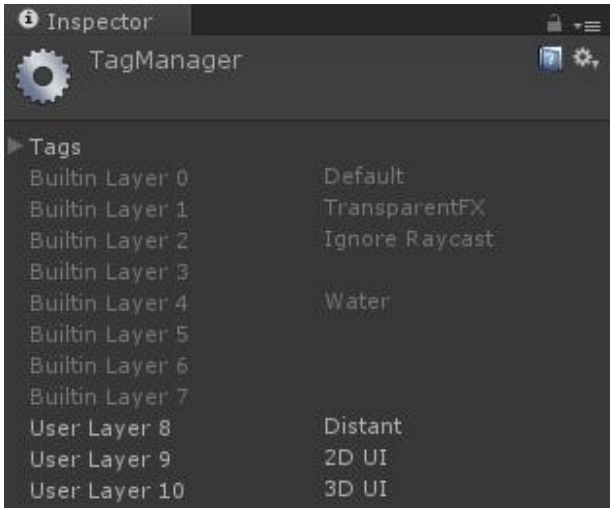
Installation

1. Upon importing this kit into a new project, you will notice a lot of errors. Don't panic! This is because you need to import NGUI. **When importing NGUI, don't import any of the examples.** (You don't need them)
2. Importing full version of NGUI will resolve all the errors, but before moving on, create the "Distant" and "2D UI" layers used by the kit.
    a. Open up the "Edit" menu in Unity
    b. Choose "Project Settings"
    c. Choose "Tags"
    d. Add "Distant" as User Layer 8
    e. Add "2D UI" as User Layer 9
    f. Add "3D UI" as User Layer 10



3. The package is now fully usable, but if you have **TNet**, you are in luck! You get the multiplayer integration for free, complete with all the relevant menus.
    a. Import **TNet**. Don't import any of TNet's examples. (You don't need them)
    b. Double-click on the **TNet Integration** unity package and import it into the project, overwriting some files.
    c. You can quickly verify that it works by navigating to the "Direct Connect" menu. You should see your internal and external IP addresses.
    d. You will want to check "Run in Background" option. **Multiplayer won't function properly without it!**
        i. File menu
        ii. Build Settings
        iii. Player Settings
        iv. Resolution and Presentation
        v. Check "Run in background"

Menu Scene

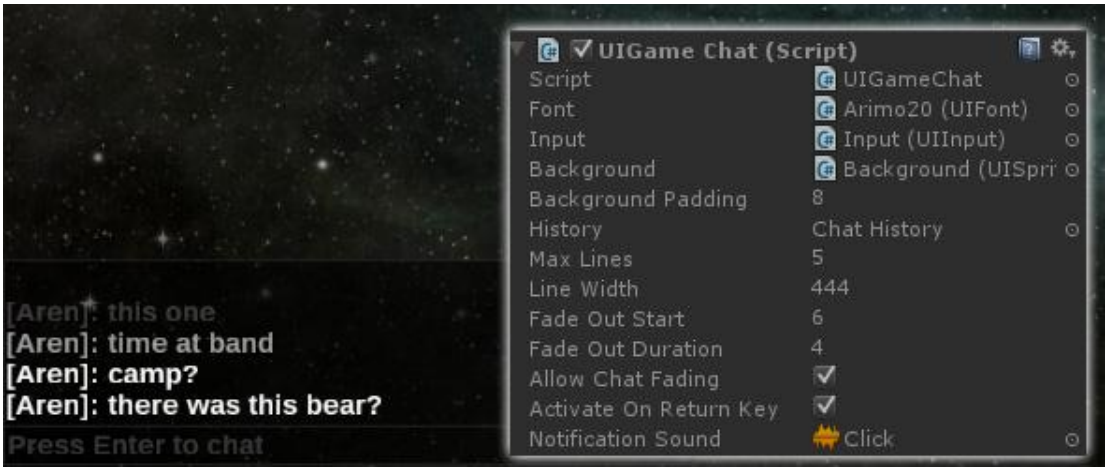
Menu Scene contains the most of the functionality in the kit. Some of the custom scripts include:

- **UIWindowButton** is used to navigate from one window to the next and back. To use it, simply attach it to any button and make it point to a panel that it should show, or choose a different action, such as “hide” or “go back”. You also have an option to restrict it to require a full version of the game (if you wanted some areas to be off-limits to the LITE version of your game).
- **UINews** script is used to load the news. To use it, attach it to any label and specify a URL to download the text from.
- **UIShowIfNotFullVersion** hides the “LITE” box on the logo. This script automatically disables the children below the game object it’s attached to if the **Profile.fullVersion** evaluates to ‘true’. Useful if you want to have one version of the game with the ability to upgrade it without having to download a new version.
- **UIRankings** can be found on the grid for the Stats window to populate the leaderboard by downloading data from a remote server, then instantiating prefabs, thus populating the list. Columns allow sorting by clicking on them.
- **UIWifiRequired** is used on the Profile button to disable it if not connected to Wifi.
- **UIOptionWifi** on a checkbox in Options allows using cell phone data for communication when checked. Players tend to appreciate when you give them an explicit choice in this matter and default it to “off”. This way if they do make a mistake, it’s theirs, and not yours.
- **UIOptionCaps** in Options toggles between English and Caps languages, showing that the entire UI is fully localized (with the exception of the news).
- **UIOptionUpgrade** in Options toggles between the Lite and Full versions of the game.
- **UIOptionPower** allows toggling between the power-saving mode (caps at 30 FPS if VSync is disabled), and the performance mode (60 FPS). This is a very useful option to have for mobile devices as it often doubles the battery life while playing your game.
- **UIFullScreen** is used on the UIRoot to listen to a button that can toggle between full screen and windowed mode. It also happens to remember the last state, so the next time you launch the game it will restore it for the player’s convenience. Works best with a resizable window.
- **UILimitedSavedOption** script can be found on many of the drop-down lists in the New Game window, restricting some of the choices only to a full version of the game. For example, Lite version of the game can only choose the Small and Medium map sizes.
- **UIPlayButton** can be found on each of the buttons inside the New Game window, with options to set the current game type, and options to disable the buttons in the web player (for example, no hosting from the web player), as well as require wifi (for multiplayer).
- **UINameField** found on the Input Area saves and loads the **PlayerProfile.playerName** that can be used throughout the game.

