Creating a Better Lobby

Why we need a better lobby

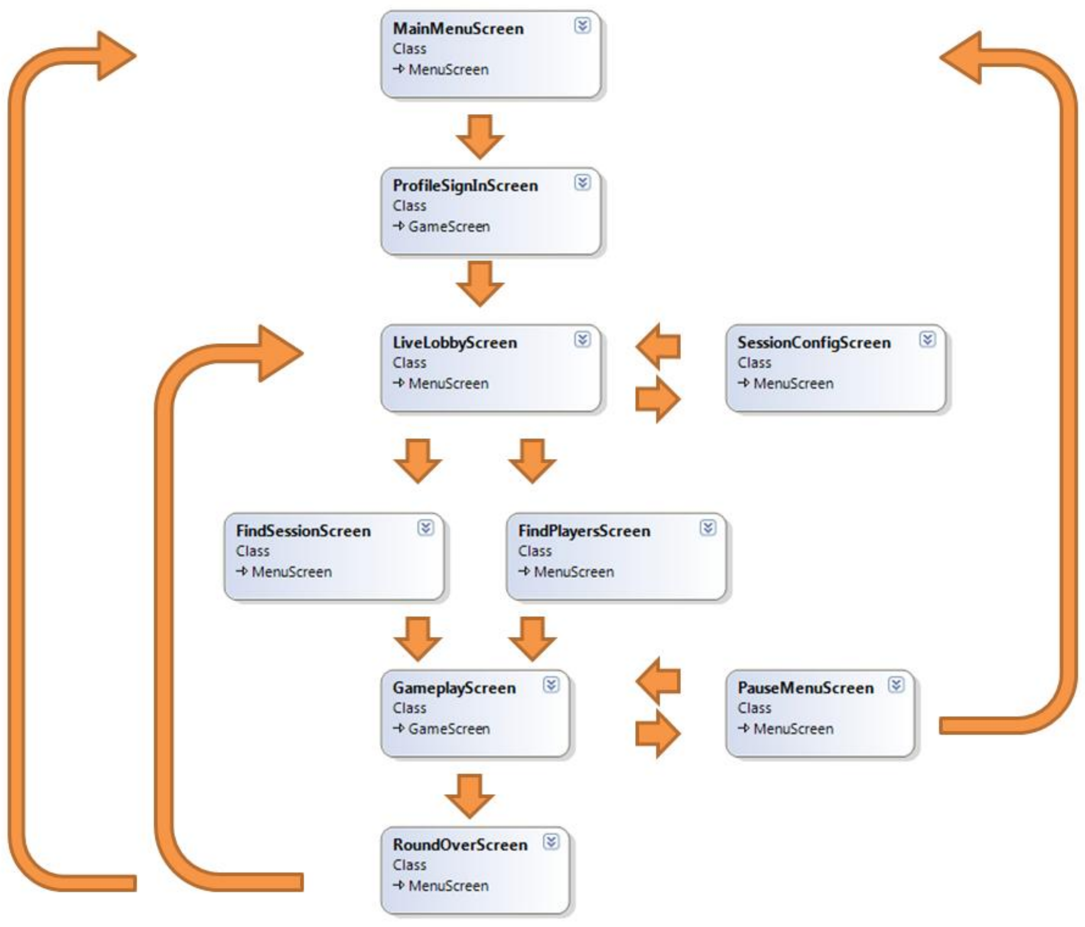
To understand why we need something better, we first need to understand current system. Most people starting XNA Networking jump to samples. Samples are great; they show us how to connect the pieces of the API to get a stable working result. Samples are also bad; they set peoples way of thinking about a problem and most people use the sample in there finished game with little or no rework.

The XNA team released a *Best Practices for Community Games* which talks about having a quick match, custom game and pre lobby for game mode settings.

I have built my example on top of the *NetworkStateManagementSample*. This is the sample that show both how to create a menu system around moving the player into a lobby, then game play, then back. As well this sample show how to create sessions, find sessions, accept invites, etc.

This current flow is a very manual process and requires the Player to create, find and pick sessions. That's 3 steps to many. Why do players care about sessions? They want to play a game instead of managing a sessions. This is why we need to rethink the current sample.

Note: I was only thinking about one of 3 networking modes Local, SystemLink and PlayerMatch. Local lobby is about setting game modes and player options, not about networking. SystemLink works very well under the current mode; this is because there is normally a very small number of SystemLink sessions and the player normally knows and cares which session they what to join.



Arriving at a solution

So we want to make the process of creating, finding and joining sessions happen auto-magically. This is no simple task. The first task is to figure out what stags are involved in this process. It starts with configuration; which involves the player setting the game mode and round options. Players then collect local players and/or are inviting friends. Next the auto-magic happens. After all players are ready, gamers are grouped and the game starts, when the party size is reached.

Simple words lead to complex problems. The major issue that makes this problem so complex is that XNA networking does not let you find other sessions when you have one created. But how do we configure a session and invite friends before we create or find a session. Well the simple answer is we can't. First we need to create a session in the lobby to configure it and invite players before the auto-magic can happen. People could argue that you don't need to create a session, but they forget that most people like to invite friends and this is best place to do it.

To start the auto-magic process we need a few pieces of information: local gamers, friends, and session properties. This is a purpose of the LiveLobbyScreen. It lets local players join quickly, lets the gamer invite friends, and lets the host configure the session properties which are needed to find the correct sessions. The SessionConfigScreen helps the host set the session properties.

