Dirty Pixel

TWIGLIGHT
REQUIREMENT ANALYSIS REPORT

1395045 – Anil Yigit Filiz

1394642 – Berkehan Altinkaya

1394600 – Derya Akpinar

1394980 – Gunes Efe

6th November 2006

TABLE OF CONTENTS

1. Project Overview

Twilight is a massive multi-player 3D game that is intended to be played by a massive number of players sharing the same server. It will be a standalone program which will run on the players machines. The game is space simulation game, set in a distant future. Players begin with creating a character that they will play throughout the game. Their characters will gain experience points and improve their skills by completing some missions or engaging in battles. There will be a couple of races and a couple of spaceship types of these races from which the players may select their ships. The game will never end. When the players dies he/she will lose some experience points and their spaceship. They will respawn from their base. It will have a consistent universe which will be running on the server. There will be also NPC's in the game besides the players. These NPC's will have different levels of intelligence depending on the mission types. Easy to use user interface, easy control and sound effects will improve the playability of the game.

2. Requirements

Hardware Requirements

Hardware requirements for the users and developers are nearly the same. They are stated as follows:

- P3 or Equivalent Processor
- 3D Graphics Card of 32 MB Memory with OPENGL support
- 128 MB RAM
- Sound Blaster of Compatible Sound Card
- Keyboard and Mouse

Software Requirements

Software requirements for the developers are:

- Object 3DS Loader (Open Source)
- GLUT Library (Open Source)
- 3D Studio Max
- Windows XP Operating System
- Dev C++ IDE (Free)
- A Free DBMS

3. Project Specifications

1. General Approach

The programming language that we will use is C++. We will use opengl and glut libraries for graphics rendering. We will use 3D Studio Max to create our models and then we will use the open source Object3DS library to import these models into opengl. We will be creating the project in Dev C++ IDE with an object oriented approach. We have divided the project into main modules and we will implement these modules as independent classes. Details of these modules are described below.

2. Main Game Loop

The main game loop will be the heart of the project and will run in client machines. It will be the primary process. It will be used to call the necessary methods and change the program state accordingly. Also the calculations for the timing of the display and user input will be done in the main loop. The main loop manages the network interaction with the server and decides when to update the server data according to the data that is produced by the game engine.

3. Game Engine

The game engine is responsible for calculating the movement of the clients objects according to physics laws and the user input. It processes the data that is received from the server. It performs collision detection between the clients objects and other objects. It modifies the data of the client such as health, experience, shield status etc. according to the actions clients take. It then passes the processed data to the network modules to be sent to the server.

4. Graphics Engine

During gameplay the graphics engine takes its input from the received data from the server and renders it accordingly. It also renders graphics objects such as lighting and background. The user interface is also rendered by the graphics engine according to the game state.

5. Network Modules

The main constraint in developing online games is the server-client architecture. We have (after a lot of research) decided in having a non-dedicated server. We decided that the processing of the game should be divided between clients and the server. To achieve this the clients will process the part of the data they are responsible for. That is, the coordinates of their spaceship and the missiles they fire and the collision detection between their missiles and the other objects. They will then put this data in a send buffer. A separate process will send the contents of the buffer to the server using TCP/IP. On the server side there will also be two modules for sending and receiving packets to the clients. There will also be a database on the server where the information about users such as username, password, experience, last health, owned spaceships etc. are stored. When the user exits these information will be flushed into the database. These information will also be flushed into the database when clients clients connection encounters a time-out.

6. Sound

The sound engine will be invoked by the game engine. The sound engine will also be responsible for the playing the game music. There will be changes in the game music according to the game state. There will be some AI programming for determining the current mood of the game for changing the music to an appropriate one.

7. User Interfaces

As stated above user interfaces will be rendered by the graphics engine according to the state of the game. These states are the login screen, character creation, purchasing screen, options screen, death of a player, a connection problem etc. In our market research we analyzed some other games similar to the one we intend to create. We decided on which elements we'll use in the gameplay screen. These elements are

- Map
- Aim Lock
- Quick Access Tray
- Health, Shield, Ammo, Fuel, Experience Information
- Chat (Used for interactive gameplay for own race.)

