

Unreal Networking Overview



This lecture is based off the info from udn networking article
<http://udn.epicgames.com/Three/NetworkingOverview.html>

Topics

- Types o Networks
 - Roles
 - Relevance
 - NetPriority
 - Replication
 - Simulated
 - Debugging Hints
-
-

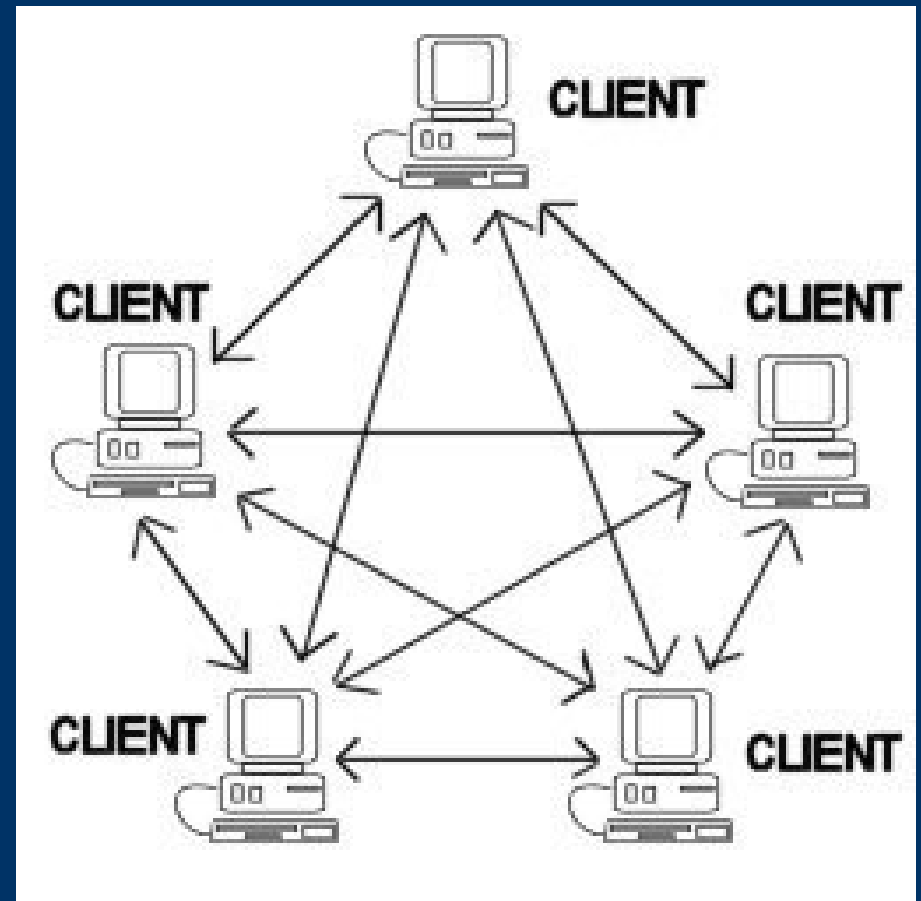
One important thing to realize is that if you plan to support networked multiplayer in your game, build it in and test it as you develop your game! Building an efficient networking implementation will have a significant impact on your game object design decisions. Retrofitting a solution is a hard and costly effort! -Tim Sweeney (Technical Director Epic)

Random Stuff

- This lecture is an INTRO to unreal Networking.
 - It is meant only as a starting point.
 - Debugging multi player games is hard.
 - Debugging multi player games is frustrating.
 - The game state of a level refers to the complete set of all actors that exist in that level and the current values of all of their variables at a time when a tick operation is not currently in progress.
-
-

