

# Section 1 - Puzzle Platforms

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](https://bit.ly/multiplayerslides)*





# Introduction to Puzzle Platforms



# Connecting Two Players

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

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

```
Listen is important.
```

# Connecting Two Players

```
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.SetInputModeGameOnly();
```

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

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

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

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

```
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()
```

```
{
```

```
Super::LoadMainMenu();
```

```
auto Engine = GetEngine();
```

```
Engine->SetInputMode(FInputModeGameOnly);
```

```
Engine->SetInputMode(FInputModeGameOnly);
```

```
CoreObjectDelegates::PreLoadMap.AddLambda([this, &]() {
```

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

```
{
```

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

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

```
LoadJoinMenu();
```

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

```
{
```

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



# Under The Hood

1. Unreal loads the **Map**
2. The **Map** specifies a **GameMode**
3. The **PlayerController** joins the **Map**
4. It ask the **GameMode** to spawn a **Pawn**
5. The **Pawn** is linked to the **PlayerController**.



# Get Set Up.

- Update Unreal
- Create a 3rd person project
- Test with multiple players
- BONUS: set up version control.

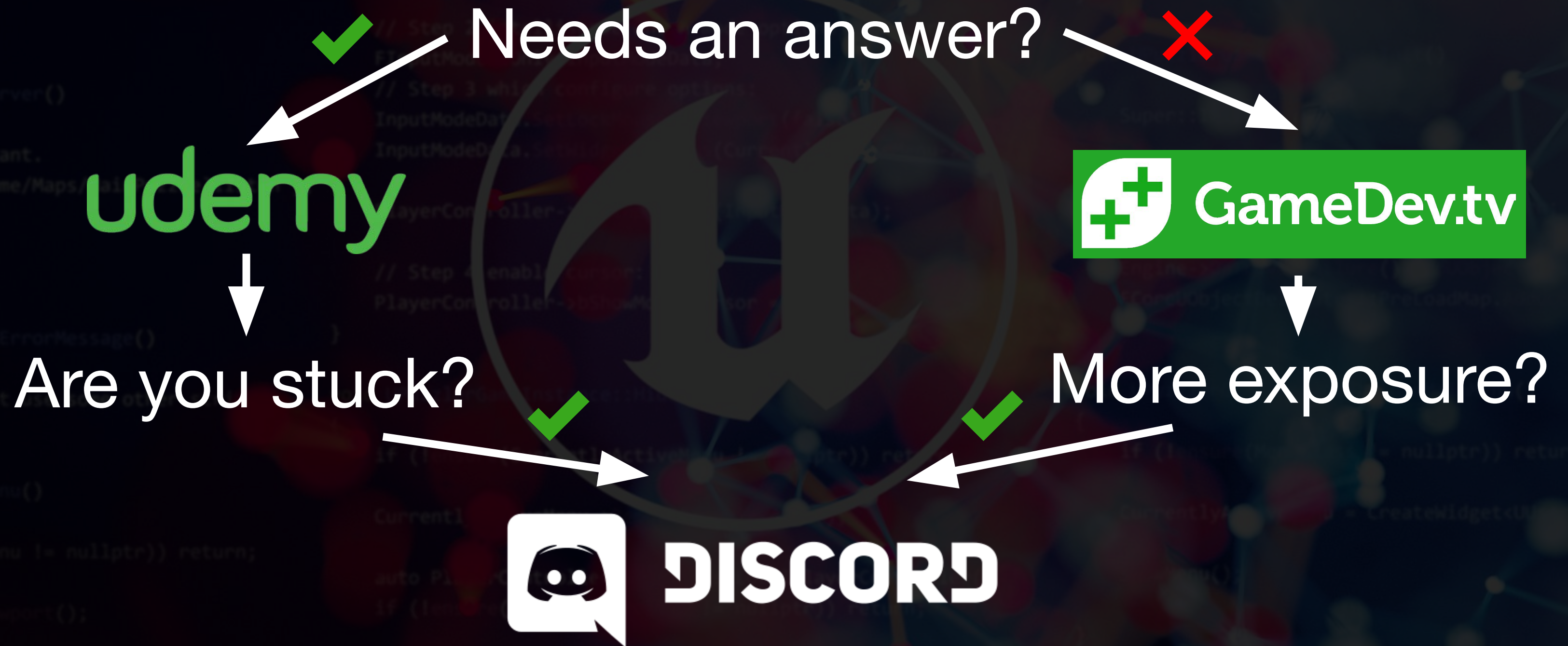




# How To Be An Active Student



# Navigating The Community



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Say “Hello World”!