4. Project Management

1. Team Organization

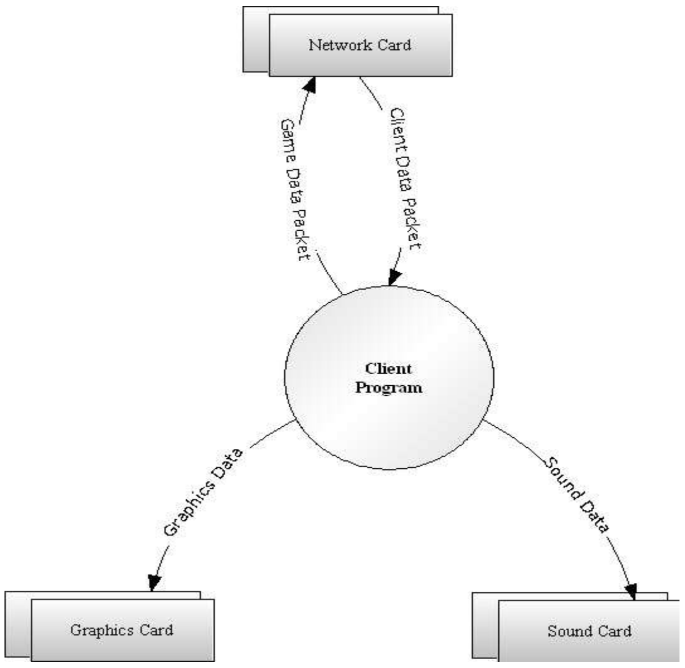
The team will be a democratic centralized group. Our leader will control the whole progress and assign tasks to the group members. But the main decisions of the game will be decided by the whole group. Every week we'll regroup, check the progress and reassign the tasks among the members. We'll also provide a progress report to our supervisor according to these meetings.

2. Management Strategy

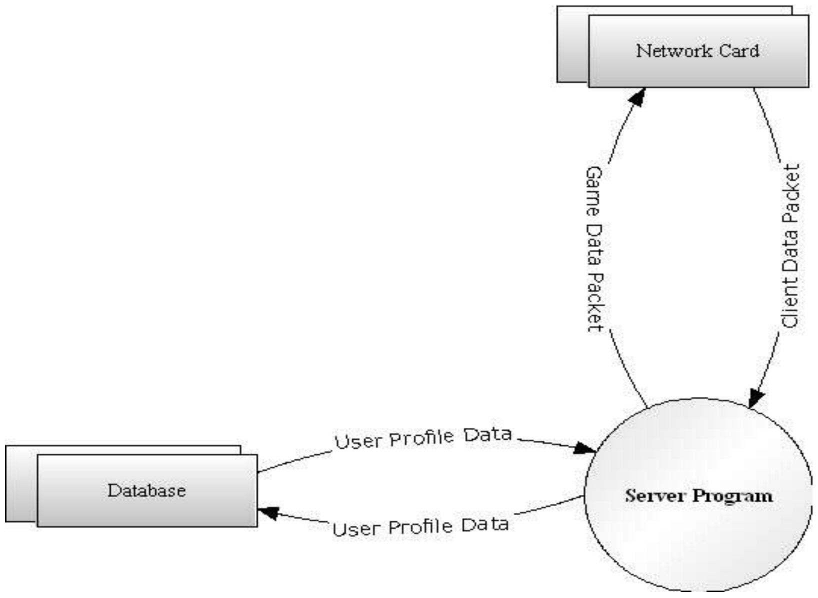
This project will be managed by waterfall process model. Iteratively the project team will go through planning, modeling, construction and deployment stages. Since the requirements of the project won't change there is no need for an evolutionary or incremental process model. Even though we'll use the waterfall model we'll build a working prototype just to see if we can manage network synchronization. The classes we'll use in this prototype will be a base for the project.

