

**Client Classes:**

**Game Controller Class:** The game controller class runs the game and tells every other class what to do. Most of the important data will be in the game controller to include current game rules, the statements and cards in play,

**Network Manager Class:** The network manager establishes a socket connect to the server. When then game controller needs to send information it uses the network manager. Information coming from the server will be received by the network manager, and relayed to the game controller.

**Profile Manager Class:** The profile manager loads and stores profiles locally. Each profile will be its own class which will be stored in a list in the profile manager.

**Screen Manager Class:** The screen manager draws everything to the screen and takes input from the user. The screen states will be controlled with an enum in the game controller.

**Asset Manager Class:** The asset manager is primarily responsible for loading files at run time and providing data to other classes. All of the textures and sprites used by the screen manager class will come from the asset manager. Additional sound files for music and sound effects will also be stored in the asset manager. The loading screen is used at the start of the game running and waits until the asset manager has finished loading files.

**Profile Class:** The profile class will need to consist of all of the variables that define the current user. This means that the profile will need a name and some form of icon to distinguish this profile from others. Additionally profiles should contain a list of friends for use in connecting users directly to game lobbies with friends already in them.



**Client Screens:**

**Loading Screen:**

The loading screen serves the purpose of giving the player something to look at while files are loaded. Since the game is really simple the need for the loading screen will probably only be at the initial launch of the client program.

**Profile Selection Screen:**

The profile selection screen is the first screen that users are able to interact with. Users must first select or create a profile to start using the client program. If there are no profiles, then the screen will only have the option to go to the profile setup screen.

**Profile Setup Screen:**

The profile setup screen is linked to the profile selection screen, and is used to either create a new profile, or edit an existing profile.

**Main Menu Screen:**

The main menu screen serves as kind of a logical central hub to the program; from it users can connect to lobbies, go to the options screen, or go back to the profile selection screen. If the user quits out of a game they will be returned to the main menu screen.

**Options Screen:**

The options screen is for the user to change aspects of how the program works. Music and sound effect volume can be changed or completely muted. If the game gets animated sprites, then the option screen will allow that feature to be turned off.

**Direct Connect Screen:**

The direct connect screen needs to allow the user to connect to a particular. Ideally the user can directly connect to a friend’s game lobby, but the minimum would be the ability to enter a lobby ID to join the related game lobby.

**Game Finder Screen:**

The game finder screen will allow the user to choose several options and automatically connect them to a lobby that matched the selected options. If a game cannot be found in a timely manner, then the game will switch to the lobby setup screen.

**Lobby Setup Screen:**

The lobby setup screen allows a user to create a new game on the server, which others can join. The options to include in the lobby setup screen include player limit, ready button use, lobby max time auto start, game mode, win condition threshold, turn limit, turn time limit, player kick threshold, card discard options. Once the user is done the game will shift to the game lobby screen.

**Game Lobby Screen:**

The game lobby screen is like a waiting area for users to see information about the match rules and other players, while people are joining and getting ready. The screen will likely have a ready button for each user, so the game will only start if everyone wants to play. A time limit could be specified such that the game will start after so much time has passed irrespective of the ready button.

**Play Fill Card Screen:**

The play fill card screen is the first screen of the actually game playing screens. In this screen the game displays the statement to fill, and the cards the user can play. The user should be able to drag their cards onto the statement card, which will change the text on that card to the statement with the fill card replacing the blank. Once the user has selected their fill card they will hit a submit button, and transition to the vote phase screen.

**Vote Phase Screen:**

The vote phase screen is similar to the play fill card screen except that this time the user is selecting from their opponents fill cards. The user will not be able to tell who played any of the cards that they are voting on. Once the player votes, the game moves onto the turn summery screen.

