

Intro to PlaneShift

PS Engine Internals

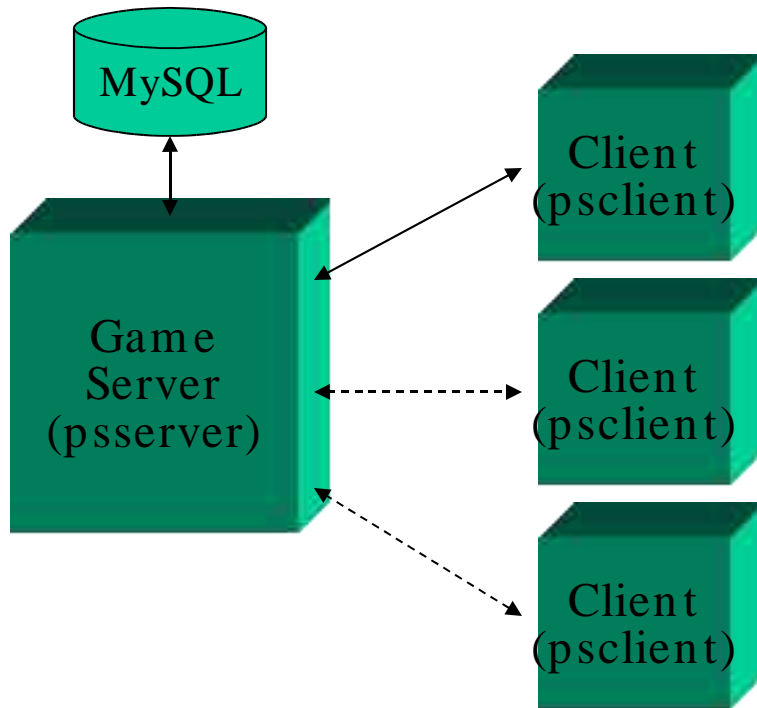
Objectives

Introduce overall structure of PS

Explain certain design decisions

Equip you to modify and add to engine consistent with existing structure

Process Structure (MB)



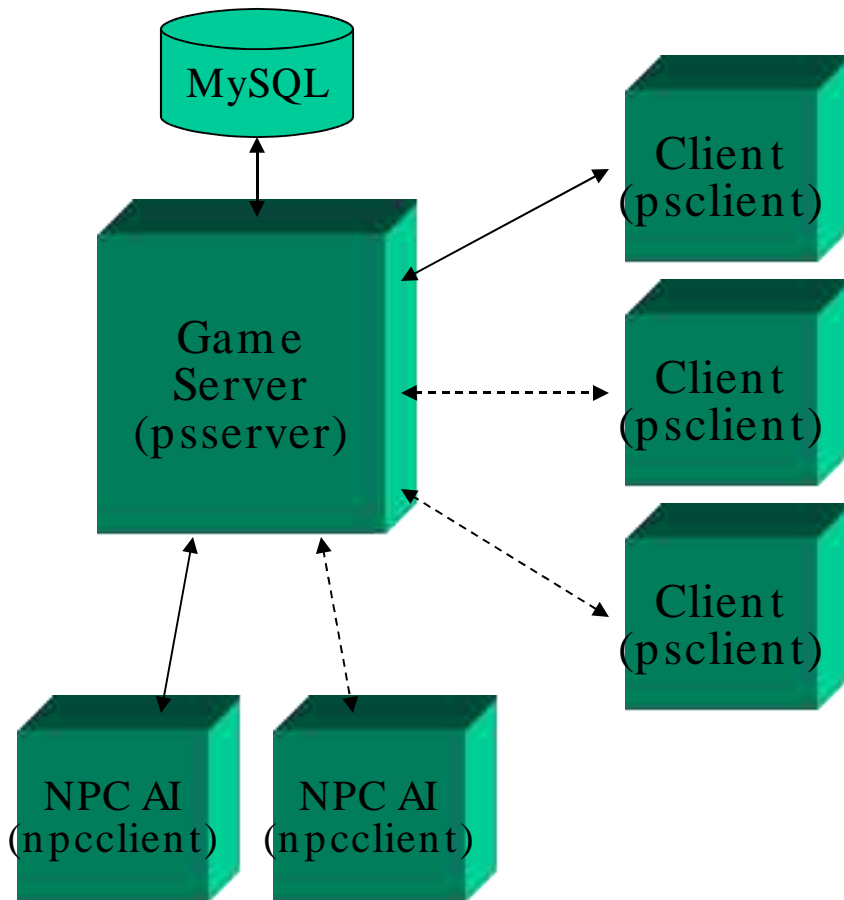
Single server handles all game activity.

