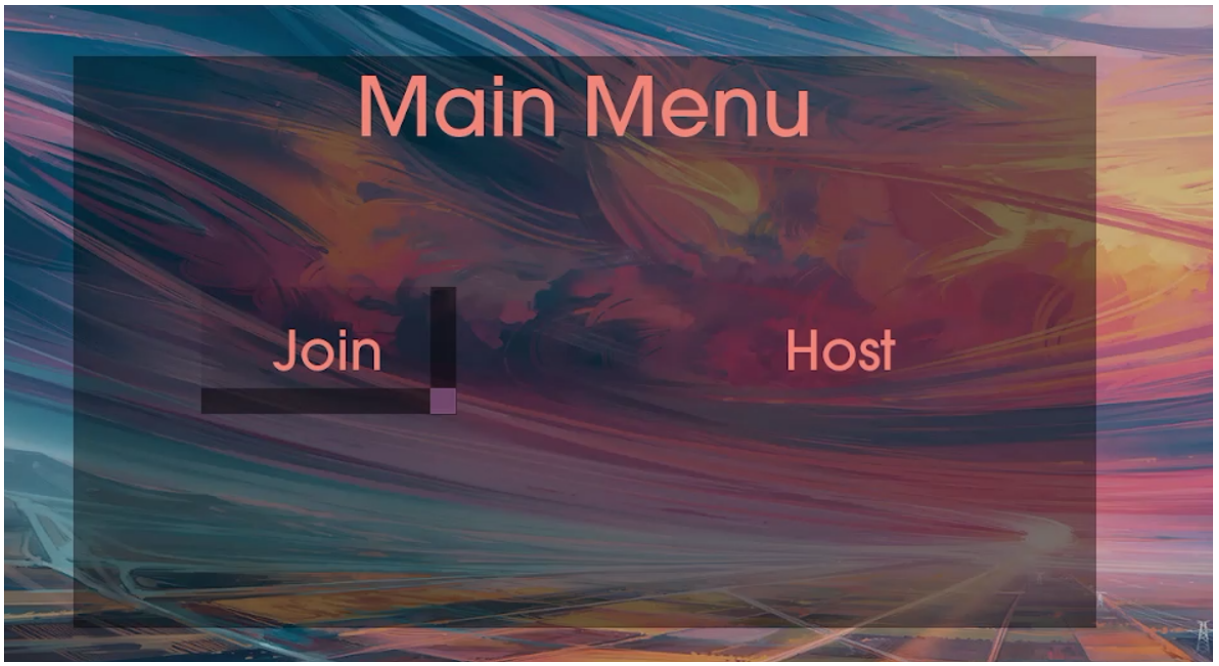
Please contact me to make changes or to share elsewhere. If you have questions or improvements, I'd be happy to hear. (Lütfen değişiklik yapmak için veya başka yerlerde paylaşmak için iletişime geçiniz. Sorularınız veya geliştirmeleriniz varsa duymaktan memnun olurum.)
Github: https://github.com/creosB
Youtube: https://www.youtube.com/channel/UCqWVCirXu-1fTdGRI73lCzA
The channel where we do this and more: https://www.twitch.tv/creosb

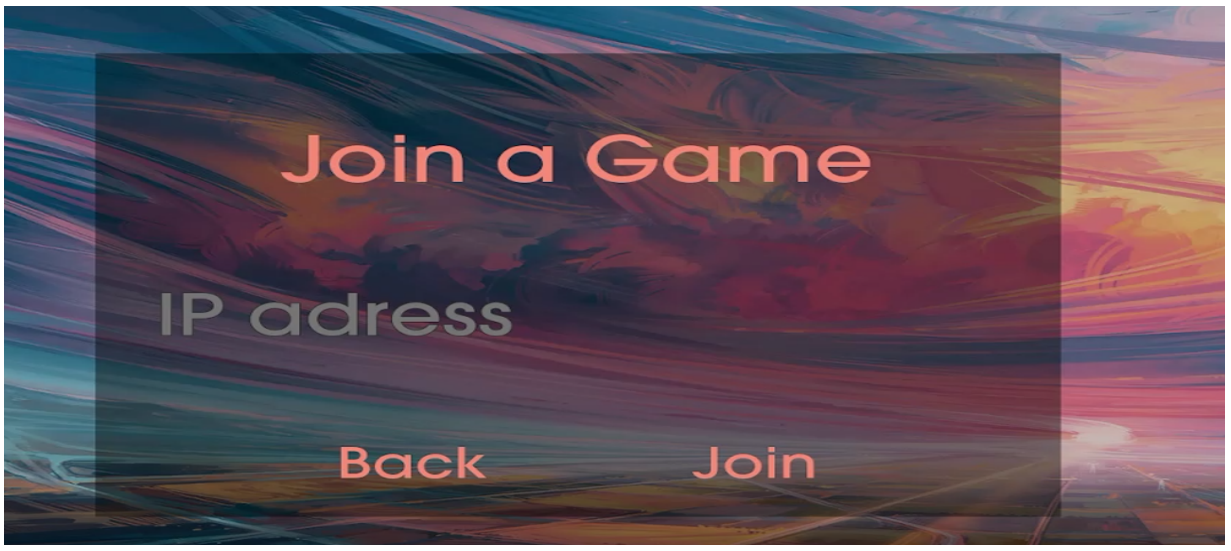
1) **Multiplayer lobby (Multiplayer with hamachi or any IP address)**
2) **Puzzle Platform (As in puzzle games, while someone is standing in a place, the mechanism starts to move, and the other player can cross)**
3) **BP Create Collectable Object**
4) **Multiplayer Server Type Selection**
5) **Steam 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**
14) **Unreal Engine Environment (light, effects, materials, etc.)**
15) **Unreal Engine Optimizing (GPU, LOD, DataSmith,UV, etc.)**

Multiplayer Lobby

Aşağıdaki gibi bir widget oluşturuyoruz.



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



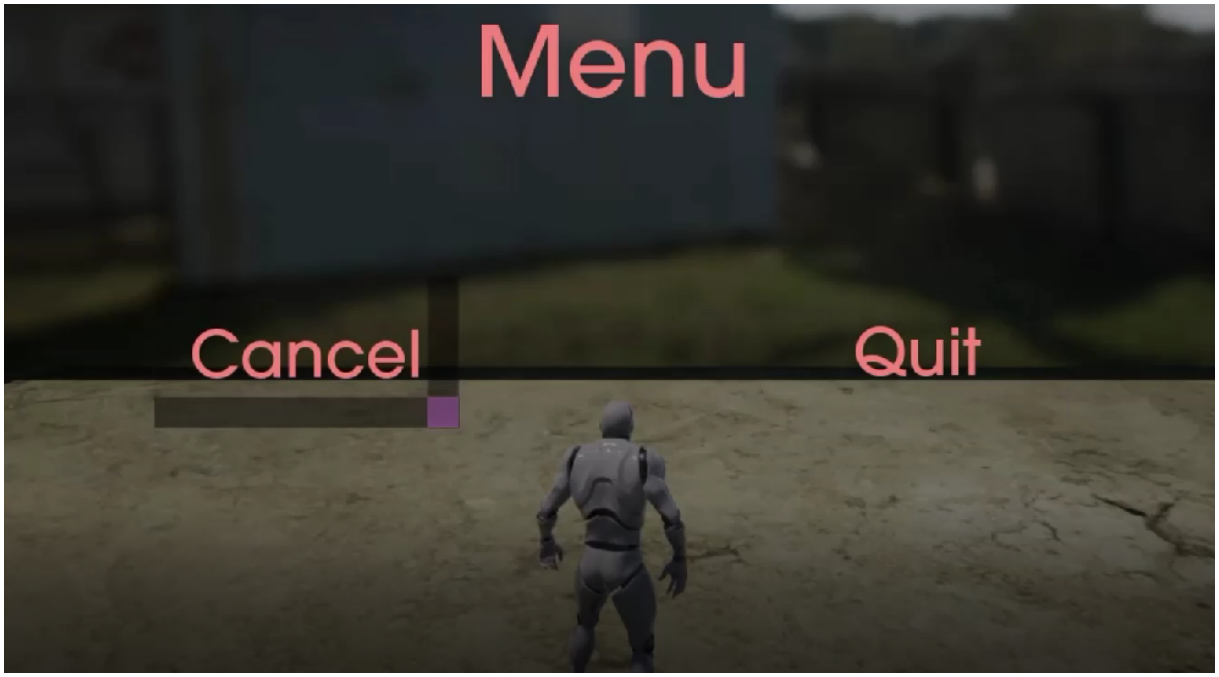
Sayfalar arasında veriyi iletebilmek için Inferface yaratıyoruz.

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;
```

MenuWidget.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 (MainMenuInterface != nullptr)
    {
        MainMenuInterface->Host();
    }
}
// connect - join the server
void UMainMenu::JoinServer()
{
    if(MainMenuInterface != nullptr && IPWriteBox != nullptr)
    {
        MainMenuInterface->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 içerisine 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 verdiğimiz 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 IMenuInterface
{
    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;
```

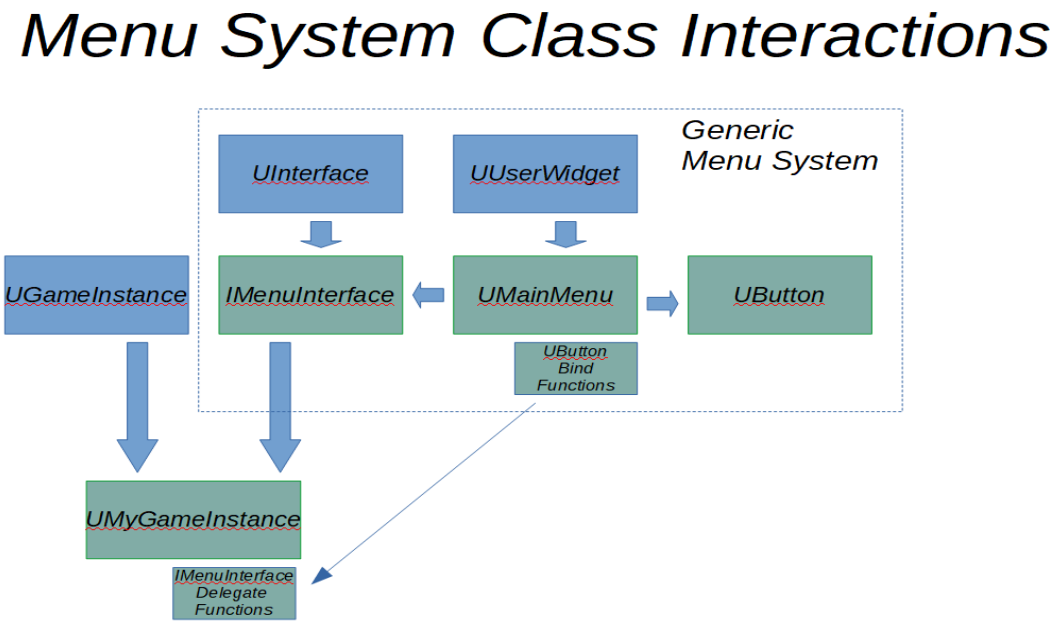
```
    UFUNCTION(Exec)
    virtual void Join(const FString& Adress) override;

    virtual void LoadMainMenu() override;

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

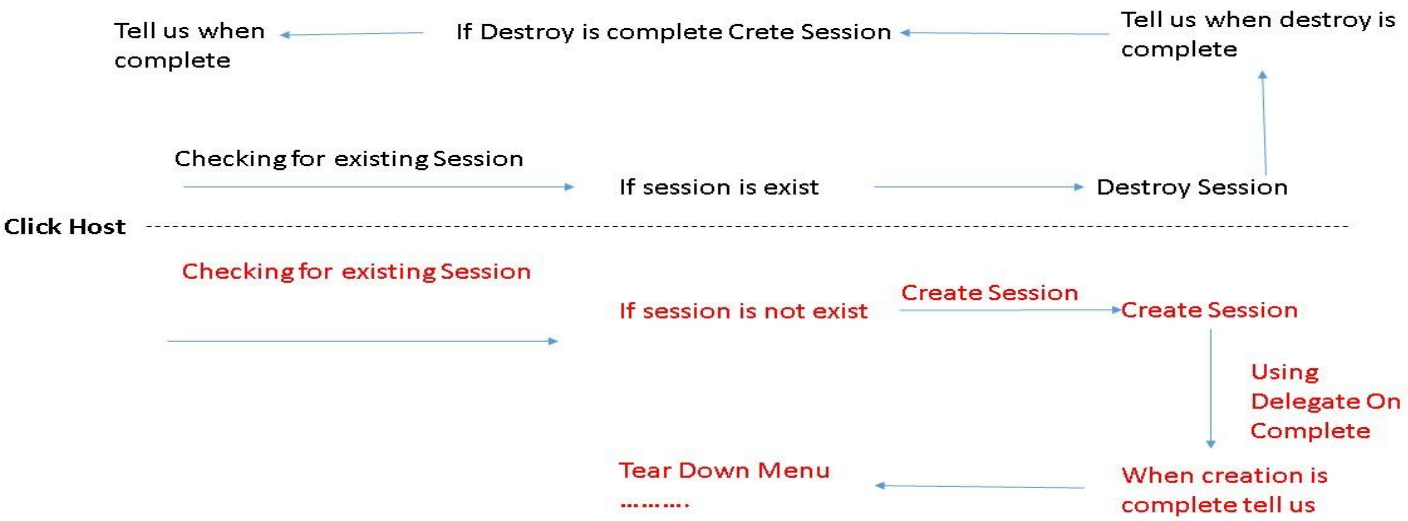
GameInstance.cpp

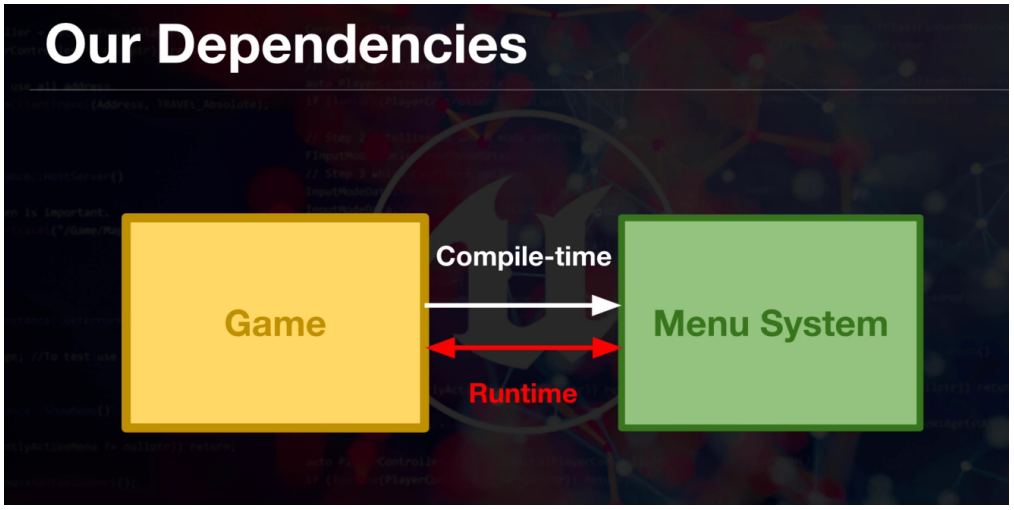
```
UYourGameInstance::UYourGameInstance(const FObjectInitializer& ObjectInitializer)
{
    // find menu widget class and define to MenuClass.
    ConstructorHelpers::FClassFinder<UUserWidget>
MenuBPClass(TEXT("/Game/Blueprints/Widgets/WBP_MainMenu"));
    if (!ensure(MenuBPClass.Class != nullptr)) return;
    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
```