5. Data Flow Diagrams

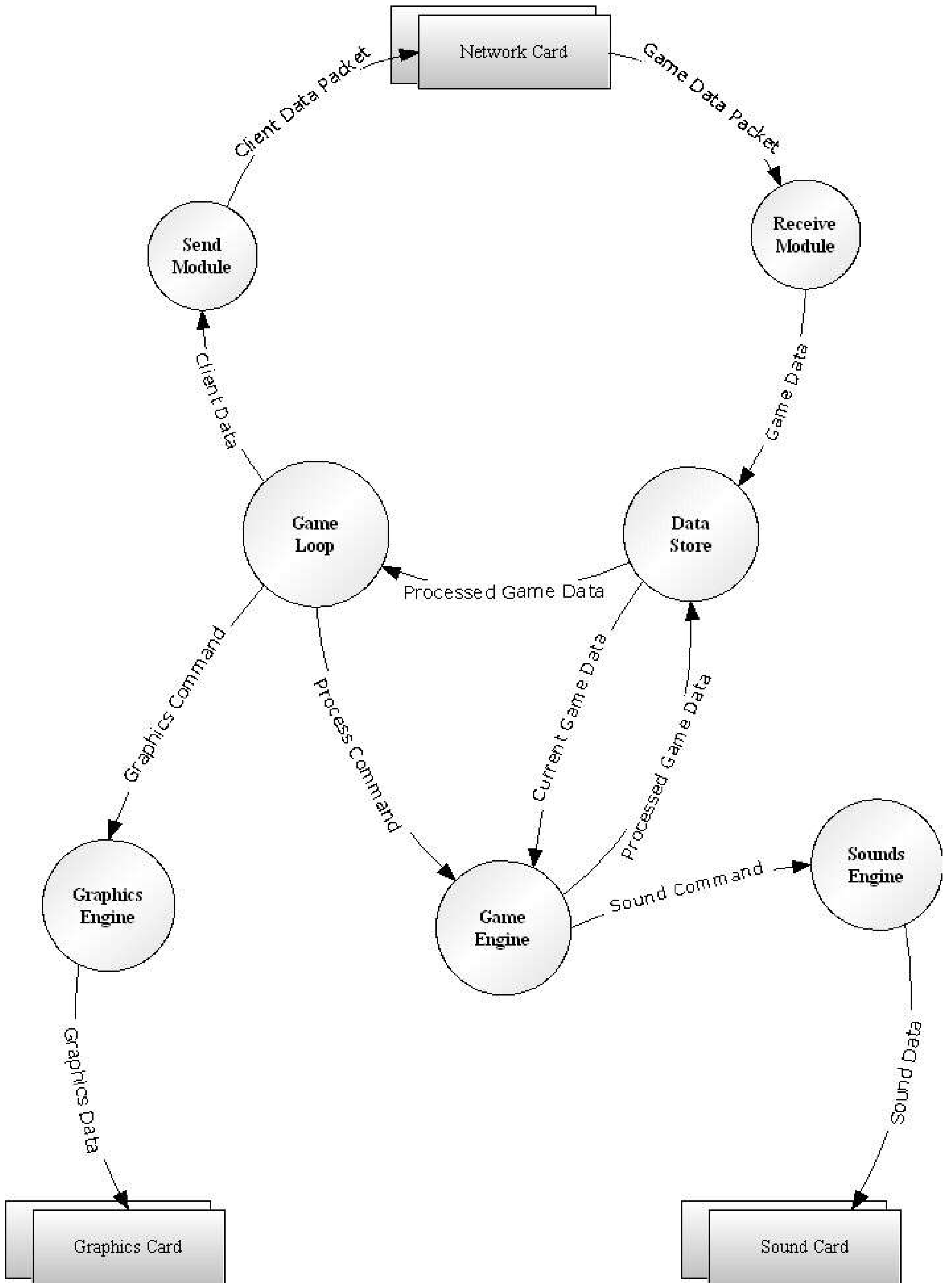
Client Level 0 DFD



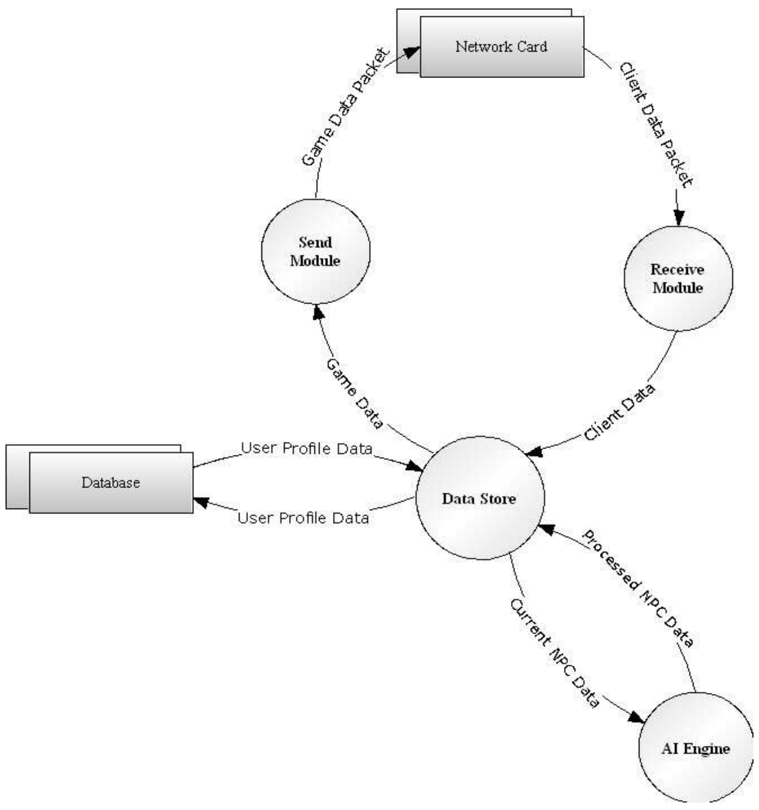
Server Level 0 DFD