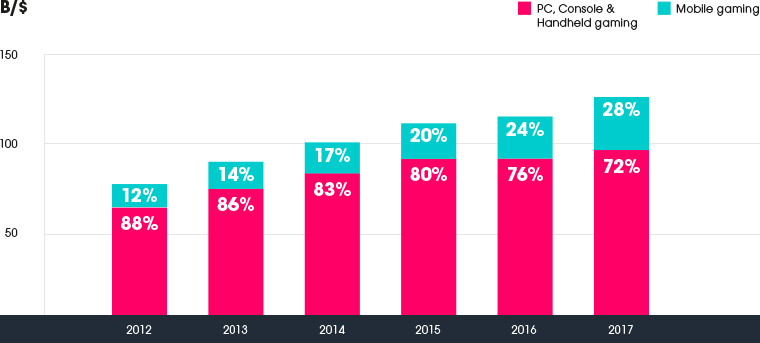
**Turn Summery Screen:**

The turn summery screen serves as a place holder for rounds, and lets the players know who played each fill card, and the points awarded. Once the user is done with the turn summery screen the game will go back to the play fill card screen to start the next round, or it will go to the end game screen.

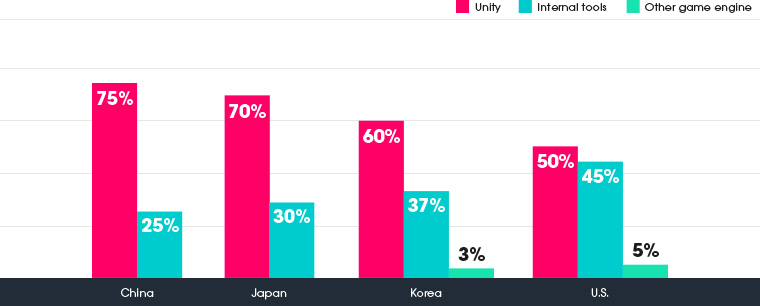
**End Game Screen:**

The end game screen is similar to the turn summery screen except that summarizes the entire game and declares the winner(s). Once the user is done viewing the screen they are given the option to go back to a lobby with the same settings as the last game, or just go back to the main menu screen.

**Customer Values – Mobile Advantages:**



Based on the graph above from unity3d.com the market share of mobile games has been rapidly growing since 2012. This trend demonstrates the clear advantage of developing for mobile platforms as it looks to be future proof.



In terms of using Unity 3d to develop an app, the above graph from unity3d.com shows the market share of 3d mobile games based on game engine. At least in the four countries in the graph, 3d mobile games made with Unity 3d represent the majority of the market.

**Technology:**

**Tools:**

**Unity 3d**

What is Unity 3d? Well based on their website “Unity Technologies is revolutionizing the game industry with Unity, the breakthrough development platform for creating games and interactive 3D and 2D experiences like training simulations and medical and architectural visualizations, across mobile, desktop, web, console and other platforms.”( unity3d.com). This highlights the major consideration for using Unity 3d, as it should allow us to easily port the client from PC to mobile. Additional Unity 3d can automatically work with Microsoft Visual Studio if installed on the same machine.

**C#**

Microsoft describes C# as “C# is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework.” (msdn.microsoft.com). Unity 3d allows users to write scripts for games in either java script or C#, and since C# is similar to C/C++ it was the natural choice since everyone in the group has programmed in C/C++. C# can use the .net socket framework for networking, so that will be our client side network framework.

**Microsoft Visual Studio**

According to Microsoft “Visual Studio is a complete set of development tools for building ASP.NET Web applications, XML Web Services, desktop applications, and mobile applications. Visual Basic, Visual C#, and Visual C++ all use the same integrated development environment (IDE), which enables tool sharing and eases the creation of mixed-language solutions.” (msdn.microsoft.com). Visual studio will primarily used to program the scripts used by the client application.

**Personal:**

**Role:**

Primary client programmer. Implementing the game features and interactivity in the client application. Migration of client from PC to android.

**Skills:**

Programming in C, C++, C#, and python. Experience working with Unity 4 and Unity 5 editors. Experience working with Microsoft Visual Studio.

**Works Cited:**

<https://unity3d.com/public-relations>

<https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx?f=255&MSPPError=-2147217396>

<https://msdn.microsoft.com/en-us/library/fx6bk1f4%28v=vs.90%29.aspx?f=255&MSPPError=-2147217396>