Maturaarbeit 2004

Programmierung eines Onlinespiels

Planung und Umsetzung

Nicolas Perrenoud, Klasse 1c

Betreut von Andreas Locher

VORWORT

Liebe Leserin, lieber Leser

Sie halten hier meine Maturaarbeit zum Thema „Programmierung eines Onlinespiels" in den Händen. Computer faszinieren mich seit Jahren, und ich verbringe viel Zeit damit, Programme zu erstellen, Webseiten zu gestalten, Systeme zu konfigurieren und ab und zu auch ein Spiel zu spielen. Schon lange hat es mich gereizt, selber ein Computerspiel zu programmieren, und dies konnte ich nun mit dieser Arbeit verwirklichen.

Ich möchte an dieser Stelle Andreas Locher danken, dass er meine Arbeit betreut hat. Ich danke auch allen Mitspielern des BETA-Tests. Durch ihre konstruktive Kritik haben sie mir geholfen, das Spiel fehlerfreier zu machen und zu erweitern. Ebenfalls möchte ich allen Menschen danken, die ihre Freizeit dafür einsetzen, um freie Software zu programmieren und zu verbessern. Ohne sie gäbe es keine so guten Programme wie den Mozilla Browser, den Apache Webserver, das MySQL Datenbanksystem oder die PHP Skriptsprache.

Ich wünsche Ihnen nun viele lehrreiche und interessante Augenblicke beim Lesen meiner Maturaarbeit.

Nicolas Perrenoud

INHALT

1 Einleitung 4
1.1 Fragestellungen 4

2 Von der Idee zum fertigen Produkt 5
2.1 Idee 5
2.1.1 Was ist ein Browsergame? 5
2.1.2 Wie funktioniert ein Browsergame? 5
2.2 Vorüberlegungen 6
2.2.1 Informationsbeschaffung 6
2.2.2 Wahl der Programmiersprache und der Datenbank 6
2.2.3 Testspiel 7
2.3 Detailkonzept 8
2.4 Programmierung 9
2.4.1 Dinge, die man beachten muss 9
2.4.2 Schwierigkeiten und Lösungen 10
2.4.2.1 Die Aktualisierung von Ereignissen 10
2.4.2.2 Die Punkteaktualisierung 11
2.4.2.3 Das Kampfsystem 13
2.4.3 Ressourcenschonendes Programmieren 13
2.5 Testphase 14
2.5.1 Eigene Tests 14
2.5.2 Der Beta-Test 15
2.5.2.1 Was ist ein Beta-Test? 15
2.5.2.2 Wie macht man das Spiel bekannt? 16
2.5.2.3 Beurteilung des Beta-Tests 16
2.6 Das fertige Spiel 17
2.6.1 Spielablauf 17
2.6.2 Features des Spiels 18
2.6.3 Wie geht es weiter? 18

3 Zusammenfassung 19

4 Quellen 20
4.1 Internetquellen 20
4.1.1 Informationen 20
4.1.2 Programmierhilfen 20
4.1.3 Verwendete Software 20

5 Anhang 21
5.1 Über Datenbanken und Programmiersprachen 21
5.1.1 Was ist eine Serverskriptsprache? Wie funktioniert sie? 21
5.1.2 Funktion und Aufbau einer Datenbank 22

5.2 Detailkonzept 24
5.2.1 Über die Idee des Spiels 24
5.2.1.1 Geschichte 24
5.2.1.2 Ablauf des Spiels 24
5.2.1.3 Ziel des Spiels 25
5.2.2 Funktionsweise des Programms 26
5.2.2.1 Gebäude, Forschung, Schiffbau etc. 26
5.2.2.2 Flottenflug 27
5.2.2.3 Kampfsystem 28
5.2.3 Layout 28
5.2.3.1 Die genaue Anordnung der Frames 29
5.2.3.2 Sternenkarte 29
5.2.4 Datenbankstruktur 30
5.2.5 Programm-/Verzeichnisstruktur 30
5.2.6 Regeln für die Namensgebung 31
5.2.7 Objekte 31
5.2.7.1 Galaxie, Sektoren 31
5.2.7.2 Sonnensysteme, Sterne 32
5.2.7.3 Planeten 32
5.2.7.4 Wurmlöcher 33
5.2.7.5 Rassen 33
5.2.7.6 Bewohner, Crew, Arbeiter 34
5.2.7.7 Gebäude 34
5.2.7.8 Forschungen 36
5.2.7.9 Schiffe 36
5.2.7.10 Verteidigung 37
5.2.8 Funktionen 38
5.2.8.1 Nachrichtensystem 38
5.2.8.2 Statistik 38
5.2.8.3 Spieleroptionen 38
5.2.9 Prioritätenliste 39
5.2.10 Quellen 39
5.3 Screenshots – Bilder des Spiels 40