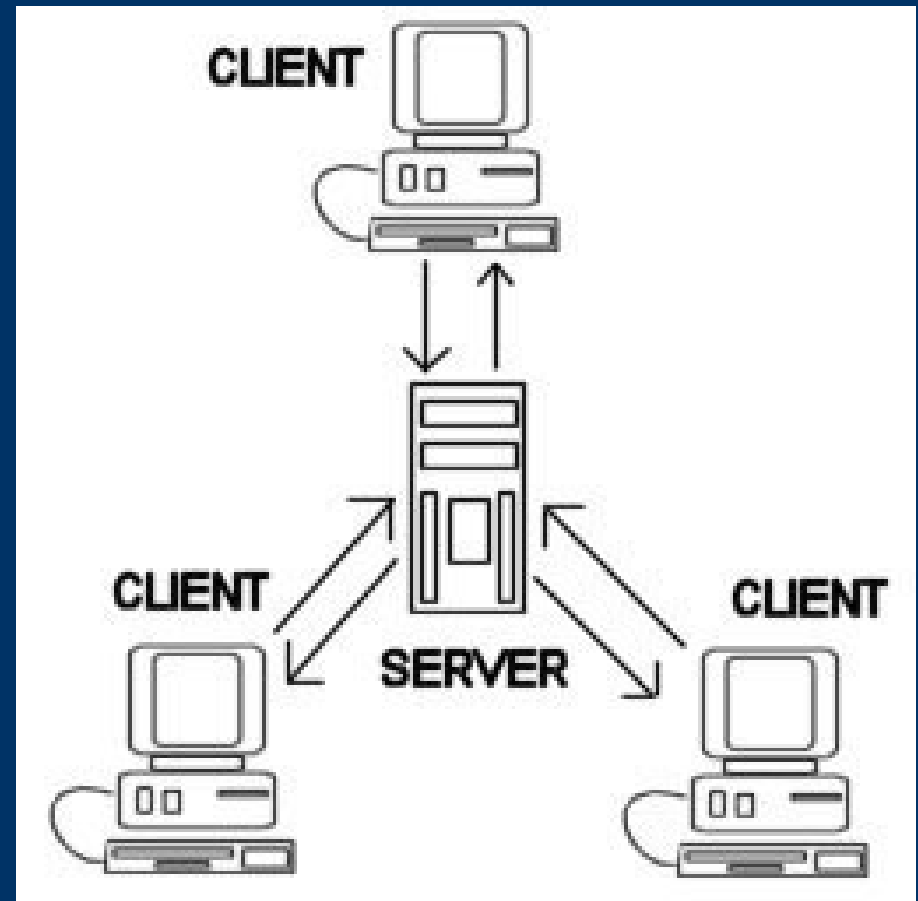
Peer to Peer (the old days)

- The Good
 - File sharing
 - Telephony
 - RTS
- The bad
 - No Computer really knows the game state
 - Cheating (faking resources)



Client Server (traditional)

- The Good
 - Less data on the wire
 - Authoritative computer with game state
- The Bad
 - Server dies, Game dies
 - Cheating (sniffing packets from server)



Generalized Client Server Model

- Differs from Client Server
 - Client Maintains accurate subset of Game State
 - Allows client to predict what is going on in the game with minimal info from server.
 - Clients talk to the server through replicated functions for which the Client is the owner of the actor.
-
-

Actor Roles on a Server

- Actors Role on the server is ROLE_Authority
 - Actors on the server may have as their RemoteRole:
 - ROLE_AutonomousProxy is used for Actors that are controlled on a client machine that owns that actor.
Examples: PlayerControllers and the Pawn they control
 - ROLE_SimulatedProxy is used for all other replicated actors.
Example: Slow Volumes, Teleporters
 - ROLE_None are Never Replicated
-
-

Client side roles

- On the client side the role and remote role are the opposite of what they are on the server.
 - Server
 - Role = Role_Authority
 - RemoteRole = ROLE_SimulatedProxy
 - Client
 - Role = ROLE_SimulatedProxy
 - RemoteRole = Role_Authority
-
-

Relevance

- Rules for determining which clients get data about which actors
 - The idea behind this is to limit the amount of data the server is sending out.
 - If a client does not need to know, it does not get the information.
-
-

Relevance Rules

- If the Actor has `bStatic=true` or `bNoDelete=true`, then it is relevant.
 - If the Actor belongs to the `ZoneInfo` class, then it is relevant.
 - If the Actor is owned by the player (`Owner==Player`), then it is relevant.
 - If the Actor is a `Weapon` and is owned by a visible actor, then it is relevant.
-
-

Relevance Rules

- If the Actor is hidden (`bHidden=true`) and it doesn't collide (`bBlockPlayers=false`) and it doesn't have an ambient sound (`AmbientSound==None`) then the actor is not relevant.
 - If the Actor is visible according to a line-of-sight check between the actor's Location and the player's Location, then it is relevant.
-
-

Relevance Rules

- If the Actor was visible less than 2 to 10 seconds ago (the exact number varies because of some performance optimizations), then it is relevant.
 - Actors whose RemoteRole!=ROLE_None are checked for relevancy for each client.
 - Virtual function AActor::IsNetRelevantFor() is called to determine whether the Actor is relevant to that client.
 - bAlwaysRelevant (relevant to all clients, always).
 - bOnlyRelevantToOwner (used for Inventory).
 - Relevancy based on visibility (using line checks).
 - If skeletally attached, relevancy of base is used.
-
-

