

Please contact me to make changes or to share elsewhere. If you have questions or improvements, I'd be happy to hear. (Lütfen değişiklik yapmak için veya başka yerlerde paylaşmak için iletişime geçiniz. Sorularınız veya geliştirmeleriniz varsa duymaktan memnun olurum.)

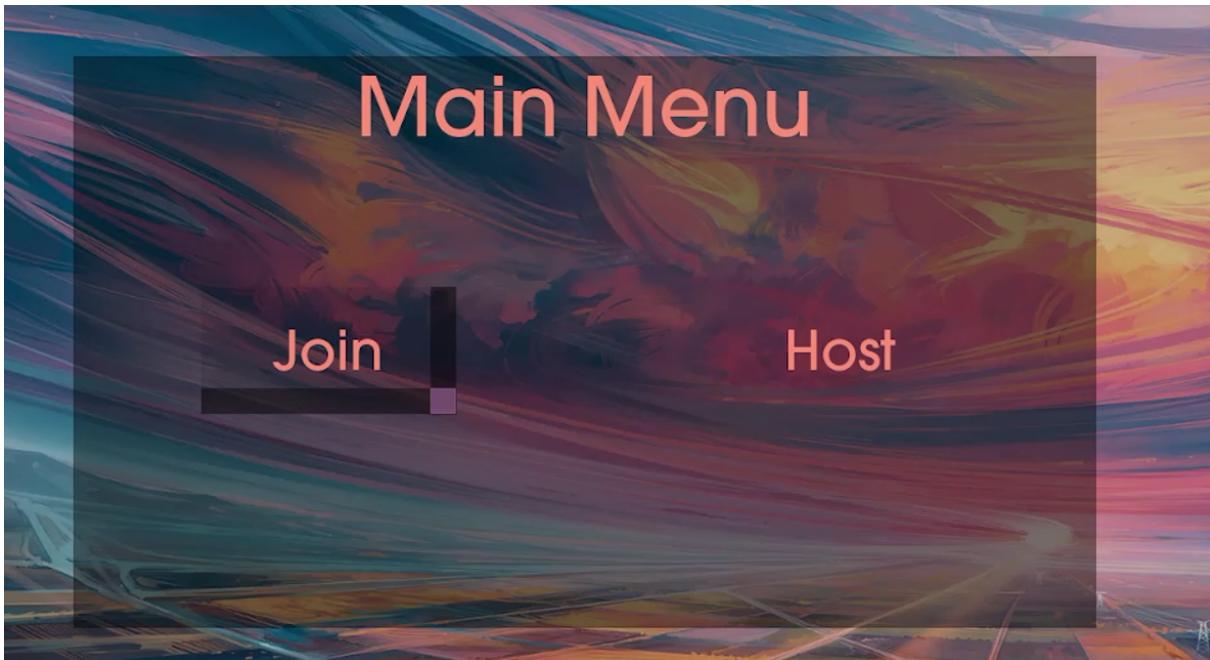
Github: <https://github.com/creosB>

Youtube: <https://www.youtube.com/channel/UCqWVCirXu-1fTdGRI73ICzA>

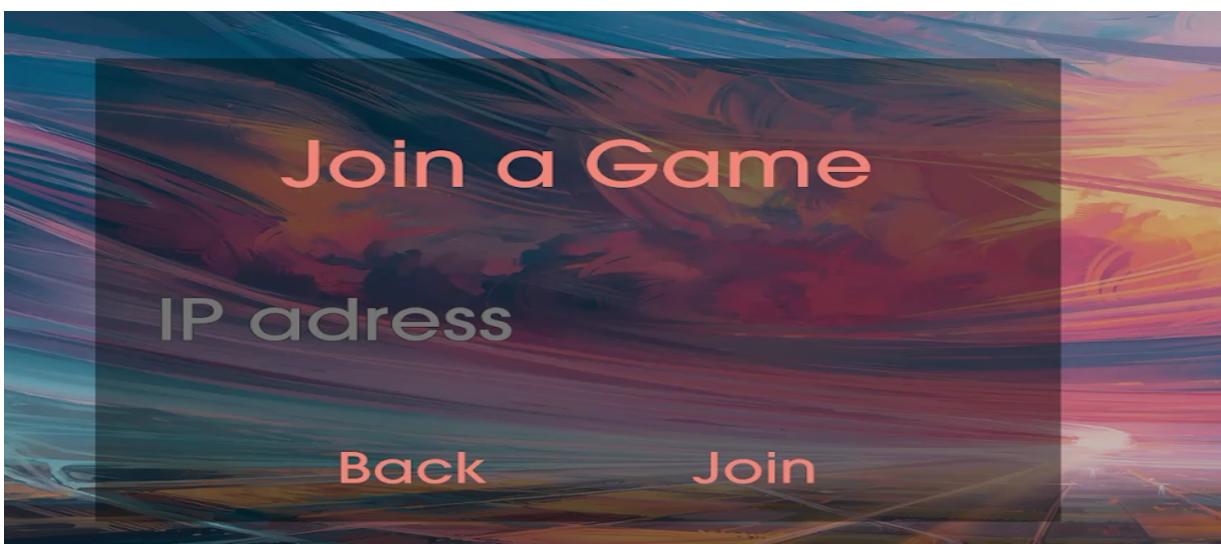
The channel where we do this and more: <https://www.twitch.tv/creosb>

- 1) [**Multiplayer lobby**](#) (Multiplayer with hamachi or any IP address)
- 2) [**Puzzle Platform**](#) (As in puzzle games, while someone is standing in a place, the mechanism starts to move, and the other player can cross)
- 3) [**BP Create Collectable Object**](#)
- 4) [**Multiplayer Server Type Selection**](#)
- 5) [**Steam Lobby System C++ \(GameDevTV Course\)**](#)
 - a. [**Server Type Check**](#)
- 6) [**BP Save - Load System**](#)
- 7) [**Loading Screen and Load Map Methods**](#) (Unreal Engine Diving Levels Course)
- 8) [**Gameplay Ability Plugin**](#)
- 9) [**Asset Manager**](#)
- 10) [**Game Start with CMD**](#)
- 11) [**Garbage Collection on Unreal Engine**](#)
- 12) [**Replication Refactor for movement bindings**](#)
- 13) [**Multiplayer Replication - LAG - Package Loss**](#)
 - a. [**Replication Actor Roles**](#)
 - b. [**What's the Lag ?**](#)
 - c. [**What's the Package Loss ?**](#)
 - d. [**Replication Refactor for movement bindings**](#)

Aşağıdaki gibi bir widget oluşturuyoruz.



Join butonuna basıldığında widget'lar arasında geçiş yapabilmek için WidgetSwitcher kullanıyoruz ve 2. Widget'ımızı aynı blueprint içerisinde oluşturuyoruz.



Sayfalar arasında veriyi iletebilmek için Interface yaratıyoruz.

Interface.h

```
public:  
    // pure virtual function - no implementation  
    virtual void Host() = 0;  
    virtual void Join(const FString& Adress) = 0;  
    virtual void LoadMainMenu() = 0;
```

.cpp dosyasına herhangi bir tanımlama yapmamıza gerek yok.

Oyun içinden de aynı şekilde main menu'ye gitmek için widget oluşturuyoruz.



Artık butonlara tıklayabilmek ve leveller arası geçiş yaparken widget'ların açık kalmamasını sağlamak kaldı. Bunu yapmak için UUserWidget tanımlıyoruz.

MenuWidget.h

```
public:  
    void Setup();  
    void SetMenuInterface(IMenuInterface* MenuInterface);  
protected:  
    IMenuInterface* MainMenuInterface = nullptr;  
    APlayerController* PlayerController = nullptr;  
    FInputModeUIOnly InputModeData; // call the struct  
    FInputModeGameOnly InputModeGameOnly; // call the struct  
  
protected:  
    virtual void OnLevelRemovedFromWorld(ULevel* InLevel, UWorld* InWorld) override; // delete widget  
and set controller
```

MenuWidget.cpp

```
#include "MenuWidget.h"  
  
// Add default functionality here for any IMenuInterface functions that are not pure virtual.  
void UMenuWidget::SetMenuInterface(IMenuInterface* MenuInterface)  
{  
    MainMenuInterface = MenuInterface;  
}  
  
// show widget on the screen  
void UMenuWidget::Setup()  
{  
    // Show menu widget  
    this->AddToViewport();  
  
    UWorld* World = GetWorld();  
    if (!ensure(World != nullptr)) { return; }
```

```

// Show cursor on the menu for the click to button.
PlayerController = World->GetFirstPlayerController();
if (!ensure(PlayerController != nullptr)) return;
//PlayerController->SetShowMouseCursor(true); // alternative for the show cursor

InputModeData.SetWidgetToFocus(this->TakeWidget()); // show in the widget
InputModeData.SetLockMouseToViewportBehavior(EMouseLockMode::DoNotLock); // set lock mode

PlayerController->SetInputMode(InputModeData);
PlayerController->bShowMouseCursor = true;
}

// When switched the level, it will activate input and remove the widget.
void UMenuWidget::OnLevelRemovedFromWorld(ULevel* InLevel, UWorld* InWorld)
{
    this->RemoveFromViewport();

    UWorld* World = GetWorld();
    if (!ensure(World != nullptr)) { return; }

    PlayerController = World->GetFirstPlayerController(); // call player controller
    if (!ensure(PlayerController != nullptr)) return;

    PlayerController->SetInputMode(InputModeGameOnly);
    PlayerController->bShowMouseCursor = false;
}

```

Şeklinde tanımlamalarımızı yaparak artık widget'lara tıklarken ki mouse gözükmeme ve leveller arasında geçiş yaparken yaşanan sorunları da önlemiş oluyoruz.

Artık butonlara bastığımız zaman gerekli işlemleri çalıştırmasını sağlamak kaldı bunun için oluşturduğumuz MenuWidget'ın alt sınıfını oluşturuyoruz.

UMainMenu : public UMenuWidget

MainMenu.h

```

private:
    // Looks for a widget in Blueprint with the same name as the property.
    UPROPERTY(meta = (BindWidget))
    class UButton* Host; // host button in main menu
    UPROPERTY(meta = (BindWidget))
    class UButton* Join; // join button in main menu
    UPROPERTY(meta = (BindWidget))
    class UButton* JoinButton; // join button in join a game menu
    UPROPERTY(meta = (BindWidget))
    class UButton* Back; // back button in main menu
    UPROPERTY(meta = (BindWidget))
    class UButton* ExitButton; // back button in main menu
    UPROPERTY(meta = (BindWidget))
    class UWidgetSwitcher* WidgetSwitch; // wrap with widget switcher on the editor
    UPROPERTY(meta = (BindWidget))
    class UWidget* JoinMenu; // join menu widget
    UPROPERTY(meta = (BindWidget))

```

```

class UEditableTextBox* IPWriteBox; // ip write box on join menu widget

UFUNCTION()
void HostServer(); // when you pushed the button, it will call this func.
UFUNCTION()
void JoinServer();
UFUNCTION()
void OpenJoinMenu(); // when you pushed the button, it will call this func.
UFUNCTION()
void Exit();

protected:
    virtual bool Initialize() override;

```

MenuItemWidget.cpp

```

#include "MainMenu.h"
#include "Components/Button.h"
#include "Components/WidgetSwitcher.h"
#include "Components/EditableTextBox.h"

bool UMainMenu::Initialize()
{
    bool Success = Super::Initialize();
    if (!Success) { return false; }
    if (!ensure(Host != nullptr)) { return false; }
    Host->OnClicked.AddDynamic(this, &UMainMenu::HostServer);
    if (!ensure(Join != nullptr)) { return false; }
    Join->OnClicked.AddDynamic(this, &UMainMenu::OpenJoinMenu);
    if (!ensure(Back != nullptr)) { return false; }
    Back->OnClicked.AddDynamic(this, &UMainMenu::OpenJoinMenu);
    if (!ensure(JoinButton != nullptr)) { return false; }
    JoinButton->OnClicked.AddDynamic(this, &UMainMenu::JoinServer);
    if (!ensure(ExitButton != nullptr)) { return false; }
    ExitButton->OnClicked.AddDynamic(this, &UMainMenu::Exit);
    return true;
}

// host the server
void UMainMenu::HostServer()
{
    if (MainMenulnface != nullptr)
    {
        MainMenulnface->Host();
    }
}

// connect - join the server
void UMainMenu::JoinServer()
{
    if (MainMenulnface != nullptr && IPWriteBox != nullptr)
    {
        MainMenulnface->Join(IPWriteBox->GetText().ToString());
    }
}

// switch between main menu

```

```

void UMainMenu::OpenJoinMenu()
{
    if (!ensure(WidgetSwitch != nullptr)) { return; }
    if (!ensure(JoinMenu != nullptr)) { return; }

    if (WidgetSwitch->GetActiveWidget() == JoinMenu)
    {
        WidgetSwitch->SetActiveWidget(this); // this = main menu - widget 0 index
    }
    else
    {
        // you can active this method (it's safety) or you can active with widget index.
        WidgetSwitch->SetActiveWidget(JoinMenu);
    }
}

// Exit to the game
void UMainMenu::Exit()
{
    UWorld* World = GetWorld();
    if (!ensure(World != nullptr)) { return; }

    PlayerController = World->GetFirstPlayerController(); // call player controller
    if (!ensure(PlayerController != nullptr)) return;

    PlayerController->ConsoleCommand("quit");
}

```

Bu şekilde tanımlamaları yaparak widget içerisinde koyduğumuz her şeye işlevini kazandırmış oluyoruz.

Not: Widget içerisinde bulunan isimleri ile fonksiyonun ismi aynı olmak zorunda.

Aynı şekilde hemen PauseMenu de yaratıyoruz.

PauseMenu.h

```

UCLASS()
class YourGAME_API UPauseMenu : public UMenuWidget
{
    GENERATED_BODY()

public:
    UPROPERTY(meta = (BindWidget))
    class UButton* CancelButton;
    UPROPERTY(meta = (BindWidget))
    class UButton* QuitButton;

private:
    UFUNCTION()
    void Quit();
    UFUNCTION()
    void Cancel();

protected:
    virtual bool Initialize() override;
};

```

PauseMenu.cpp

```
bool UPauseMenu::Initialize()
{
    bool Success = Super::Initialize();
    if (!Success) { return false; }
    if (!ensure(CancelButton != nullptr)) { return false; }
    CancelButton->OnClicked.AddDynamic(this, &UPauseMenu::Cancel);
    if (!ensure(QuitButton != nullptr)) { return false; }
    QuitButton->OnClicked.AddDynamic(this, &UPauseMenu::Quit);

    return true;
}

void UPauseMenu::Quit()
{
    if(MainMenuInterface != nullptr)
    {
        OnLevelRemovedFromWorld(GetWorld()->GetCurrentLevel(),GetWorld()); // show main menu widget
        and set cursor
        MainMenuInterface->LoadMainMenu(); // travel to the main menu map
    }
}

void UPauseMenu::Cancel()
{
    OnLevelRemovedFromWorld(GetWorld()->GetCurrentLevel(),GetWorld()); // clear viewport and get
    controller
}
```

Artık butonlar ile verdigimiz emir ile haritalar arası geçiş yaparken, hem geçiş yapmamızı hem de bilgileri taşımamıza yarayan Instance oluşturuyoruz.