1 EINLEITUNG

Das Thema der hier vorliegenden Maturaarbeit ist die Programmierung eines Onlinecomputerspiels. Ich setzte mir das Ziel, sowohl die Planung als auch die Umsetzung eines solchen Spieles zu behandeln. Darum programmierte ich als praktische Arbeit von Grund auf das Computerspiel *Escape to Andromeda*, ein Science-Fiction Strategiespiel. In diesem Bericht blicke Ich auf die sechsmonatige Arbeit zurück, gehe auf die verschiedenen Aspekte der Planung und Umsetzung ein und veranschauliche diese mit Beispielen aus meiner praktischen Arbeit. Ich will so den Weg von der Idee bis zum fertigen Spiel aufzeigen. In der Zusammenfassung werte ich die Arbeit aus und schaue, welche Teile des Detailkonzepts ich verwirklichen konnte und welche nicht. Der Anhang beinhaltet das Detailkonzept und eine kurze Erklärung zur Datenbank und zur Programmiersprache.

1.1 Fragestellungen

Ich habe mir folgende zwei Fragestellungen für meine Arbeit formuliert und versuche nun im Folgenden, diese zu beantworten:

- Wie läuft der Prozess bei der Programmierung eines Spieles ab (Idee, Konzept, Planung, Umsetzung, Veröffentlichung)?
- Wie kann man ein Spiel möglichst ressourcenschonend programmieren (bei Internet-Spielen besonders wichtig)?

2 Von der Idee zum fertigen Produkt

2.1 Idee

Ich spiele selber ab und zu Computerspiele, vor einiger Zeit vor allem auch ein Browsergame namens OGame. Da mir bei diesem Spiel viele Mängel aufgefallen sind und es Sachen gibt, die ich gerne anders machen würde, kam mir die Idee, selber so etwas zu programmieren. Mein Spiel hat darum auch mit Science-Fiction zu tun; ein weiterer Grund ist, dass ich mich sehr für Astronomie und Raumfahrt interessiere. Das Spiel ist ein reines Multiplanerspiel (Mehrspielerspiel), dass über das Internet mit Hilfe eines normalen Internetbrowser erreichbar ist.

2.1.1 Was ist ein Browsergame?

Es ist ein Computerspiel, dass über ein grosses Netzwerk (damit ist vor allem das Internet gemeint) gespielt wird; jeder kann mitspielen, man braucht dazu nur einen normalen Webbrowser. Das Spiel läuft serverseitig, und alle Daten des Spielers werden in einer Datenbank abgespeichert, d.h. der Spieler klickt z.B. auf einen Button, dieser startet eine Anfrage an den Webserver, welcher die Anfrage bearbeitet, die nötigen Daten aus der Datenbank holt, daraus eine Seite generiert und an den Browser zurückschickt. Beispiele für sehr populäre Browsergames sind Scherbenwelten[1] (Fantasy), Wogen des Schicksals[2] (Fantasy), GalaxyWars[3] (Science-Fiction), Ogame[4] (Science-Fiction) oder Syndicates[5] (Wirtschaft der Gegenwart).

2.1.2 Wie funktioniert ein Browsergame?

Bei den meisten Spielen geht es darum, seine Ressourcen zu vermehren (z.B. Geld, Metall, Holz, usw.), Städte und Objekte zu bauen (Gebäude, Fahrzeuge, Schiffe, Flugzeuge, Armeen, Raumschiffe....), neue Sachen zu erforschen, mit

[1] http://www.scherbenwelten.de

[2] http://www.wogen-des-schicksals.de

[3] http://www.galaxywars.de

[4] http://www.ogame.de

[5] http://syndicates.4players.de

anderen Spielern zu handeln und sich durch strategische Eroberung von Gebieten oder Lösen von bestimmten Aufgaben Punkte zu holen. Bei vielen Browsergames wird auch das Rollenspiel sehr betont: man hat seinen Charakter, seine Figur, die bestimmte Eigenschaften hat und in Kontakt mit anderen Spielern tritt. Das Spielen eines solchen Spiels kann sehr zeitaufwändig sein, wenn man zu den Besten gehören will. Ein gutes Spiel zeichnet sich aber auch dadurch aus, dass schlechtere Spieler genauso mithalten können.

2.2 Vorüberlegungen

Den grössten Fehler, den man beim Erstellen eines Programms machen kann, ist, einfach drauflos zu programmieren, ohne überhaupt irgendwelche Vorabklärungen gemacht zu haben. Es kann sein, dass auch so ein funktionierendes Programm entsteht, aber meistens mit chaotischem Aufbau des Codes, und es kann sein, dass man mitten im Programmierprozess merkt, dass etwas Grundsätzliches falsch durchdacht worden ist.
Bei meinem Spiel hatte ich viele verschiedene Abklärungen zu treffen. Zum einen musste ich schauen, wie ich zu Informationen kam, die mir beim Programmieren und beim Ideensammeln weiterhelfen konnten, zum anderen war da die Wahl der richtigen Programmiersprache und das Erstellen eines Testspiels.