NetPriority

- Situation will occur, where there is still too much data the server would like to send to the clients.
 - Unreal deals with this using a NetPriority variable.
 - This lets the server know how important the data is.
 - `NetPriority = 3.3`
 - The higher the value the more important the data is
 - Exists in `defaultproperties`
-
-

Replication

- Replication is the mechanism in which the server interacts with the clients.
- 3 types
 - Actor
 - Variable
 - Function
- Replication occurs when a tick is completed

Actor Replication

- Server identifies a list of relevant actors. The list is sent to the client. The client then creates a Replicated copy of the actor locally.
- This has the implication that the client may have an approximate version to the real actor.

Actor replication properties

- `bNetDirty` is true if any replicated properties have been changed by UnrealScript (or this flag has been set in C++). This is used as an optimization (no need to check UnrealScript replication conditions, or to check whether properties which are only modified in script have been changed if `bNetDirty` is false). Don't use `bNetDirty` to manage replication of frequently updated properties!
 - `bNetInitial` remains true until initial replication of all replicated Actor properties is complete.
 - `bNetOwner` is true if the top owner of the Actor is the `PlayerController` owned by the current client.
-
-

Variable Replication

- When ever the value of a var changes on the server. The server sends the new value to the client.
 - The client may have a different value of the var, the clients old value is overwritten with the server value.
 - If a variable has the keyword Repnotify in front of it, then the Replicated event is called.
 - Usefull for running simulated functions when a var changes
-
-

- ParticleModuleSpawnPerUnit.uc
- ParticleModuleSubUV.uc
- ParticleModuleSubUVBase.uc
- ParticleModuleSubUVDirect.uc
- ParticleModuleSubUVSelect.uc
- ParticleModuleTrailBase.uc
- ParticleModuleTrailSource.uc
- ParticleModuleTrailSpawn.uc
- ParticleModuleTrailTaper.uc
- ParticleModuleTypeDataBase.uc
- ParticleModuleTypeDataBeam.uc
- ParticleModuleTypeDataBeam2.uc
- ParticleModuleTypeDataMesh.uc
- ParticleModuleTypeDataMeshNF
- ParticleModuleTypeDataNxFluid.uc
- ParticleModuleTypeDataTrail.uc
- ParticleModuleTypeDataTrail2.uc
- ParticleModuleUberBase.uc
- ParticleModuleUberLTISIVCL.uc
- ParticleModuleUberLTISIVCLIL.uc
- ParticleModuleUberLTISIVCLILIR
- ParticleModuleUberRainDrops.uc
- ParticleModuleUberRainImpacts.uc
- ParticleModuleUberRainSplashA.uc
- ParticleModuleUberRainSplashB.uc
- ParticleModuleVelocity.uc
- ParticleModuleVelocityBase.uc
- ParticleModuleVelocityInheritParent
- ParticleModuleVelocityOverLifetime
- ParticleSpriteEmitter.uc
- ParticleSystem.uc
- ParticleSystemComponent.uc
- PathNode.uc
- PathRenderingComponent.uc
- Pawn.uc
- PhysicalMaterial.uc

```

Pawn.uc
// (cpptext)
// (cpptext)
// (cpptext)
// (cpptext)
// (cpptext)
// (cpptext)
// (cpptext)
replication
{
    // Variables the server should send ALL clients.
    if( bNetDirty && (Role==ROLE_Authority) )
    FlashLocation, bSimulateGravity, bIsWalking, PlayerReplicationInfo, HitDamageType,
        TakeHitLocation, DrivenVehicle, Health;

    // variables sent to owning client
    if ( bNetDirty && bNetOwner && Role==ROLE_Authority )
        InvManager, Controller, GroundSpeed, WaterSpeed, AirSpeed, AccelRate, JumpZ, AirControl;

    // sent to non owning clients
    if ( bNetDirty && (!bNetOwner || bDemoRecording) && Role==Role_Authority )
        bIsCrouched, FlashCount, FiringMode;

    // variable sent to all clients when Pawn has been torn off. (bTearOff)
    if( bTearOff && bNetDirty && (Role==ROLE_Authority) )
        TearOffMomentum;

    // variables sent to all but the owning client
    if ( (!bNetOwner || bDemoRecording) && Role==ROLE_Authority )
        RemoteViewPitch;
}

/** Check on various replicated data and act accordingly. */
simulated event ReplicatedEvent( name VarName )
{

