ОСОБЕННОСТИ РАЗРАБОТКИ МНОГОПОЛЬЗОВАТЕЛЬСКОГО КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ НА ГРАФИЧЕСКОМ ДВИЖКЕ UNITY 2D

В.В. Иванцов, А.Т. Зиганшин, Т.М. Катышева, П.А. Хаустов
Томский политехнический университет
ziganshin@sibmail.com

На сегодняшний день индустрия компьютерных игр является бурно развивающимся сектором мировой экономики. Разработкой игр занимаются как крупные компании мирового уровня, так и небольшие команды разработчиков, известные только узкому кругу пользователей. Различаются и их продукты: игры, поражающие своим размахом, производящие фурор в игровой индустрии, и совсем небольшие игры, способные воспроизводиться даже на смартфонах.

Среди всего разнообразия компьютерных игр особое место занимают многопользовательские онлайн игры. Давно прошли те дни, когда люди нуждались в личных встречах с друзьями, чтобы играть в игры. Теперь можно играть когда угодно, и где угодно. Некоторые игры настолько просты, но в то же время увлекательны, что в них можно играть в обеденный перерыв, или по пути на работу. Многие из них созданы по мотивам известных настольных или азартных игр. Подобные игры популярны среди людей, которые хотят отвлечься от повседневной суеты или провести своё свободное время в общении с друзьями.

В ходе изучения теоретического материала по разработке клиент-серверных приложений на языке C#, было принято решение использовать в качестве базового класса класс Socket [1] платформы .NET Framework. На основе базового были разработаны два класса (Server и Client), обеспечивающих сетевое взаимодействие и отвечающие требованиям данного проекта.

Серверное приложение работает в многопоточном режиме, что позволяет одновременно обрабатывать множество подключений (для каждого нового подключения создаётся отдельный поток). Взаимодействие обеспечивается в двустороннем режиме по типу «запрос-ответ». Примерная схема сетевого взаимодействия представлена на рис. 1.

Реализованы различные программные средства, увеличивающие стабильность и надёжность клиент-серверного взаимодействия. Ошибки, возникающие в ходе работы приложения, отлавливаются и корректно обрабатываются как на сервере, так и на клиенте. Кроме того, в случае непредвиденного отключения пользователя во время игрового процесса, текущая сессия игрока сохраняется максимально долго, что позволяет пользователю продолжить игру с момента разъединения. Таким образом, неполадки соединения не влияют на работоспособность сервера или игровой процесс пользователей.

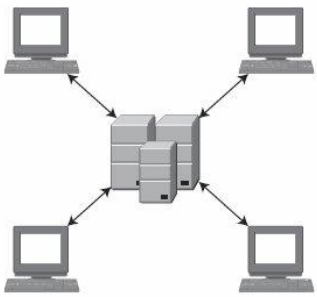

Рис.1. Сетевое взаимодействие

Так как проект является многопользовательским приложением, возникает необходимость в хранении необходимых данных о пользователях (логин, пароль, электронная почта и т.д.). Для этого используется локальная база данных, которая хранит всю необходимую информацию. Сама база на данный момент состоит из одной таблицы и хранится на жёстком диске в виде sdf-файла. Взаимодействие серверного приложения и базы данных обеспечивается с помощью класса SqlCeConnection [2]. Схема базы данных приведена ниже (Рис. 2):

Пользователи		
Никнейм	varchar(15)	
Пароль	varchar(15)	
Вопрос для восстановления пароля	varchar(30)	
Ответ на вопрос	varchar(30)	
Деньги	int(15)	
Add field		

Рис.2. Схема базы данных

В ходе разработки игровой логики для удобства были реализованы различные классы. Свойства класса Gamer содержат информацию о конкретном игроке, которая используется для обеспечения взаимодействия с другими пользователями. Класс GameRoom содержит методы, обеспечивающие взаимодействие игроков внутри игровой комнаты (обмен сообщениями, занятие игровых мест), а также инструменты для подключения и отключения игроков от комнаты. Для простоты был введён класс Place, который содержит свойства, необходимые непосредственно для игрового процесса. Использование класса Place делает игрока независимым от места за столом, что позволяет сменить его в любой момент.

Сам игровой процесс реализован в классе Game. Этот класс содержит методы, описывающие ходы игроков и этапы игры, согласно правилам покера Texas Hold’em.