GameInstance.h

```
UCLASS()
class YourGAME_API UYourGameInstance : public UGameInstance, public IMenueInterface
{
    GENERATED_BODY()

public:
    UYourGameInstance(const FObjectInitializer& ObjectInitializer);

    virtual void Init() override;

    UFUNCTION(BlueprintCallable)
    void LoadMenu();
    UFUNCTION(BlueprintCallable)
    void PauseLoadMenu();

    UFUNCTION(Exec) // exec = console command
    virtual void Host() override;
```

```

UFUNCTION(Exec)
virtual void Join(const FString& Adress) override;

virtual void LoadMainMenu() override;

private:
TSubclassOf<class UUserWidget> MenuClass = nullptr;
UMainMenu* Menu = nullptr;
TSubclassOf<class UUserWidget> PauseMenuClass = nullptr;
UPauseMenu* PauseMenu = nullptr;
};

```

GameInstance.cpp

```

UYourGameInstance::UYourGameInstance(const FObjectInitializer& ObjectInitializer)
{
    // find menu widget class and define to MenuClass.
    ConstructorHelpers::FClassFinder<UUserWidget>
    MenuBPClass(TEXT("/Game/Blueprints/Widgets/WBP_MainMenu"));
    if (!ensure(MenuBPClass.Class != nullptr)) return;
    MenuClass = MenuBPClass.Class;
    // find pause menu widget class and define to PauseMenuClass.
    ConstructorHelpers::FClassFinder<UUserWidget>
    PauseMenuBPClass(TEXT("/Game/Blueprints/Widgets/WBP_PauseMenu"));
    if (!ensure(PauseMenuBPClass.Class != nullptr)) return;
    PauseMenuClass = PauseMenuBPClass.Class;
    //UE_LOG(LogTemp, Warning, TEXT("Game Instance Constructor"));
}

void UYourGameInstance::Init()
{
    //UE_LOG(LogTemp, Warning, TEXT("Game Instance Init"));
}

void UYourGameInstance::Host()
{
    // load to the map
    UWorld* World = GetWorld();
    if (!ensure(World != nullptr)) { return; }
    World->ServerTravel(TEXT("/Game/Maps/Level2?listen")); // ?listen server
}

void UYourGameInstance::Join(const FString& Adress)
{
    if (GEngine)
        GEngine->AddOnScreenDebugMessage(-1, 15.0f, FColor::Yellow, FString::Printf(TEXT("Joining %s"),
    *Adress));
    // join to the input adress.
    APlayerController* PlayerController = GetFirstLocalPlayerController();
    if (!ensure(PlayerController != nullptr)) { return; }
    PlayerController->ClientTravel(Adress, ETravelType::TRAVEL_Absolute);
}
// create main menu widget

```

```

void UYourGameInstance::LoadMenu()
{
    // Create menu widget
    if (!ensure(MenuClass != nullptr)) return;
    Menu = CreateWidget<UMainMenu>(this, MenuClass); // create menu widget with MenuClass
    if (!ensure(Menu != nullptr)) return;

    Menu->Setup(); // add to viewport and show cursor
    Menu->SetMenuItemInterface(this);
}

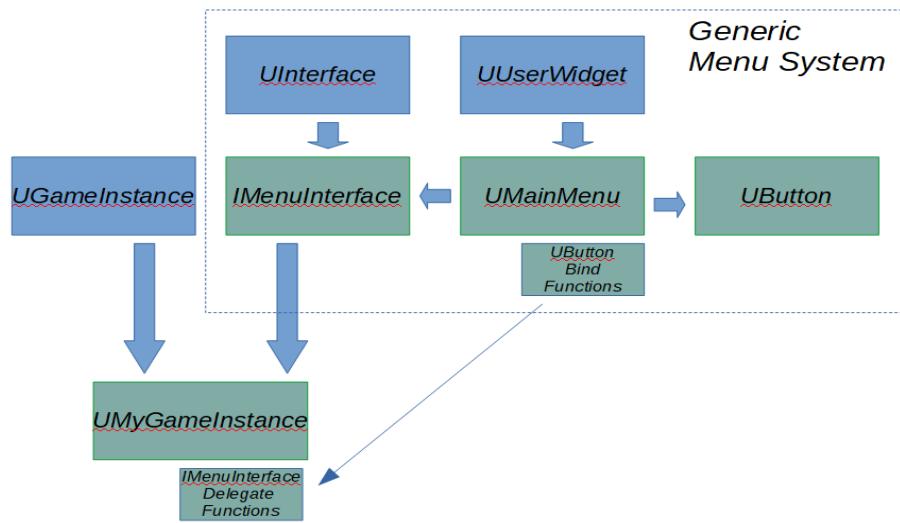
// create pause menu widget
void UYourGameInstance::PauseLoadMenu()
{
    if (!ensure(PauseMenuClass != nullptr)) return;
    PauseMenu = CreateWidget<UPauseMenu>(this, PauseMenuClass); // create menu widget with
    MenuClass
    if (!ensure(PauseMenu != nullptr)) { return; }
    PauseMenu->Setup(); // add to viewport and show cursor
    PauseMenu->SetMenuItemInterface(this);
}

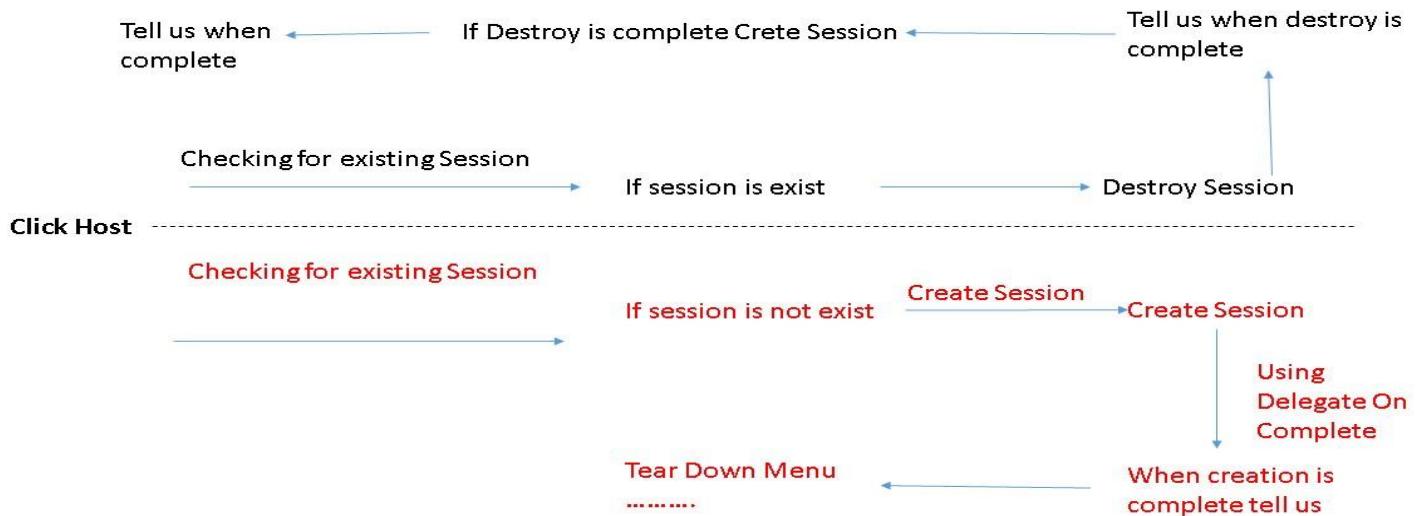
// open the main menu
void UYourGameInstance::LoadMainMenu()
{
    APlayerController* PlayerController = GetFirstLocalPlayerController();
    if (!ensure(PlayerController != nullptr)) { return; }
    PlayerController->ClientTravel("/Game/Maps/MainMenu", ETravelType::TRAVEL_Absolute); // return to
    the main menu
}

```

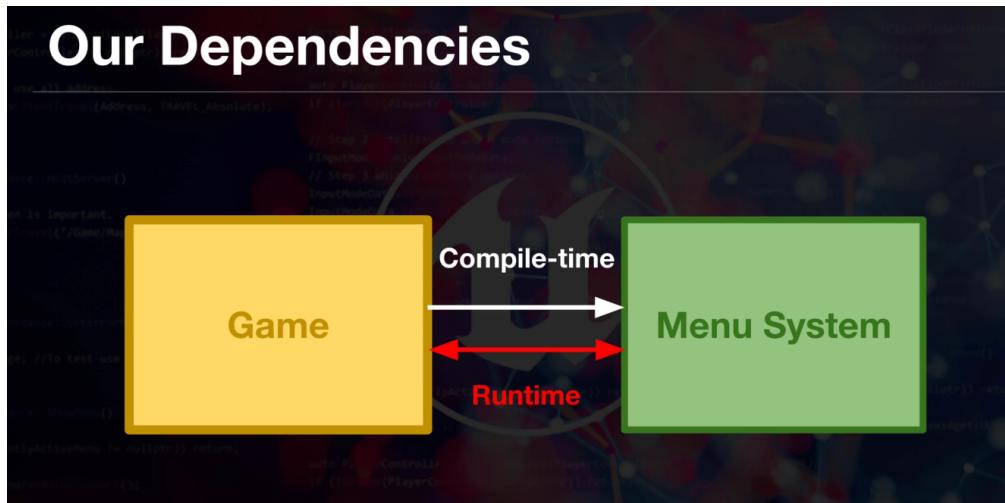
Let's visualize the menu system

Menu System Class Interactions



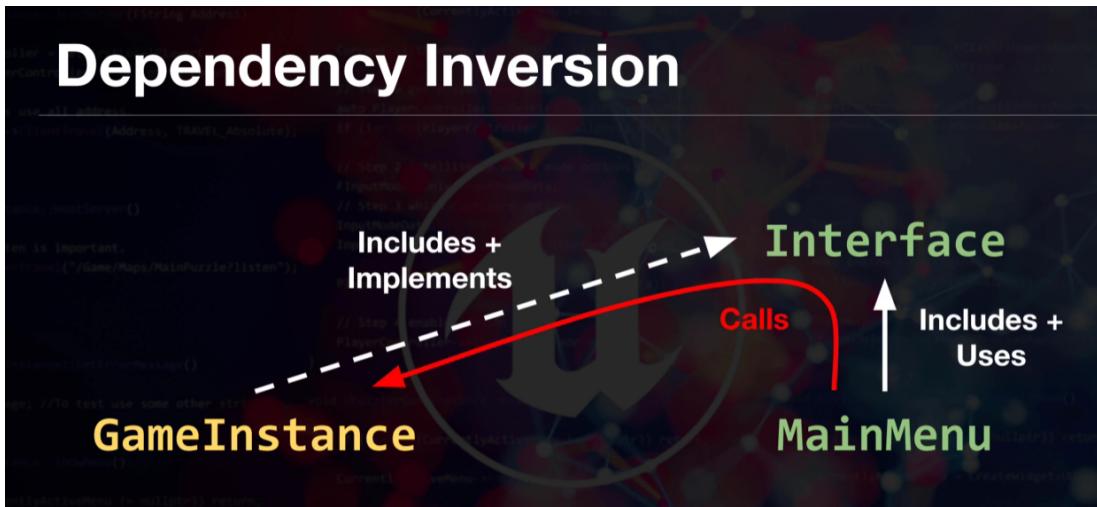


Our Dependencies



Dependency Inversion





Artık tüm işlemlerimizi halleğimize göre birisi host olup diğeri onun vereceği IP adresine bağlanabilir.

Ör: <https://www.youtube.com/watch?v=vXsprCUXH68>

Puzzle Platform

Öncelikle platformu çalıştırduğumızda StaticMeshActor'un ne yapacağını yazıyoruz.

MovingPlatform.h

```

public:
    AMovingPlatform();
    void MovementActive();
    void MovementDeactive();

private:
    void Movement(float DeltaTime);

    // MakeEditWidget command is working for the set location with visual (gizmo)
    UPROPERTY(EditAnywhere, Category= "Movement Settings", meta = (MakeEditWidget = true))
    FVector TargetLocation;
    FVector GlobalTargetLocation;
    FVector GlobalStartLocation;
    UPROPERTY(EditAnywhere, Category= "Movement Settings")
    float Speed = 20.0f;
    UPROPERTY(EditAnywhere)
    int ActivatePlatforms = 1;

protected:

    virtual void Tick(float DeltaTime) override;
    virtual void BeginPlay() override;

```

MovingPlatform.cpp

```

AMovingPlatform::AMovingPlatform()
{
    PrimaryActorTick.bCanEverTick = true;
}

```

```

// set platform movable object
SetMobility(EComponentMobility::Movable);
}

void AMovingPlatform::BeginPlay()
{
    Super::BeginPlay();

    if (HasAuthority())
    {
        // it's opening replicate with movement
        SetReplicates(true);
        SetReplicateMovement(true);
    }

    GlobalStartLocation = GetActorLocation();
    GlobalTargetLocation = GetTransform().TransformPosition(TargetLocation);
}

void AMovingPlatform::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

    if (ActivatePlatforms > 0)
    {
        // check if is true, it's server. Else, it's client
        if (HasAuthority())
        {
            Movement(DeltaTime);
        }
    }
}

void AMovingPlatform::Movement(float DeltaTime)
{
    FVector Location = GetActorLocation();
    if ((GlobalTargetLocation - Location).IsUnit(Speed * DeltaTime))
    {
        Swap(GlobalTargetLocation, GlobalStartLocation);
    }
    FVector Direction = (GlobalTargetLocation - GlobalStartLocation).GetSafeNormal();
    Location += Speed * DeltaTime * Direction;
    SetActorLocation(Location);
}

void AMovingPlatform::MovementActive()
{
    ActivatePlatforms++;
}

void AMovingPlatform::MovementDeactive()
{
    if (ActivatePlatforms > 0)
    {
        ActivatePlatforms--;
    }
}

```

```
}
```

Ne yapacağını bildiğimize göre bunu etkinleştirmesi için bir AActor yaratıyoruz.

