

Section 2 - Menu System

These are the slides that accompany the Unreal Multiplayer course.

Looking for something, try searching [our GitHub repo.](#)

Enjoy your stay!

*Sam & Ben
GameDev.tv*

Live Google Slides at bit.ly/multiplayerslides



Introduction to Menu System

Create A Menu Blueprint

Create A Menu

- It should have a...
- Title.
- **Host** button.
- **Join** button.



Accessing UI Classes in C++

Save The Menu Class

- Create a **MenuClass** field.
- Use the class finder to save to it.
- Find out what type it should have.
- Log the name found in init.
- Try building without UMG.



Load And Display UMG In C++

Create A Scalable Layout

Main Menu

Host

Join



Create A MainMenu Level

- Use an empty level.
- Make the **LoadMenu BlueprintCallable**.
- Load the menu automatically.
- A good place is BeginPlay.



Changing UI Input Modes

```
Instance::JoinServer(FString Address)  
  
Controller = GetFirstLocalPlayerController();  
PlayerController != nullptr)) return;  
  
// All players use all address.  
PlayerController->ClientTravel(Address, TRAVEL_Absolute);
```

```
Instance::HostServer()
```

```
Listen is important.
```

Changing UI Input Modes

```
Instance::GetErrorMessage()
```

```
Message; //To test use some other string
```

```
Instance::ShowMenu()
```

```
CurrentlyActiveMenu != nullptr)) return;
```

```
Menu->AddToViewport();
```

```
Player controller.
```

```
Controller = GetFirstLocalPlayerController();
```

```
PlayerController != nullptr)) return;
```

```
if (!ensure(CurrentlyActiveMenu != nullptr)) return;
```

```
CurrentlyActiveMenu->AddToViewport();
```

```
// Step 0 get player controller.
```

```
auto PlayerController = GetFirstLocalPlayerController();
```

```
if (!ensure(PlayerController != nullptr)) return;
```

```
// Step 2 Intellisenet which mode options are.
```

```
FInputModeGameOnly InputModeData;
```

```
// Step 3 which configures options.
```

```
InputModeData.SetInputMode(FInputModeGameOnly);
```

```
InputModeData.SetInputMode(FInputModeGameOnly);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```


Rediscover the API

- Set the input mode.
- Make the mouse cursor visible.
- Use the documentation to help you.
- Try not to peak at the code.



Advanced UMG Widget Layout

Custom Buttons And Fonts

Make A Pretty UI

- Add a background.
- Customize the buttons.
- Choose a custom font.



Solution: Custom Buttons And Fonts

Connecting UMG to C++

```
Instance::JoinServer(FString Address)
{
    Controller = GetFirstLocalPlayerController();
    if (PlayerController != nullptr) return;

    // Means use all address.
    PlayerController->ClientTravel(Address, TRAVEL_Absolute);
}
```

```
Instance::HostServer()
```

```
Listen is important.
```

```
ServerToClient
```

```
Instance::GetErrorMessage()
```

```
Message; //To test use some other string
```

```
Instance::ShowMenu()
```

```
CurrentlyActiveMenu != nullptr)) return;
```

```
Menu->AddToViewport();
```

```
player controller.
```

```
Controller = GetFirstLocalPlayerController();
```

```
PlayerController != nullptr)) return;
```

```
if (ensure(CurrentlyActiveMenu != nullptr)) return;
```

```
CurrentlyActiveMenu->AddToViewport();
```

```
// Step 0 get player controller.
```

```
auto PlayerController = GetFirstLocalPlayerController();
```

```
if (ensure(PlayerController != nullptr)) return;
```

```
// Step 2 Intellisece which mode options are.
```

```
FInputModeGameOnly InputModeData;
```

```
// Step 3 which confides options;
```

```
InputModeData.SetInputMode(FInputModeGameOnly);
```

```
InputModeData.SetInputMode(FInputModeGameOnly);
```

```
// Step 4 enable cursor;
```

```
PlayerController->bShowMouseCursor = true;
```

```
}
```

```
void UPuzzlerGameInstance::HitMenu()
```

```
if (ensure(CurrentlyActiveMenu != nullptr)) return;
```

```
CurrentlyActiveMenu->RemoveFromParent();
```

```
auto PlayerController = GetFirstLocalPlayerController();
```

```
if (ensure(PlayerController != nullptr)) return;
```

```
FInputModeGameOnly InputModeData;
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->bShowMouseCursor = false;
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
ConstructorHelpers::FClassFinder<UserWidget> MenuClass;
```