```cpp
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->SetMenuInterface(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->SetMenuInterface(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
}
```

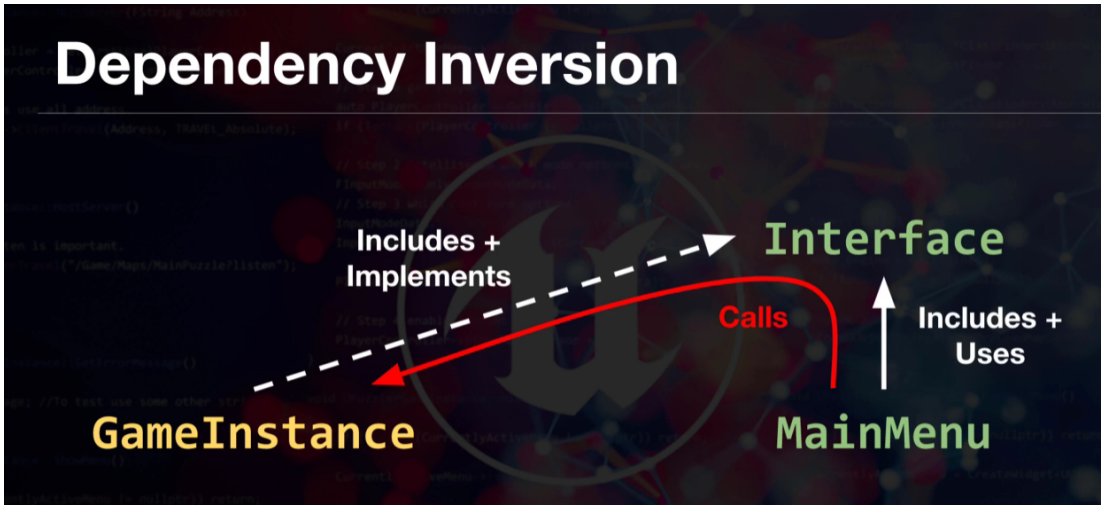
Let's visualize the menu system




Tell us when complete
If Destroy is complete Crete Session
Tell us when destroy is complete
Checking for existing Session
If session is exist
Destroy Session
Click Host
Checking for existing Session
If session is not exist
Create Session
Create Session
Using Delegate On Complete
Tear Down Menu
……..
When creation is complete tell us


Our Dependencies
Game
Compile-time
Runtime
Menu System


Dependency Inversion
Includes + Calls
GameInstance
MainMenu



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

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

Puzzle Platform

Öncelikle platformu çalıştırdığımızda 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();
}
```

```cpp
// 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

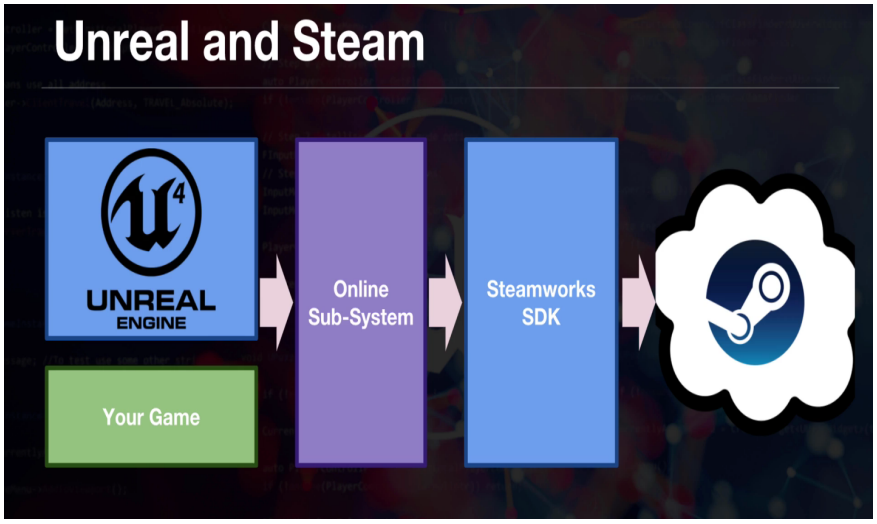
```
IOnlineSubsystem* SubSystem = IOnlineSubsystem::Get();
if (SubSystem != nullptr)
{
	UE_LOG(LogTemp, Warning, TEXT("Found subsystem %s"), *SubSystem->GetSubsystemName().ToString());
}else
{
	UE_LOG(LogTemp, Warning, TEXT("Found no subsystem %s"));
}
```

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 yürümesi için delegate kullanıyoruz.

void UPuzzlePlatformsGameInstance::OnCreateSessionComplete(FName SessionName, bool Succes)

void UPuzzlePlatformsGameInstance::OnDestroySessionComplete(FName SessionName, bool Succes)

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 çağırdığımız host fonksiyonu bunlardan hangisinin çağrılacağını belirliyor.

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

Steam bağlantısı için steam online subsystem'i aktif ediyoruz.

Build.cs içerisine 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 gerekli.

SessionSettings.bUsesPresence = true;

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

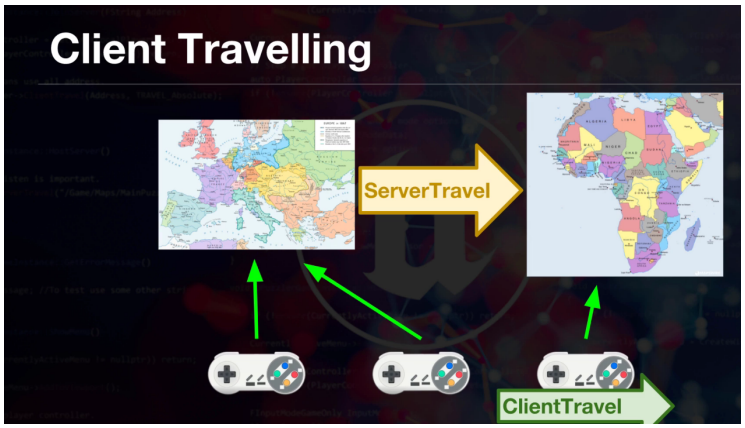
-nonsteam yazarak çalıştırırsak steam ihtiyaç duymadan test edebiliriz.

```
// 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 akıcılaşır.

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 gönderidiği 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 çağrılacak 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 iletişimi sağlamış oluyoruz.

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

Server içerisinde ki 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 contructor 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/OyunİsmiHaritaDosyası/Mapİsmiolarakgameöncesineyazıpbaşlatıyoruz.

Multiplayer server'e bağlanmak için -game yerine -server yazıp başlatıyoruz.

Multiplayer oyunu başlatmak için local ip adresini yazip 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

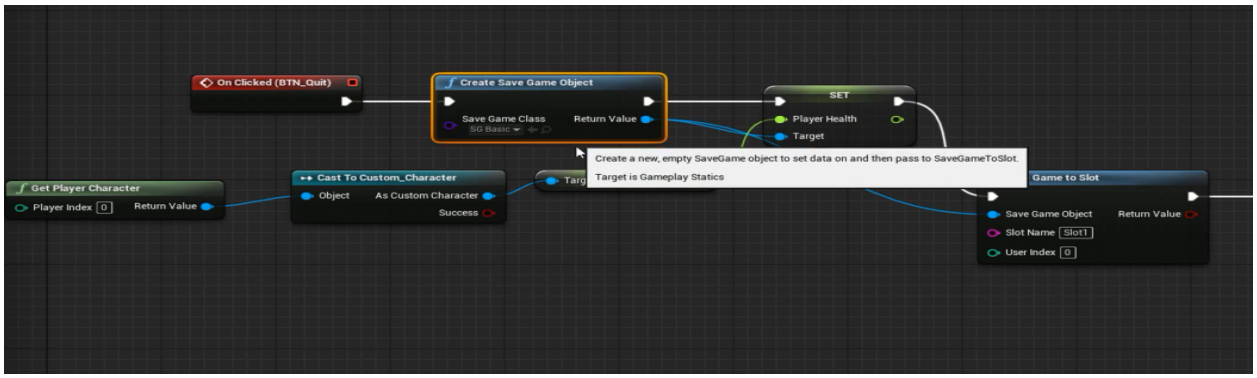
```
// check if is true, it's server. Else, it's client
if (HasAuthority())
{
FVector Location = GetActorLocation();
Location += FVector(5 * DeltaTime, 0, 0);
SetActorLocation(Location);
}
```

Gameplay Ability Plugin:

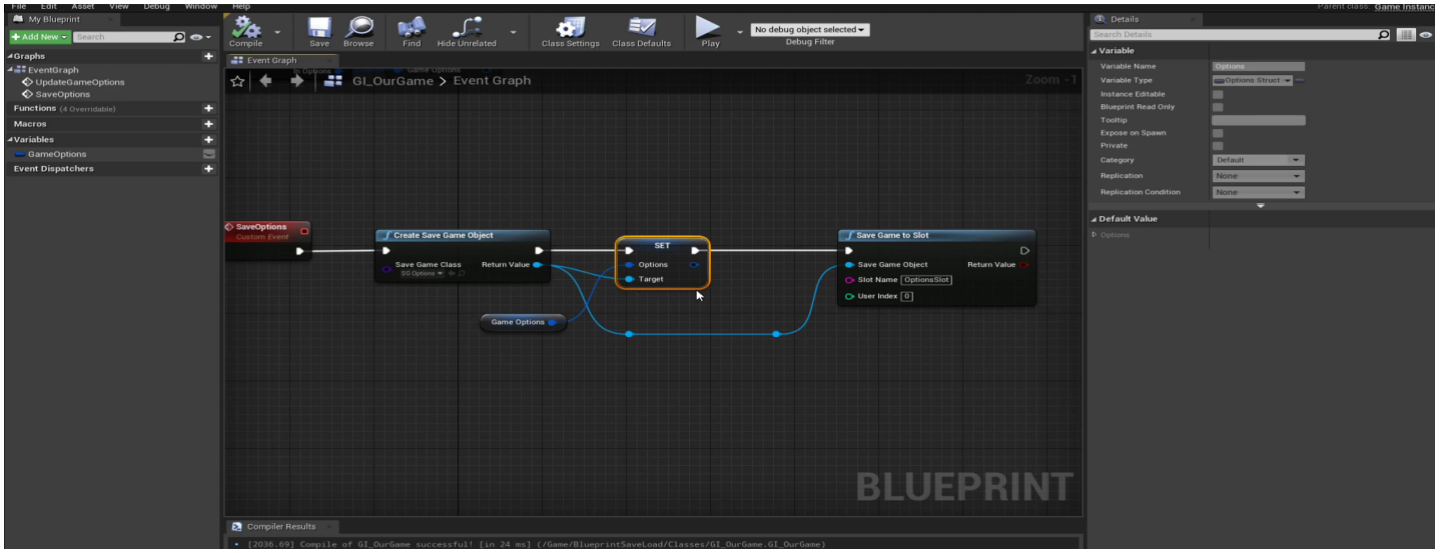
Oyun içi mana, can veya güç veren iksir, yetenekleri yönetmek için gameplay ability plugin kullanıyoruz. Buna herhangi bir iksir vs. eklemek için sadece data asset oluşturmak yeterli.

Asset Manager: Oyun içinde itemleri, iksirleri vb. itemleri tutmak için c++ FPrimaryAssetType ile asset manager oluşturup defaultEngine.ini dosyasına AssetManagerClassName = /Script/GameName.DosyaIsmiAssetManager şeklinde ekleme yapıyoruz. Artık Project Settings -> Asset Manager kısmında eklediğimizi görebiliriz. Oyun içinde bunları kullanmak için Bluepint 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.