PlatformTrigger.h

```
public:  
    // Sets default values for this actor's properties  
    APlatformTrigger();  
  
private:  
    UPROPERTY(EditAnywhere)  
    class UBoxComponent* Trigger;  
  
    UFUNCTION()  
    void OnOverlapBegin(class UPrimitiveComponent* OverlappedComp, class AActor* OtherActor,  
                        class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep,  
                        const FHitResult& SweepResult);  
    UFUNCTION()  
    void OnOverlapEnd(class UPrimitiveComponent* OverlappedComp, class AActor* OtherActor,  
                      class UPrimitiveComponent* OtherComp, int32 OtherBodyIndex);  
    // find all platform on scene and select on editor  
    UPROPERTY(EditAnywhere)  
    TArray<class AMovingPlatform*> TriggerToPlatform;  
  
protected:  
    // Called when the game starts or when spawned  
    virtual void BeginPlay() override;  
    // Called every frame  
    virtual void Tick(float DeltaTime) override;
```

PlatformTrigger.cpp

```
// Sets default values  
APlatformTrigger::APlatformTrigger()  
{  
    // Set this actor to call Tick() every frame. You can turn this off to improve performance if you don't need  
    it.  
    PrimaryActorTick.bCanEverTick = true;  
  
    Trigger = CreateDefaultSubobject<UBoxComponent>(TEXT("Trigger"));  
    if (!ensure(Trigger != nullptr)) return;  
    RootComponent = Trigger;  
  
    Trigger->OnComponentBeginOverlap.AddDynamic(this, &APlatformTrigger::OnOverlapBegin);  
    Trigger->OnComponentEndOverlap.AddDynamic(this, &APlatformTrigger::OnOverlapEnd);  
}  
  
// Called when the game starts or when spawned  
void APlatformTrigger::BeginPlay()  
{  
    Super::BeginPlay();  
}
```

```

// Called every frame
void APlatformTrigger::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
}

void APlatformTrigger::OnOverlapBegin(UPrimitiveComponent* OverlappedComp, AActor* OtherActor,
                                      UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep,
                                      const FHitResult& SweepResult)
{
    if (GEngine)
        GEngine->AddOnScreenDebugMessage(-1, 3.0f, FColor::Yellow, TEXT("Activated"));
    // you can do with int i value but this way more easier than.
    for (AMovingPlatform* Platform : TriggerToPlatform)
    {
        // Calling other page function
        Platform->MovementActive();
    }
}

void APlatformTrigger::OnOverlapEnd(UPrimitiveComponent* OverlappedComp, AActor* OtherActor,
                                    UPrimitiveComponent* OtherComp, int32 OtherBodyIndex)
{
    if (GEngine)
        GEngine->AddOnScreenDebugMessage(-1, 3.0f, FColor::Yellow, TEXT("Deactivated"));
    for (AMovingPlatform* Platform : TriggerToPlatform)
    {
        Platform->MovementDeactive();
    }
}

```

Platformun üzerine geldiğimiz zaman çalışmaya, bıraktığımız zaman ise durmaya başlayacak.

Create OnlineSubsystem

multiplayer steam sdk

build.cs içerisinde "OnlineSubsystem" ekliyoruz

<https://docs.unrealengine.com/4.27/en-US/API/Plugins/OnlineSubsystem/IOnlineSubsystem/>
---> koda ekleme

<https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Online/#interfaces>
---> inceleme

instance dosyasında init kısmında

```

IOnlineSubsystem* SubSystem = IOnlineSubsystem::Get();
if (SubSystem != nullptr)
{

```

```

        UE_LOG(LogTemp, Warning, TEXT("Found subsystem %s"),
*SubSystem->GetSubsystemName().ToString());

    }else
    {
        UE_LOG(LogTemp, Warning, TEXT("Found no subsystem %s"));
    }

```

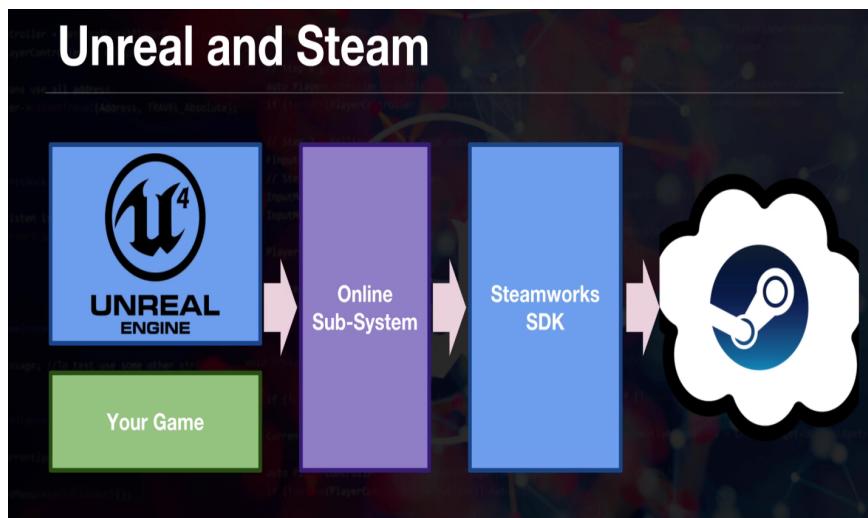
ile subsystem'imizi tanımlamış oluyoruz. Bunun sayesinde inceleme kısmında verdiğim linkte bulunan kısımlara erişimimiz oluyor.

DefaultGameEngine.ini dosyasına

[OnlineSubsystem]

DefaultPlatformService=NULL

ekliyoruz.



[--- Create Session Interface](#)

subsystem null değer döndürmüyorsa oturumumuzu oluşturabiliriz bunun için:

```
const IOnlineSessionPtr SessionInterface = SubSystem->GetSessionInterface();
```

```

    // check session is valid
    if(SessionInterface.IsValid())
    {
        UE_LOG(LogTemp, Warning, TEXT("Found session interface"));
    }

```

Stack is a linear data structure whereas Heap is a hierarchical data structure

<https://docs.unrealengine.com/4.27/en-US/API/Runtime/Core/Templates/TSharedPtr/>

Not: NULL subsystem geliştirme kısmında test etmemize yarar.

--- Create Session

<https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/ProgrammingWithCPP/UnrealArchitecture/Delegates/Multicast/>

<https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Online/SessionInterface/>

Server cevap beklerken diğer işlemlerin de aynı şekilde yürütmesi için delegate kullanıyoruz.

```
void UPuzzlePlatformsGameInstance::OnCreateSessionComplete(FName SessionName, bool Success)
```

```
void UPuzzlePlatformsGameInstance::OnDestroySessionComplete(FName SessionName, bool Success)
```

şeklinde 2 delegate oluşturup eğer session yoksa oluşturuyoruz. Var ise oyunu kuruyoruz (host oluyoruz).

Delegate kısımlarında bunu hem kontrol hem oluşturma kısmını yaparken host butonuna bastığımızda çağrıdığımız host fonksiyonu bunlardan hangisinin çağrılacağını belirliyor.

Yani direkt host fonksiyonunda interface doğruluğunu kontrol edip ardından session oluşturma-silme işlemini çağrıyoruz.

Steam bağlantısı için steam online subsystem'i aktif ediyoruz.

Build.cs içerisinde OnlineSubsystemSteam'i ekliyoruz.

DefaultEngine.ini içerisinde DefaultPlatformService kısmını Steam yapıyoruz.

TOptional: Veriyi üye değişken olarak saklar.

TSharedPtr: Veriyi başka yerde saklar ve onu işaret eder.

OnlineSessionInterface ile bağlanma sıralaması:

HostSession FindSession JoinSession GetResolvedConnectString ClientTravel

Log LogOnline Verbose: Online bağlanırken bunun log'unu yazdırır.

Steam OSS bağlantısı için: Steam OSS Plugin – OnlineSubsystemSteam modülü gereklidir.

SessionSettings.bUsesPresence = true;

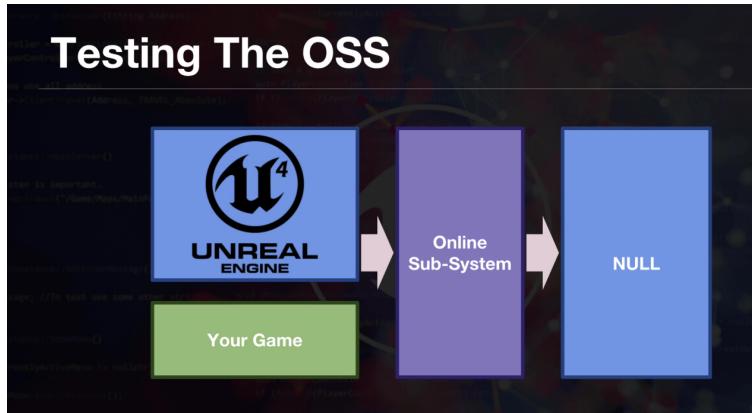
Presence'yi aktif etme sebebimiz Steam lobilerini kullanmak için eğer kullanmazsak lan game olarak oynayabiliriz.

-nonsteam yazarak çalıştırırsak steam ihtiyaç duymadan test edebiliriz.

```

// if you want to test NonSteam and Steam version, you need to add this conduction.
// for the online game, delete this conduction and add to only false version.
if(IOnlineSubsystem::Get()->GetSubsystemName() == "NULL")
{
    SessionSettings.bIsLANMatch = true; // local gameplay is true
}
else
{
    SessionSettings.bIsLANMatch = false; // local gameplay is false
}

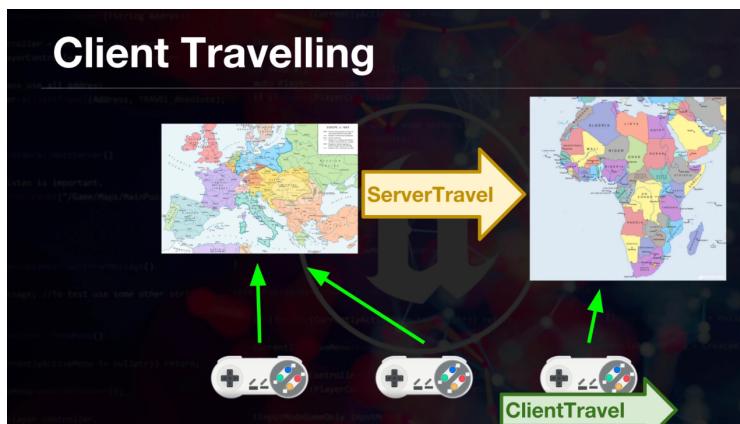
```



Non-Seamless Travel: Tüm kullanıcıları ilk önce oyundan çıkarır (disconnect) sonra tekrar bağlar. Bundan dolayı haritayı yüklerken oyun donar ve yüklenince tekrar akıçılır.

Seamless Travel: Kullanıcıları alıp tekrar yüklerken geçici (transition) map kullanır bu sayede donma yaşanmaz. Bu geçiş haritası 2 büyük harita arasında yükleme yaparken bir loading map veya küçük harita olabilir. Aktif etmek için:

bUseSeamlessTravel = true; kullanılır.



UFUNCTION(NetMulticast, unreliable, WithValidation)

Validation kısmı oyuncunun gönderdiği değerin null olmaması için işlemi keserek önlem sağlıyor.

Unreliable kısmı uzaktan müdahale edilmesini engellemek için.

NetMultiCast: suncudan bilgiyi alıp tüm oyunculara dağıtmak için genel kullanım.

Server: bilgi client'de çağrılmak fakat suncuda yürütülecek.

Client: Sunucudan bilgiyi alıp client'de yürütme işlemini yapar.

Not: Bunu herhangi bir fonksiyona ekledikten sonra o fonksiyonun _Implementation

Tipini de yapmamız gerekiyor. Bunun sayesinde iletişimini sağlamış oluyoruz.

Not2: Validation işlemi için de _Validate gerekiyor. Aynı zamanda bunları yapmak hile koruması da sağlar.

Server içerisindeki görevimizi (rol) görmek için:

```
 FString GetEnumText(ENetRole Role)
{
    switch (Role)
    {
        case ROLE_None:
            return "None";
        case ROLE_Authority:
            return "Authority";

        case ROLE_AutonomousProxy:
            return "AutonomousProxy";
        case ROLE_SimulatedProxy:
            return "SimulatedProxy";
        default:
            return "ERROR";
    }
}
```

```
DrawDebugString(
    GetWorld(), // world space
    FVector(0,0,100), // start vector
    GetEnumText(GetLocalRole()), // end vector
    this,
    FColor::White, // color
    DeltaSeconds
);
```

Değerlerin replicate edilmesi için:

Öncelikle aktörün replikasyonunu açıyoruz bunun için constructor kısmına

```
bReplicates = true;
ardından değeri ve replicated fonk. yazıyoruz.
```

```
UPROPERTY(ReplicatedUsing = OnRep_ReplicatedTransform)
FTransform ReplicatedTransform;
```

```
void AsquidgameCharacter::OnRep_ReplicatedTransform()
{
    //UE_LOG(LogTemp, Warning, TEXT("Replicated Transform"));
    SetActorTransform(ReplicatedTransform);
}
```

```
void AsquidgameCharacter::GetLifetimeReplicatedProps(TArray<FLifetimeProperty>& OutLifetimeProps) const
{
    Super::GetLifetimeReplicatedProps(OutLifetimeProps);

    DOREPLIFETIME(AsquidgameCharacter, ReplicatedTransform); // registering the value
}
```

Sunucu testi yaparken:

Sanal bir şekilde packet loss, lag vb. şeyler için host olduğumuz oyunda konsolu açıp NetEmulation sonrasında

Setting	Description
PktLag	Delays the sending of a packet by the amount of time specified in milliseconds
PktLagVariance	Provides some randomness to the amount of time a packet is delayed, +/- the amount specified in milliseconds
PktLoss	Specifies a percentage chance of an outbound packet being discarded to simulate packet loss
PktDup	Specifies a percentage chance to send a duplicate packet
PktOrder	Sends packets out of order when enabled (1 = enabled, 0 = disabled)

birini kullanıyoruz.

Game Start with CMD

Unreal Engine dosyasında engine/binaries içerisinde [UE4Editor.Exe](#) adresi ve projenin [.uproject](#) doyasının adresinin arasına boşluk koyup -game dediğimiz zaman konsola direkt oyunu başlatıyor. “-game” sonrası [-log](#) koyup başlatırsak oyun oynanırken olan tüm işlemleri gösteriyor.