```
MenuClass = MenuClassFinder.Class;
```

```
ConstructorHelpers::FClassFinder<UserWidget> JoinMenuClass;
```

```
JoinMenuClass = JoinMenuClassFinder.Class;
```

```
void UPuzzlerGameInstance::LoadMainMenu()
```

```
{
```

```
if (ensure(MenuClass != nullptr)) return;
```

```
CurrentlyActiveMenu = CreateWidget<UserWidget>(this, MenuClass);
```

```
CurrentlyActiveMenu->AddToViewport();
```

```
void UPuzzlerGameInstance::LoadJoinServerMenu()
```

```
{
```

```
if (ensure(JoinMenuClass != nullptr)) return;
```

```
CurrentlyActiveMenu = CreateWidget<UserWidget>(this, JoinMenuClass);
```

```
CurrentlyActiveMenu->AddToViewport();
```

```
void UPuzzlerGameInstance::LoadJoinServerMenu()
```

```
{
```

```
if (ensure(JoinMenuClass != nullptr)) return;
```

```
CurrentlyActiveMenu = CreateWidget<UserWidget>(this, JoinMenuClass);
```

```
CurrentlyActiveMenu->AddToViewport();
```

```
void UPuzzlerGameInstance::LoadJoinServerMenu()
```

```
{
```

```
if (ensure(JoinMenuClass != nullptr)) return;
```

```
CurrentlyActiveMenu = CreateWidget<UserWidget>(this, JoinMenuClass);
```

```
CurrentlyActiveMenu->AddToViewport();
```

```
void UPuzzlerGameInstance::LoadJoinServerMenu()
```

```
{
```

```
if (ensure(JoinMenuClass != nullptr)) return;
```

```
CurrentlyActiveMenu = CreateWidget<UserWidget>(this, JoinMenuClass);
```

```
CurrentlyActiveMenu->AddToViewport();
```


How Does BindWidget Work?

- Look at errors.
- Try to describe how it functions.



Initialisers and UButton Callbacks

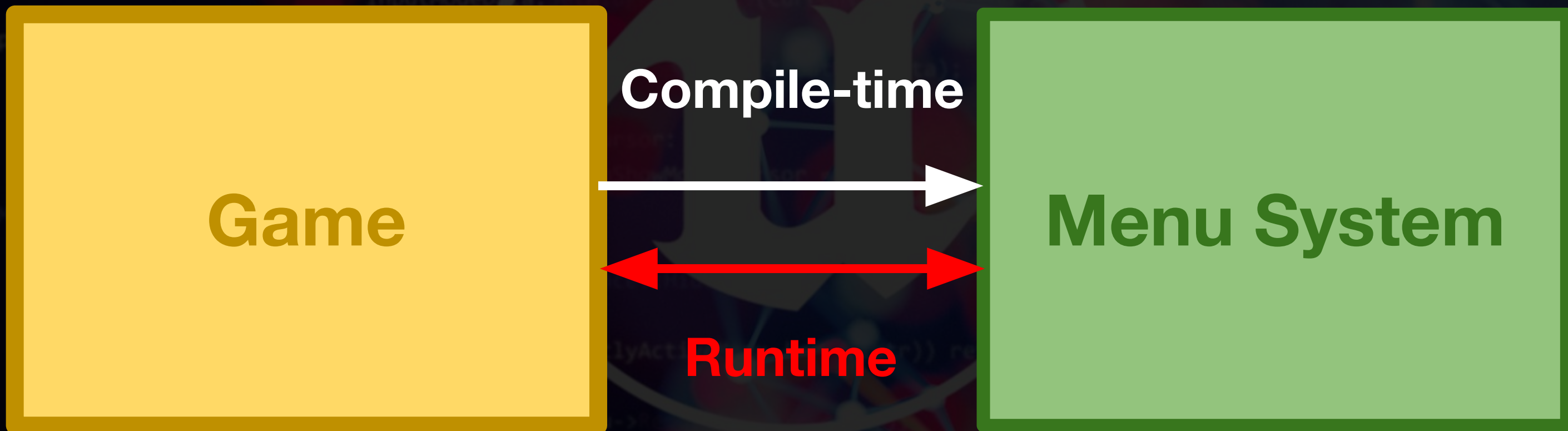
Bind the OnClick

- Just like other dynamic events.
- Print out to the console.
- Test!



Interfaces To Invert Dependencies

Our Dependencies



Live Google Slides at bit.ly/multiplayerslides

Dependency Inversion

GameInstance ← Includes + **Calls** **MainMenu**

Live Google Slides at bit.ly/multiplayerslides



Dependency Inversion



Live Google Slides at bit.ly/multiplayerslides



Inject The **MenuInterface**

- Set the **MenuInterface**.
- Remember best practices about setting.
- What happens Interface isn't fully implemented.
- Test that “hosting” works.



Solution: Injecting Dependencies

Deactivating Menus