- Introduce yourself to the community
- Use the map to guide you
- Share your “Why”.





# Surveying The Multiplayer Space



# Types Of Multiplayer Game

	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

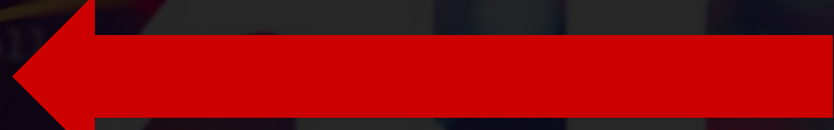
Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Session-Based Stages

1. Discovery

2. Connection  We are here

3. Synchronisation

Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# What Will You Build?

- Review the outline of game types
- What background are you coming from?
- Tell us what you want to make
- What difficulties do you foresee?





# Meet The Client-Server Model



# Input And State

State #1

Tick

State #2

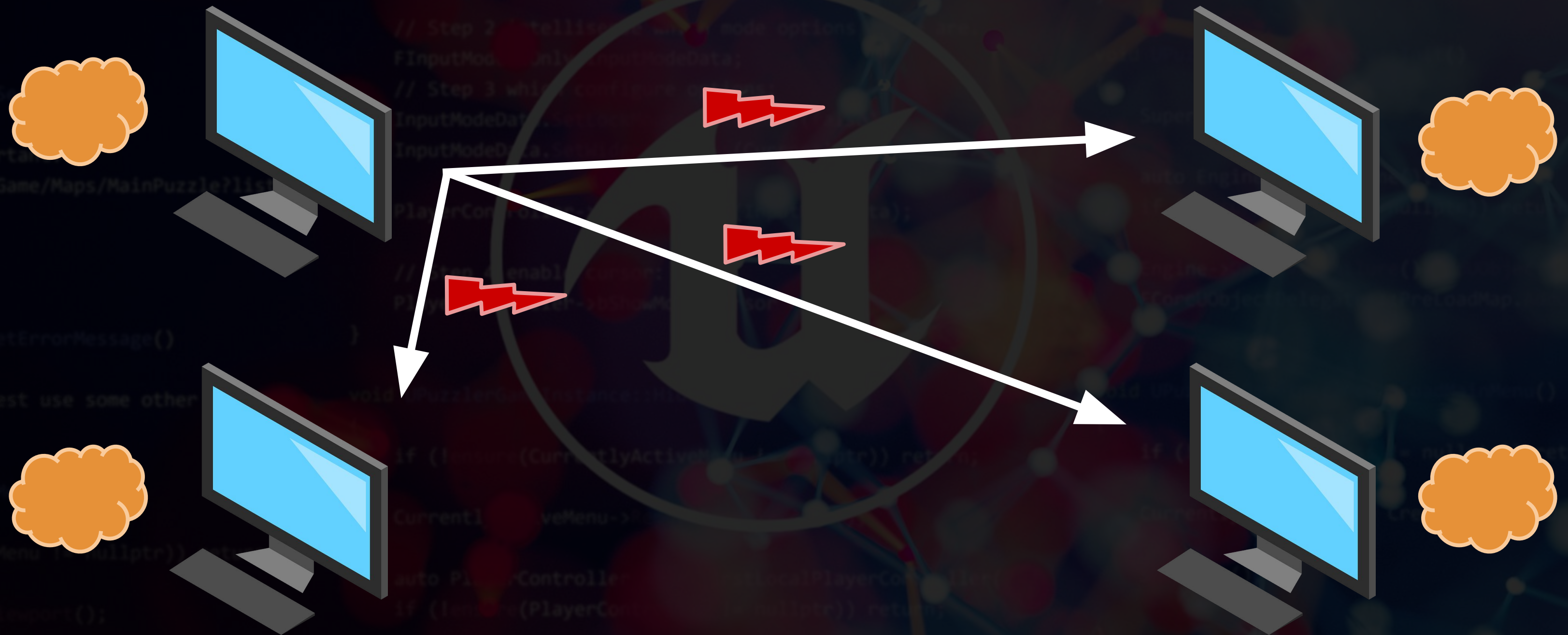
Actions

Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Peer-To-Peer



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)



# Client-Server



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)



# Connect Your Mini-Network

- Find your IP address
- Launch a server
- Launch some clients
- Ensure they are connected to the server.





# Detecting Where Code Is Running



# Move Only On Server

- Should only move on Server
- Abstract speed to variable.