Таким образом, игроки (Gamer), подключенные к игровой комнате (GameRoom) и занявшие место за столом (Place), могут начать игру. Игра начинается с раздачи карт игрокам (по правилам покера две карты каждому игроку) с помощью метода ShuffleDeck. Далее определяется дилер (игрок, с которого начинается круг) с помощью метода FindDealer(). По правилам игры два игрока ставят обязательные ставки (блайнды). Эта процедура производится автоматически методом Blind(). Игрок, которому принадлежит следующий ход, определяется методом FindNextGamer(). Текущему игроку, в зависимости от ситуации, доступны следующие действия: повысить ставку (метод Raise()), поддержать ставку или, если нет в этом необходимости, передать ход следующему игроку (метод Call_Check()), скинуть карты (метод Fold()). Все выше перечисленные методы реализованы в классе Game.

Игра Texas Hold'em имеет несколько кругов, каждый из которых имеет своё название. За каждый круг отвечает одноимённый метод из класса Game. По завершению последнего круга определяется победитель.

Как известно, в покере существует десять комбинаций, каждая из которых имеет свою силу. Комбинации в порядке возрастания силы: старшая карта, пара, две пары, тройка, стрит, флэш, фулхаус, каре, стрит-флэш, роял-флэш.

Каждый из игроков имеет в распоряжении семь карт: две карты на руках и пять карт на столе. Из этих семи карт алгоритм выбирает пять таких, что они образуют комбинацию с наибольшей силой. Если наиболее сильную комбинацию можно собрать несколькими способами, то правилами покера оговариваются дополнительные критерии сравнения силы комбинаций. Таким образом, выбирается пять карт, которые образуют наилучшую возможную комбинацию игрока.

После этого комбинации всех игроков сравниваются с учетом силы комбинации и дополнительных критериев, игроки ранжируются по местам. Причем сразу несколько игроков могут поделить одно место. Все игроки, которые заняли первое место, делят банк поровну. Возможна ситуация, когда не все игроки поставили одинаковые ставки. В таком случае правилами покера оговорен принцип, по которому победители делят лишь часть денег, после чего деньги начинают распределяться между последующими местами.

Для определения силы комбинаций, мест участников и выигрыша каждого из игроков реализован класс PokerLogic. Который имеет метод PlayRound(), получающий на вход карты и ставки каждого и игроков, возвращающий силу комбинации каждого из игроков и количество денег, которое он получает в результате игры.

Для простоты и наглядности реализации используются класс карты Card и класс комбинации Combination, каждый из которых реализует интерфейс IComparable для возможности сортировки коллекций классов с такими объектами. Так, например, можно сортировать комбинации по силе или выбирать наиболее сильную комбинацию из набора, используя метод CompareTo().

Внутри класса PokerLogic также реализованы вспомогательные методы для определения силы комбинации, наиболее сильных комбинаций игроков, распределения выигрыша, которые инкапсулируют всю логику определения победителей и распределения выигрышей.

Также следует отметить, что в целях защиты от мошенничества вся игровая логика обрабатывается на сервере.

В ближайшем будущем планируется разработка искусственного интеллекта для интеграции в игровое приложение. Это позволит добавить дополнительные режимы игры. К примеру, пользователь может играть с компьютером, не ожидая других игроков, или же играть в оффлайн-режиме. Также будет возможен смешанный режим, где за столом будут присутствовать как реальные люди, так и игрок, под управлением искусственного интеллекта.

Кроме того, планируется использовать кросс-платформенность, как одну из особенностей графического движка Unity [3-4]. Это позволит расширить аудиторию возможных пользователей.

Ещё одной задачей развития игрового приложения является оптимизация игровой логики и клиент-серверного взаимодействия. Отдельное внимание будет уделено безопасности клиент-серверного взаимодействия: для повышения его безопасности, передаваемые данные будут шифроваться.

Литература

1. Socket – класс. [Электронный ресурс]. - Режим доступа: http://msdn.microsoft.com/ruru/library/system.net.sockets.socket.aspx, свободный.
2. Класс SqlCeConnection [Электронный ресурс]. –Режим доступа: http://msdn.microsoft.com/ru-ru/library/system.data. sqlserverce. sqlceconnection/, свободный.
3. Unity [Электронный ресурс]. Режим доступа: http://unity3d.com/, свободный.
4. Unity3D [Электронный ресурс]. Режим доступа: http://habrahabr.ru/hub/unity3d/, свободный