Another thing most people tend to forget is that some players need to host and some players need to search. So which players go looking for sessions and which players sit tight waiting to be found. Well based on 3 pieces of information we collected in the lobby local gamers, friends and session properties; we know that there are 3 types of sessions single, simple, and complex. Single involves a solo Live account. Simple involves a Live account with guests. Complex involves a Live account with guests and friends. So which of these groups should search and which should host? Well since we can not recreate a complex session we should really try and keep it alive. As well we can easily recreate sessions with just local gamers, but to improve the split I keep simple sessions live as well. That leaves single session which then gets destroyed in order to search for a new session. Single session gets transferred to the FindSessionScreen and Simple and Complex session get transferred to FindPlayerScreen. As well have to remember to set a session property that indicates if the session is ready to be found. Since we don't want to find session that are still in the lobby and not searching.

LiveLobbyScreen

LoadContent

```
if (this.Session == null)
{
    // Which local profiles should we include in this session?
    var localGamers =
    NetworkSessionComponent.ChooseGamers(NetworkSessionType.PlayerMatch,
    ControllingPlayer.Value);

    // Create session
    var options = new SessionOptions();
    this.Session = NetworkSession.Create(NetworkSessionType.PlayerMatch,
    localGamers, NetworkSessionComponent.MaxGamers, 0, options);
    this.Session.AllowHostMigration = true;

    // Create a component that will manage the session we just joined.
    NetworkSessionComponent.Create(ScreenManager, this.Session);
}
else
{
    if (this.Session.IsHost)
        this.Session.Options().IsReady = false; // Session not Ready!
}
```

Update

```
if (this.Session.IsEveryoneReady)
{
    foreach (var gamer in this.Session.LocalGamers)
        gamer.IsReady = false;

    ExitScreen();

    if (this.Session.AllGamers.Count == 1)
    {
        // Single player only, look for new session
        ScreenManager.AddScreen(new
        FindSessionScreen(this.Session.Options()), this.ControllingPlayer);
    }
    else if (this.Session.AllGamers.Count != this.Session.LocalGamers.Count)
    {
        // Local players and friends, cant recreate session, look for players
        ScreenManager.AddScreen(new FindPlayersScreen(this.Session),
        this.ControllingPlayer);
    }
    else if (this.Session.AllGamers.Count == this.Session.LocalGamers.Count)
    {
        // Local players only, look for more players
        ScreenManager.AddScreen(new FindPlayersScreen(this.Session),
        this.ControllingPlayer);
    }
    else
    {
        throw new NotSupportedException("Session Configuration");
    }
}
```

Note: Why would you create a session in the LiveLobbyScreen only to destroy it in FindSessionScreen? Well someone needs to search and we can't have an active session when we call Find.

The FindSessionScreen is where all the auto-magic happens. Three things need to happen in this screen. First we find all sessions that meet the SessionProperties we set out in the lobby. Second we filter the available sessions to weed out sessions we don't want to join. Third we join a session which puts the screen in a passive waiting mode. It will wait until all the slots in the session are full, then start the game.

FindSessionScreen

```
if (!this.JoinedGame)
{
    if (FindResult == null && JoinResult == null)
    {
        FindResult = NetworkSession.BeginFind(
        this.Type, this.LocalGamers, this.Options, null, null);
    }

    if (FindResult != null && FindResult.IsCompleted)
    {
        var availableSessions = NetworkSession.EndFind(FindResult);
        if (availableSessions.Count == 0)
        {
            availableSessions.Dispose();
        }
        else // If we did, Join the first one that meets the Requirements.
        {
            this.Message = "Joining Game";

            var sessions = availableSessions.Where(
            s => (s.CurrentGamerCount + this.LocalGamers.Count()) <=
            NetworkSessionComponent.MaxGamers);
            var session = sessions.First();

            JoinResult = NetworkSession.BeginJoin(session, null, null);
        }
```

```
            FindResult = null;
        }

        if (JoinResult != null && JoinResult.IsCompleted)
        {
            this.Message = "Game Joined";

            try
            {
                this.Session = NetworkSession.EndJoin(JoinResult);
                NetworkSessionComponent.Create(this.ScreenManager, this.Session);

                this.JoinedGame = true;
                JoinResult = null;
            }
            catch (NetworkSessionJoinException) { } // Try Again
        }
    }
    else
    {
        if (this.Session.SessionState == NetworkSessionState.Playing)
        {
            // Check if we should leave the lobby and begin gameplay.
            // We pass null as the controlling player, because the networked
            // gameplay screen accepts input from any local players who
            // are in the session, not just a single controlling player.
            LoadingScreen.Load(ScreenManager, true, null, new
            GameplayScreen(this.Session));
        }

        if (this.Session.IsHost &&
            this.Session.SessionState == NetworkSessionState.Lobby &&
            this.Session.AllGamers.Count == NetworkSessionComponent.MaxGamers)
        {
            // Start game if Lobby is full
            this.Session.StartGame();
        }
    }
```

The FindPlayersScreen is a lot more passive and it will wait until all the slots in the session are full then start the game.

Both FindPlayersScreen and FindSessionScreen transfer to the Gameplay Screen. When the round is over I transfer the session to the PlayerMatchingScreen. This screen gives gamers the chance to group together for the next round. If they do not they are transferred back to the lobby to create a new session. If they do want to group up, then the complex session is transferred back to the lobby.

Where to go from here

XNA team recommends creating a quick matching option. For this I created QuickMatchScreen, which is basically the LiveLobbyScreen, but does not create a session. This translates to a single session type which is passed directly on to the FindSessionScreen with default session properties.

Party Up is a common feature in modern games that lets gamers group up and play together. To handle this I create a PartyupScreen that lets gamers return to lobby the main menu, or if the session is not empty it is passed back to the FindPlayersScreen.