Client Level 1 DFD



Server Level 1 DFD

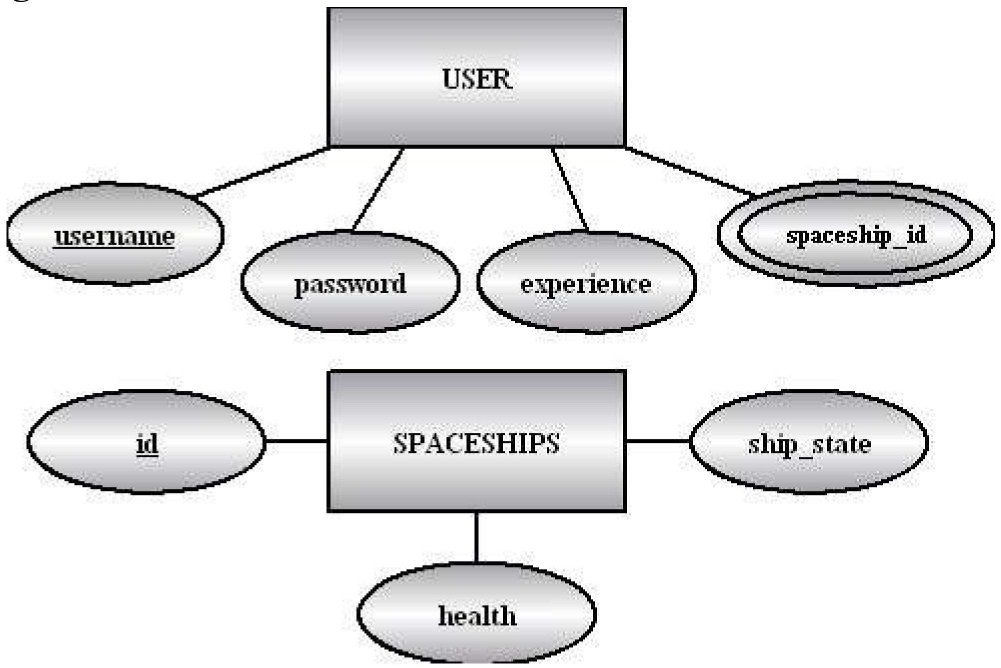


The client and server side DFD's of the project are above.