Farklı bir map ile başlatmak için [/Game/Oyunİsmi/HaritaDosyası/Mapİsmi](#) olarak -game öncesine yazıp başlatıyoruz.

Multiplayer server'e bağlanmak için [-game](#) yerine [-server](#) yazıp başlatıyoruz.

Multiplayer oyunu başlatmak için [local ip adresini](#) yazıp daha sonrasında -game yazarak başlatıyoruz.
(eğer 2 kere aynı komutu girersek aynı dünyaya bir oyuncuyu daha gönderir)

Bunun sayesinde crash vs. yediğimizde oyunun nerede hata verdiğiniz bulup düzeltebiliyoruz.

- Server Type Check

```
// check if is true, it's server. Else, it's client
```

```
if (HasAuthority())
```

```
{
```

```
FVector Location = GetActorLocation();
```

```
Location += FVector(5 * DeltaTime, 0, 0);
```

```
SetActorLocation(Location);
```

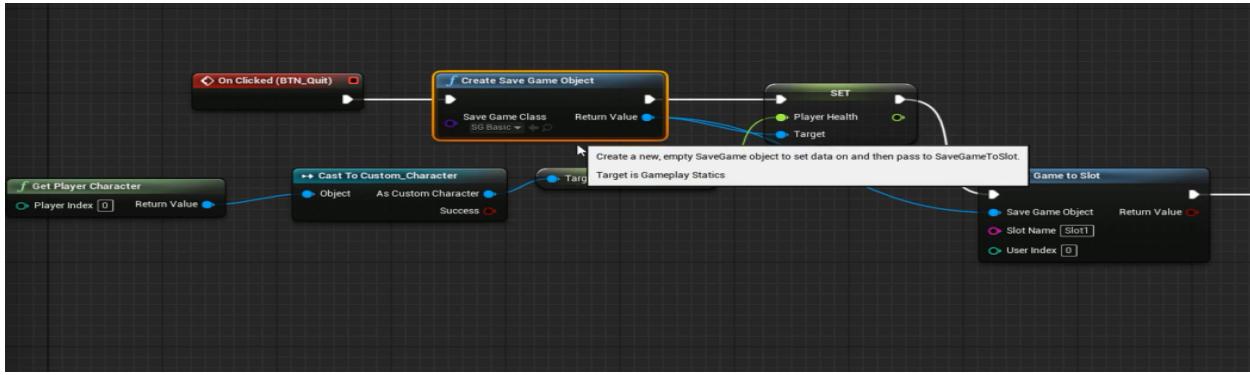
```
}
```

Gameplay Ability Plugin:

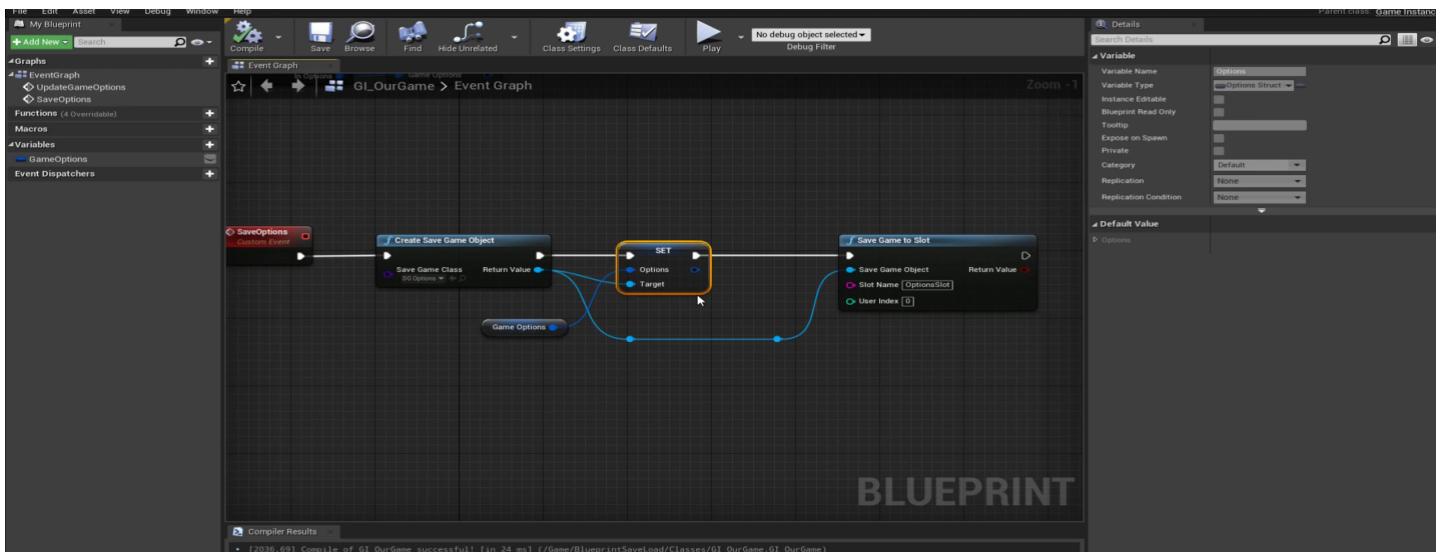
Oyun içi mana, can veya güç veren iksir, yetenekleri yönetmek için gameplay ability plugin kullanıyoruz. Buna herhangi bir iksir vs. eklemek için sadece data asset oluşturmak yeterli.

Asset Manager: Oyun içinde itemleri, iksirleri vb. itemleri tutmak için c++ FPrimaryAssetType ile asset manager oluşturup defaultEngine.ini dosyasına AssetManagerClassName = /Script/GameName.DosyalsmAssetManager şeklinde ekleme yapıyoruz. Artık Project Settings -> Asset Manager kısmında eklediğimizi görebiliriz. Oyun içinde bunları kullanmak için Blueprint kısmında tüm itemleri for each loop döngüsü ile alıp Get Primary Id List ile isimleri eşleşiyormu diye kontrol ederiz ve çıktıları async load primary asset list ile tekrar for each loop'a alıp ona göre stored items listesine ekleyip tutarız. Bunun sayesinde tüm itemlerimizi bilmiş oluruz.