```
Instance::JoinServer(FString Address)  
  
Controller = GetFirstLocalPlayerController();  
PlayerController != nullptr)) return;  
  
// Ensures all address.  
PlayerController->ClientTravel(Address, TRAVEL_Absolute);
```

```
Instance::HostServer()
```

```
Listen is important.
```

```
ServerToClient
```

```
Instance::GetErrorMessage()
```

```
Message; //To test use some other string
```

```
Instance::ShowMenu()
```

```
CurrentlyActiveMenu != nullptr)) return;
```

```
Menu->AddToViewport();
```

```
Player controller.
```

```
Controller = GetFirstLocalPlayerController();
```

```
PlayerController != nullptr)) return;
```

```
if (ensure(CurrentlyActiveMenu != nullptr)) return;
```

```
CurrentlyActiveMenu->AddToViewport();
```

```
// Step 0 get player controller.
```

```
auto PlayerController = GetFirstLocalPlayerController();
```

```
if (ensure(PlayerController != nullptr)) return;
```

```
// Step 2 Intellisenet which mode options are.
```

```
FInputModeGameOnly InputModeData;
```

```
// Step 3 which configures options.
```

```
InputModeData.SetInputMode(FInputModeGameOnly);
```

```
InputModeData.SetInputMode(FInputModeGameOnly);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
PlayerController->SetInputMode(InputModeData);
```

```
// Step 4 enable cursor.
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
PlayerController->bShowMouseCursor = false;
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()
```


Reverse The Setup

- Use setup as a base.
- Hide the cursor.
- Set the input mode.
- Remove from the viewport.



Sub-Menus With Widget Switchers

Menus With Style!

- Style the Text box.
- Improve the layout.
- Ensure everything is legible.
- Add a heading.



Solution: Styling Our Sub-Menu

Navigating Menus In C++

Create A Back Button

- Put it in a sensible place.
- Bind it.
- Create a callback.
- Switch to the main menu.



Reading Text Fields From C++

Make It Functional!

- Hook up the join button.
- Tweak the interface.
- Call the join function.
- Test.



Challenge: Loading In-Game Menus

In Game Menu

- Create a separate widget.
- With a C++ class.
- Add **cancel** and **main menu** buttons.
- Load the menu on **ESC** key pressed.



Challenge: Leaving A Server

Cancel And Quit

- Create function in the game instance.
- Implement cancel.
- Implement quit to main menu.



Quitting A Game From C++

Quit The Game

- Add a new button to the Main Menu.
- Bind the button.
- Implement quit.



Menu System Wrap-up