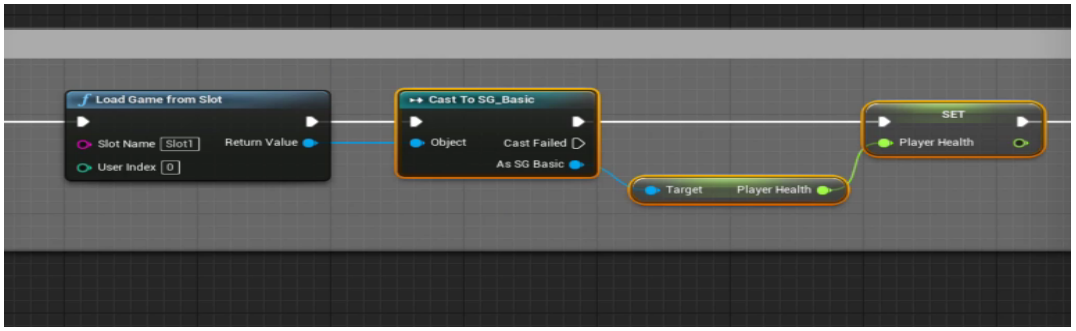
BP Create Save



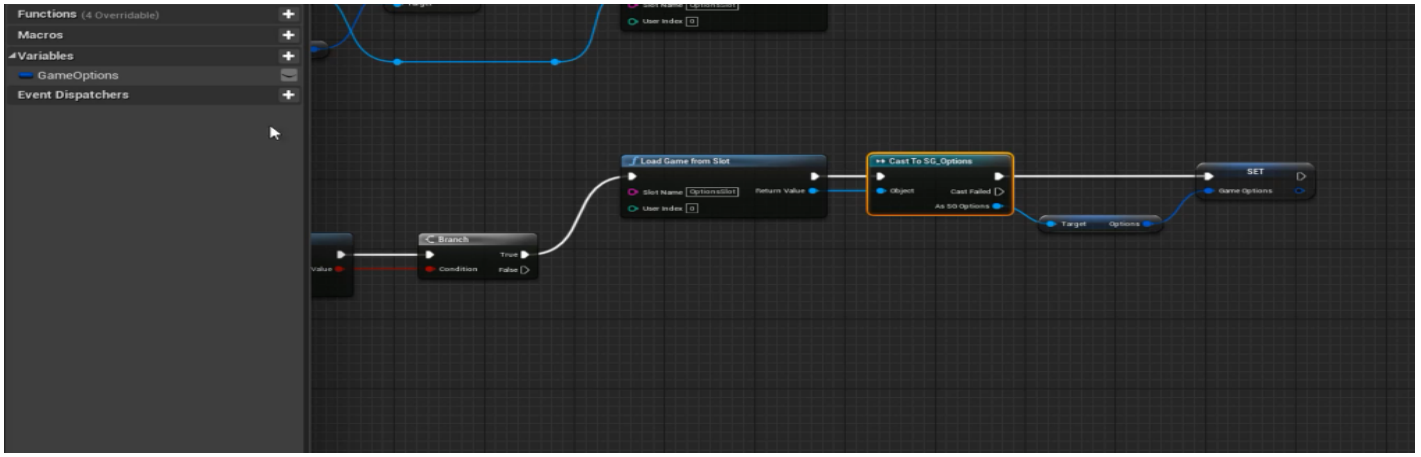
BP Create Game Options



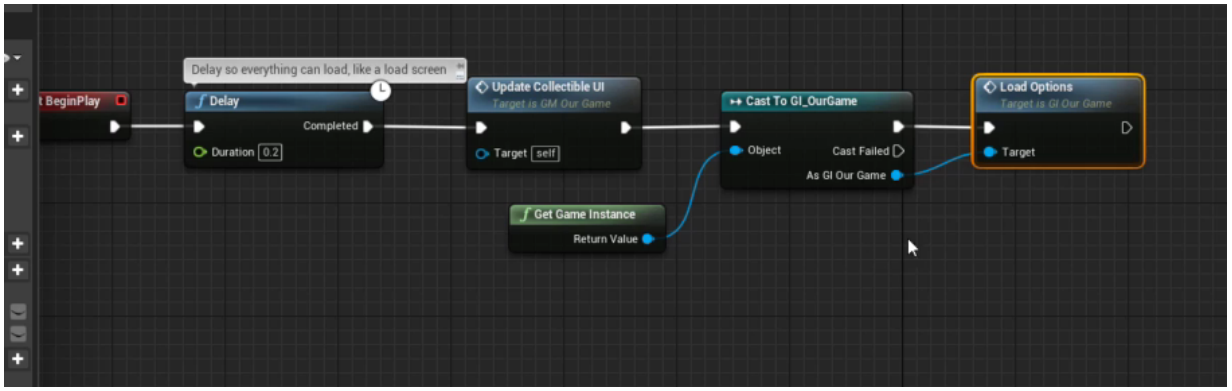
BP Load Save



BP Load Options

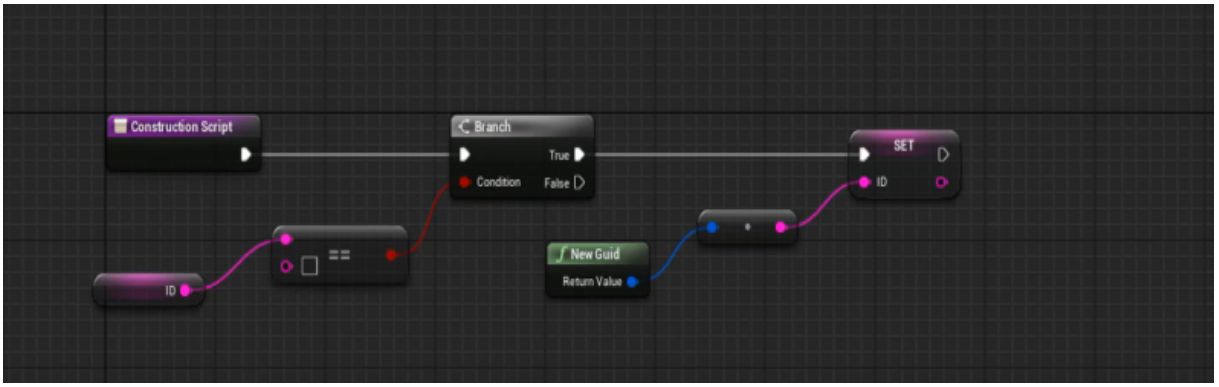


BP Set Game Options

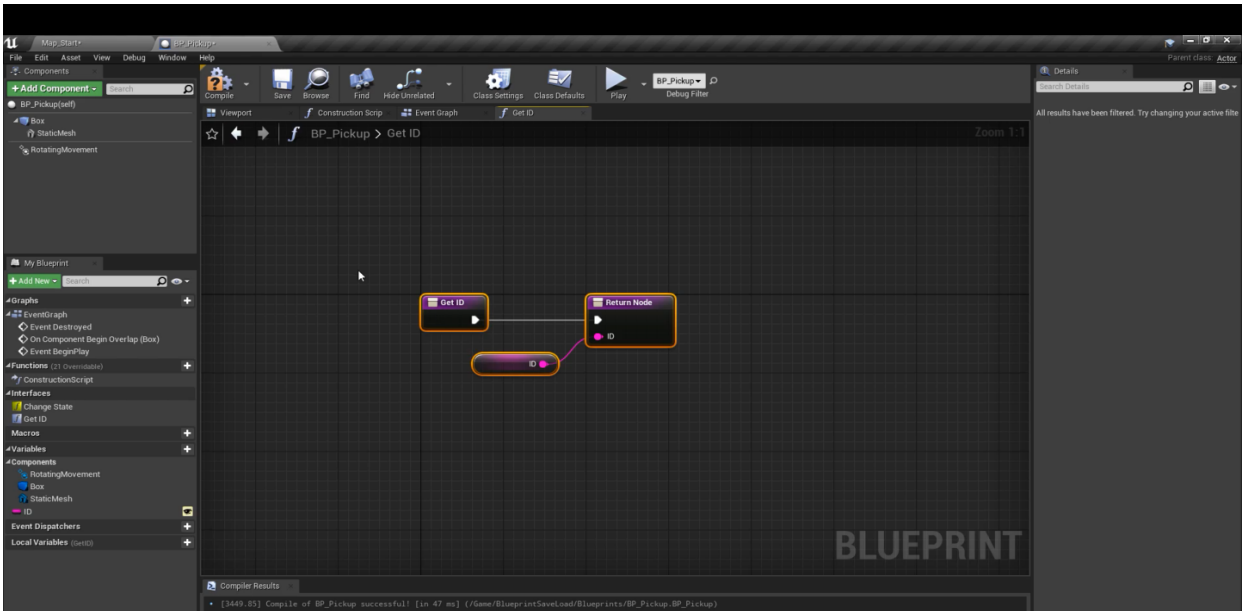


BP Create Collectable Object

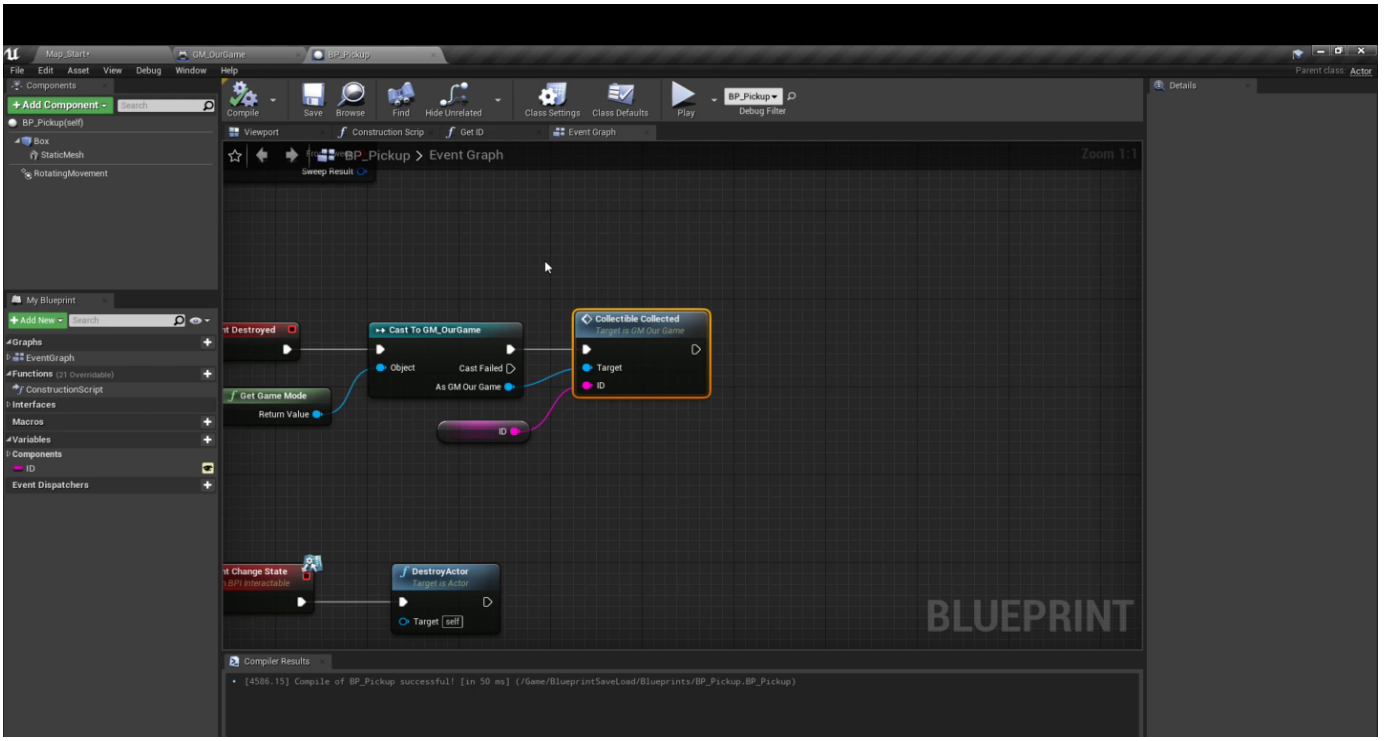
1. **Create UniqueID**



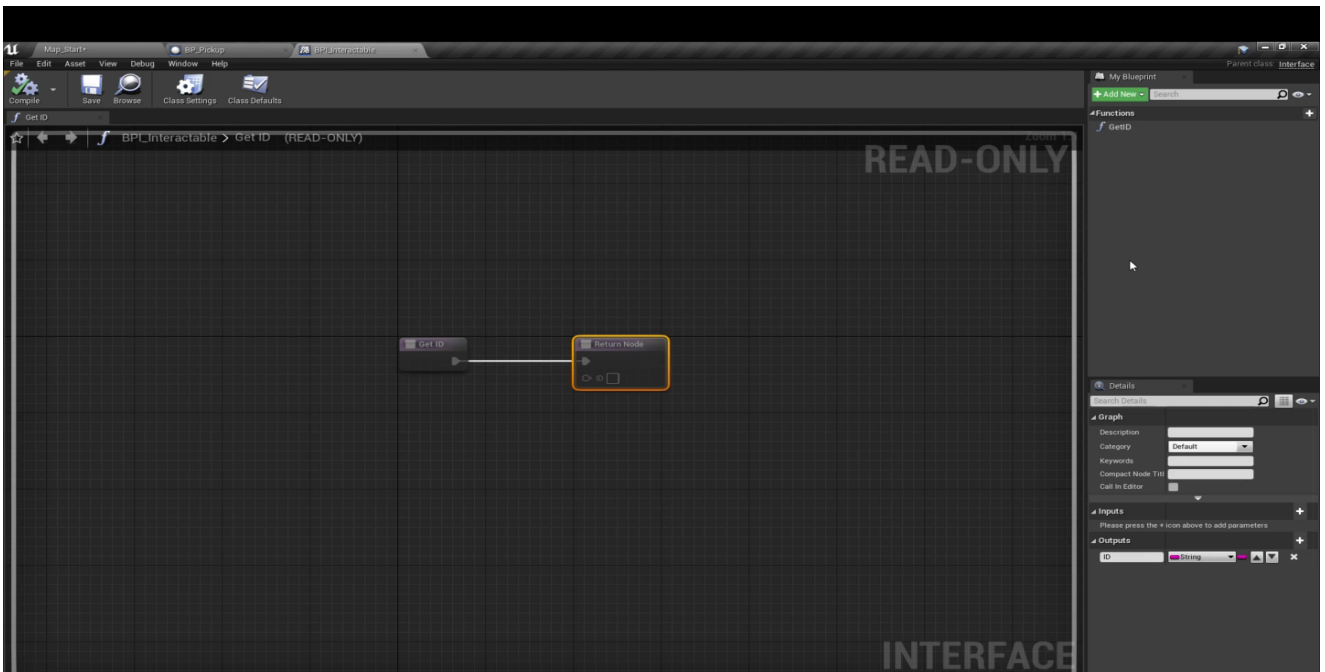
2. **Return UniqueID**



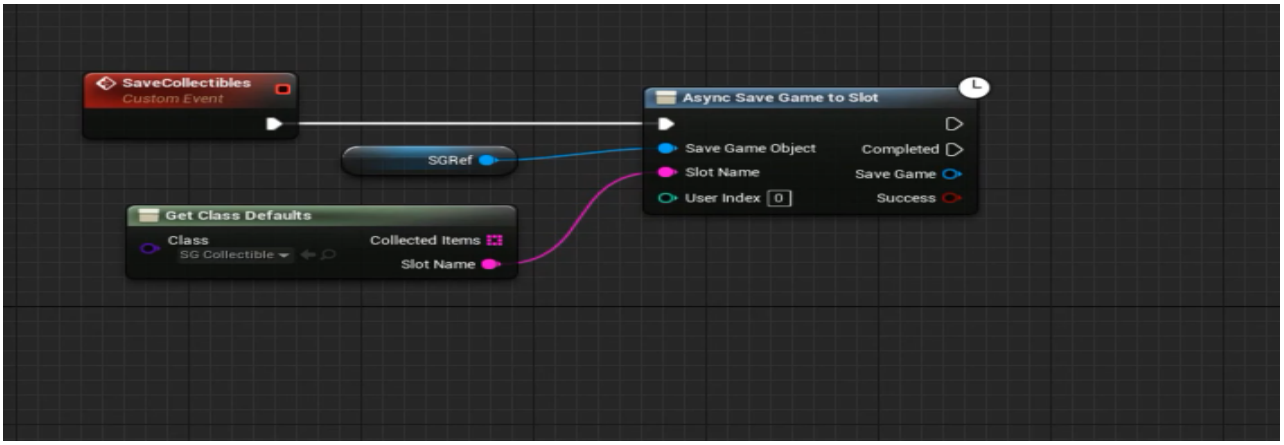
3. **Save Collectable UniqueID (when you hit the object)**



