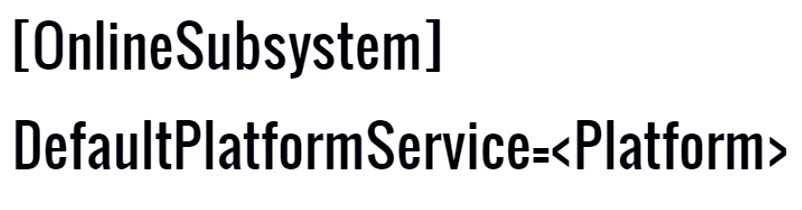
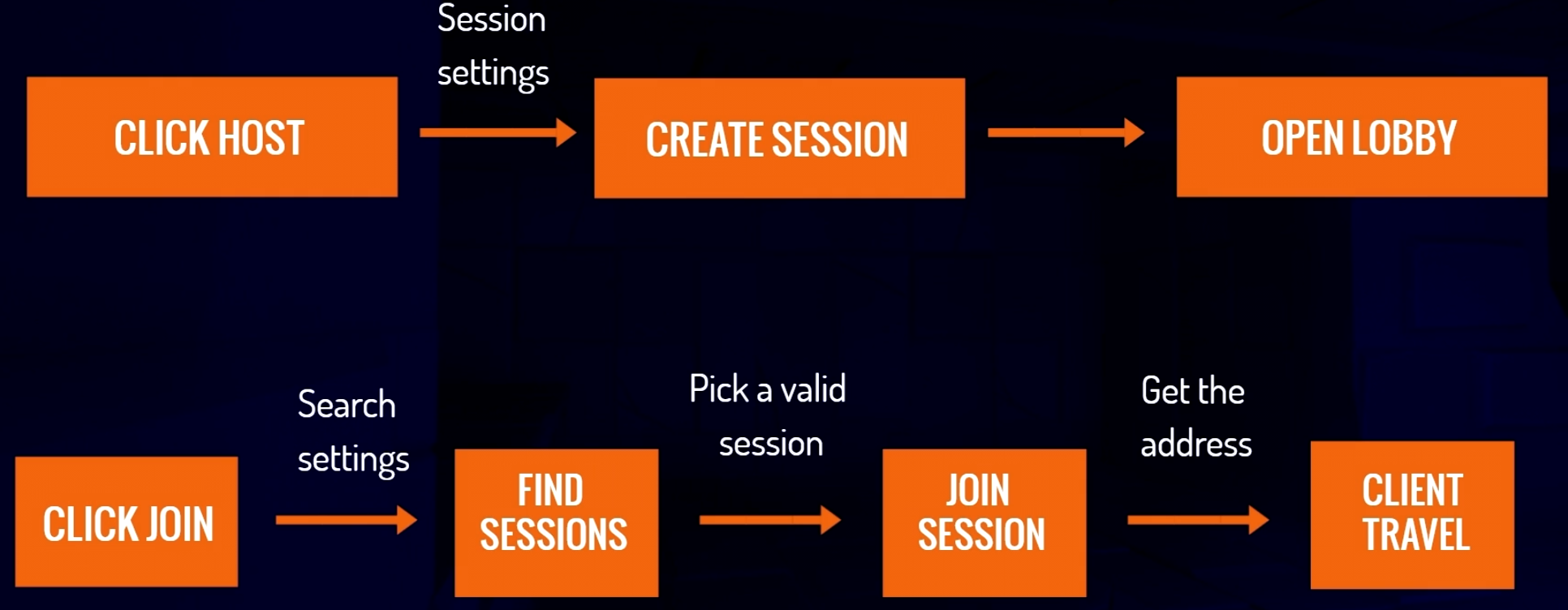
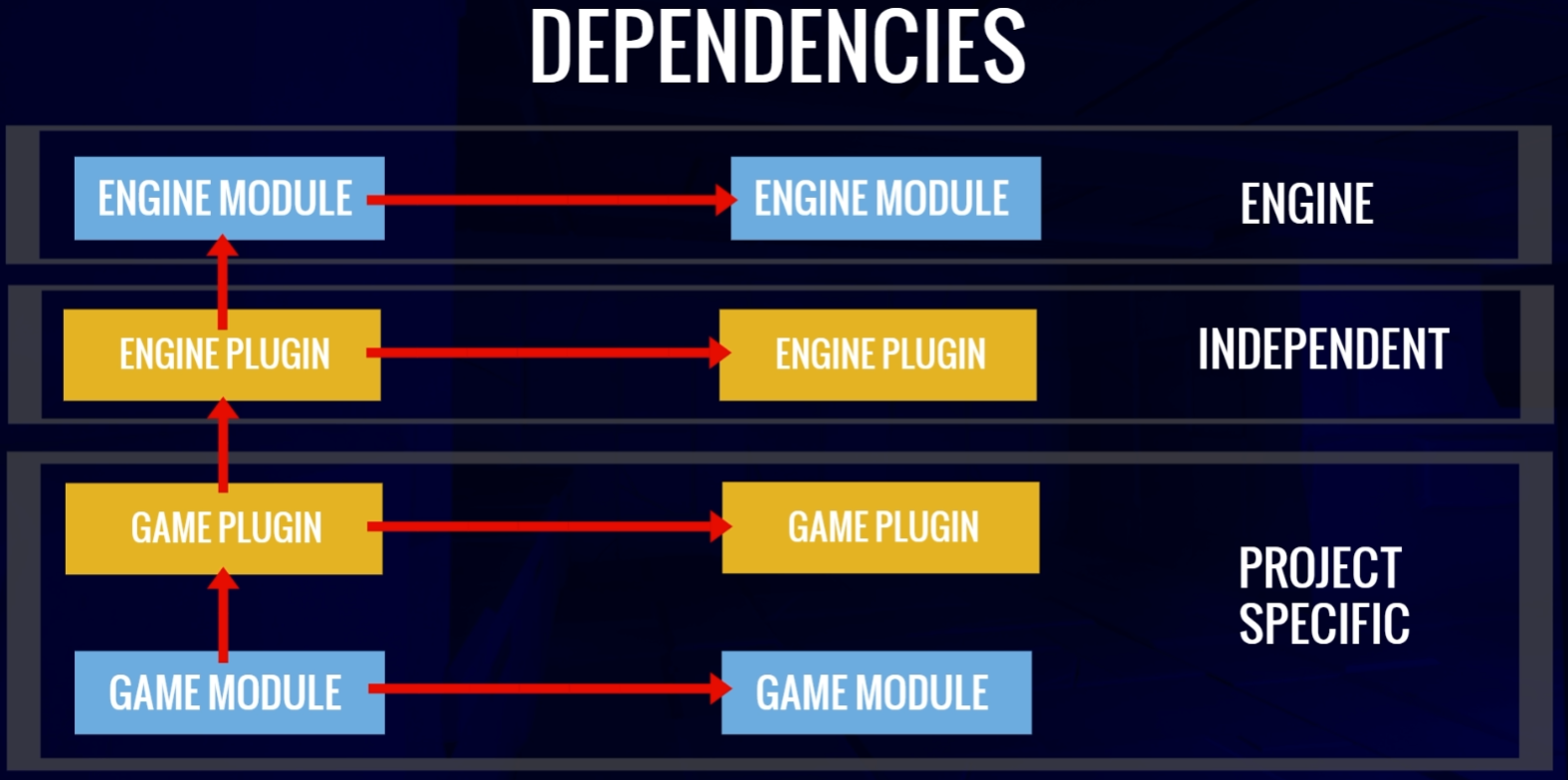
Blaster – Multiplayer Shooter

Unreal Engine 5 Multiplayer Tutorial by Stephen Ulibarri – documentation

* C++ objects
  + Multiplayer Plugin
    - UGameInstanceSubsystem
      * UMultiplayerSessionsSubsystem
    - UUserWidget
      * UMenu
  + Blaster
    - ACharacter
      * ABlasterCharacter
    - UDataAsset
      * UInputConfigDataAsset
    - UAnimInstance
      * UBlasterAnimInstance
* UE objects
  + Multiplayer Plugin
    - UMenu
      * WBP\_Menu
* Course Topics
  + Online Subsystem
    - Provides a way to access functionality of online platform (steam, xbox live, etc.) services
    - Each of platform has its own set of services upport for things like friends, achievements, setting up matchmaking sessions, etc. Online Subsystem contains a set of interfaces designed to handle these diefferent services for each platform
    - We can use Online Subsystem to handle our use of these interfaces no matter which service we choose, all we must do is configure our project for a particular platform
      * We specify default platform service in Engine.ini file
    - The Online Subsystem is a class of type IOnlineSubsystem, which can be accessed through the static IOnlineSubsystem\* IOnlineSubsystem::Get() function
    - Session Interface
      * Handles creating, managing, destroying, searching game sessionsand other matchmaking functionality
      * Session – An instance of the game running on the server.
        + Advertised (players can join)
        + Private (invite only)
      * Lifetime of a session
        + Create session
        + Wait for players to join
        + Register each one as they come in
        + Start session
        + Play
        + End session
        + Unregister players
        + Update/Destroy session
    - Game plan
  + Plugins
    - Collection of code and data
    - Easy to enable/disable per project
    - Runtime gameplay functionality
    - Editor functionality
    - Made of modules
      * A distinct unit of C++ code with its own build file