The client program will be in interaction with the network card, sound card and the graphics card of the client computer. There will be a game loop which will send commands to the game engine and the graphics engine. It will also send the client data to the send module which is to be sent to the IP of the server. The packets from the server will be received by the receive module and these data will be stored for the computation of the game engine. The game engine will process this data and update it accordingly. Then this updated data will be sent to the server side via game loop and the send module. The game engine will also invoke sound events. These events will be handled by sound engine. Sound engine will also decide which music to play regarding to the state of the game. Finally the graphics engine will update the display frame buffer according to the commands received from the game loop.

The server program will be in interaction with the network card of the server and the DBMS on the server. It will be responsible for the management of the whole packages in the system as well as the NPC data. It will include an AI module which controls the NPC's and other free objects in the environment. This AI module will process the data in a time interval like 40 ms in order to achieve the frame display rate of 25 fps. After the process of data, it will update the whole game data in the server. The data store module in the server will be responsible for the database transactions as well as the packet distribution to the clients.

6. ER Diagram



The ER diagram of the database in the server is above. This database is needed for logging the user information. Since the game will have a consistent universe it is very important to store the up to date information about the users. The user table will hold username, password which are defined by the user at the very beginning of the game; the user experience which is gained accomplishing missions or destroying other players; and a couple of spaceship ids depending on the number of spaceships user has. These spaceship ids will be looked up from another table "spaceships" where the user spaceships information are held. Since the transaction load is not that much of a finance institution software there is no need for an expensive DBMS like oracle. Instead we are thinking of using a free DBMS.

7. Appendix

1. Market Report

INTRODUCTION

Massively multiplayer online game market became popular in the late 90s with games like Ultima Online(1997) and EverQuest(1999) . The growing trend in massive multiplayer online games is still keeping the market alive. Depending on the user reviews one of the most popular games in our time is Eve Online by CCP games which is very similar to the game we would like to produce. Other popular examples of sci-fi MMO games are: Freelancer by Microsoft and Jumpgate by NetDevil.

EVE ONLINE

EVE Online is a persistent world multiplayer online game set in space. It is developed by the Icelandic company CCP Games and was published from May to December 2003 by Simon &

Schuster Interactive, after which CCP purchased the rights back and began a digital distribution scheme. Players pilot customizable spaceships in an immense online environment. Players have a wide array of ship designs available to them, each of which is suited for specific uses. Eve Online is one of the few MMOGs in which all players exist in one large virtual universe at the same time, without the "sharding" (distributing the players to different servers) common in other MMOGs.

JUMPGATE: THE RECONSTRUCTION INITIATIVE

Jumpgate: The Reconstruction Initiative is an MMORPG in a science fiction setting for the PC, released However, only months after 3D0 released the game to stores, their contract with NetDevil was broken or cancelled. NetDevil then hired Themis Group to provide technical support and provide in-game events. In the game, jumpgates are used to travel from sector to sector. All sectors, without a station, consist of asteroids and a beacon. Many have Player Owned Stations and may have items related to events run by GMs. Station sectors all have a planet within the sector. The player is the pilot of a spacecraft, acquiring wealth and status by engaging in trade, mining, and/or combat. Players may work on their own or together as members of a collaborative squad. Players can upgrade their craft in various ways, including adding weapons, engines, shields, and other modifications and upgrades.

FREELANCER

Freelancer is a space simulation computer game developed by Digital Anvil and published by Microsoft in March 2003 for Microsoft Windows. The primary distinguishing feature of the game is that it does not end when the last in-game mission is completed. At that time the player has only explored a small part of the game universe, and is then free to continue to explore the rest of the region.