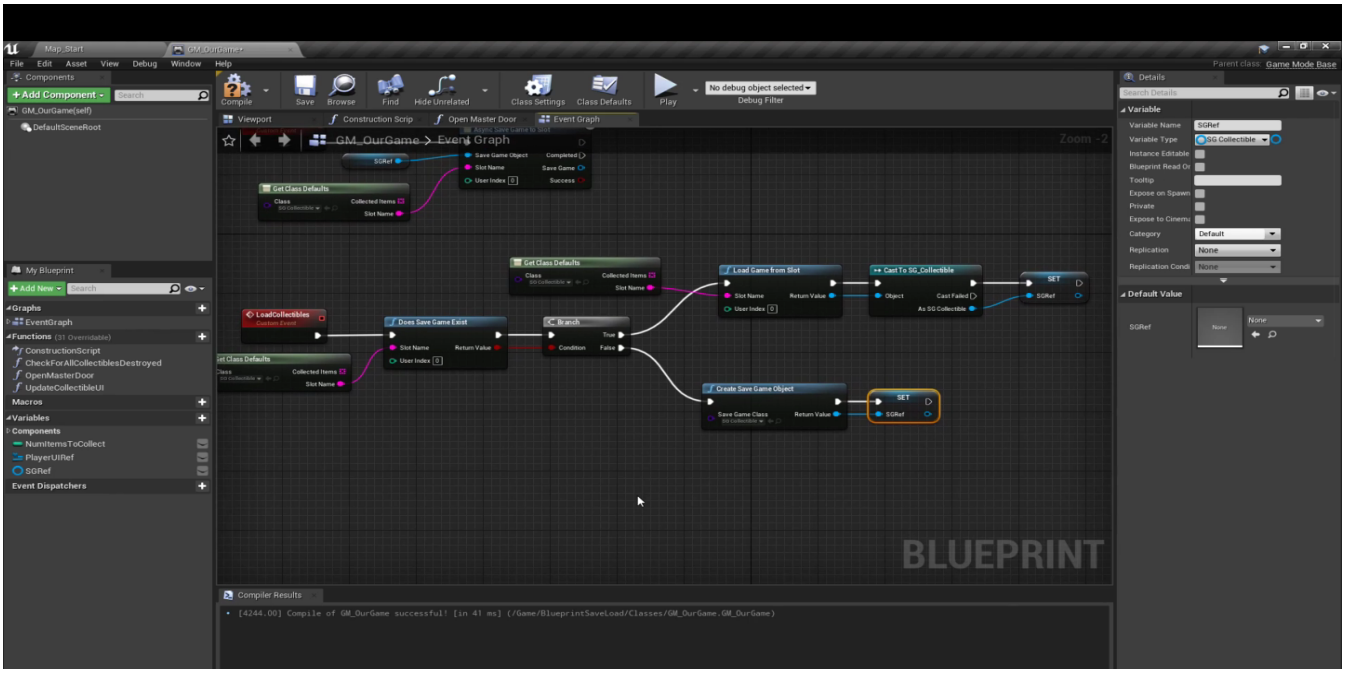
4. **Get UniqueID**



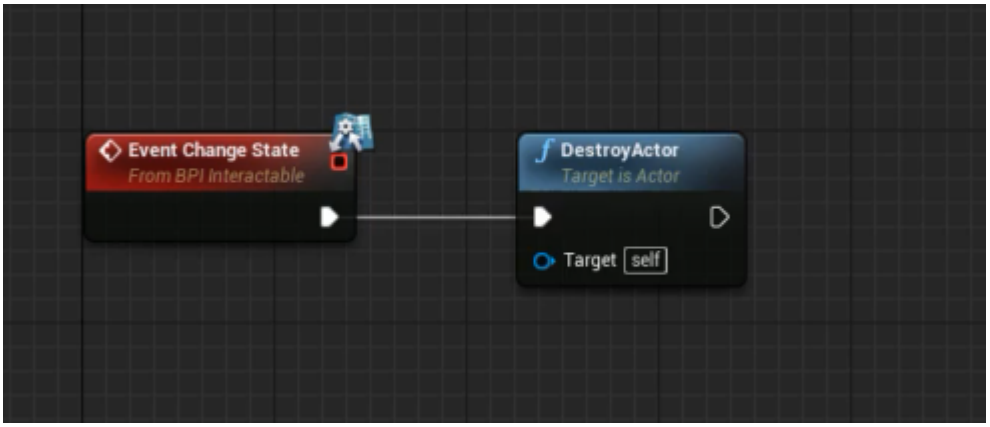
5. **Save Collected Object**



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/

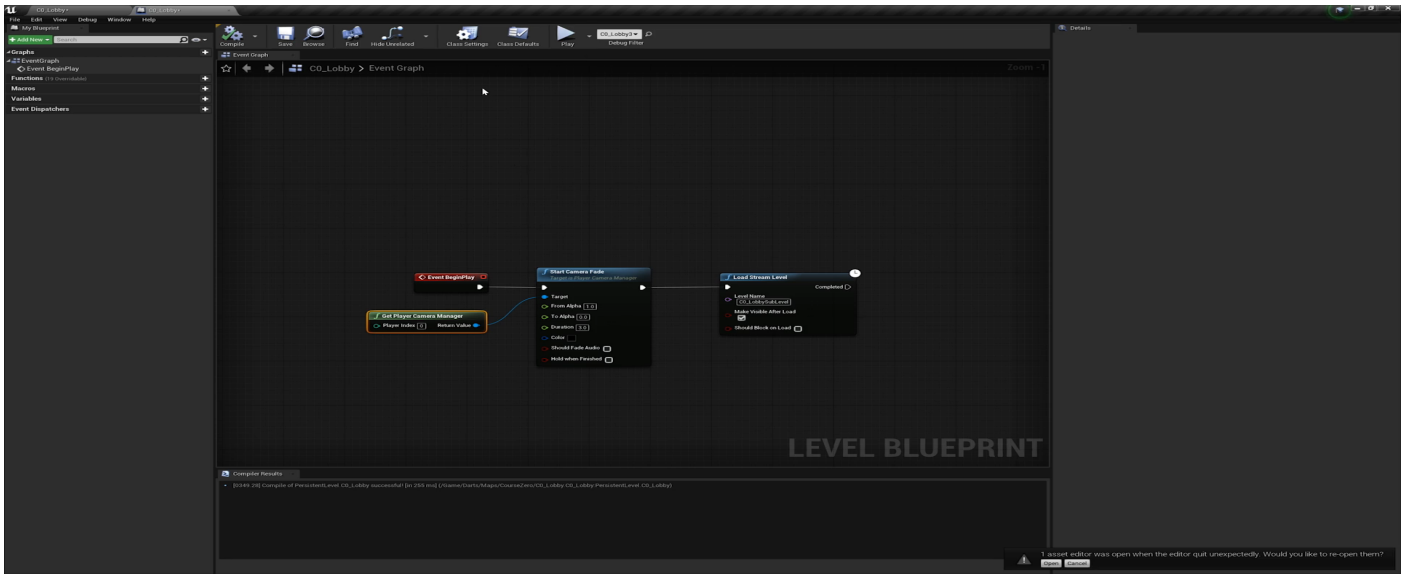
3) Show one or more stereo layers (most powerful option but might also take the most work)

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 wating 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 thhe 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.)

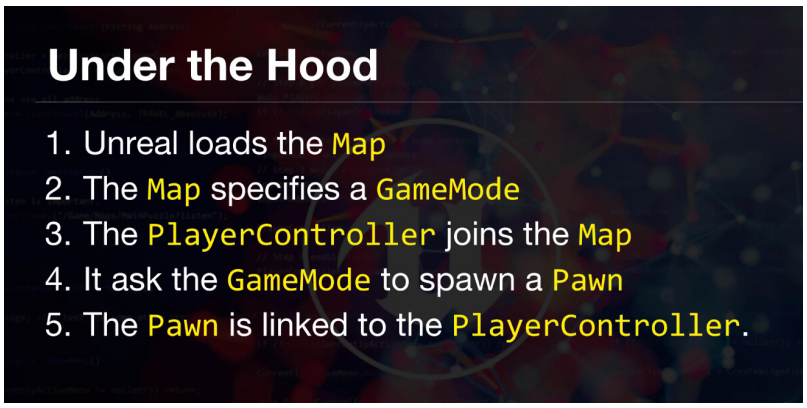
9) Open the PSOCaching folder and copy ProjectName_GLSL_ES3_1.stablepc.csv file -> open the ProjectName/build/Android folder and create PipelineCaches -> paste the file in here

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.

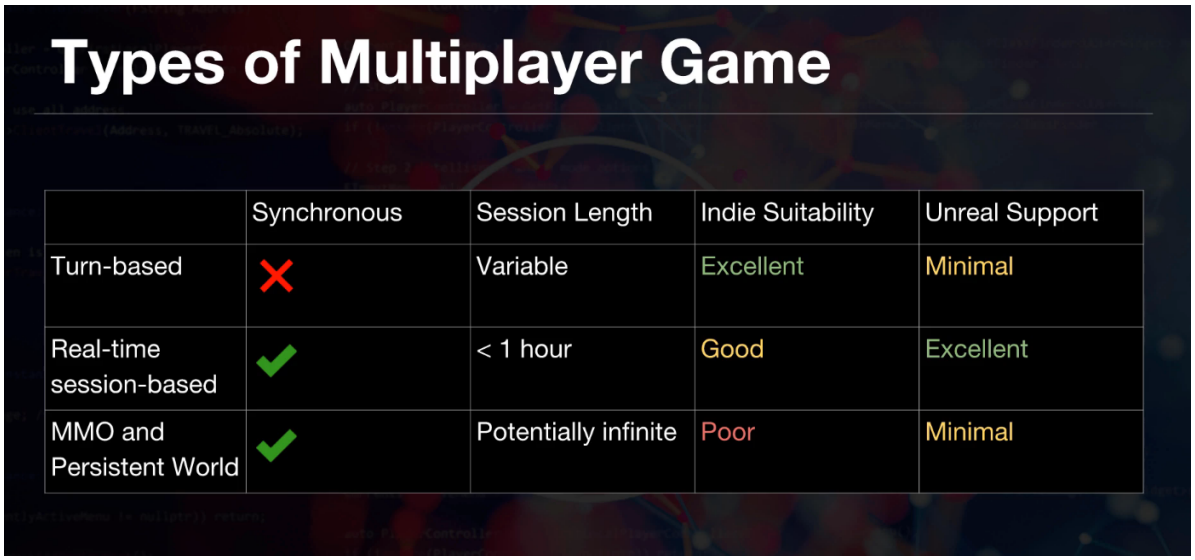
Note: Before product ships or major milestones, re-run steps 2 to 5 to update the stablepc.csv file to make sure all shaders used are included, as well as make sure there are no unused shaders in the PSO cache.