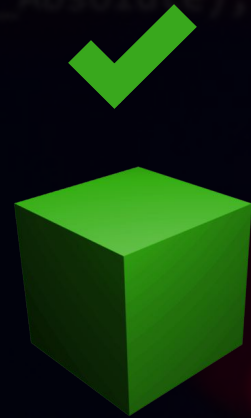




# Authority And Replication



# Actors And Replication



pos = 10 ✓  
color = green



pos = 10  
color = red



pos = 10  
color = green



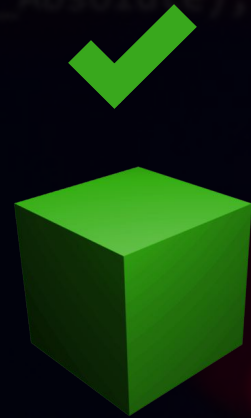
pos = 10  
color = red

Live Google Slides at [bit.ly/multiplayerslides](http://bit.ly/multiplayerslides)





# Actors And Replication



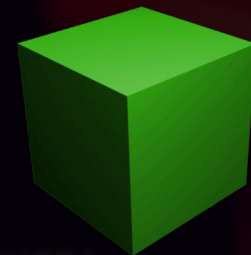
pos = 20 ✓  
color = green



pos = 10  
color = red



pos = 20



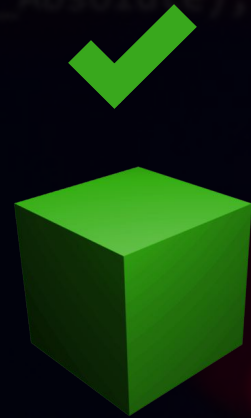
pos = 10  
color = green



pos = 10  
color = red



# Actors And Replication



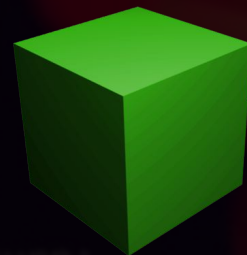
pos = 20 ✓  
color = green



pos = 10  
color = red



pos = 20



pos = 20  
color = green



pos = 10  
color = red



# What If?

- Update the cube on the client only
- What happens?
- Can you explain the phenomenon?





# Widgets For FVector Properties



# Revising Vectors



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Move The Platform

- Implement an **FVector** property
- Calculate the direction
- Move in that direction
- Don't worry about returning yet.

