**Browsing available servers** **To browse the available servers, call MasterServer.RequestHostList. This takes one single parameter: the game type name (this is the same game type name you passed to RegisterHost).** **This does not return anything, instead the result will be asynchronously down- loaded, and the last known list of servers can be accessed via MasterServer.PollHostList. Additionally, to ensure you aren't using old data, you can call MasterServer.ClearHostList. For example, if the user hits the Refresh but- ton in the lobby you might clear the host list and then request a new list from the Master Server.** **The following script shows a lobby for users to browse available servers and con- nect to them:**

**using UnityEngine;**

**using System.Collections;**

**public class ExampleUnityNetworkingBrowseServers : MonoBehavior {**

**// are we currently trying to download a host list?**

**private bool loading = false;**

**// the current position within the scrollview**

**private Vector2 scrollPos = Vector2.zero;**

**void Start()   {**

**// immediately request a list of hosts**

**refreshHostList();   }**

**void OnGUI()**

**{     if( GUILayout.Button( "Refresh" ) )**

**{       refreshHostList();     }**

**if( loading )     {       GUILayout.Label( "Loading..." );     }     else     {       scrollPos = GUILayout.BeginScrollView( scrollPos, GUILay- out.Width( 200f ), GUILayout.Height( 200f ) );**

**HostData[] hosts = MasterServer.PollHostList();**

**for( int i = 0; i < hosts.Length; i++ )       {**

**if( GUILayout.Button( hosts[i].gameName, GUILayout.- ExpandWidth( true ) ) )         {           Network.Connect( hosts[i] );         }       }       if( hosts.Length == 0 )       {         GUILayout.Label( "No servers running" );       }         GUILayout.EndScrollView();**

**}   }     void refreshHostList()   {     // let the user know we are awaiting results from the master server     loading = true;     MasterServer.ClearHostList();     MasterServer.RequestHostList( "GameTypeNameHere" );   }     // this is called when the Master Server reports an event to the client – for example, server registered successfully, host list received, etc   void OnMasterServerEvent( MasterServerEvent msevent )   {     if( msevent == MasterServerEvent.HostListReceived )**

**{       // received the host list, no longer awaiting results       loading = false;     }   } }**

**}**

**}**

**The preceding code will list available servers registered to the Master Server. Clicking one of the buttons will call the Network.Connect function and connect to the corresponding server, and clicking on Refresh will display a Loading... message while results are fetched from the Master Server. There are a number of improvements and other tweaks that can be made to this code, left as an exercise for the reader:**

**Refresh the host list every few seconds. This should be done transparently, without displaying a Loading message.**

**Allow the user to add servers to a "favorites" list (possibly saved as CSV to**

**PlayerPrefs), if your game allows players to run dedicated servers.**

**•If the user attempts to connect to a password-protected game**

**(HostData.passwordProtected is true), display a password entry field.**

**•Save game information such as map, mode, and so on in the Comments field**

**when registering a server, and allow the user to filter server results.**