All player clients connect to it the same way.

Single DB for all.

No NPC Management or AI.

Process Structure (CB)



Single server handles all game activity.

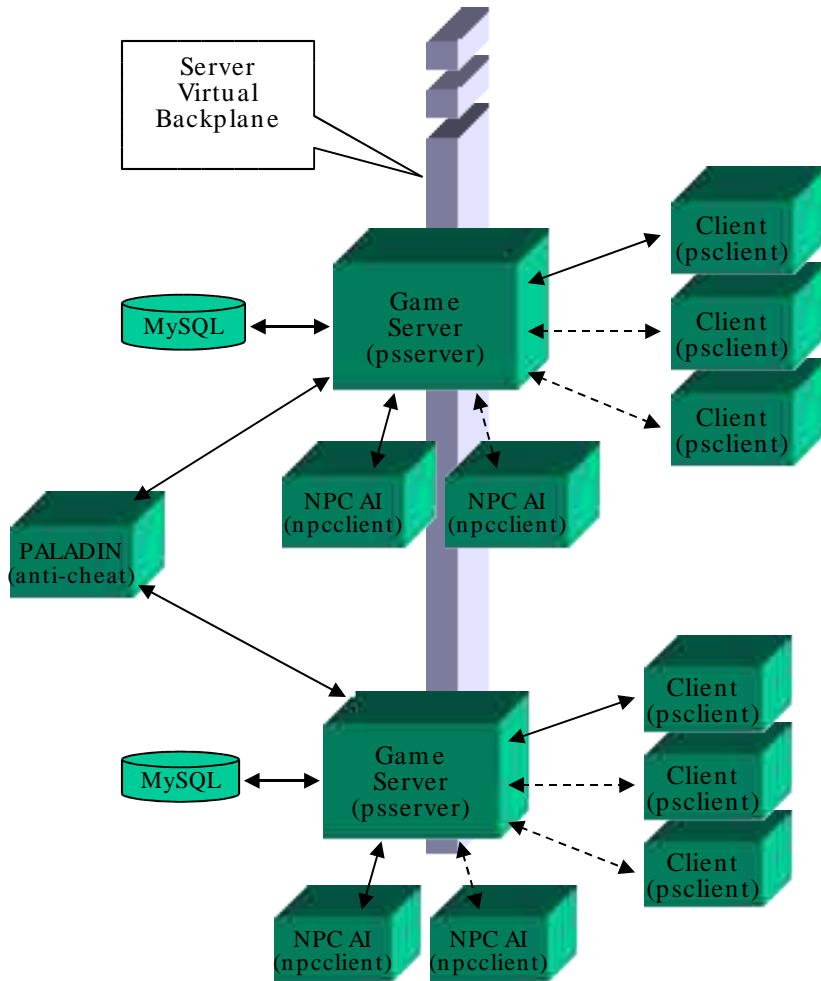
All player clients connect to it the same way.

Single DB for all.

AI managed in separate process, connected to server through network.

Multiple AI clients supported, but not required.

Process Structure (Planned)



Multiple interconnected game servers through a virtual backplane.

Local database to each server.