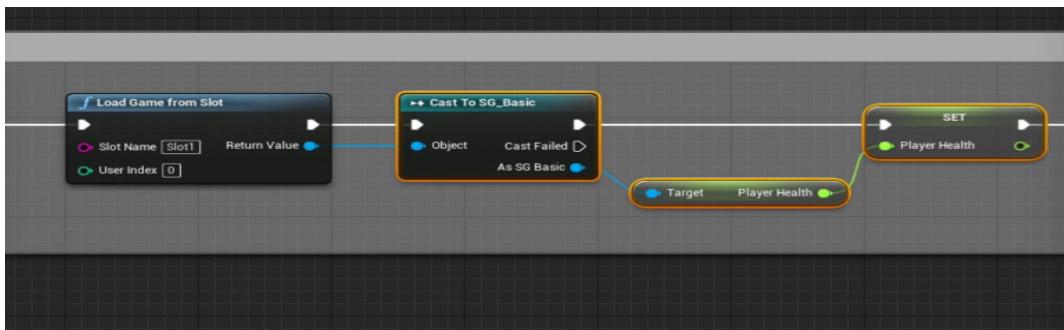
BP Create Save



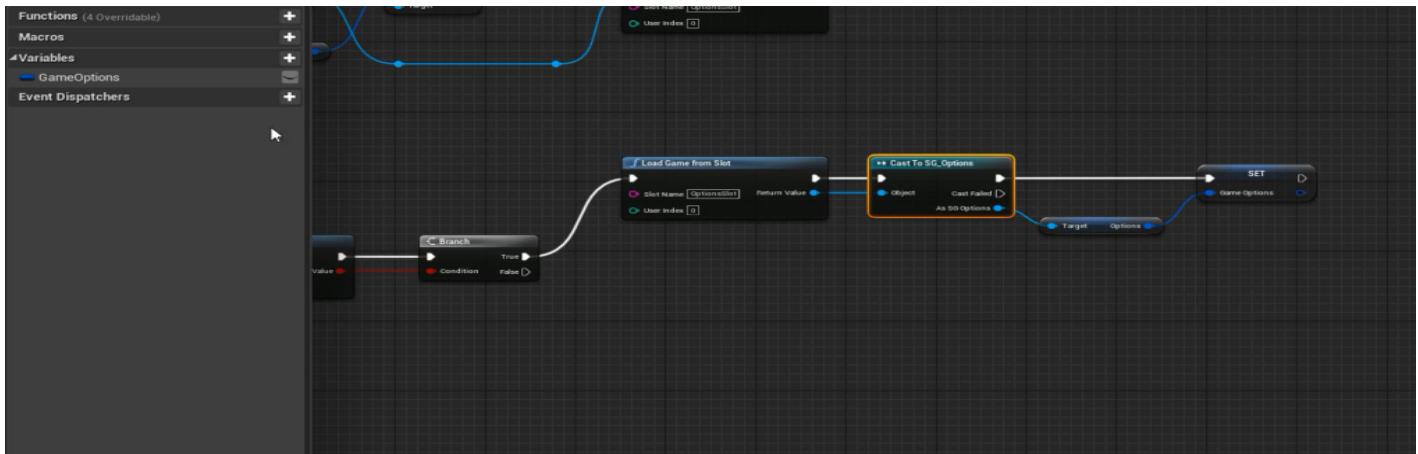
BP Create Game Options



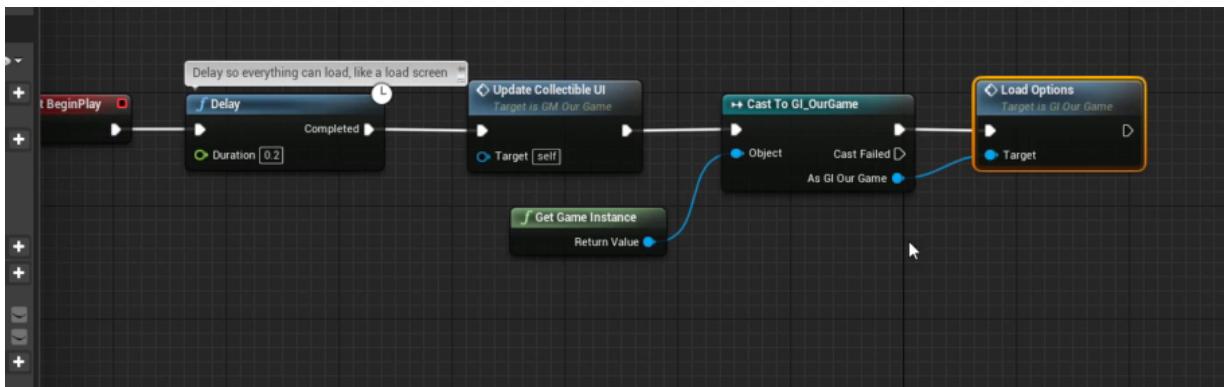
BP Load Save



BP Load Options

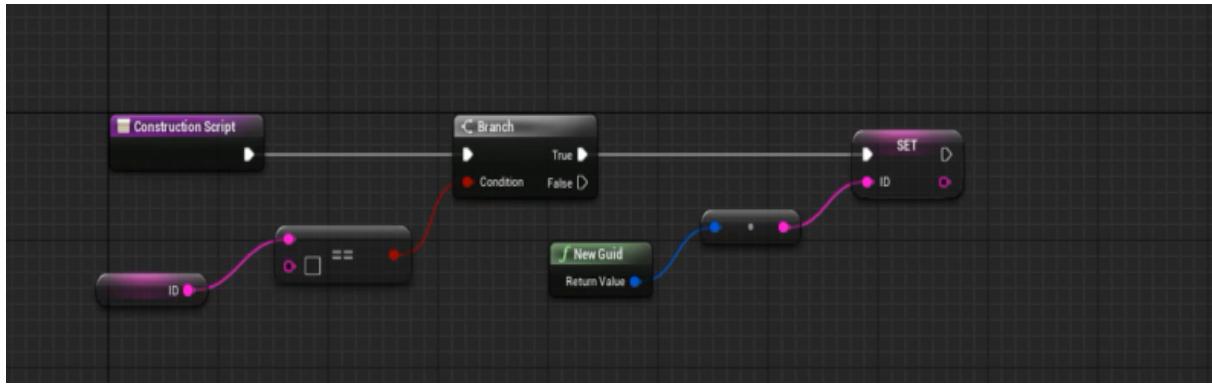


BP Set Game Options

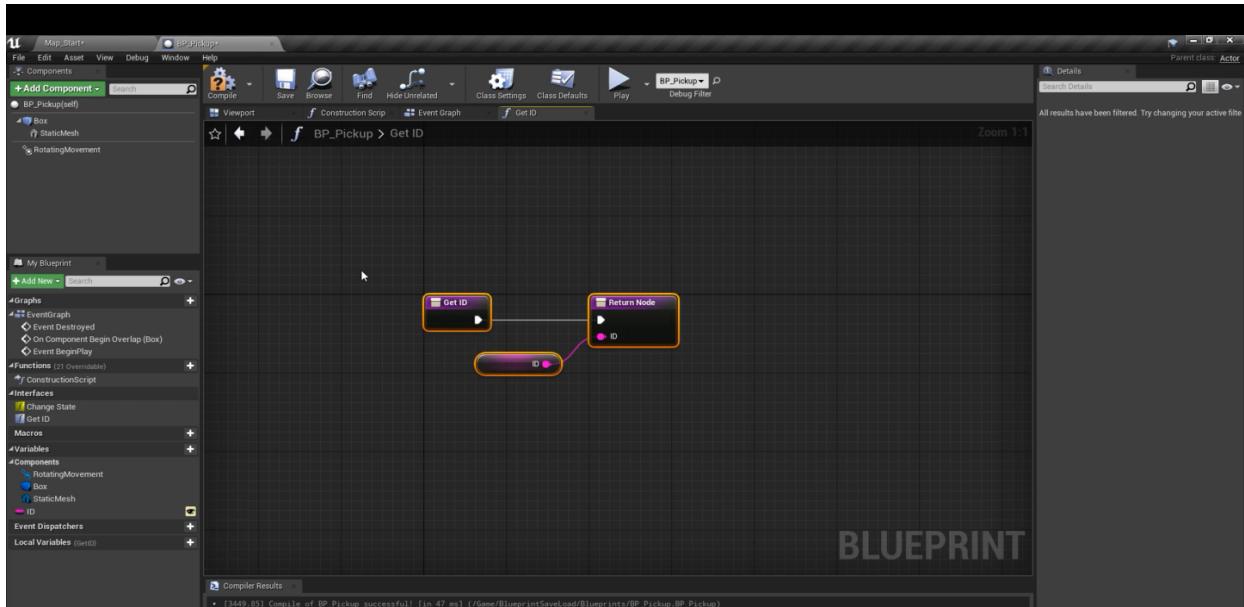


BP Create Collectable Object

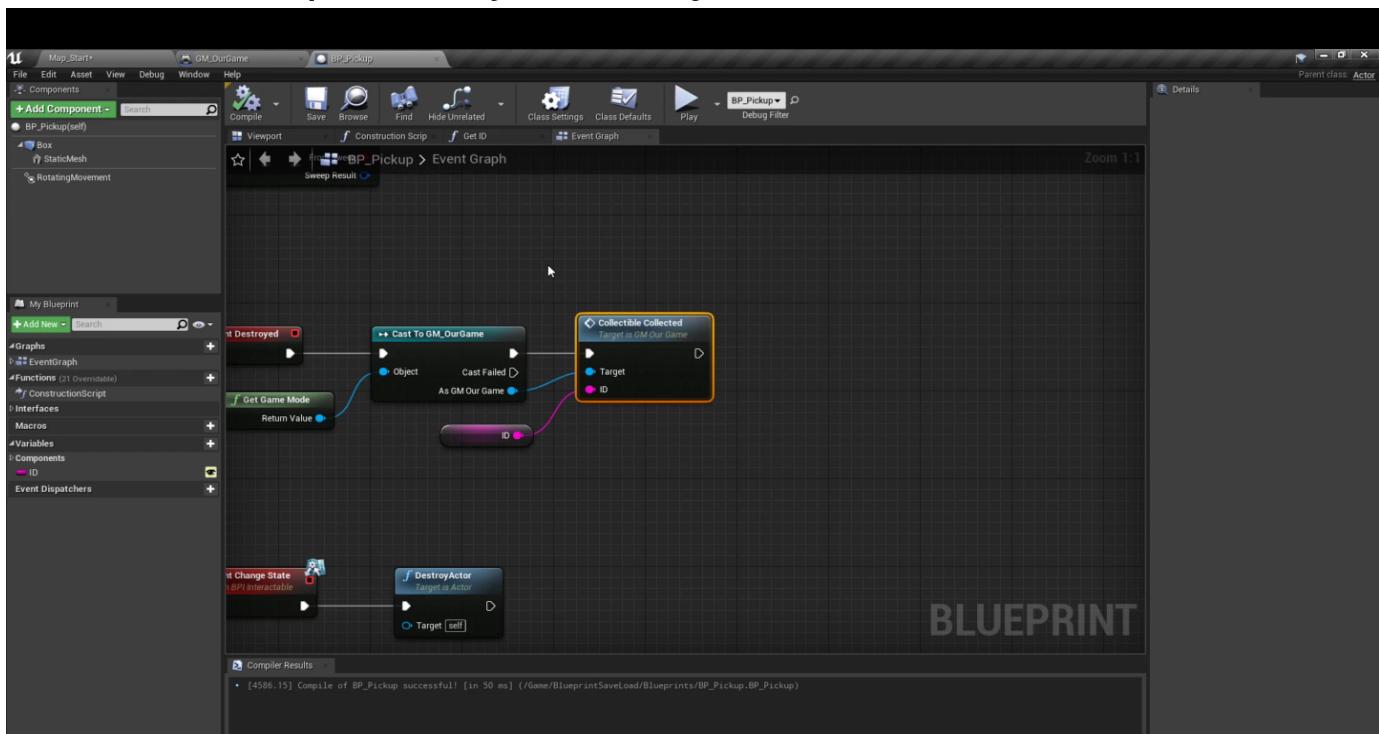
1. Create UniqueID



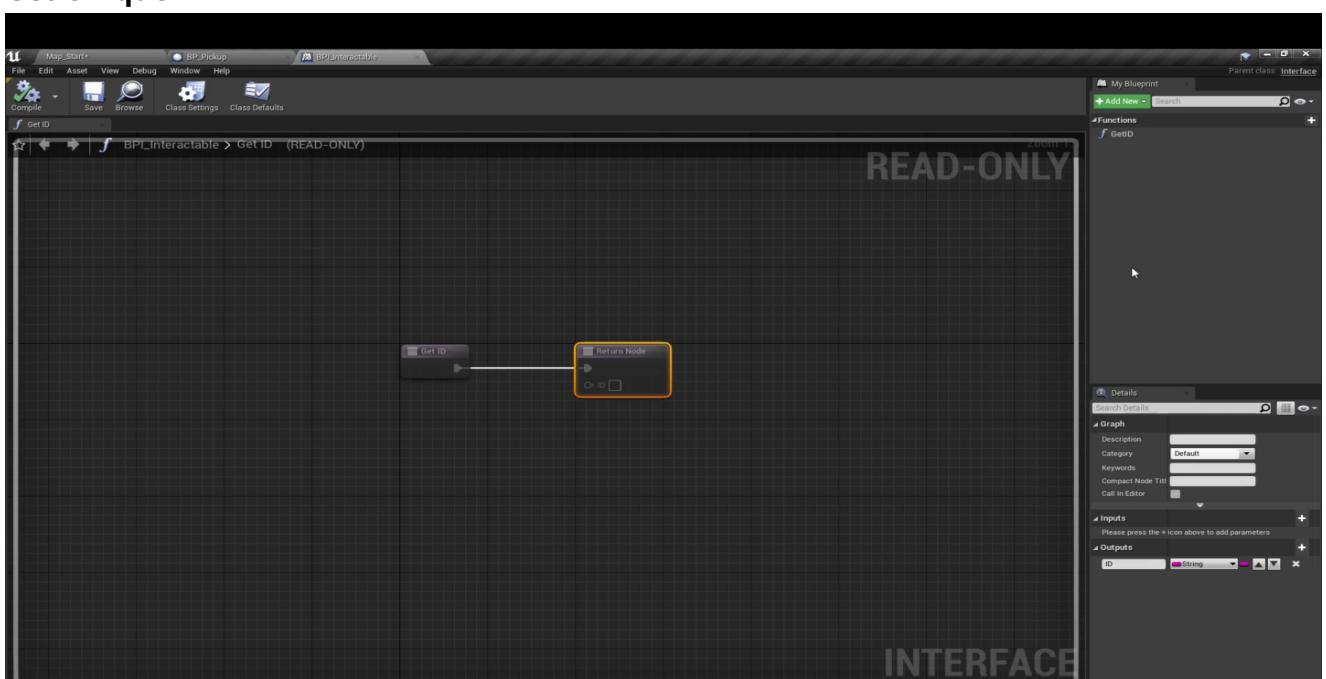
2. Return UniqueID



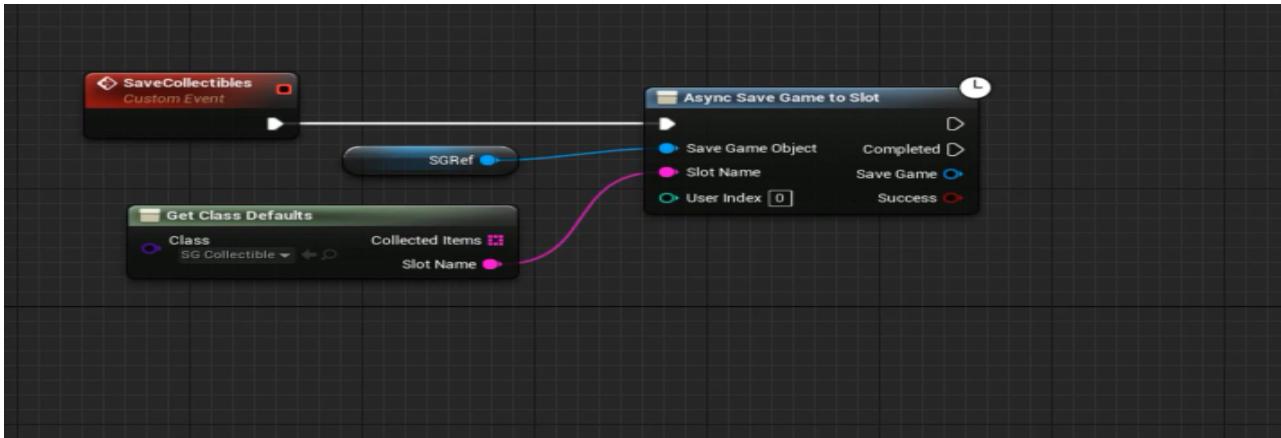
3. Save Collectable UniqueID (when you hit the object)



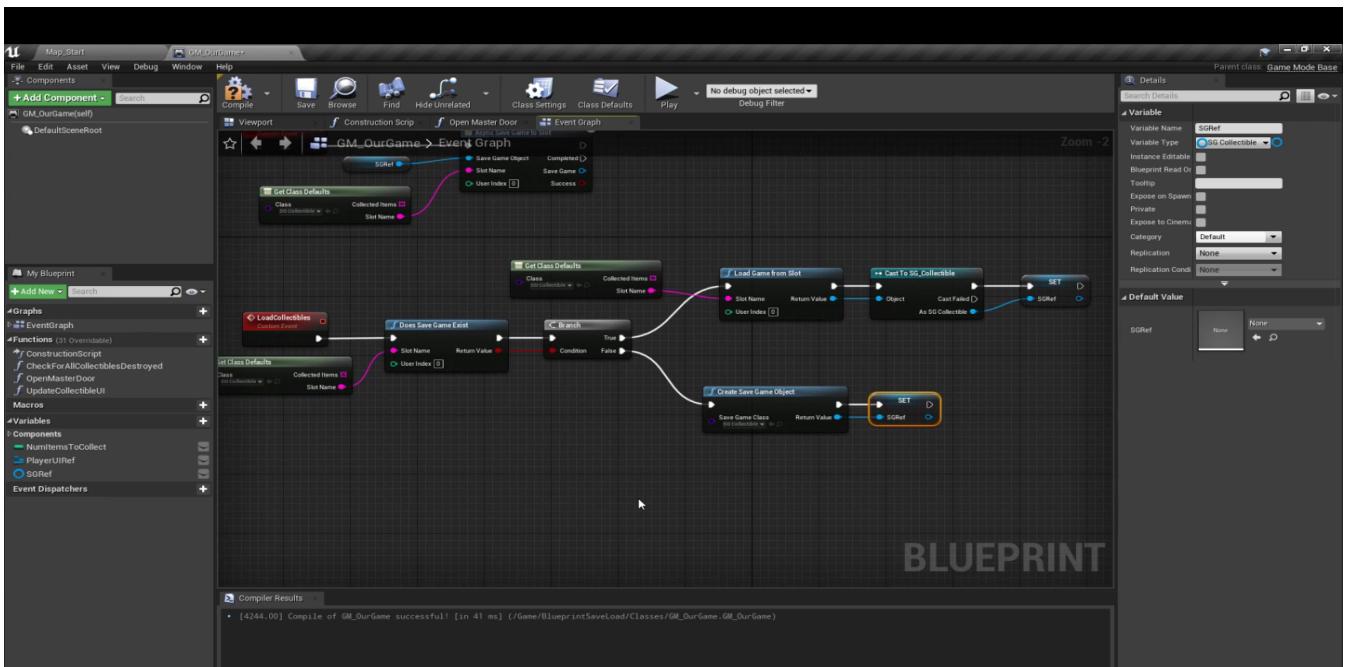
4. Get UniqueID



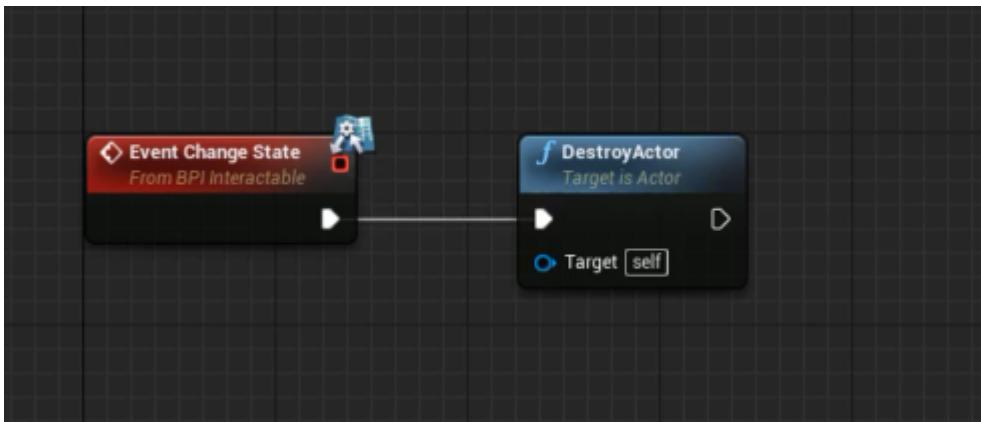
5. Save Collected Object



6. Load Collected Object



7. Level Blueprint Change State



Unreal Engine Diving Levels Course

This method for the load or unload environment.

- 1) windows -> levels menu
- 2) levels -> new -> select empty level -> level name C0_sublevel
- 3) select all meshes -> right click to c0_sublevel on levels menu -> select move

Now when you click the eye icon, all static meshes will hide or unhide.

---- Loading Screens

Options for the loading screens:

- 1) Fade to solid color (most common solution but users shouldn't stay in a faded state for more than 5 seconds or they might think the application has frozen.)

<https://docs.unrealengine.com/en-US/BlueprintAPI/CameraFades/StartCameraFade/index.html>

- 2) Show VR splash screen (Engine-supported, easy to use as long as animation or interaction isn't required during the loading screen.)