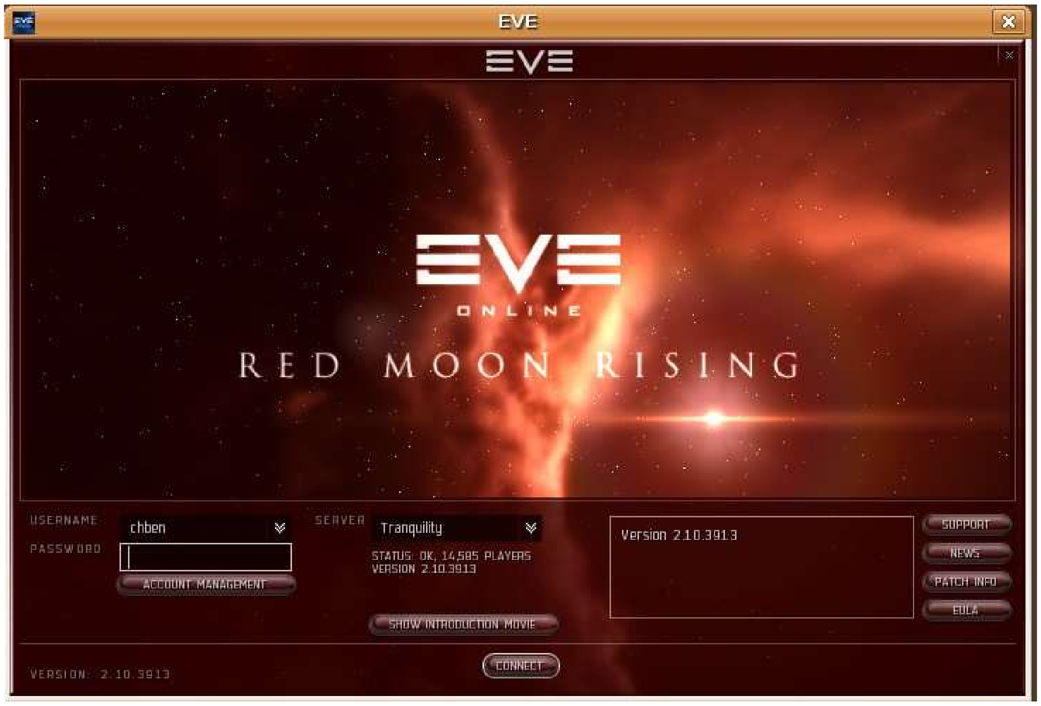
COMPARISON TABLE

	Eve Online	**Jumpgate**	**Freelancer**	**Our Game**
Released by	CCP games	NetDevil	Microsoft	Dirty Pixel Studios
Release date	December 2003	September 2001	March 2003	June 2007 (expected)
Multiplayer support	Yes	Yes	Yes	Yes
Single player support	No	No	Yes	No
Log-in to a server provided by the company	Yes	Yes	No	No
Client sale method	Download	Download	Retail sale	Unknown
Client purchase price	Not for sale	Not for sale	45.99$	Unknown
Monthly fee	14.99$	9.95$	None	None
Platform	Windows	Windows	Windows	Windows
Genre	Science Fiction	Science Fiction	Science Fiction	Science Fiction
Camera views	First Person	First Person	First Person and Third Person	First Person
Persistent universe	Yes	Yes	Yes	Yes
Open-ended universe	No	No	Yes	No
No. of players	125,625	69,311	Unknown	Unknown

Customizable user interface	Yes	No	No	No
Graphics rendering	Directx 9	OpenGL	Directx 9	OpenGL

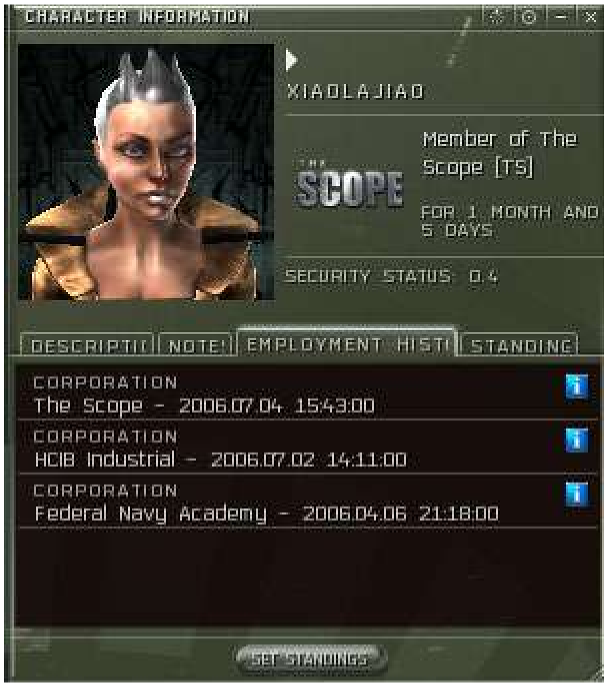
Screenshots From EVE Online

Login Screen



USERNAME chben
SERVER Tranquility
STATUS: OK, 14,585 PLAYERS
VERSION 2.10.3913
Version 2.10.3913
VERSION: 2.10.3913