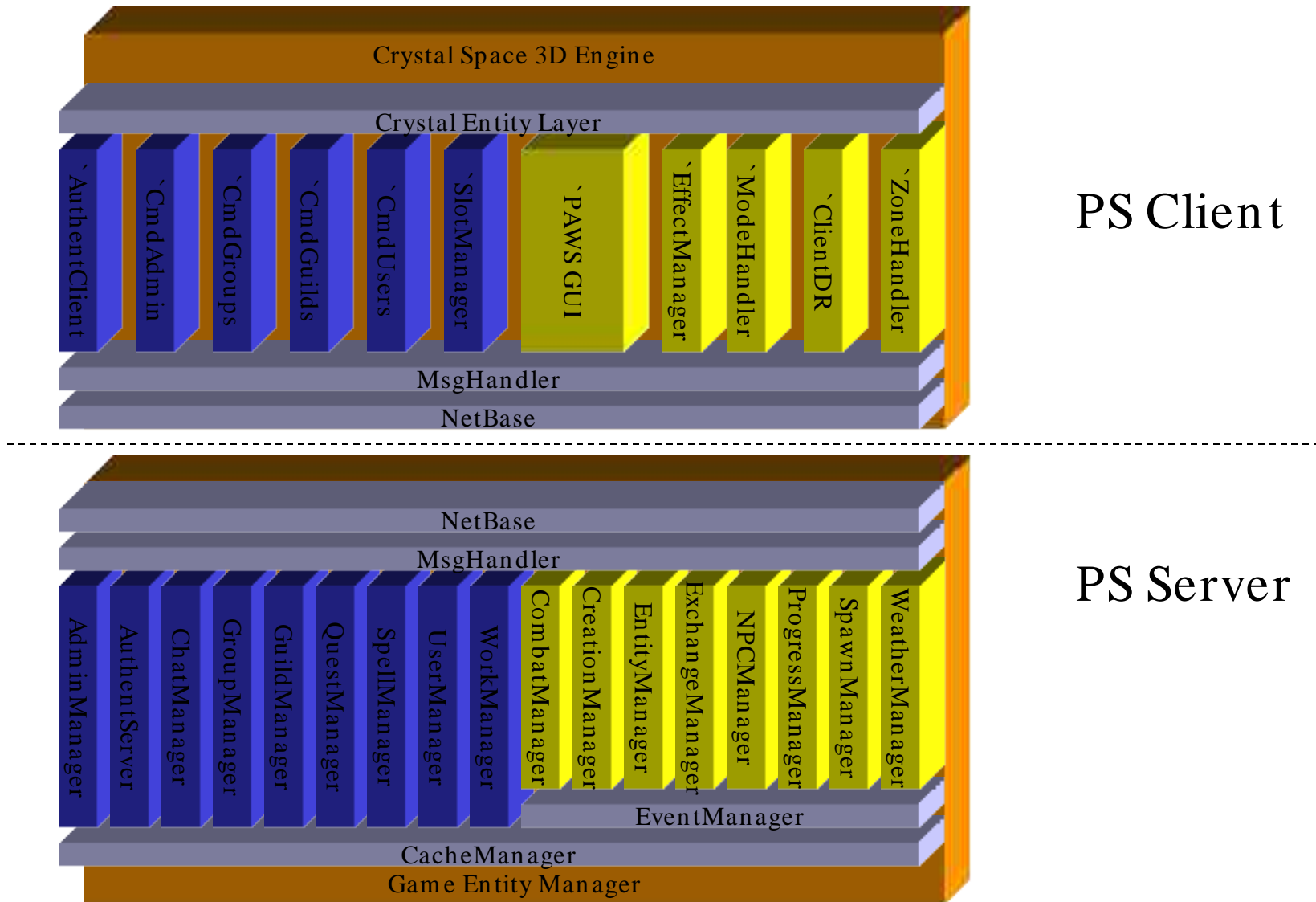
Master copy of player owned by single server during lifespan.

Game servers not divided by zone, but dynamically load-balanced.

Multiple servers can talk to single client in some cases.

PALADIN is separate process doing anti-cheat detection for CD, speed hacks, macroing, etc.

Client/ Server Overview



Client/ Server Overview (cont'd)

Shared networking code handles all communications between client and server. Game Logic objects do not have to understand the network to use it.