```

Find and Replace

Quick Find Quick Replace

Find what: replication

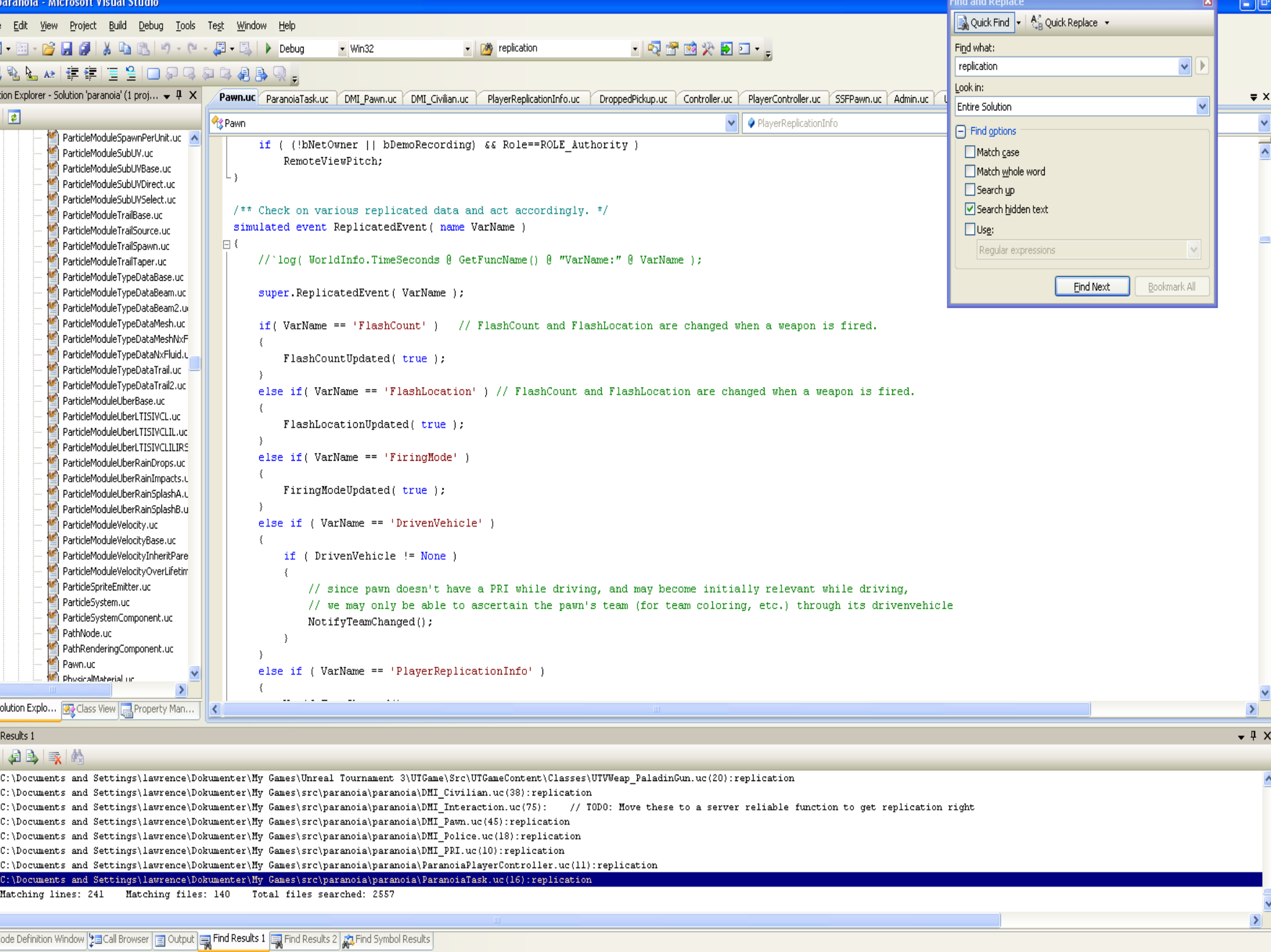
Look in: Entire Solution

Find options

- Match case
- Match whole word
- Search up
- Search hidden text
- Use: Regular expressions