Character Information



CHARACTER INFORMATION

XIAOLAJIAO

Member of The Scope [TS]

FOR 1 MONTH AND 5 DAYS

SECURITY STATUS: 0.4

DESCRIPTI | NOTE | EMPLOYMENT HIST | STANDING

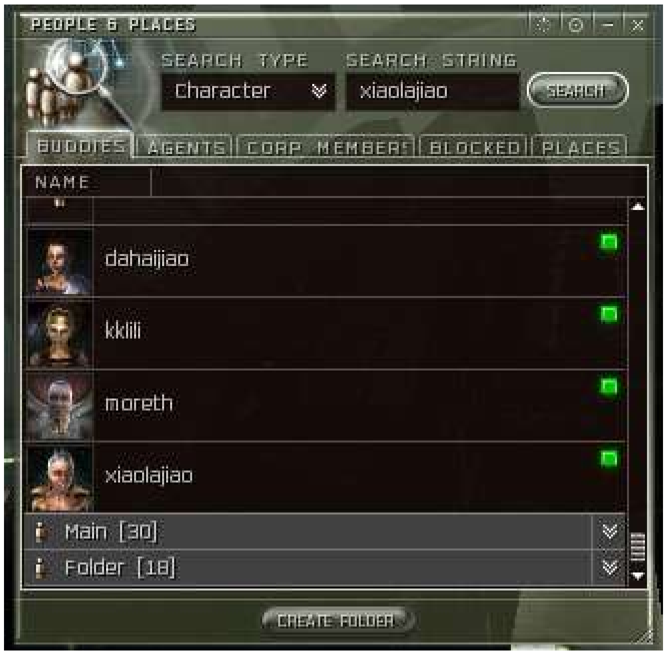
CORPORATION
The Scope - 2006.07.04 15:43:00