<https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/XRDevelopment/VR/VRHowTos/XRLoadingScreens/>

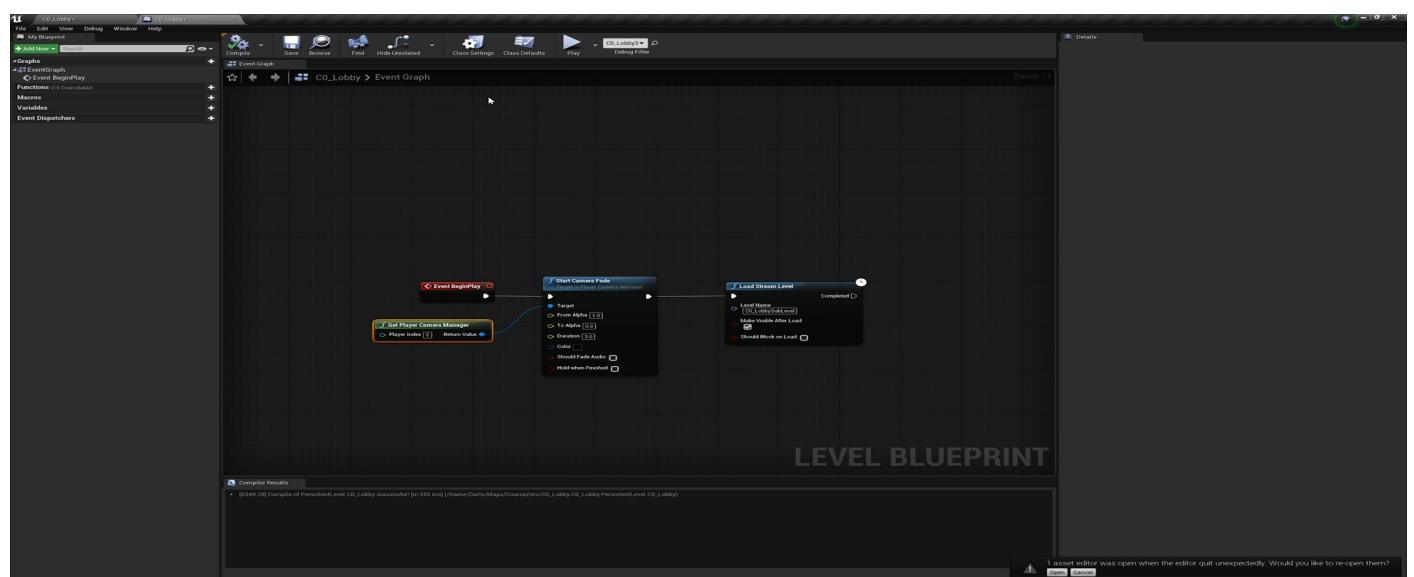
- 3) Show one or more stereo layers (most powerful option but might also take the most work)

<https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/XRDevelopment/VR/VRHowTos/StereoLayers/>

Example of fade to solid color method:

c0_lobbymap

Open Level Blueprint



Example of VR Splash Screen:

c4_lobby

BP_TravelTo (for the switch between levels)

When you move the level load are it will show the splash screen. After waiting the splash screen, it will load level.

Example of Cache method:

On Quest, shaders are not actually compiled until they are used in a draw call. This causes the first frame to hitch as the first frame gets rendered.

One single shader compilation can cause the app to fail the oculus certification requirement of 13.88 ms.

- 1)** Open the engine/source folder -> search shaderpipelinecache.cpp and open -> Under the FShaderPipelineCache bool method, change FPipelineFileCache::SetGameUsage... to FShaderPipelineCache.
- 2)** In the editor go to edit -> project settings -> set "share shader material code" and "shared material native libraries" option the true.
- 3)** Windows -> Developer tools -> open device profiles -> click the android -> Under the Console Variables click to Rendering + icon -> search "r.ShaderPipelineCache.Enabled" -> set the value 0 to 1 -> close editor.
- 4)** open Engine/config/android/AndroidEngine -> save and close.

5) In the editor, open the project launcher -> press the + icon -> project section click the browse -> select project.uproject -> cook section select "by the book" than set true "Android", "Android_ASTC" -> scroll down and cooked maps section, set true all of the map. -> Scroll down to launch section and under the initial map section , select c0_lobby (starting level this course) -> under the "additional command line parameters" type "-logPSO -NoVerifyGC" -> Scroll up and rename the profile "Android PSO Cache".

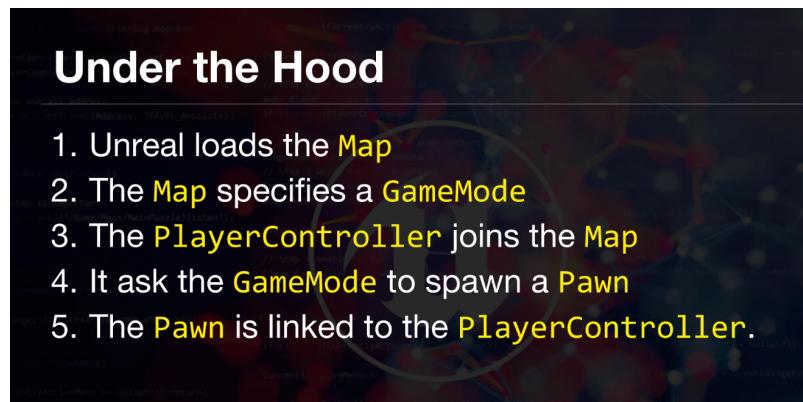
Note: Before the pressing the launch this profile button, make sure that you have your target device setup and connected your dev. PC.

- 6)** Open the command line and write "adb shell mkdir -p yourproject/Saved/CollectedPSOs"
- 7)** Create PSOCaching folder -> open yourprojectfolder/saved/cooked/android_astc/projectname/metadata/PipelineCaches and copy two ShaderStableInfo file into created PSOCaching folder.
- 8)** Open the project/saved/CollectedPSOs -> copy UE4-DEV... cache file and paste PSOCaching folder -> open the engine/binaries/win64 -> Create UE4Editor-Cmd.exe shortcut and move PSOCaching folder -> press the right click and open the properties than in the target section, add Z:/ProjectName -run=ShaderPipelineCacheTools expand C:/PSOCaching/*.rec.upipelinecache C:/PSOCaching/*.scl.csv ProjectName_GLSL_ES3_1_ANDROID.stablepc.csv and press the ok button -> run the shortcut file -> it will create csv file on this folder. (if it didn't create file on this folder, you need the check engine binaries folder.)
- 9)** Open the PSOCaching folder and copy ProjectName_GLSL_ES3_1.stablepc.csv file -> open the ProjectName/build/Android folder and create PipelineCaches -> paste the file in here
- 10)** In the editor, build with File-> PackageProject -> Android -> Android (ASTC)

Recommendations: Check in the stablepc.csv file into source control , there is no need to update this file during development.

Note: Before product ships or major milestones, re-run steps 2 to 5 to update the stablepc.csv file to make sure all shaders used are included, as well as make sure there are no unused shaders in the PSO cache.

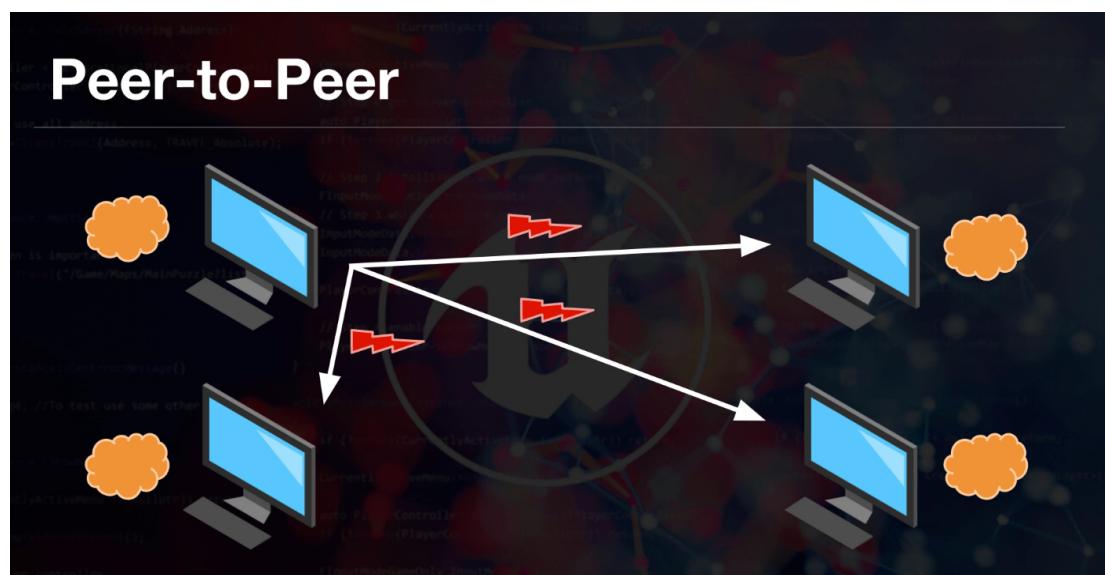
Unreal Load Game Hierarchy



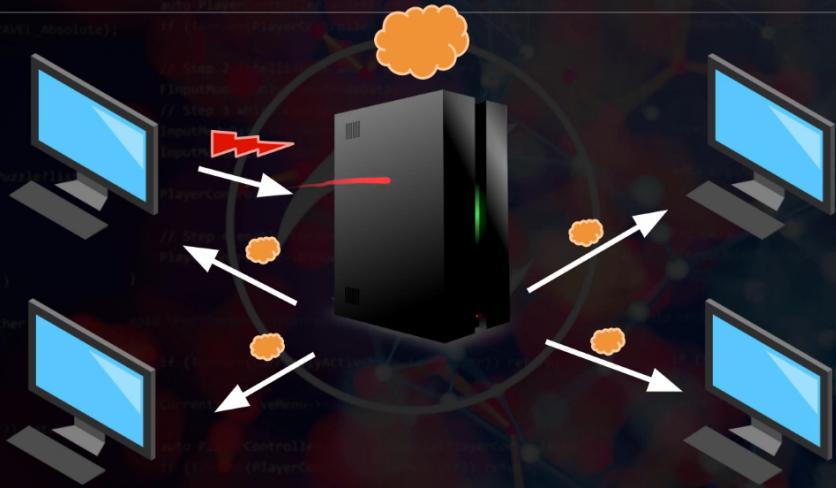
Multiplayer Server Type Selection

Comparison for multiplayer game Unreal Engine support.

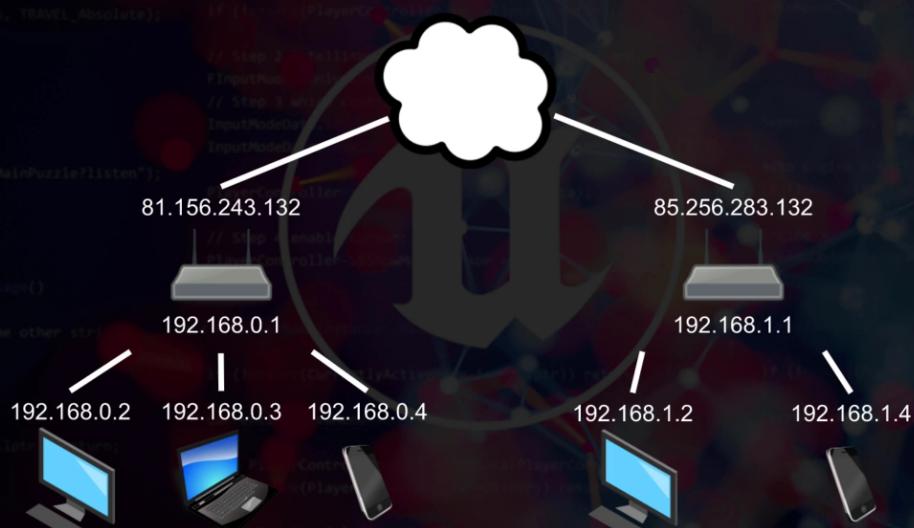
	Synchronous	Session Length	Indie Suitability	Unreal Support
Turn-based	✗	Variable	Excellent	Minimal
Real-time session-based	✓	< 1 hour	Good	Excellent
MMO and Persistent World	✓	Potentially infinite	Poor	Minimal



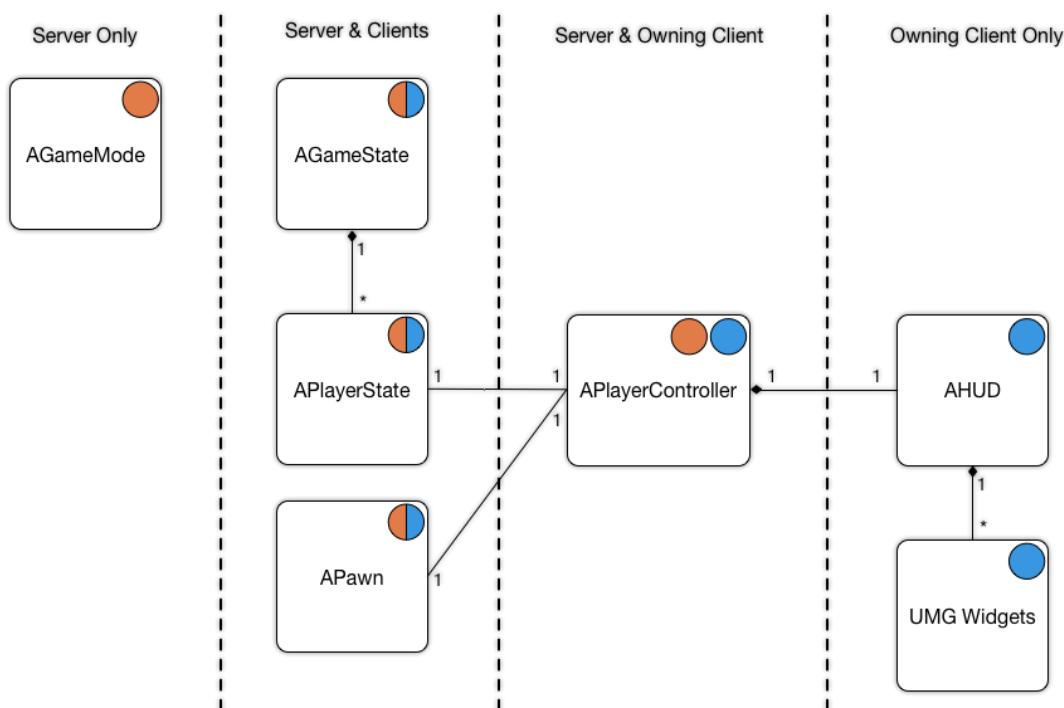
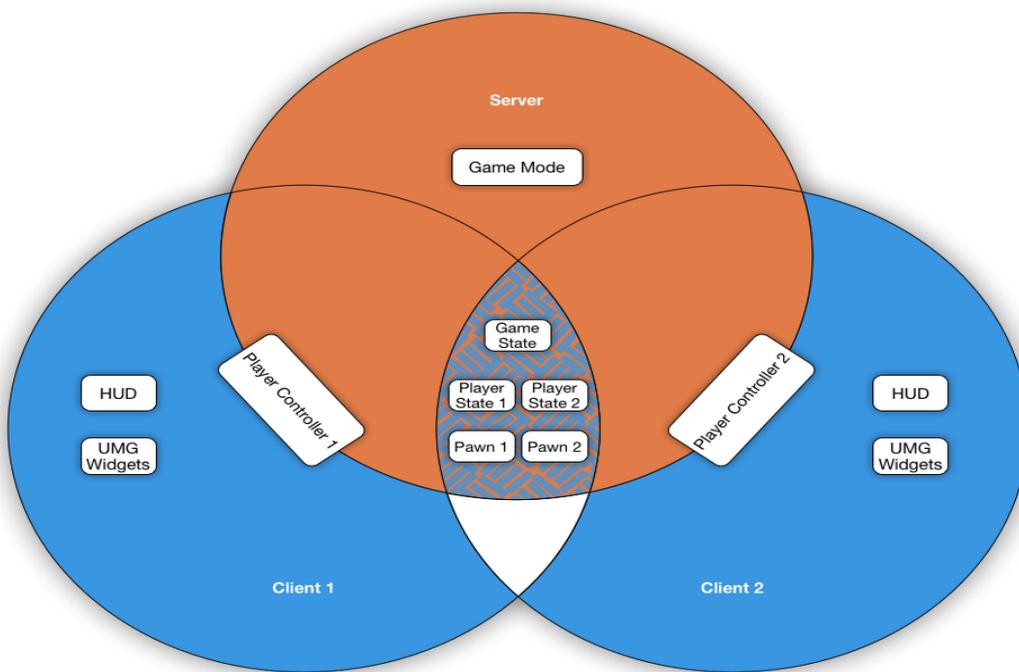
Client-Server