Game Scene

Menu Scene contains more game-centric functionality, such as chat, progression widget, and the ability to pause the game.

- **UIPlayerExperience** can be found on the Progress object, and is what controls the widget showing the player's progress towards next level. It also makes it easy to animate experience gain as it simply watches the **PlayerProfile.experience** and does everything under the hood, including level-up notifications. You can specify any custom animation of your choice – even one on a completely different game object if you wanted to get creative and fancy with an animated window that would fly in and sparkles everywhere. Why not!
- **UIPlayerList** is used on the Player List object to instantiate the list of players found on the left hand side of the screen. The list updates as the players join and leave the game. It's designed to automatically grow both up and down so that it stays centered.
- **UIPauseButton** on the Pause Button game object allows the player to pause the game at any time (and you can see the timer pause when you do so). A checkbox controls whether the button will animate when paused by pulsing softly, and a Button Key Binding allows it to be activated by hitting the Space Bar.
- **UIGameChat** script on the Chat Window drives the chat window that's a bit fancier than the one that comes with NGUI. Messages pop in by scaling up and pushing older messages upwards, and fade out automatically, minimizing the amount of space taken on the screen. When the player starts typing, the history is automatically brought back for the player's convenience.



TNet Integration (1/2)

If you happen to have the **TNet Networking Library**, you get even more content with the kit. Import TNet into the project (no need to import the examples folder), then import the provided **TNet Integration** package.

NOTE: Don't forget to check "Run In Background" in the PlayerSettings as explained on the first page!

- **UICreateServer** script found on Window – LAN reacts to a remote button click, starting or stopping a LAN server with the same button (and it updates the button's text accordingly).
- **UIDirectConnect** script found on the **Window – Direct** uses TNet's **Tools.ResolveIPs** function in order to automatically determine the local and external IP addresses, displaying them to the player. Note that the external IP address discovery can be more reliable if you use your own URL instead of relying on the provided one. This is the **ip.php** script used on the Tasharen server:

```
<?php
echo 'Current IP Address: ' . $_SERVER['REMOTE_ADDR'];
?>
```

UIDirectConnect also makes it possible to connect to a remote server by using its IP, as well as to start a local server easily (and will connect to it automatically).

TNet Integration (2/2)

- **LobbyManagers** script on the **_Network Manager** object makes it possible to have both TCP (internet) and UDP (local network) discovery clients in one game. Since both derive from the same class and update the same list of players, LobbyManagers script ensures that only one is active at a time. **UIEnableLobby** script found on the menu's buttons calls into LobbyManager's functions, making it possible to easily switch from one to the other.
- **UIServerList** found on LAN and Internet grid objects takes care of instantiating UI prefabs for all known servers by checking the **TNLobbyClient.knownServers**. For this script to work there must be a lobby client active, so it's a good idea to attach the **UIEnableLobby** script to the same button that opens the server list.
- **UIChannelList** found on the Scrollable Panel's Grid object is responsible for populating the list of channels. It queries the TNet server for a list of channels, and then parses the response by instantiating channel prefabs that have a **UIChannelListItem** script on them.
- **UIChannelListItem** script continues where the UIChannelList script leaves off, parsing the channel data, filling out the information in labels, and allowing the player to join the channel when clicked on.
- **UIDisconnectOnClick**, predictably, disconnects from the server when clicked on. Simple and effective!
- **UIGameChat** is a bit different with TNet as it was modified to allow chatting with other people in the channel. On the surface it looks exactly the same though!
- **UIPlayerList** was changed as well so that it's able to properly create player name entries for all the connected players in the same channel.
- **UIMustBeConnected** script, found on the Play, Join, and Create buttons within multiplayer windows takes care of disabling the button if the connection to the server is lost for any reason.

Questions, Comments and Support

Although the package was designed to be as intuitive as possible, I know what's intuitive for some may not be for others, so I am always available to answer questions. The best way to do so is to go to the forum (http://www.tasharen.com/forum/index.php?board=6.0) and ask there. Due to the very high volume of correspondence I am no longer able to answer direct support-related emails as they can't help others – but the forum posts certainly can.

So TLDR version: have a question? Just ask on the forum. ☺