Find Next Bookmark All

- ...and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\UTGameContent\Classes\UTVWeap_PaladinGun.uc(20):replication
 - ...and Settings\lawrence\Dokumenter\My Games\src\paranoia\paranoia\DMI_Civilian.uc(38):replication
 - ...and Settings\lawrence\Dokumenter\My Games\src\paranoia\paranoia\DMI_Interaction.uc(75): // TODO: Move these to a server reliable function to get replication right
 - ...and Settings\lawrence\Dokumenter\My Games\src\paranoia\paranoia\DMI_Pawn.uc(45):replication
 - ...and Settings\lawrence\Dokumenter\My Games\src\paranoia\paranoia\DMI_Police.uc(18):replication
 - ...and Settings\lawrence\Dokumenter\My Games\src\paranoia\paranoia\DMI_PRI.uc(10):replication
 - ...and Settings\lawrence\Dokumenter\My Games\src\paranoia\paranoia\ParanoiaPlayerController.uc(11):replication
 - ...and Settings\lawrence\Dokumenter\My Games\src\paranoia\paranoia\ParanoiaTask.uc(16):replication
- Matching lines: 241 Matching files: 140 Total files searched: 2557

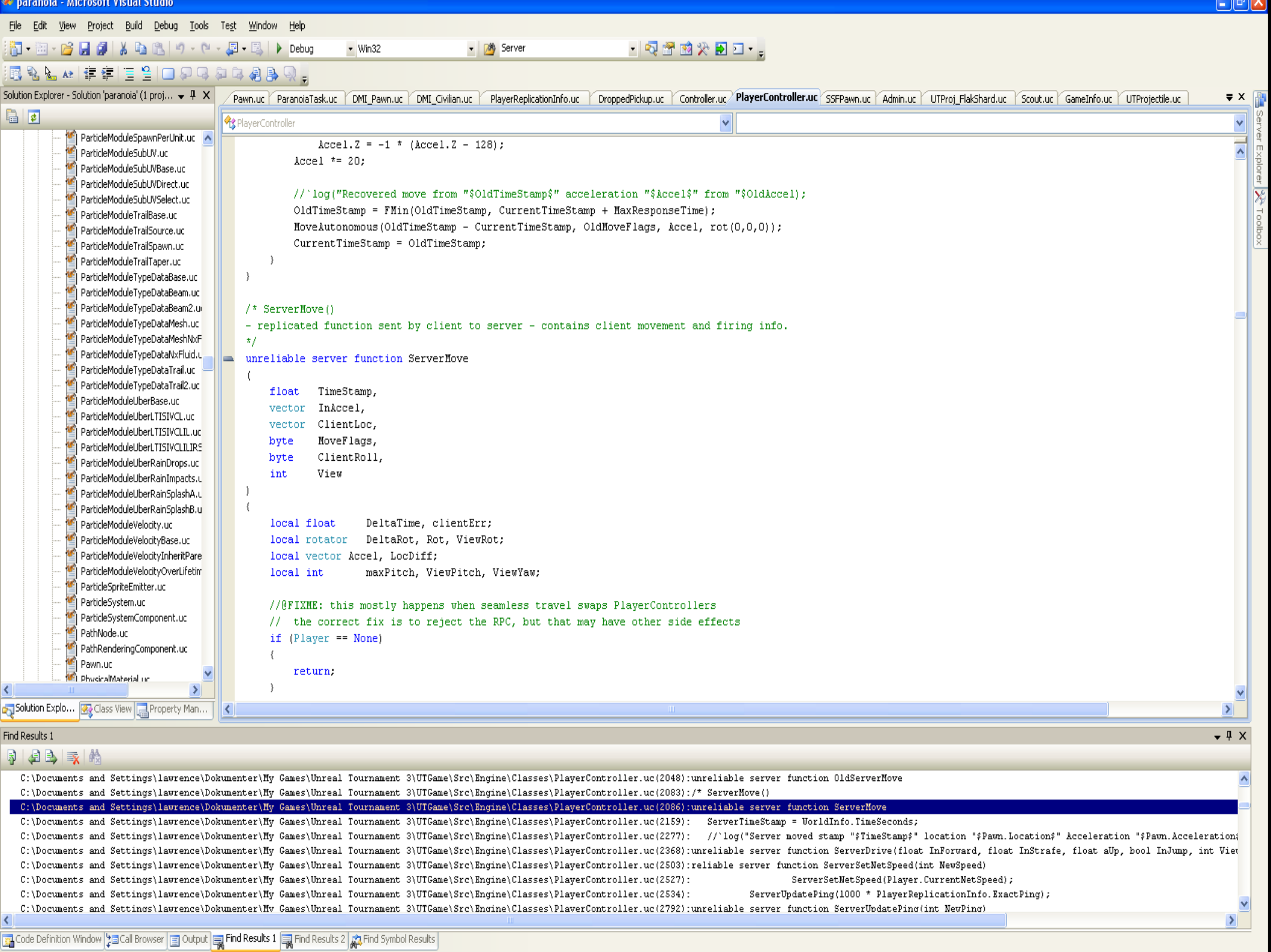


Function Replication

- The server tells the client to run a particular function.
 - The server packs up the function name, and any variables passed into the function and sends it in a packet to the client for execution.
 - If the function returns a value it will be set to 0. This is to avoid deadlock.
 - Can not be multicast
-
-