The Ugliness Of NATs

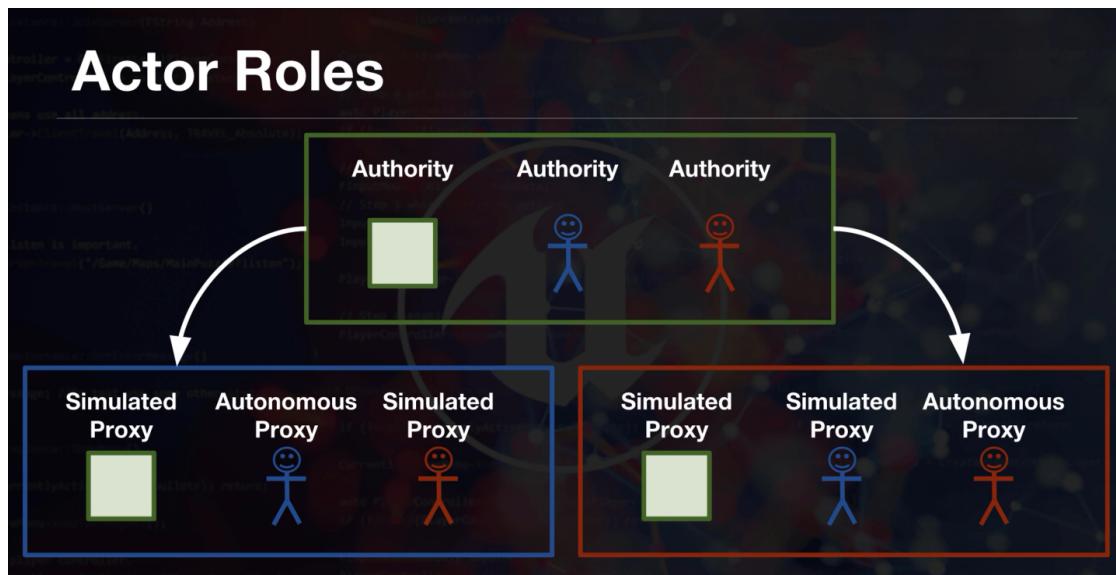


Let's divide section by section

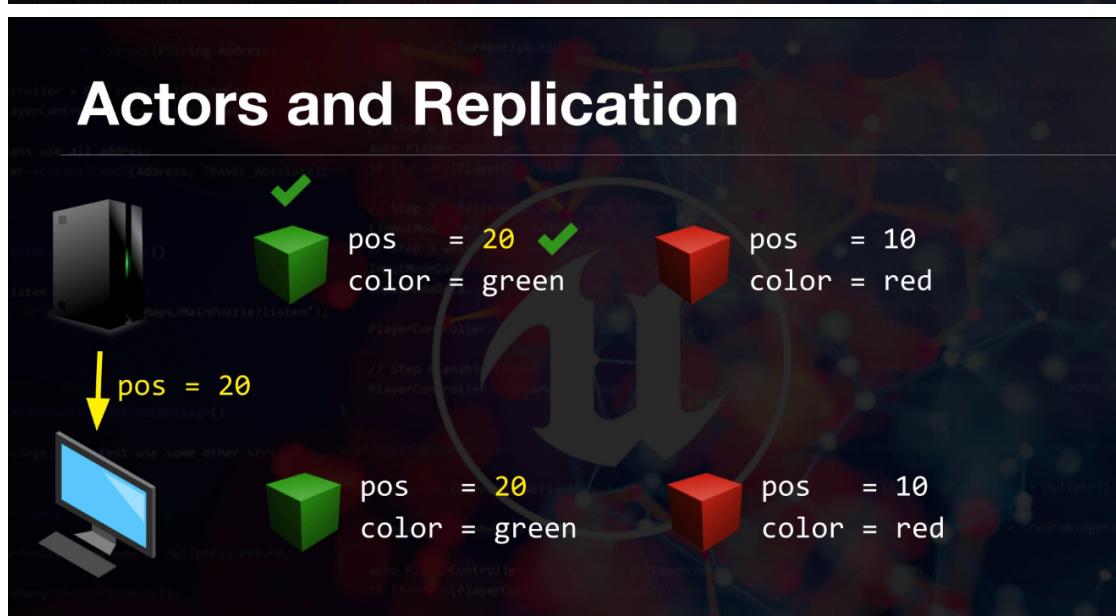


Multplayer Replication - LAG - Package Loss

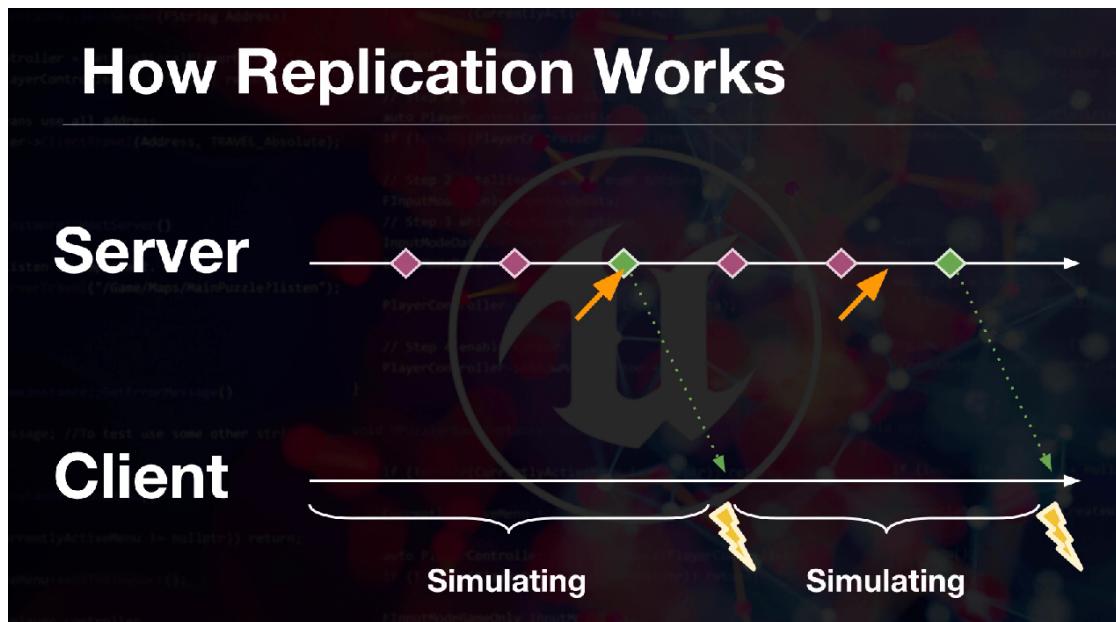
Replication Actor Roles



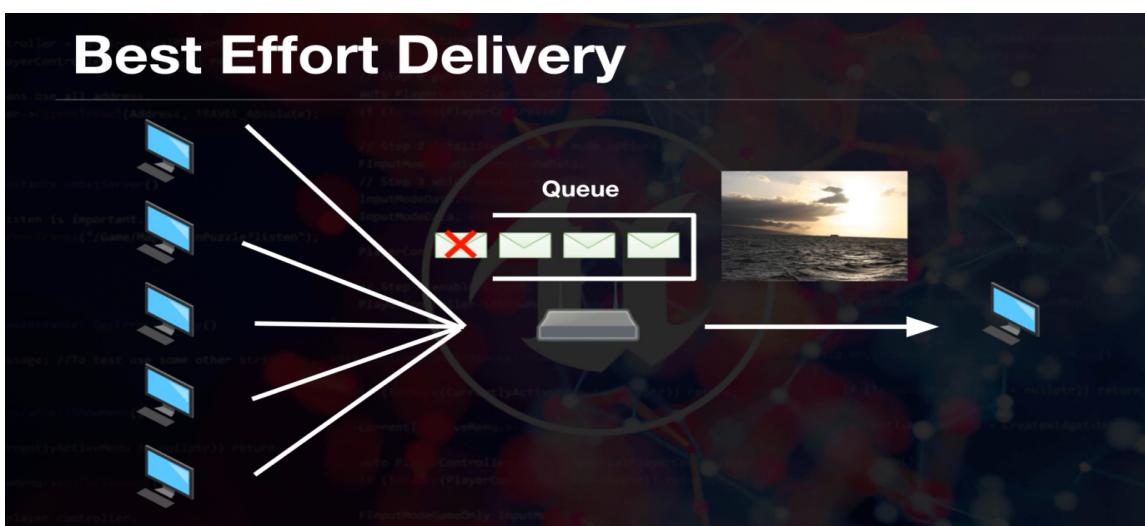
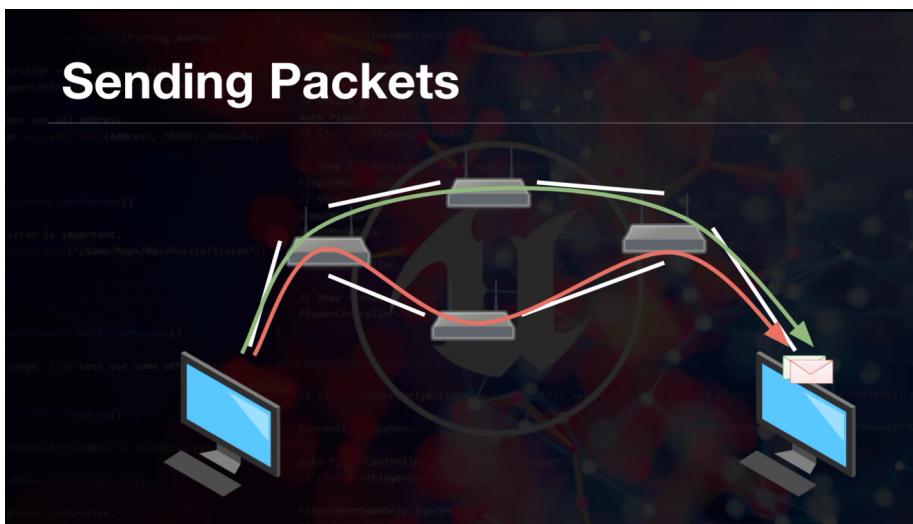
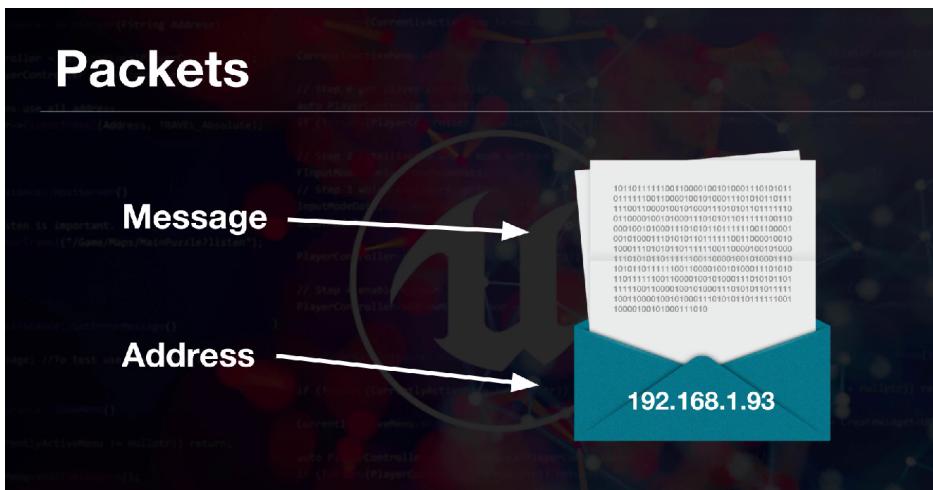
Actors and Replication