CORPORATION
HCIB Industrial - 2006.07.02 14:11:00

CORPORATION
Federal Navy Academy - 2006.04.06 21:18:00

SET STANDINGS

People and Places



PEOPLE & PLACES

SEARCH TYPE: Character

SEARCH STRING: xiaolajiao

SEARCH

BUDDIES | AGENTS | CORP MEMBERS | BLOCKED | PLACES

NAME

dahaijiao

kklili

moreth

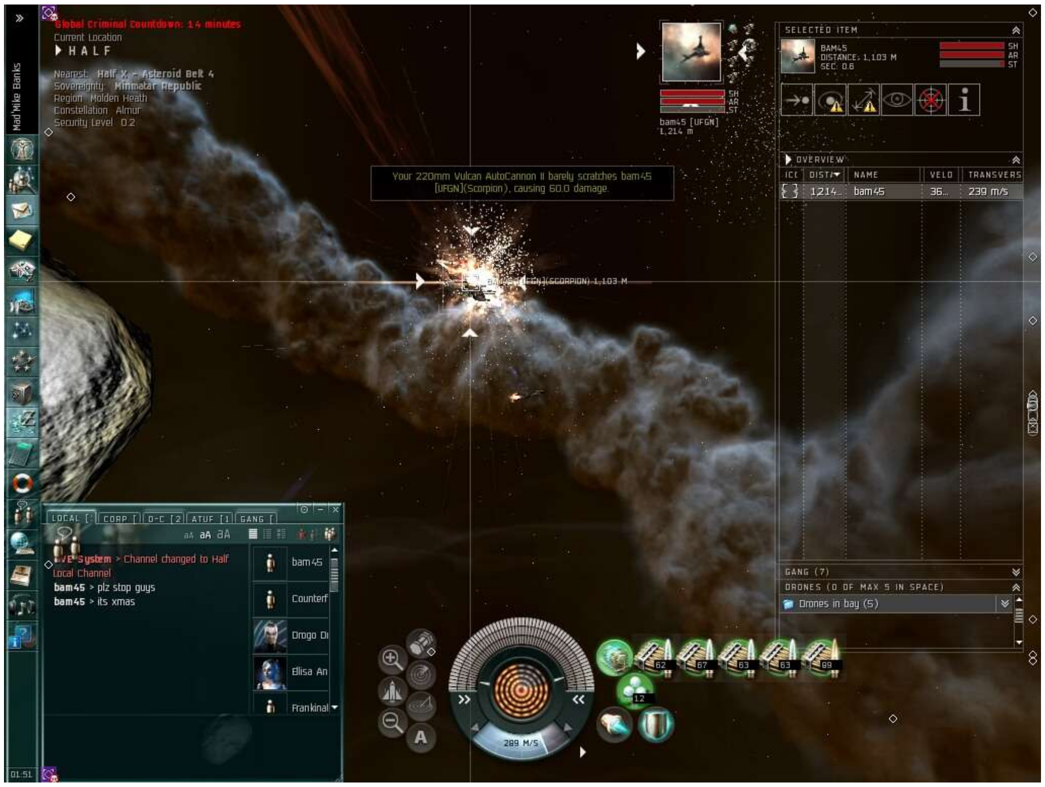
xiaolajiao

Main [30]

Folder [18]

CREATE FOLDER

Main Game Screen



Global Criminal Countdown: 14 minutes
Current Location
HALF

Nearest: Half X - Asteroid Belt 4
Sovereignty: Minmatar Republic
Region: Molden Heath
Constellation: Almur
Security Level: 0.2

Your 220mm Vulcan AutoCannon II barely scratches bam45 [UFGN](Scorpion), causing 60.0 damage.

bam45 [UFGN]
1,214 m

BAM45
DISTANCE: 1,103 M
SEC: 0.6

ICI	DIST	NAME	VELO	TRANSVERS
	1,214...	bam45	36...	239 m/s

EVE System > Channel changed to Half Local Channel
bam45 > plz stop guys
bam45 > its xmas

Task Name	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17	Week 18	Week 19	Week 20	Week 21	Week 22	Week 23	Week 24
Game Concept Development																								
* Storyline	▓	▓	▓	▓	▓	▓	▓	▓																
* Gameplay design					░	░	░	░	¬															
* Interfaces design									▓	▓	▓	▓	▓	▓	▓	▓								
* Sketching	░	░	░	░	░	░	░	░	¬															
* Objects design									░	░	░	░												
* Objects modelling											▓	▓	▓	▓	▓	▓	▓	▓	▓					
Graphics Engine Development																								
* Engine basics creation	▓	▓	▓	▓	▓	▓	¬																	
* Displaying 3ds objects							░	░	░	░	░	░												
* Displaying animations											▓	▓	▓	▓										
* Displaying lights													░	░	░	░								
Network Modules Development																								
* Chat module	░	░	░	░																				
* Packet creation													▓	▓	▓	▓	▓	▓						
* Packet parsing													░	░	░	░	░	░						
* Packet transfer			▓	▓	▓	¬																		
* Send and Receive modules						░	░	░	░	░	░	░	░	░	░	░	¬							
* Synchronization																	▓	▓	▓	▓	▓	▓	▓	
Data Structures Development																								
* Organization of data	▓	▓	▓	▓	▓	▓	▓	▓	¬															
* Encapsulation of data									░	░														
* Database tables creation											▓	▓												
* Database interaction classes											░	░												
Artificial Intelligence Development																								
* NPC design					░	░	░	░																
* NPC behavior modelling													▓	▓	¬									
* NPC behavior programming															░	░	░	░						
Game Flow Development																								
* Maps design															░	░	░	░						
* Maps programming																	▓	▓	▓	▓				
* Quests design									░	░	░	░	░	░	░	░	░	¬						
* Quests programming																		▓	▓	▓	▓			

* Game flow programming																			░	░	░	▒		
Deployment and Testing																								
* Deployment of classes													░	░	░	░	░	░	░	░	░	▒	▒	
* Designing test cases																							▒	
* Testing																							▒	▒