Unreal Load Game Hierarchy

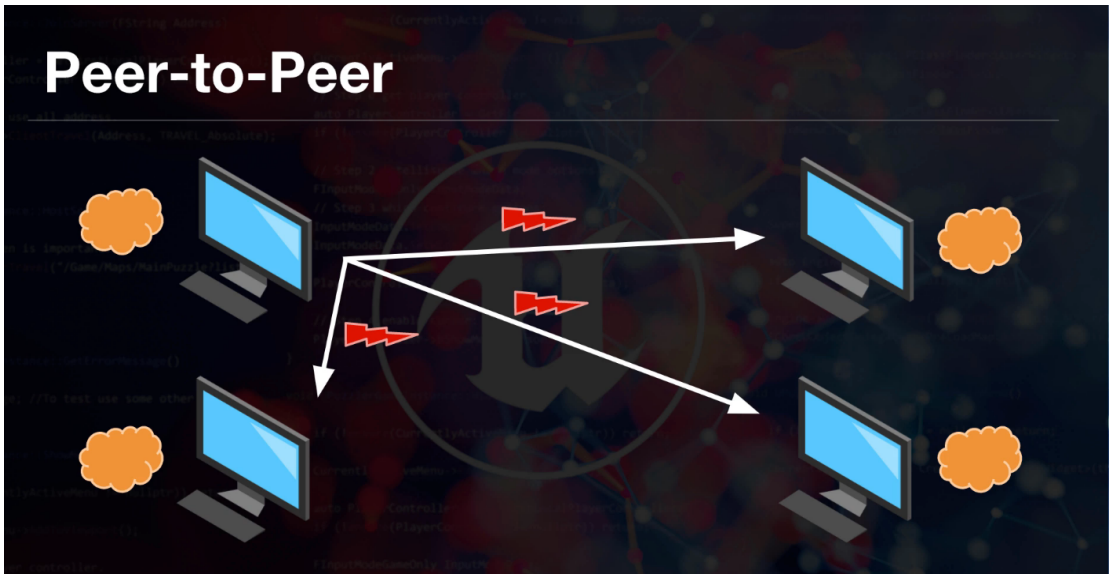


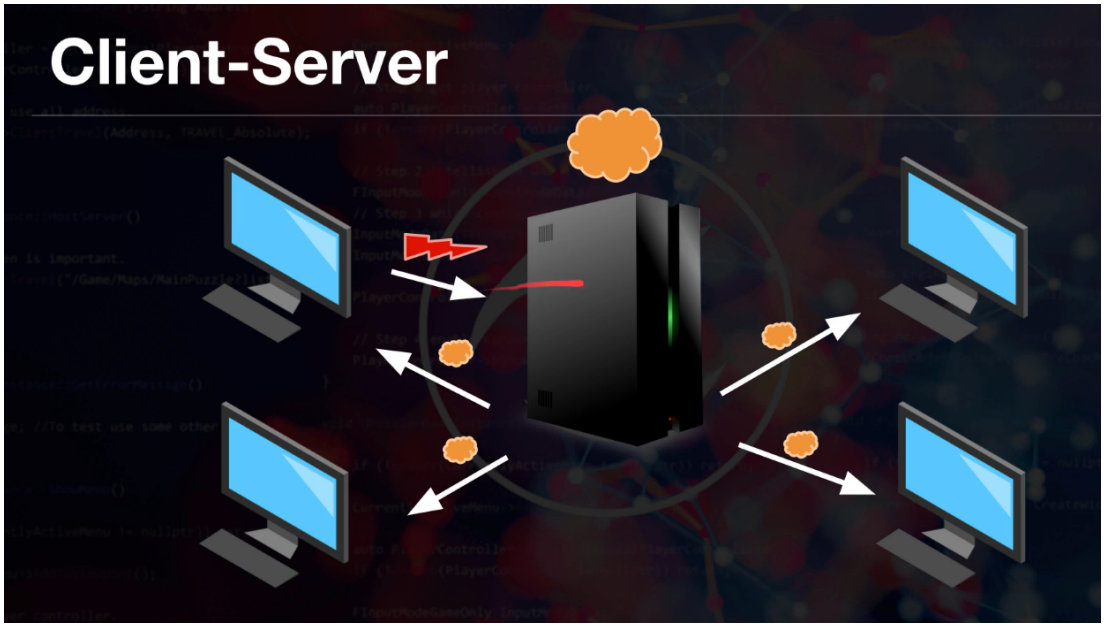
Multiplayer Server Type Selection

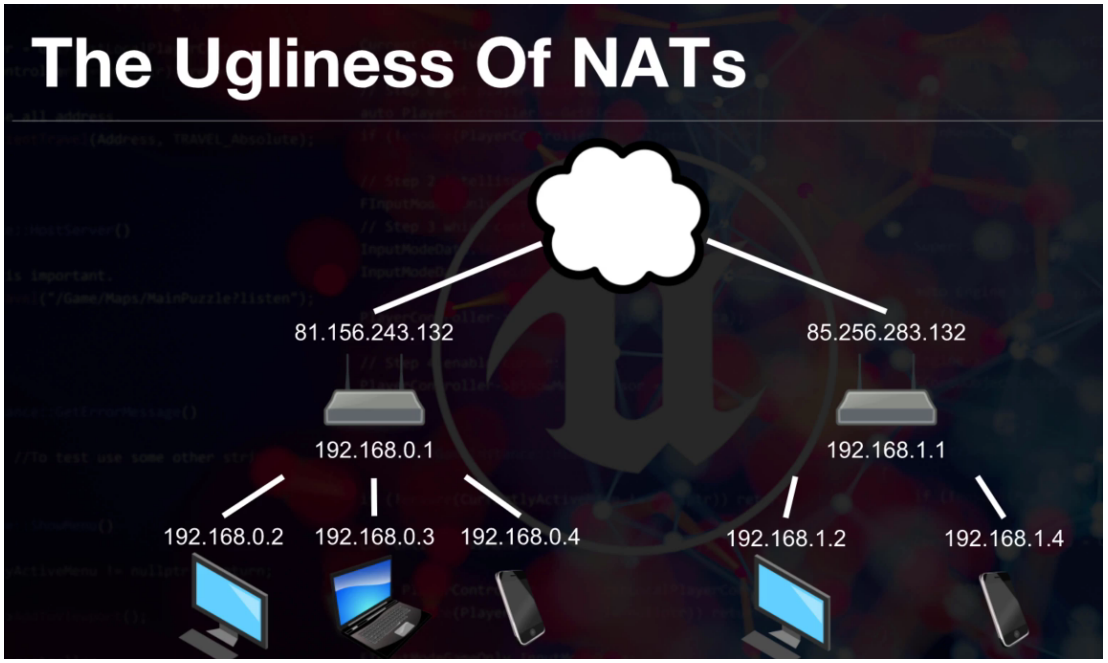
Comparison for multiplayer game Unreal Engine support.



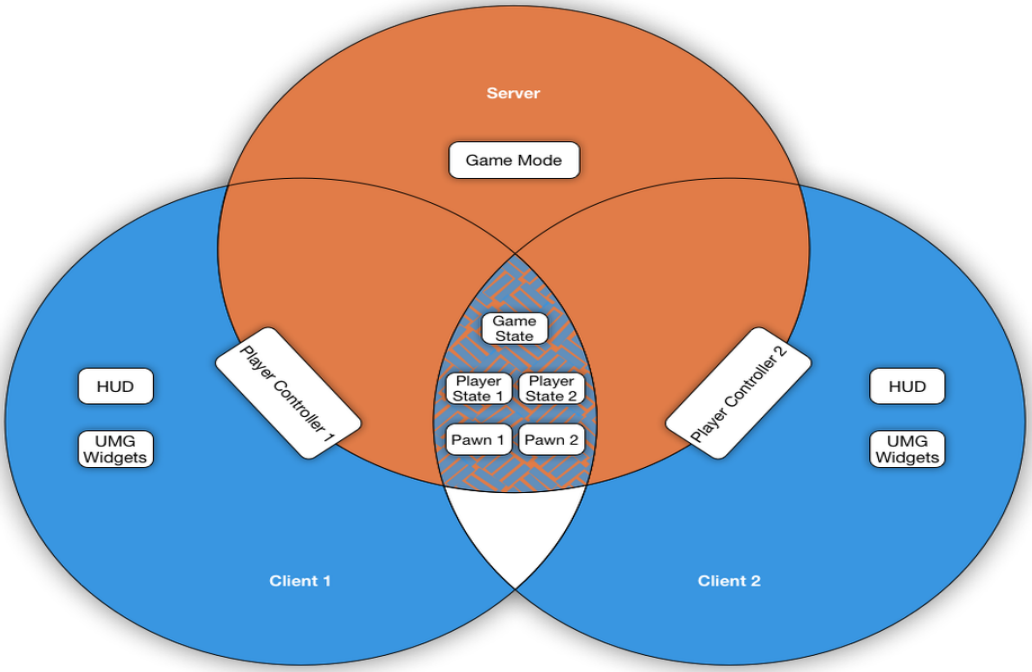
	Synchronous	Session Length	Indie Suitability	Unreal Support
Turn-based	✗	Variable	Excellent	Minimal
Real-time session-based	✓	< 1 hour	Good	Excellent
MMO and Persistent World	✓	Potentially infinite	Poor	Minimal

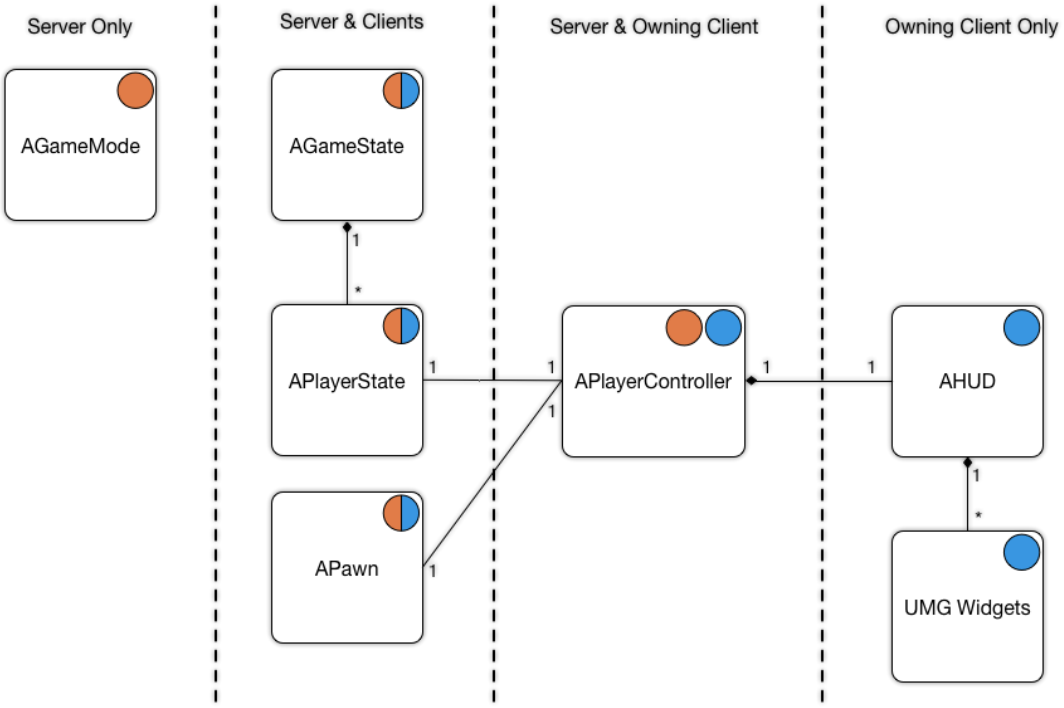



Client-Server


The Ugliness Of NATs
81.156.243.132
85.256.283.132
192.168.0.1
192.168.1.1
192.168.0.2
192.168.0.3
192.168.0.4
192.168.1.2
192.168.1.4

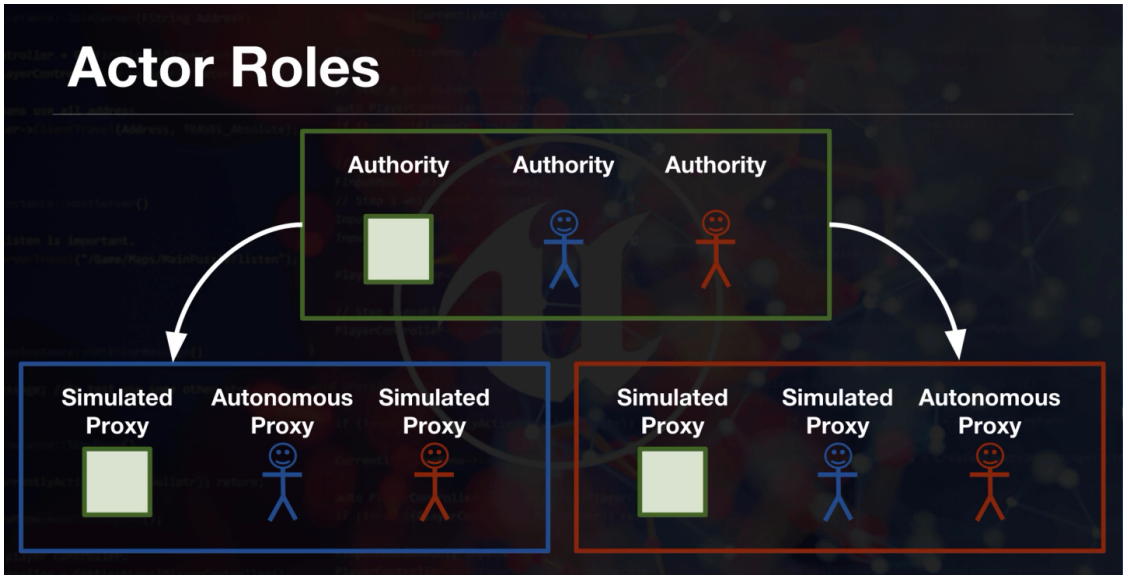
Let’s divide section by section

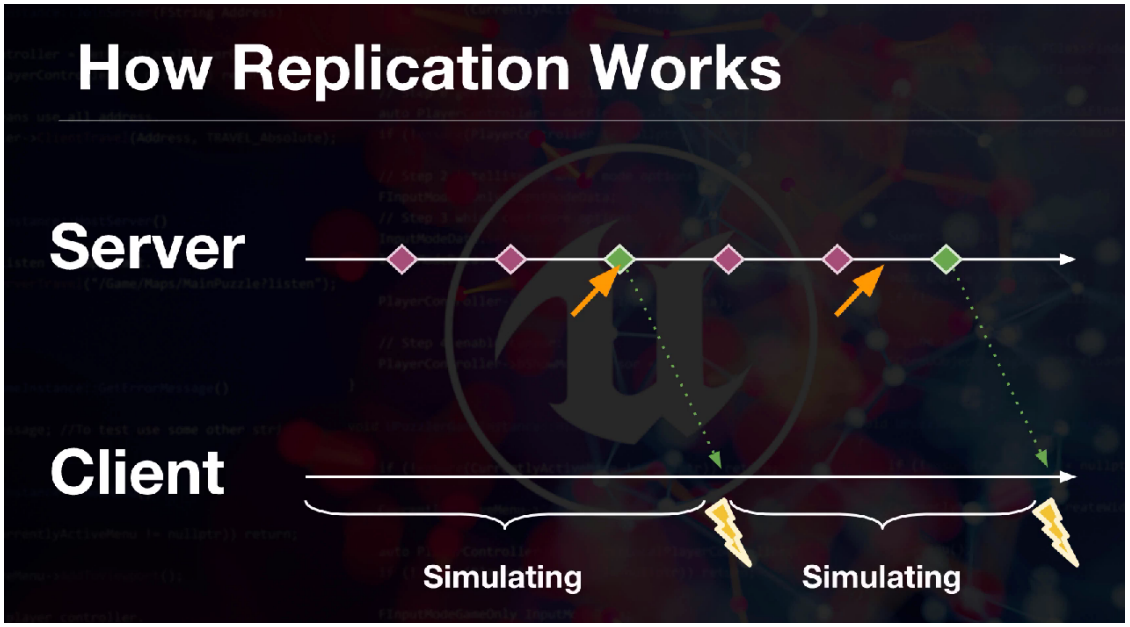




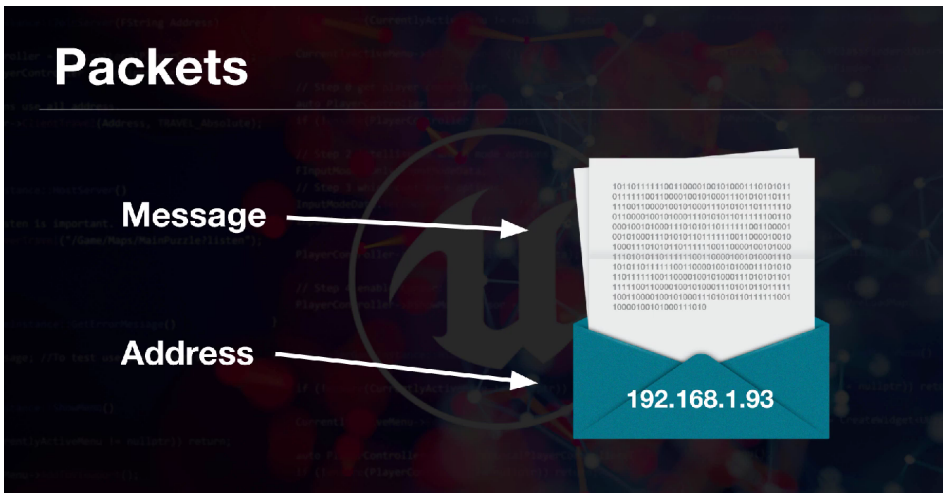
Multiplayer Replication - LAG - Package Loss