2.2.1 Informationsbeschaffung

Viele Ideen zum Spiel bekam ich durch ähnliche Spiele, die ich früher mal gespielt habe. In den Foren dieser Spiele werde häufig Vorschläge zur Verbesserung geschrieben, und so konnte ich dort Ideen für mein Projekt sammeln. Weitere Hilfen waren Online-Enzyklopädien und Astronomie-Seiten.
Da ich gerne Science-Fiction-Literatur und Filme lese bzw. anschaue, konnte ich auch Ideen von diesen Quellen verwenden.

2.2.2 Wahl der Programmiersprache und der Datenbank

Als Programmiersprache verwendete ich die serverseitige Skriptsprache PHP[6] und als Datenbank MySQL. Serverseitig bedeutet, dass das Programm auf einem

[6] http://www.php.net

zentralen Rechner ausgeführt wird, und der Benutzer bekommt nur noch die generierten Daten zu sehen. Ich wählte PHP, weil ich diese Sprache relativ gut kenne und schon seit einigen Jahren dynamische Webapplikationen damit programmiere. Dies waren im Vergleich zu diesem Projekt nur kleine Sachen wie z.B. ein Gästebuch, ein Chat oder ein Online-Shop. Das Spiel, dass nun programmiert werden sollte, war viel komplexer und der Code musste sehr gut durchdacht werden, da viele Objekte miteinander agieren. Das eine Datenbank mit einbezogen werden musste war klar, weil es viel umständlicher ist, viele verschiedene Informationen aus Textdateien herauszulesen.

2.2.3 Testspiel

Da ich mir vornahm, wirklich alles von Grund auf zu erstellen, musste ich zuerst überprüfen, ob das Spiel so funktionieren würde wie ich mir das vorstellte, und ob ich die Funktionsweise von Browsergames begriffen hatte. Ich erstellte darum vor dem Beginn der eigentlichen Arbeit ein „Minispiel" mit Funktionen, die im richtigen Spiel dann auch so funktionieren sollten.
Ich konnte so den Aufwand und die Möglichkeiten abschätzen und mir genaue Überlegungen zur Struktur der Datenbank machen. Dieses Spiel hatte kein Konzept, es war ja auch nur zum Test da. Es hatte den selben Aufbau wie viele bekannte Sci-Fi Browsergames. Es hatte eine Login- und Registrierungsfunktion, man konnte Gebäude bauen und Rohstoffe sammeln.

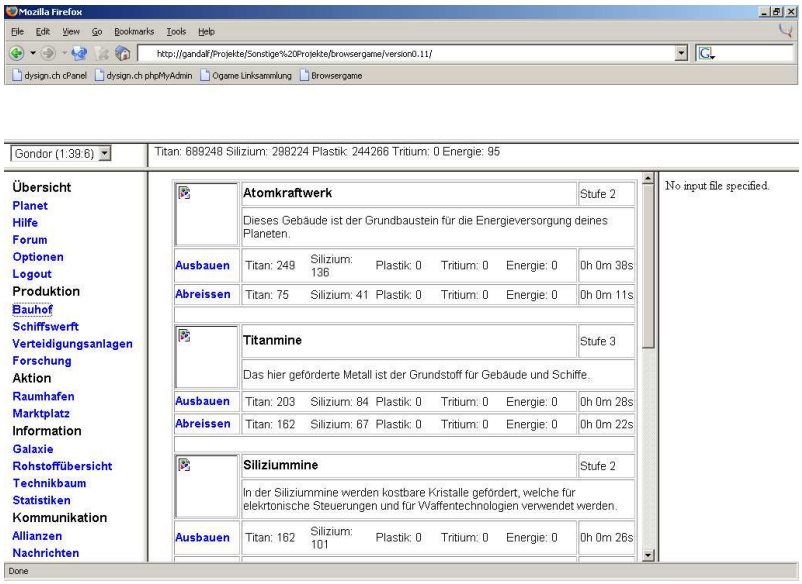

Abb. 1:
Gebäudebaumenü des Testspiels

Das Spiel funktionierte wie es sollte, und darum ging ich nun über zur Planung meines richtigen Spieles; es ist hier noch zu sagen, dass ich für das richtige Spiel keinen Code vom Testspiel wiederverwendete, weil der zu wenig gut aufgebaut und strukturiert war.