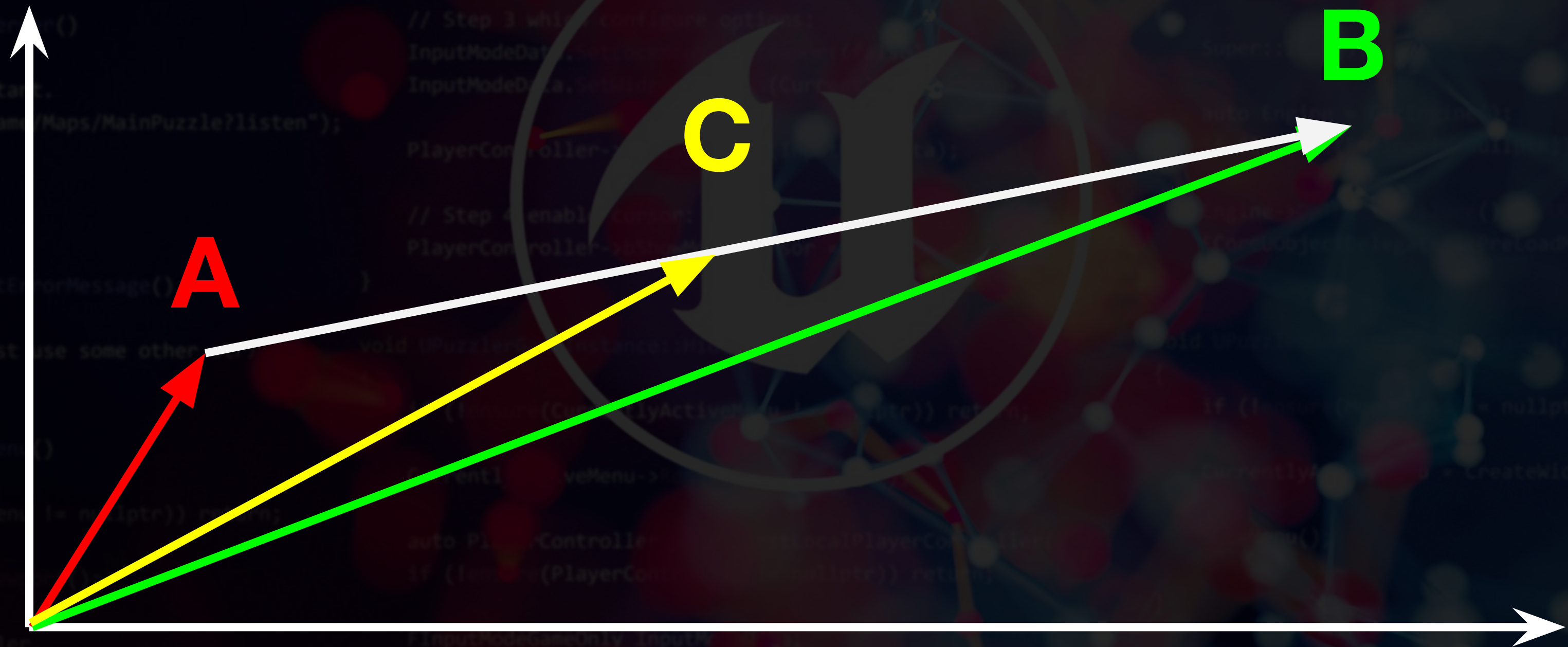




# Sending The Platform Back



# Are We There Yet?



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)



# Swapping Start and Target

- What condition should you use?
- Research the **FVector** API.
- Implement the condition.
- Swap the variable.





# Set Up A Simple Puzzle



# Set Up Your Puzzle

- Add multiple platforms.
- Allow access to new areas.
- Have at least 3 hops.
- Allow some safe zones.

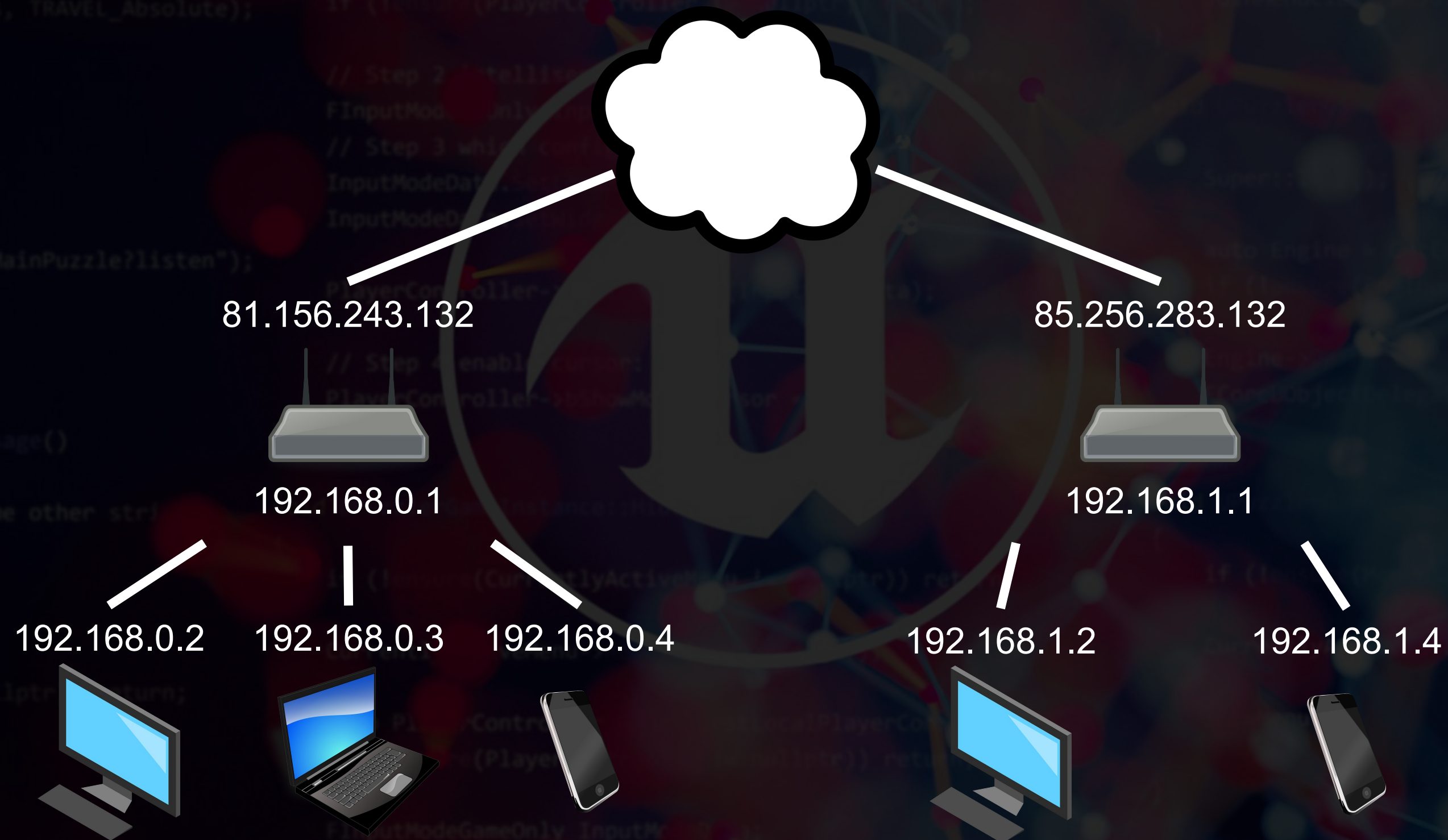




# Playing Over The Internet



# The Ugliness Of NATs



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)