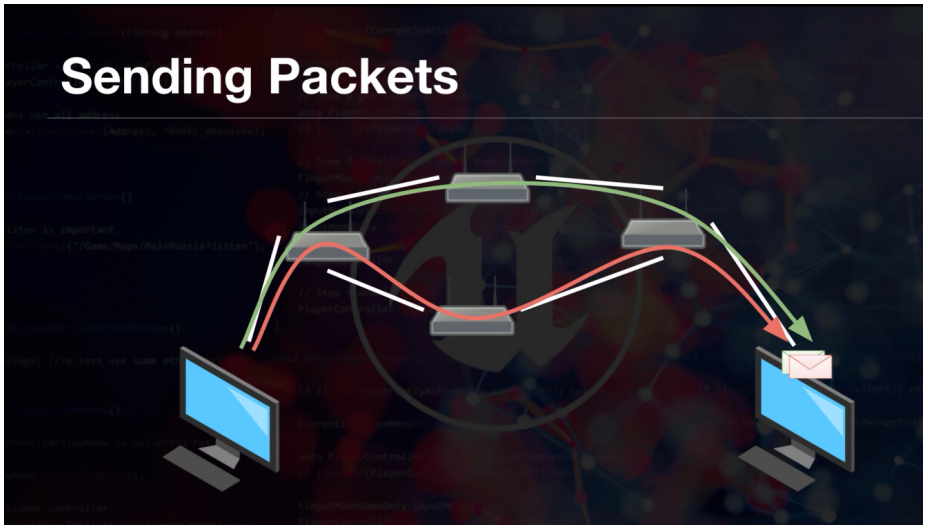
Replication Actor Roles

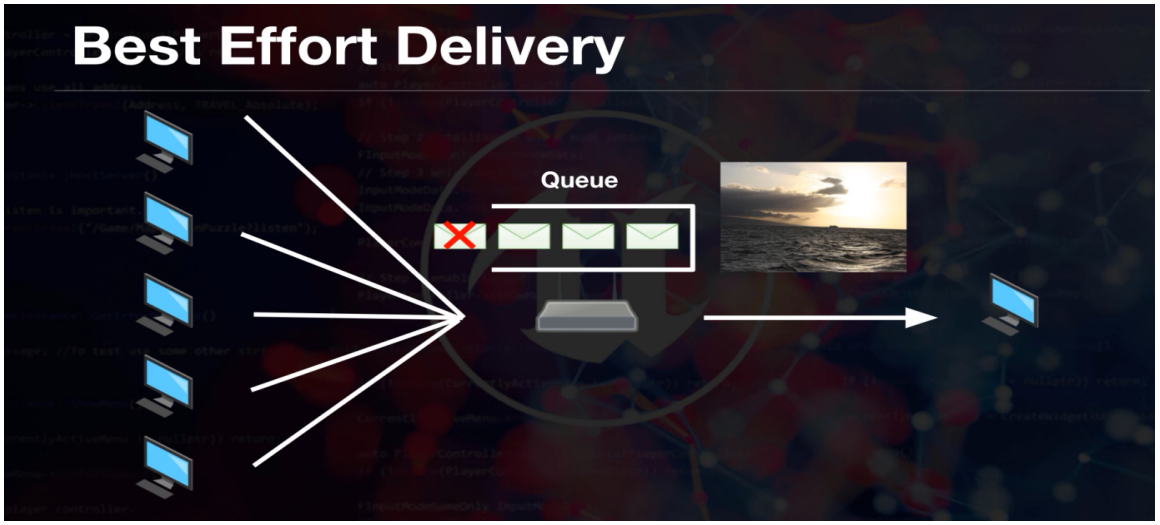




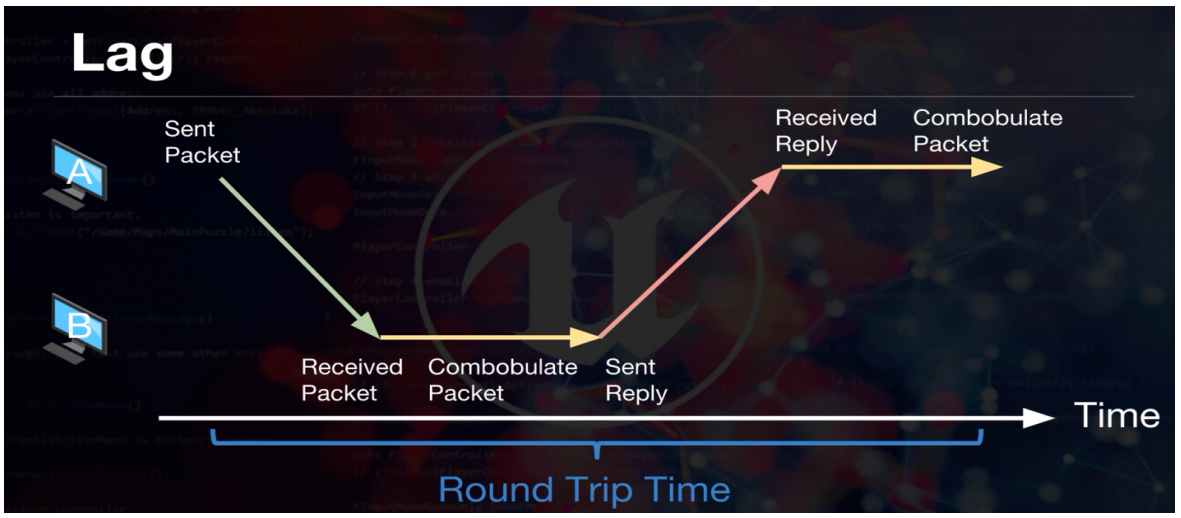
What is the packet loss ?



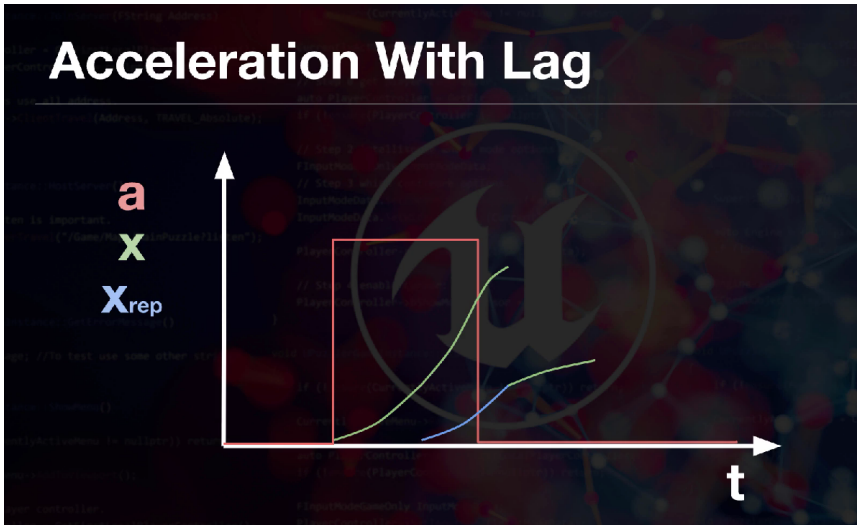


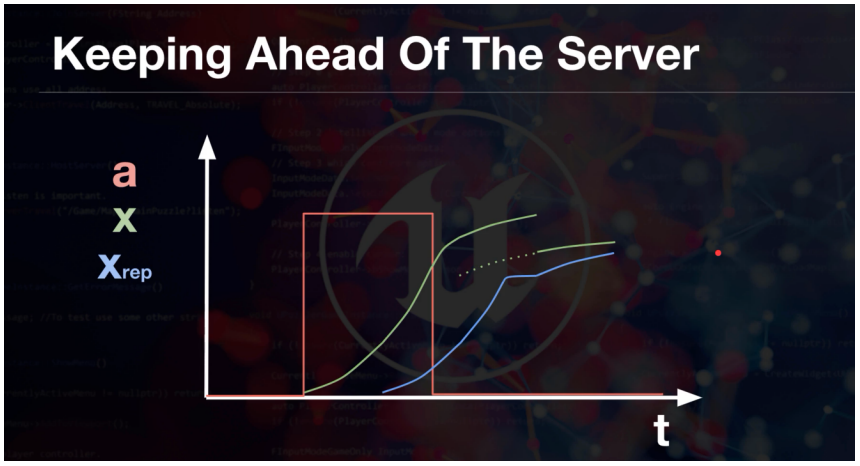


What is the LAG ?



Let’s visualize the lag



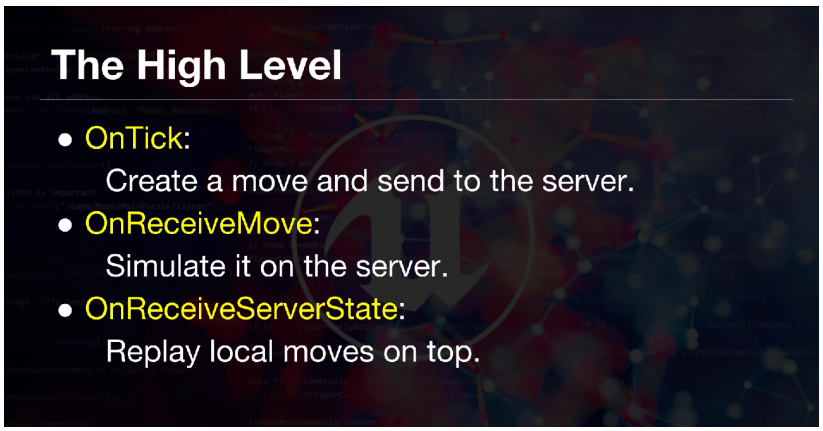


Fix the LAG Methods

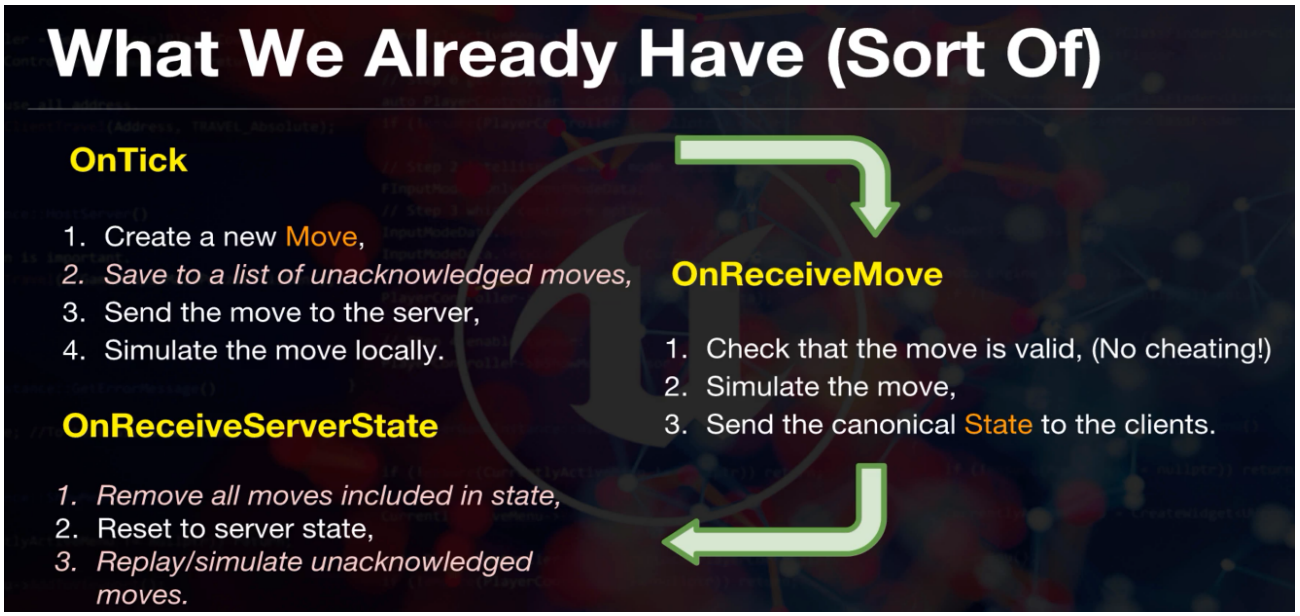
Comparison Of Methods

	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

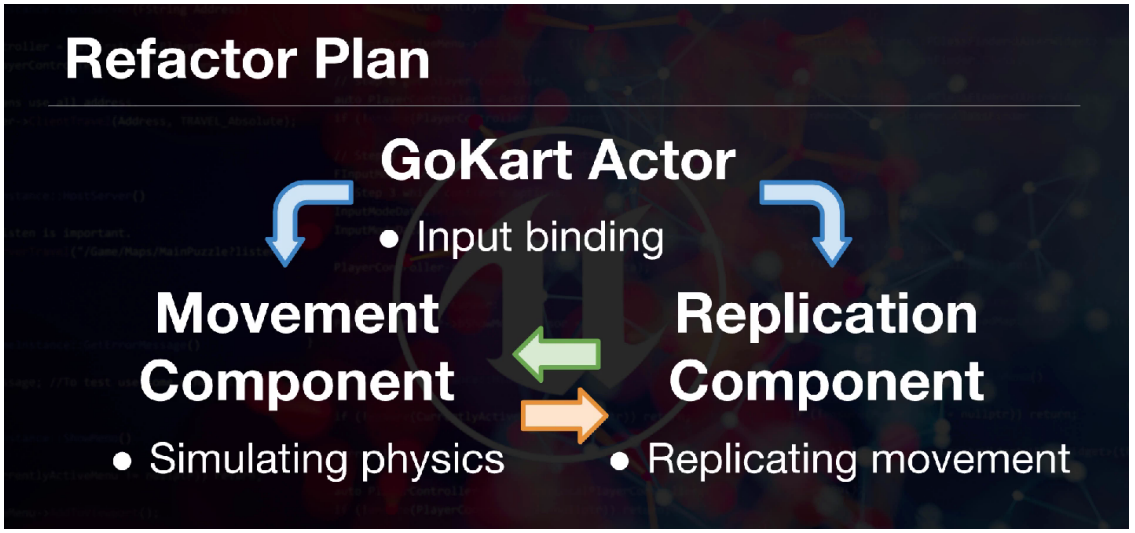
Solutions:



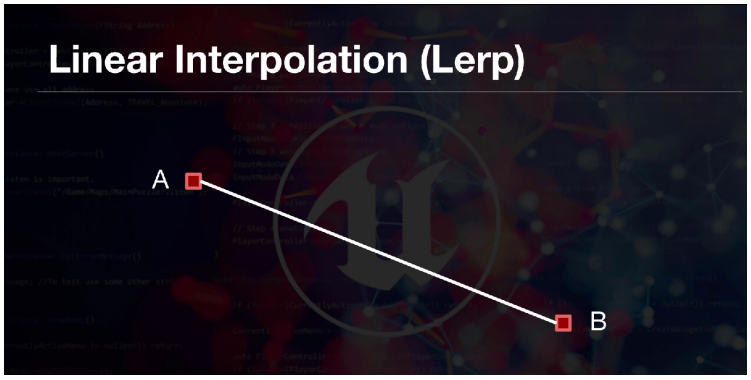
Let's start!