Function Replication

- If a function marked with the keyword `server` on a client then it is replicated on the server.
 - If a function is marked with the keyword `client` and is run on the server it will be replicated on the client.
 - Function replication has priority over variable replication. Therefore you can flood the net with function replication calls.
 - `unreliable`, `reliable` keywords designate the type of connection (think `udp`, and `tcp`)
-
-



```
        Accel.Z = -1 * (Accel.Z - 128);
        Accel *= 20;

        //`log("Recovered move from "$OldTimeStamp$" acceleration "$Accel$" from "$OldAccel");
        OldTimeStamp = FMin(OldTimeStamp, CurrentTimeStamp + MaxResponseTime);
        MoveAutonomous(OldTimeStamp - CurrentTimeStamp, OldMoveFlags, Accel, rot(0,0,0));
        CurrentTimeStamp = OldTimeStamp;
    }
}

/* ServerMove()
- replicated function sent by client to server - contains client movement and firing info.
*/
unreliable server function ServerMove
(
    float   TimeStamp,
    vector  InAccel,
    vector  ClientLoc,
    byte    MoveFlags,
    byte    ClientRoll,
    int     View
)
{
    local float   DeltaTime, clientErr;
    local rotator DeltaRot, Rot, ViewRot;
    local vector  Accel, LocDiff;
    local int     maxPitch, ViewPitch, ViewYaw;

    //@FIXME: this mostly happens when seamless travel swaps PlayerControllers
    // the correct fix is to reject the RPC, but that may have other side effects
    if (Player == None)
    {
        return;
    }
}
```

Find Results 1

- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2048):unreliable server function OldServerMove
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2083):/* ServerMove()
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2086):unreliable server function ServerMove
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2159): ServerTimeStamp = WorldInfo.TimeSeconds;
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2277): //`log("Server moved stamp "\$TimeStamp\$" location "\$Pawn.Location\$" Acceleration "\$Pawn.Acceleration");
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2368):unreliable server function ServerDrive(float InForward, float InStrafe, float aUp, bool InJump, int ViewPitch, int ViewYaw, int ViewRoll, int ViewPitch, int ViewYaw);
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2503):reliable server function ServerSetNetSpeed(int NewSpeed)
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2527): ServerSetNetSpeed(Player.CurrentNetSpeed);
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2534): ServerUpdatePing(1000 * PlayerReplicationInfo.ExactPing);
- C:\Documents and Settings\lawrence\Dokumenter\My Games\Unreal Tournament 3\UTGame\Src\Engine\Classes\PlayerController.uc(2792):unreliable server function ServerUpdatePing(int NewPing)

Replication Statement / Block

- Defined in the each class.
- Can only refer to variables defined in the class.
 - Can not use variables from super classes
- Side effects are allowed but A VERY BAD idea.
 - Bad: `if(counter++ > 10)`

Simulated keyword

- The use of the simulated keyword designates a function as running on the client side.
- Lightens the load on the network and the server, because they are unaware the function is being run.



Debugging (Welcome to hell)

- In general with Unreal script is hard to figure out the flow of execution. This is do to multiple threads, events, and the lack of access to the c++ code.
 - Run the game in windowed mode
 - Use this command line option
 - -useunpublished -nomoviestartup -windowed -resx=640 -resy=480 -onethread -novsync -log -nosplash
 - ``log()` macro.
 - If you use the -log option it dumps values as they occur
 - Like cout
-
-

Debug in Unreal Script

- UnrealScript Studio
 - Break points work!
 - Lets you peak at object values



Debugging MultiPlayer

- From my experience, it is best to be at least 2 people
 - Easier to get the game running
 - 1 pair of eyes on the server
 - 1 pair of eyes on the client
 - It is a very good idea to have more than 1 client running.
 - Helps tracking down bugs related to relevant actors
 - Not sure if Debugger works. My guess would be no. (because execution of game stops. Therefore communication between server and client would stop
-
-

Thats it!

Good Luck!