      * Encapsulate their funcionality
      * Just code, no other assets like meshes or textures
    - Dependencies
      * Our project itself is a module
      * Whenever we enable a plugin, it’s added to our project’s .uproject file
      * Plugin itself can also depend on other plugins, these are added to .uplugin file
      * Plugins and modules are broken into hierarchical levels and can only depend on other plugins and modules at their same level or higher
      * Our game project module can depend on an engine module, but and engine module cannot depend on our game (the engine modulest must be able to be built without any project)
  + Game Instance class
    - Spawned at game creation
    - Not destroyed until the game is shut down
    - Persist between level
  + [Subsystems in Unreal Engine](https://dev.epicgames.com/documentation/en-us/unreal-engine/programming-subsystems-in-unreal-engine?application_version=5.3)
    - Automatically instanced classes with managed lifetimes
    - Provide easy to use extension points, where the programmers can get Blueprint and Python exposure right away while avoiding the complexity of modifying or overriding engine classes
    - Supported subsystems
      * Engine
      * Editor
      * GameInstance
      * LocalPlayer
  + Delegate handles
    - Steps of creating delegate
      * Construct delegate
      * Bind delegate
      * Add delegate to the delegate list
    - Session interface functions used to add delegate return FDelegateHandle which we can store to unbind when we don’t need it anymore
      * AddOn...Delegate\_Handle()
      * ClearOn...Delegate\_Handle()
  + Game Mode and State
    - Game Mode
      * Rules for the game
      * Moving players to levels
      * Selecting spawn locations
      * Tracking when players enters or leaves the game
        + PostLogin(APlayerController\*) - called whenever player joins the game
        + Logout(AController\*) - called when player leaves the game
    - Game State
      * Holds state information about the game not individual players
      * Contains array of player states (player specific information)
      * Accessed by Game Mode