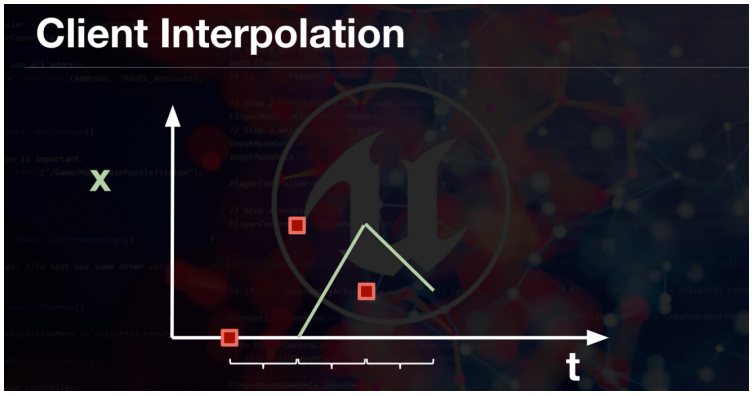
Replication Refactor for movement bindings



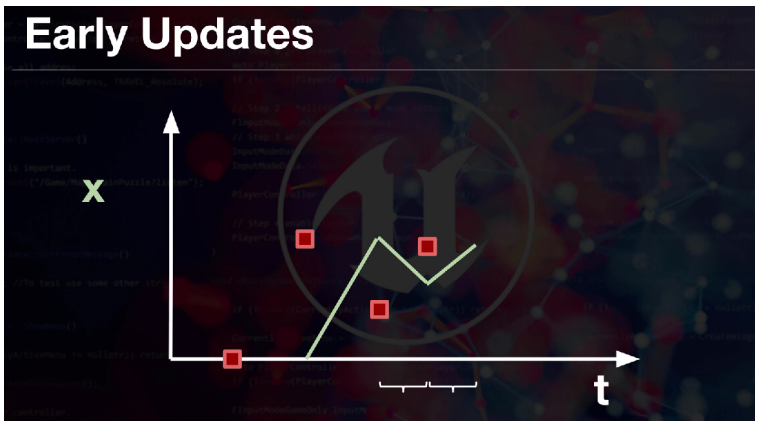
What is the correct smooth movement graph ?



Client Interpolation



Early Updates



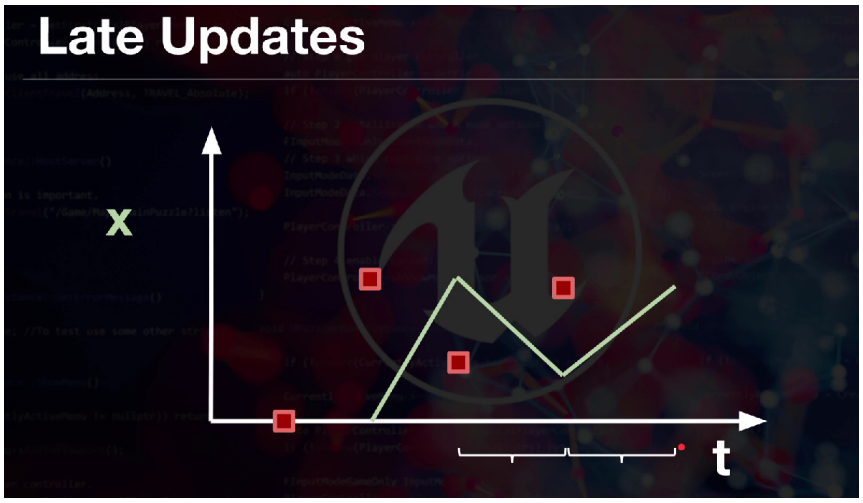
OnTick:

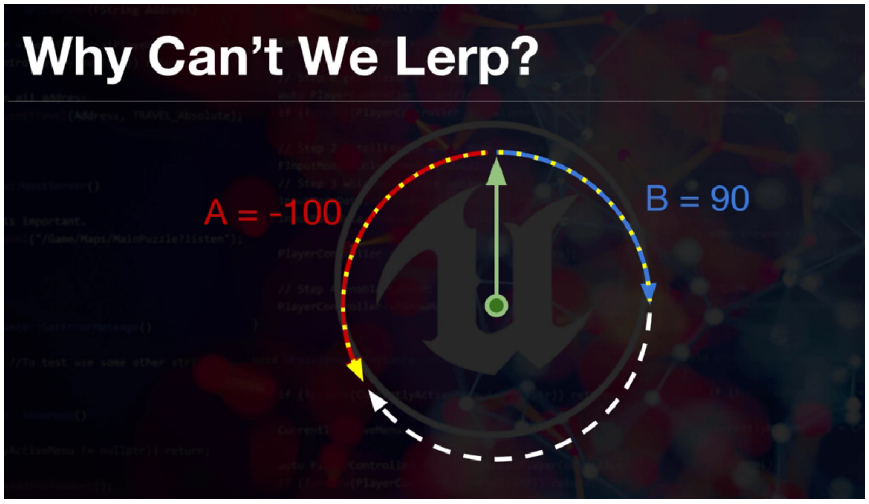
```
TargetLocation = ServerState.Location
LerpRatio = TimeSinceUpdate / TimeBetweenLastUpdates
NextLocation = Lerp(StartLocation, TargetLocation, LerpRatio)
SetLocation(NextLocation)
```

OnRep:

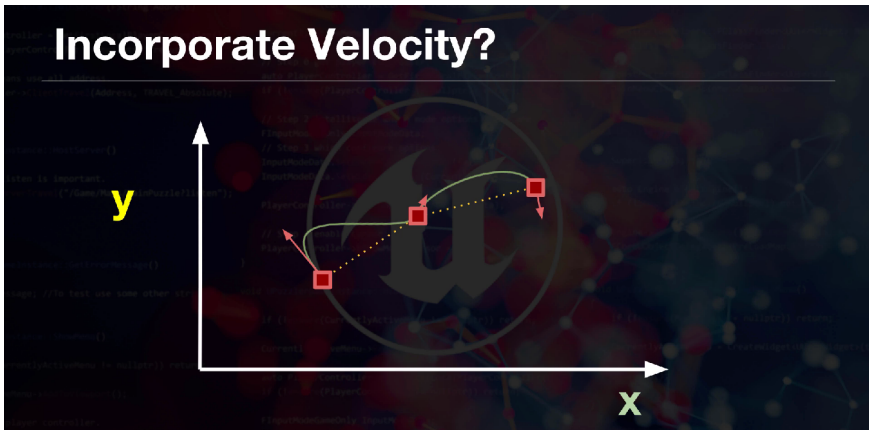
```
StartLocation = GetLocation()
```

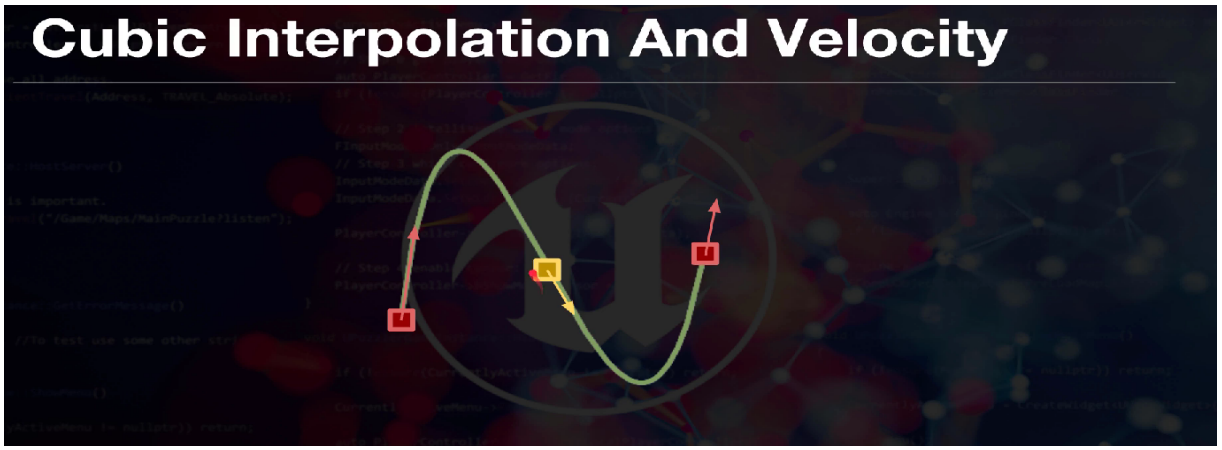
Late Updates

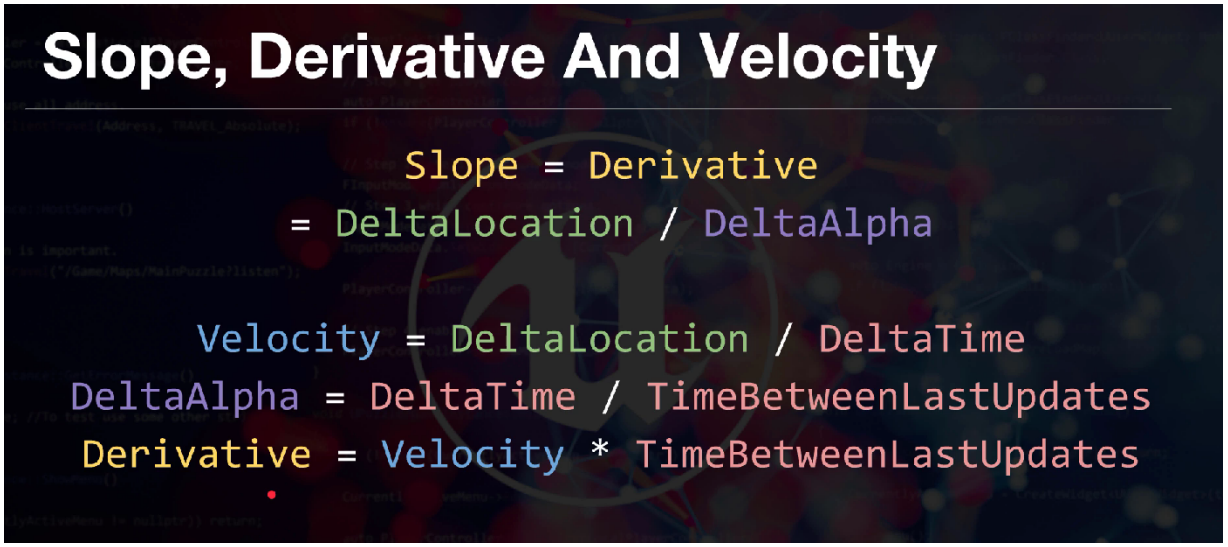




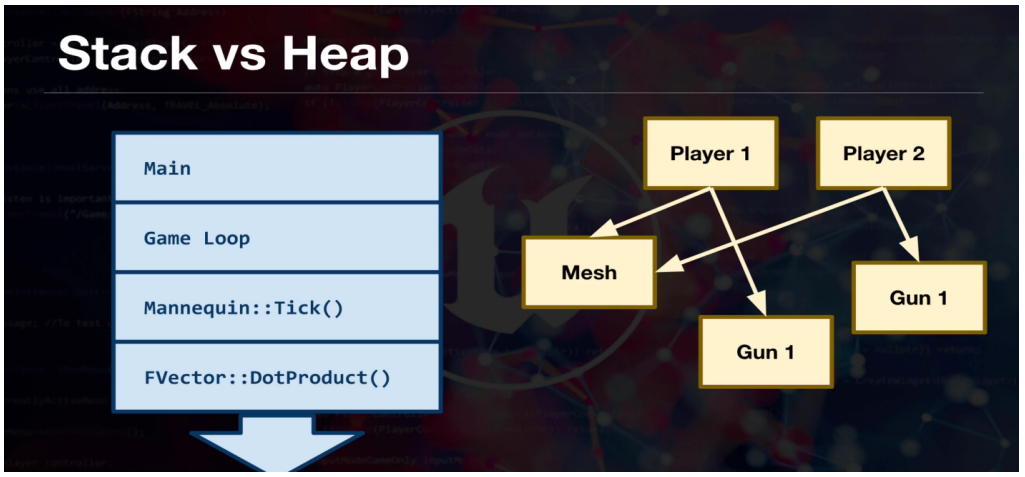
Simulate and make smooth movement

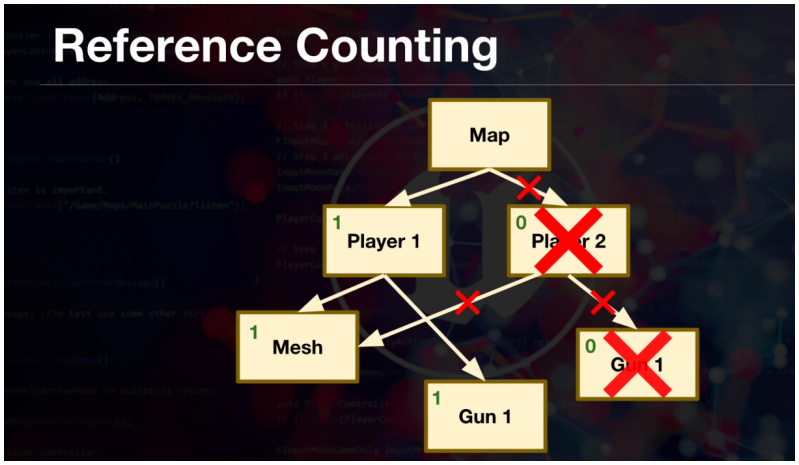


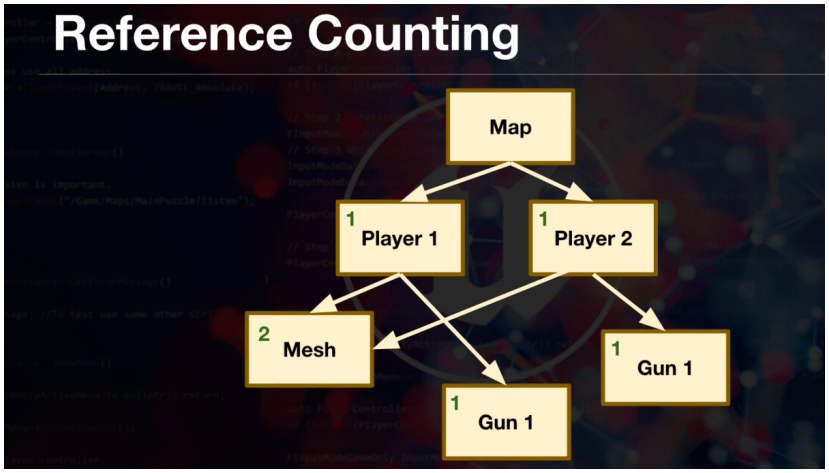




Garbage Collection on Unreal Engine





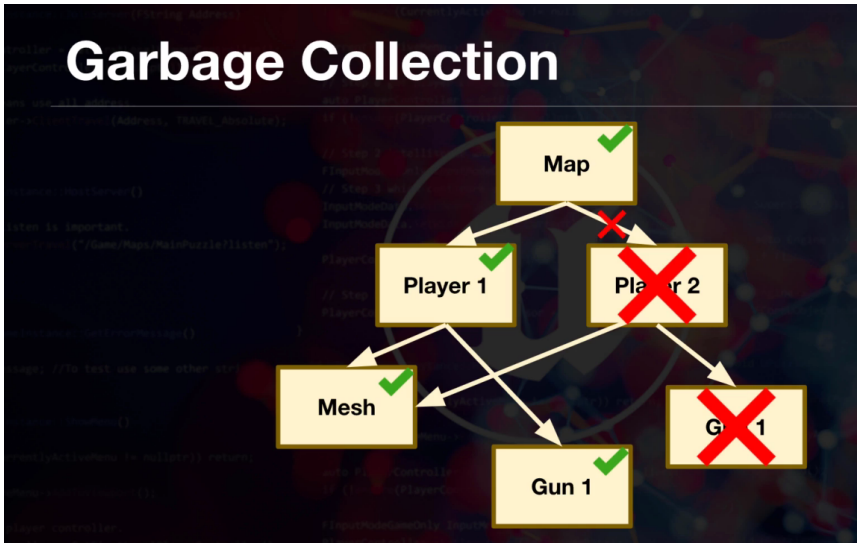


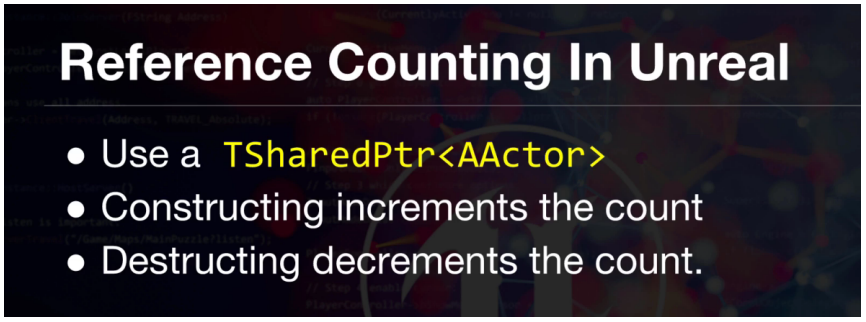
You can check multiplayer puzzle mechanic video for this.

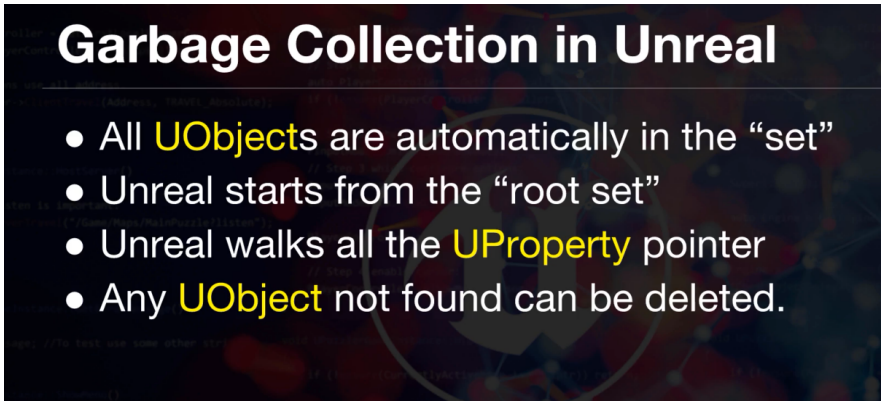
```
TArray<UObject*> Referencers;

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: Reallocating 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







Unreal Engine Environment

Collision

When you import static mesh object for example chair and if you want to add collision your own object, you have 3 methods for this situations.

1) Select object -> edit and add box, sphere or other collision.

2) Select object -> edit and add complex collision with some settings (hull count).

3) Custom collision for your object. In the modelling program add the collison and export. Make sure your own collision to use the UCX_ prefix before the asset you are working with. Also add _01 at the end of the filename for many pieces.

Example: SM_Chair would be UCX_SM_Chair_01

Material

Create 3 texture sample for your material.

1) Texture sample for base color. (diffuse)

2) Texture sample for normal. (mask)

3) Texture sample R node for ambient occlusion, G node for roughness, B node for metalic. (normal)

press the right click and select "convert to parameters" for all of them.

Now we can create material instance. Right click the material and select "create material instance" and rename with MI_ prefix.

Texture

When you import the texture, don't forget the check compression settings. For example 3 texture for your object. 1 diffuse, 1 mask, 1 normal map for your object. Mask compression settings it need to be selected "Mask (no rgb)" and other texture need to be default DXT in the editor. You can check with the right click and edit.

Open your material instance where you are created from material. You can see diffuse, mask and normal section on the right. Just drag and drop correct texture this area.

Now open your static mesh and change your default material to created material instance.

Light

Unreal Engine light types:

Directional Light: Sun or moon light. You can change sun location with Ctrl + L.

Point Light: Turn of cast shadow settings in this light because this light type too expensive than others.

Spot Light: Most common light type for room.

Rect Light: Useful light for tv screen.

Sky Light: When you check the light density, you can see some object red or other color. It need to be green or similar color. You can easily change this situation. Click the object and select edit -> right panel general settings in below you can see "Light Map Resolution" -> change the lower value. For example 256 to 128 -> In the editor Build section select "Build Lightning" and wait.

If your screen is too dark, you can change light intensity.

Extra: You can use a light profile for your lamp. Select your object -> in the details panel, find light profile -> select IES Texture.

In the World Settings you can see the light mass section. In this section, you can enable ambient occlusion and you can change light quality or set other light settings.

Lightmass Portal: for the window opening effect.

Sphere Reflection Capture: for the metallic and roughness material.

Note: If you want a reflection size you need to open Project Settings -> Rendering -> Reflection -> Reflection Capture Size. For example, change 128 to 256 values. After the changing reflection capture, in the build section -> select build reflection capture.

Post Process Volumes

Add the post process volume and atmospheric fog. You can change game looking style with post process volume. You can think like photo mode in game. You can set black and white, exposure, or other stuff.

Unreal Engine Optimizing

Identify Key Metrics

1) What hardware will the end user be using ?

2) Avoid unneeded features and effects

3) Framerate:

VR:90 FPS

Console/PC: 60 FPS

Let's visualize and track the problem.

Note: If you want to see game fps, you can open with the "Stat fps" command or press down the arrow button. If you want to see more details about GPU, Draw, Fps etc. you can open with the "Stat unit" command.

GPU

GPU visualizer shortcut: ctrl + shift + comma

You can see which one affects the GPU with loading duration and you can fix it.

In this case the post process effect -> bloom effect is affecting the GPU for a reason you need to find the post process effect -> bloom -> change the method with standard. Search the point light and delete all of them.

CPU

CPU visualizer command: Stat SceneRendering

Draw calls meaning it's going to ue source and calling the polygons.

If you want to reduce draw calls you can remove, merge objects, remove extraneous material channel or reduce the amount of shadows being cast by dynamic lights.

Thread

If you want the game thread to be the bottleneck you can type "Stat Game". This will show you how many tick operations are occuring on frame in the Viewport.

Memory

You can see the " TEXTURE STREAMING POOL OVER" warning in the Unreal Engine because we don't have enough memory allocated to the Unreal Engine to display the highest-resolution textures.

Let's fix that issue!

In the Editor, Window -> Statistics -> Click Primitive Stats and select "Texture Stats". Now you can see the fully loaded memory. Find the largest texture and let's start to reduce it. Open the largest texture -> Change "Maximum Texture Size" with power of two values (2048) -> save the texture and back to the statistics window.

Note: If any of your textures being used aren't the power of two textures, they can affect your memory usage. Because they don't have a lower mipmap texture to switch. Resolve that issue, open the texture -> in the Texture section, change "Power of Two Mode " to pad texture to a higher value. Alternative solution is you can edit your texture with simple programs and reimport to Unreal Engine.

Alternative solution of the "Texture Streaming Pool Over", you can allocate more ram with console command. Type "streaming.poolsize amount". Example usage: "streaming.poolsize 8000". Now we can check with the "stat memory" command.

Visualize the Complexity

You can check with:

1) Light Complexity (are too many overlapping dynamic lights?)

2) Shader Complexity (models are optimize or not)

3) Quad overdraw (you can see each pixels is optimize or not)

Some Additional Settings

1) Project Settings -> In the Project section, choose Target Hardware (mobile, pc)

2) Project Settings -> Engine -> Under the framerate, we can select fixed fps if we don't need to streaming extremely high frame rate -> and we can select "Smooth Frame Rate" for help stuttering if you can't avoid frame rate issues in the editor.

3) Project Settings -> Rendering -> Under the Culling section, make sure the "Occlusion Culling" is enabled. (this setting determines whether objects will be hidden or if they are visible to the camera.) -> Scroll down and under the "Reflections", "Reflection Capture Size" be sure under the power two.

4) If you don't need Ray Tracing, Global illumination disables it.

LOD

When looking at a model, the model is made up of triangles. You need to make sure that this number of triangles changes when you move away or get closer. This way you don't have to render a model that is too far away in high resolution.

If you want to change LOD settings right click to static mesh -> Details -> Under the LOD Settings, you can change "LOD Group". Example when you selected "SmallProp" you can see your mesh triangle amount is changing for your away or get closer movement.

Now if you want to modify the LOD reduction settings, under the "LOD Picker" section you can enable "custom" option. Now you can scroll down and see the LOD 1, LOD 2, LOD 3 section. Open the LOD 1 reduction settings and modify the amount of decimation you would like.

We can change every mesh LOD settings with new one. Select all mesh and press right click-> Asset Actions -> Bulk Edit via Property Matrix -> Under the LOD Settings, change the LOD Group name with "SmallProp" (case sensitive).

DataSmith Plugin

Plugins -> Enable Datasmith Importer Plugin -> Enable Dataprep Editor Plugin

Right click and select Dataprep -> Dataprep Asset -> rename with Dataprep_1

Before adding Unreal Engine, we can look at its preview to import or files to import and set up action that we would like to occur on the file here.

Press the Input + icon -> Choose datasmith file -> Press the import button and wait.

Now you can change for example triangle count. Only you need to do drag and drop the right area and give an amount (select greater than 500). After doing that, add LOD Group Node and select type. If you finished all settings now you can press the execute button and start waiting. When it's done you can press commit to add the model to your viewport and content browser.

Jacketing

Plugins -> Enable Polygon Editing

Right click the mesh and select jacketing -> You can see different settings in the pop-up.

1) Voxel prediction determines the precision of the skin.

2) Gap Max Diameter determines how large of gaps we should skip over if any are present.

3) You can choose a level or mesh scan.

a) Level: if object are inside the external casing and modify those object that are not seen from any angle outside of the exterior object. (large buildings, vehicles)

b) Mesh: it will instead consider occlusion at the level of individual triangles and delete polygons that are not visible from the exterior. If we really don't need to render complex objects (have a large amount of detail), it will be helpful. You can check on the wireframe mode (for it's necessary or not decision).

Note: This function is only available on 64 bit windows.

Auto UV Generation

When you import the geometry, don't forget the enable "generate lightmap uv" option.

UV0 is the default UV channel that was created to inform how textures would display on the mesh.

UV1 i s for the lightmap.

In the UV section, select unwrap UV option -> if you keep the angle and UV0 was blank, you need to select specify channel option -> press the proceed button.

If you have the more complicated geometry, you can use the generate UV tool. This tool will allow more editing options for the UV.

Note: You will get the best result in the 3D program for UV.