# Find Someone To Play With

- Create a post in the forum.
- Put available times in the title.
- Use this lecture's link.
- Show a screenshot of the game.
- While you wait, reply to others.
- Play your game over Hamachi.





# Set Up A Platform Trigger



# Create A Blueprint Child

- Scale the box sensibly.
- Attach a mesh.
- Place it in the world.





# Handling Overlap Events in C++



# Register For End Overlap

- Repeat the steps.
- Check with a log message.
- Remember to reload the map.





# Activating Platforms From Triggers



# Implement The Logic

- Fill out the interface.
- Activate from the callbacks.
- Conditionally activate the platforms.
- Test in the world.





# When To Use A GameInstance



# Init Or Construct?

- Implement both methods.
- Log out from both.
- Try starting the game in different ways.
- Restart the editor.
- What do you notice?





# Console Commands With Exec



# Exec Compatible Classes

- PlayerControllers
- Possessed Pawns
- HUDs
- Cheat Managers
- Game Modes
- Game Instances

Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Make A **Join** Command

- Repeat the steps for Host.
- Add an address argument.
- How does this show on the console?
- Print said address.
- HINT: **FString::Printf()**







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

```
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.SetCapture(true);  
PlayerController->SetInputMode(InputModeData);  
  
// Step 4 enable cursor:  
PlayerController->bShowMouseCursor = true;
```

```
UPuzzlerGameInstance::UPuzzlerGameInstance()  
  
ConstructorHelpers::FClassFinder<UserWidget> MenuClass(  
    "MenuClass", MenuClassFinder.Class);  
  
ConstructorHelpers::FClassFinder<UserWidget> JoinMenuClass(  
    "JoinMenuClass", JoinMenuClassFinder.Class);  
  
void UPuzzlerGameInstance::LoadMainMenu()  
  
void UPuzzlerGameInstance::LoadJoinServerMenu()  
  
void UPuzzlerGameInstance::LoadPuzzleMenu()  
  
void UPuzzlerGameInstance::LoadPuzzleMenu()
```

```
Instance::HostServer()  
  
Listen is important.  
ServerToClientMap[Address] = Instance;
```

```
void UPuzzlerGameInstance::HideMainMenu()  
  
if (!ensure(CurrentlyActiveMenu != nullptr)) return;  
CurrentlyActiveMenu->RemoveFromParent();  
  
auto PlayerController = GetFirstLocalPlayerController();  
if (!ensure(PlayerController != nullptr)) return;  
  
FInputModeGameOnly InputModeData;  
PlayerController->SetInputMode(InputModeData);  
  
PlayerController->bShowMouseCursor = false;
```

```
CoreObjectDelegates::PreLoadMap.Add(this, &UPuzzlerGameInstance::PreLoadMap);  
  
void UPuzzlerGameInstance::LoadMainMenu()  
  
if (!ensure(MenuClass != nullptr)) return;  
CurrentlyActiveMenu = CreateWidget<UserWidget>(this, MenuClass);  
CurrentlyActiveMenu->AddToViewport();  
CurrentlyActiveMenu->Init(this);  
  
void UPuzzlerGameInstance::LoadJoinServerMenu()  
  
if (!ensure(JoinMenuClass != nullptr)) return;
```

# Hosting Servers With **ServerTravel**



# Travelling



ServerTravel



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Travelling



ServerTravel



Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Test Hosting

- Host a server,
- Connect with a standalone game,
- What happens without “?listen”.





# Joining Servers With ClientTravel



# Client Travelling



ServerTravel



ClientTravel

Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Client Travelling



ServerTravel



ClientTravel

Live Google Slides at [bit.ly/multiplayerslides](https://bit.ly/multiplayerslides)





# Implement Join

- Implement travelling.
- Test hosting with joining.
- Use a standalone game.
- Celebrate!





# Sharing Your Game On Itch.io





# Share Your Game

- Build it.
- Create an Itch game.
- Upload.
- Share it with a friend.





# Puzzle Platforms Wrap-up