Publish/Subscribe Architecture used on network messages to create new managers and handlers. New vertical boxes can be added trivially without affecting others.

Client commands are added as subsystems if different category from existing. Server subsystems (managers) are added to match.

Server also has subsystems primarily driven by time-based events. These can be thought of as separate threads on the server, although they are not.

Server

3 Threads cover all logic on Server

- Network Messaging/ Sockets

- Server keyboard console

- Game Logic (99% of the work is here)

 - Handling incoming network messages

 - Commands

 - Game State Updates

 - NPC control commands

 - Timed Events

 - Weather

 - Swords in combat hitting every 4 seconds

 - Spell delay between cast and activation, buffs wearing off after 20 minutes

Primary Server Classes

MsgHandler

Gets all incoming messages from all clients and superclients. Publishes each message to any other class which subscribes to that type of message.

Client

Each instance represents a connected machine, either with a psclient or an npcclient connected. This has important data such as the gemActor and the psCharacter and the name, guild, group associated with that player.

EventManager

Maintains a queue of timed events to be executed and guarantees that they are executed in the correct order. Callers can create their own subclass of psGameEvent, pick a time in the future and throw this new object into EventManager knowing that it will fire at the right time. Objects use this to wake themselves up every so often or to timeout things, etc.

Primary Server Classes

CacheManager

The server does almost no querying of the database at runtime (except for getting info of connecting players). At startup, cachemanager preloads many tables into collections in RAM and provides functions for accessing these.

EntityManager

PS uses CEL to link game logic to CS structures. EntityManager handles most of this work.

NPCManager

PS employs 1 or more npc clients to handle AI-related functions. NPCManager handles these connections, sends world state information to all npcclients and receives all NPC commands from them—translating them into regular client commands for the rest of the server.

Game Entity Management – GEM

gemObject

All game logic (not RP logic) related to any object in the game. This is the superclass of all gem classes. Stores name, position, mesh, etc. It also can generate its own network message to propagate all necessary info for a client to create it.

gemItem

Items are defined as visible and usable objects in the game world. Swords, mugs, anvils, muffins are all Items if they are visible independent of a player.

gemActor

Every character in the game is a gem Actor. Actors can be grouped or in guilds; they have psCharacter records which hold all their RP data. They have lists of valid duels, etc.

gemNPC

Some Actors are NPCs, which are any automated character in the game such as a merchant or a monster. NPCs have slightly more capabilities than human player Actors, but not many.

Handling Client Commands

AuthentServer
AdminManager
ChatManager
CreationManager
ExchangeManager
GroupManager
GuildManager
InviteManager
SpellManager
UserManager
WorkManager

These classes all register to hear certain types of messages from the clients, all of which are textual user commands such as /say hello, /disbandguild, /attack and so on.

In many cases, these have their own local caches of data and maintain other data structures in response to these user commands.

Server Pseudo-Processes

CombatManager

The calculations involved in rolling and using character RP data to determine damage, blocking, HP, etc. is very complex and is all here. CombatManager uses EventManager to schedule the attacking swing-by-swing, one at a time, on all sides of the battle.

SpawnManager

This makes sure that dead npcs are respawned in the right place at the right time. It uses EventManager to pick a new spawn point and spawn time immediately upon death and queues it for reactivation.

WeatherManager

Lighting, day-night transitions and rain, thunder and lightning are all determined here.

ServerDR

Clients send Dead Reckoning (DR) messages frequently to the server, which reflects those messages to all clients within a certain radius. The server also updates the positions at the same time so it can track where everyone is.

Client

2 Threads cover all logic on the client*

Network Messaging/ Sockets

Game Loop and Frame Rendering

Incoming network messages from server are handled between frames.

Client has zero knowledge of anything other than what is necessary to render accurately and inform the player.

PAWS is the very extensive GUI library for the client. The main GUI is in `/src/common/paws` (libpaws) and the PS-specific parts are in `/src/client/gui`.

*CS does use additional threads for things like Sound, but they don't count in this discussion and are hidden from the PS developer.

NPC Client

Security