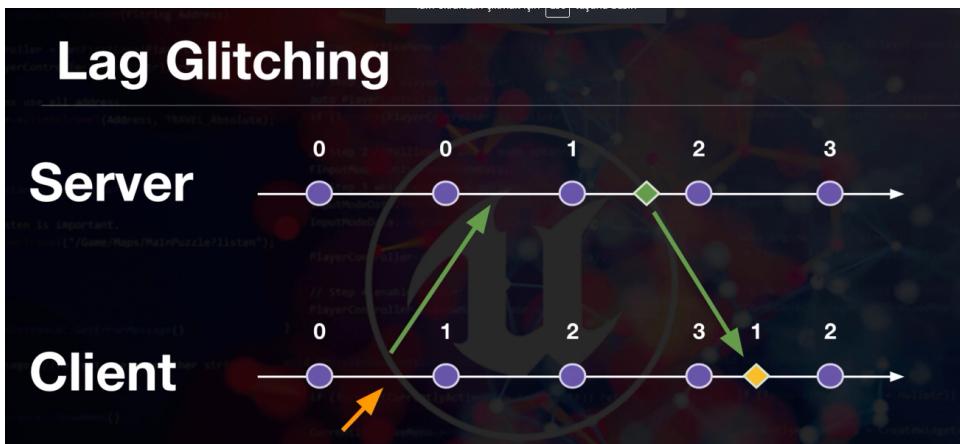
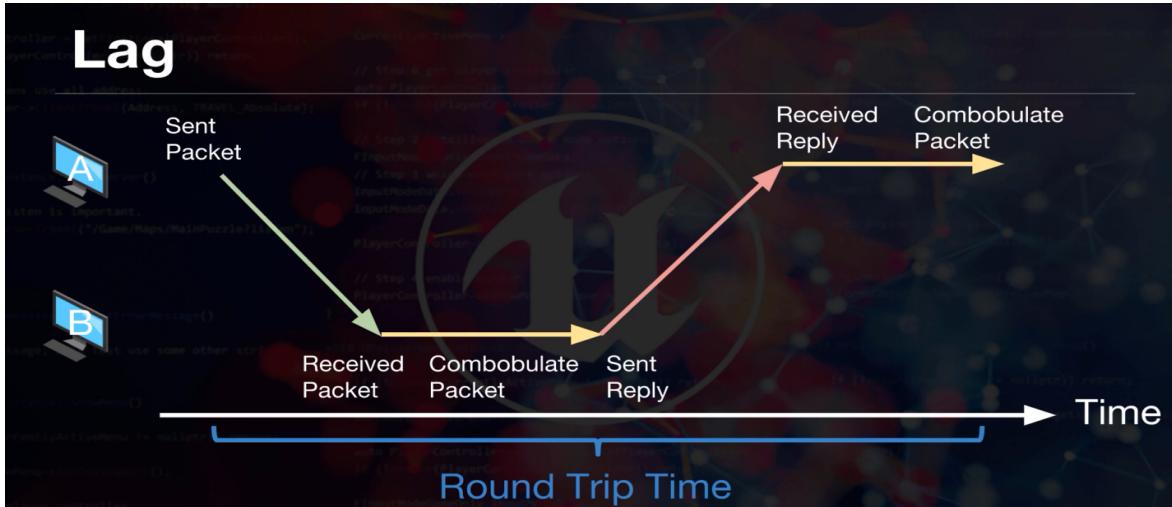
How Replication Works



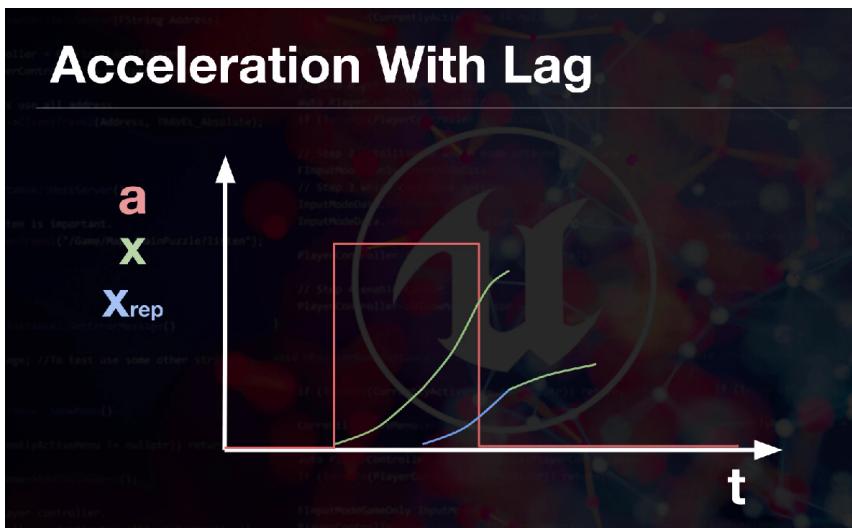
What is the packet loss ?



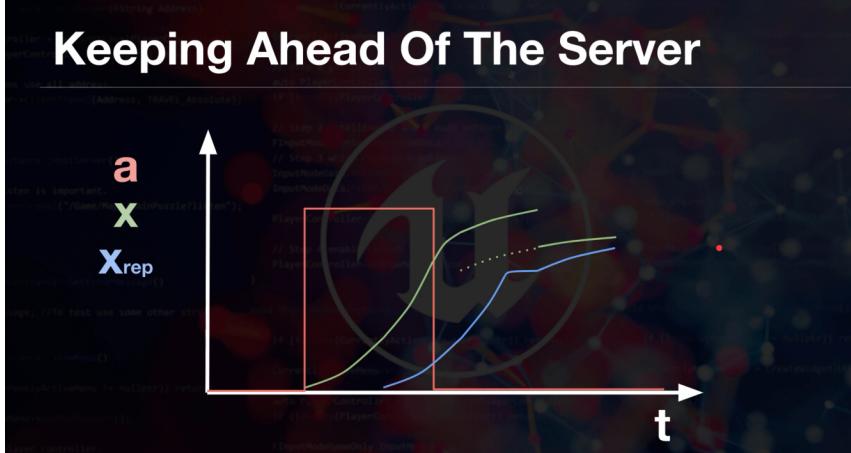
What is the LAG ?



Let's visualize the lag



Keeping Ahead Of The Server



Fix the LAG Methods

Comparison Of Methods

	v1	v2	v3
Problem	Not smooth	Lag	-
Information sent to server	Throw		
Between updates	Do nothing	Simulate	
Information received	Transform, Velocity		Transform, Velocity, ServerTime
On Receipt	Overwrite local		Replay controls since ServerTime

Solutions:

The High Level

- **OnTick:**
Create a move and send to the server.
- **OnReceiveMove:**
Simulate it on the server.
- **OnReceiveServerState:**
Replay local moves on top.

Let's start!

What We Already Have (Sort Of)

OnTick

1. Create a new Move,
2. Save to a list of unacknowledged moves,
3. Send the move to the server,
4. Simulate the move locally.

OnReceiveMove

1. Check that the move is valid, (No cheating!)
2. Simulate the move,
3. Send the canonical State to the clients.

OnReceiveServerState

1. Remove all moves included in state,
2. Reset to server state,
3. Replay/simulate unacknowledged moves.

Replication Refactor for movement bindings

Refactor Plan

GoKart Actor

- Input binding

Movement Component

- Simulating physics

Replication Component

- Replicating movement

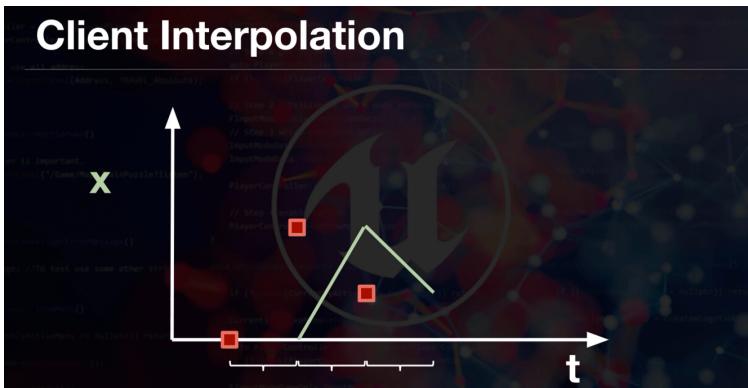
What is the correct smooth movement graph ?

Linear Interpolation (Lerp)

A

B

Client Interpolation



Early Updates



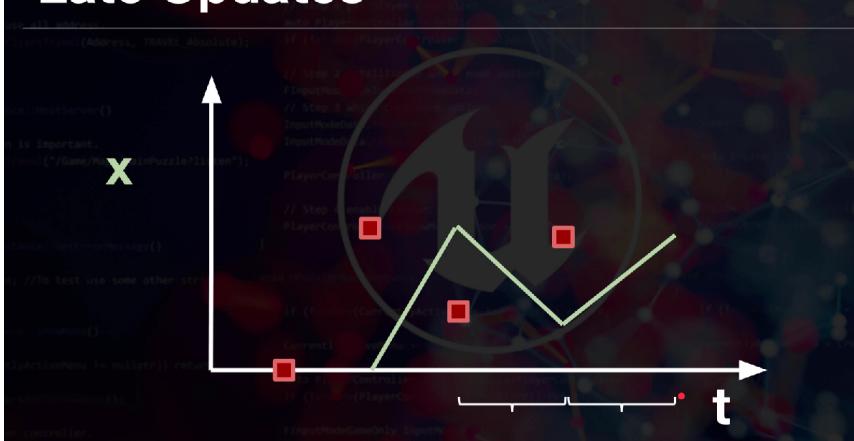
OnTick:

```
TargetLocation = ServerState.Location  
LerpRatio = TimeSinceUpdate / TimeBetweenLastUpdates  
NextLocation = Lerp(StartLocation, TargetLocation, LerpRatio)  
SetLocation(NextLocation)
```

OnRep:

```
StartLocation = GetLocation()
```

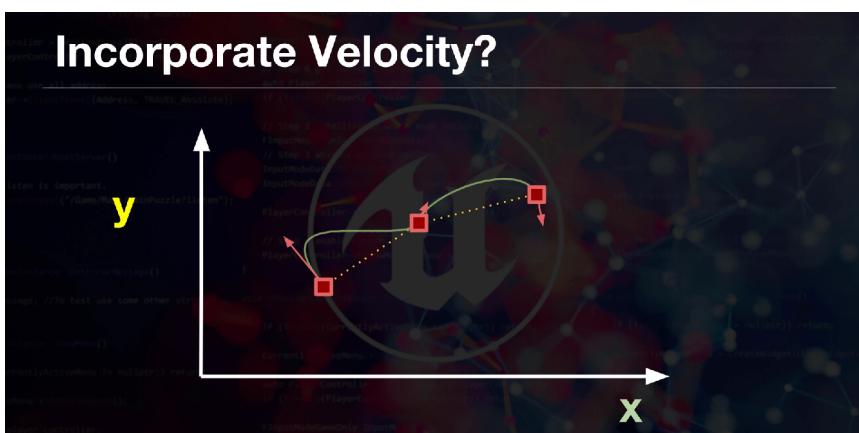
Late Updates



Why Can't We Lerp?



Simulate and make smooth movement



Cubic Interpolation And Velocity



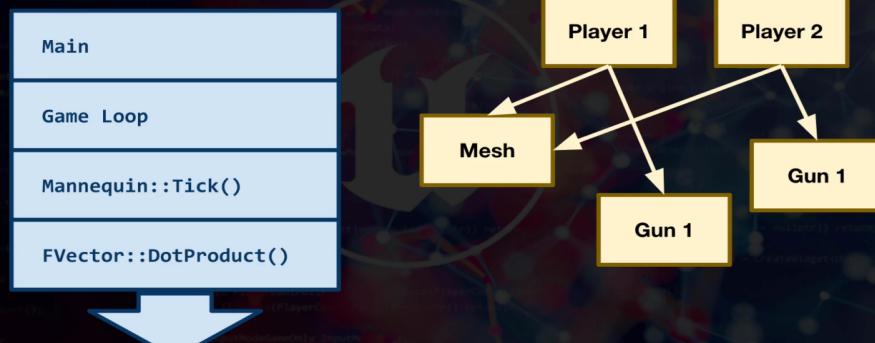
Slope, Derivative And Velocity

$\text{Slope} = \text{Derivative}$
 $= \Delta \text{Location} / \Delta \text{Alpha}$

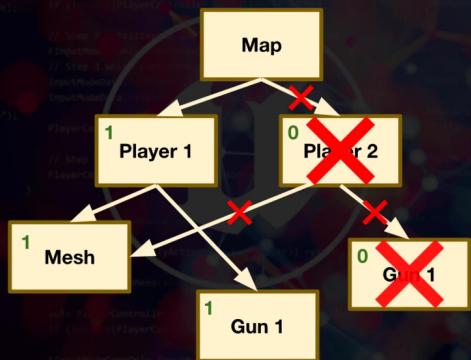
$\text{Velocity} = \Delta \text{Location} / \Delta \text{Time}$
 $\Delta \text{Alpha} = \Delta \text{Time} / \text{TimeBetweenLastUpdates}$
 $\text{Derivative} = \text{Velocity} * \text{TimeBetweenLastUpdates}$

Garbage Collection on Unreal Engine

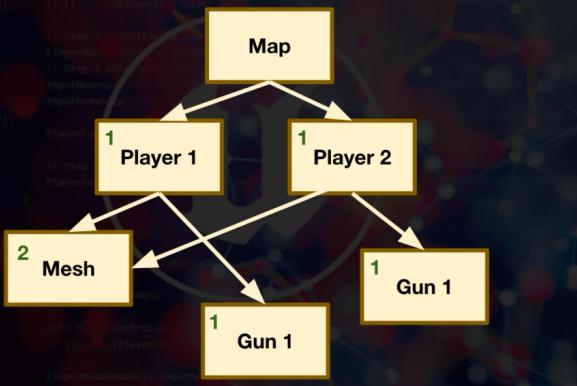
Stack vs Heap



Reference Counting



Reference Counting



You can check multiplayer puzzle mechanic video for this.

```
	TArray<UObject*> Referencers;

GetObjReferenceCount(this,&Referencers);

for(UObject* Each : Referencers)
{
    if(Each)
    {
        UE_LOG(YourLog,Warning,TEXT("%s"), *Each->GetName());
    }
}
```

[2016.03.26-19.39.53:730][2]LogRenderer: Relocating scene render targets to support 1024x768 (Frame:2).
[2016.03.26-19.39.54:728][38]Joy:Warning: AJoyPC(172): MaterialInstanceDynamic_2
[2016.03.26-19.39.54:729][38]Joy:Warning: AJoyPC(172): MaterialInstanceDynamic_1
[2016.03.26-19.39.54:743][38]Joy:Warning: AJoyPC(172): CheatManager_0
[2016.03.26-19.39.54:765][38]Joy:Warning: AJoyPC(172): PlayerInput_0
[2016.03.26-19.39.54:767][38]Joy:Warning: AJoyPC(172): TransformComponent0
[2016.03.26-19.39.54:769][38]Joy:Warning: AJoyPC(172): PC_InputComponent0
[2016.03.26-19.39.56:628][112]JoyNet:Verbose: AJoyWorld(172): Updating All Joy SMAs in single loop!
[2016.03.26-19.40.01:635][303]JoyNet:Verbose: AJoyWorld(172): Updating All Joy SMAs in single loop!

Output of ref counting function! ❤ Rama

Garbage Collection



Reference Counting In Unreal

- Use a `TSharedPtr<AAActor>`
- Constructing increments the count
- Destructing decrements the count.

Garbage Collection in Unreal

- All `UObjects` are automatically in the “set”
- Unreal starts from the “root set”
- Unreal walks all the `UProperty` pointer
- Any `UObject` not found can be deleted.