bool
    {
        return(A.getOrderNo() < B.getOrderNo());
    };
```

Deal with player entries/input:



```
//Called by Command
void APuzzleTemplate::RemoveLastPlayerEntry()
{
        if (arrayPlayerEntry != arrayAnswer)
        {
            if (arrayPlayerEntry.Num() > 0)
            {
                if (arrayPlayerEntry.Num() <= arrayAnswer.Num())
                {
                    arrayPlayerEntry.RemoveAt(arrayPlayerEntry.Num() - 1);
                }
            }
```



```
void APuzzleTemplate::AddPlayerEntryToArray(EnumAnswer key)
{
    if (arrayPlayerEntry.Num() < arrayAnswer.Num())
    {
        arrayPlayerEntry.Add(key);
        //One entry = one UI spawned
        SpawnKeyUIOverlayParticles(key);
    }
}
```