2.3 Detailkonzept

Das Detailkonzept ist etwas vom wichtigsten am Programmieren überhaupt. Im professionellen Bereich wird häufig viel mehr Zeit für die Ausarbeitung des Detailkonzepts aufgewendet als für die eigentliche Programmierung. Wenn das Konzept gut ausgearbeitet ist, muss man beim Programmieren gar nicht mehr viel überlegen und muss quasi nur noch das Konzept in eine Programmiersprache ‚übersetzen'. Das Konzept beinhaltet alle akzeptierten Ideen, die man für das Spiel umsetzen will.

> *Jede Idee des Konzepts muss hinreichend abstrakt sein und Strategien, Methoden und Masse beinhalten, die es erlauben, die Ideen auf das Programm anzuwenden, um so - quasi automatisch - das Ergebnis zu erreichen.*
> (vgl. http://www.wikipedia.de/wiki/Konzept, 10.10.2004).

Konkret auf mein Projekt angewendet heisst das, dass im Konzept folgendes vorhanden sein muss:

- Ablauf des Spiels (für den Spieler)
- ein Grundaufbau des Spiels vom Programmieren her gesehen
- alle festen Definitionen
- das Layout
- das Datenbankschema
- Beschreibung der verschiedenen Objekte
- Funktionen / Module
- Prioritätenliste / Zeitplan

Wichtig ist, dass man während dem Programmieren das Konzept immer wieder hervorholt und Vergleiche macht. Wie sich bei mir zeigt, konnte das ursprüngliche Konzept nicht zu 100% eingehalten werden, einige Sachen sind dazugekommen oder erweitert worden, andere erwiesen sich in der Umsetzung als zu zeitaufwändig und wurden abgeändert oder weggelassen. Das ganze Detailkonzept findet sich weiter hinten in dieser Arbeit.

2.4 Programmierung

Nun ging es an den aufwändigsten Teil des Projekts: die Programmierung. Ein für mich wichtiger Grundsatz ist: Programmieren beginnt nicht mit dem Eintippen von Codezeilen, sondern mit Bleistift und Papier. Bevor man überhaupt anfängt, nimmt man zuerst das Konzept zur Hand, überlegt sich den Ablauf des Programms nochmals und zeichnet dann ein ungefähres Schema auf Papier auf. Danach beginnt man mit dem Programmieren. Dieser Programmierprozess dauerte bei meinem Projekt mehrere Wochen.

2.4.1 Dinge, die man beachten muss

Einige Sachen sind mir beim Programmieren wichtig geworden:

- **Strukturierten Code schreiben**

  Schlussendlich ist es dem Anwender eines Programms gleich, wie der Code aussieht, aber sobald man als Programmierer einen Fehler suchen muss, ist man froh, wenn man den Code strukturiert hat und immer das gleiche Schema benutzt. Ich strukturiere meinen Code immer folgendermassen:

Codebeispiel Struktur

```
elseif ($_GET['action']=="ranks")
{
	if($_POST['ranknew']!="")
		dbquery($sql_query1);
	if($_POST['ranksubmit']!="")
	{
		if (count($_POST['rank_name'])>0)
		{
			foreach ($_POST['rank_name'] as $id=>$name)
			{
				if ($_POST['rank_del'][$id]==1)
					dbquery($sql_query2);
				else
					dbquery($sql_query3);
			}
		}
	}
}
```

  Bei Code mit vielen Funktionen, Bedingungen und Schleifen finde ich es wichtig, dass man den Code mit Tabulator entsprechend einschiebt, dies hilft vor allem beim Überarbeiten des Codes oder bei der Fehlersuche.

- **Kommentare dazuschreiben**

  Das Kommentieren eines Programmcodes ist sowohl für den Programmierer, aber auch für andere, welche den Code lesen und verstehen wollen, wichtig. Ich habe den Code auf folgende Art und Weise kommentiert:

Codebeispiel Kommentare

```
//////////////////////////////////////////////////////
// Escape to Andromeda - Browsergame                //
// Ein Massive-Multiplayer-Online-Spiel             //
// Programmiert von Nicolas Perrenoud               //
// www.nicu.ch | mail@nicu.ch                       //
// als Maturaarbeit '04 am Gymnasium Oberaargau     //
// ------------------------------------------       //
// Datei: index.php                                 //
// Topic: Hauptseite                                //
// Version: 1.0                                     //
// Letzte Änderung: 29.10.2004                      //
//////////////////////////////////////////////////////

// Session-Cookie setzen
ini_set('arg_separator.output',  '&amp;');
session_start();

// Globale Variabeln definieren
$_GET = $HTTP_GET_VARS;
$_POST = $HTTP_POST_VARS;

// Funktionen und Config einlesen
include("conf.inc.php");
include("functions.php");

// Mit der DB verbinden und Config-Werte laden
dbconnect();
$conf = get_all_config();

// Zufallsgenerator initialisieren
mt_srand(time());

// IP-Blockierung
$ip_deny=array();
$ip_deny["1.2.3.4"]="Der Zugriff von dieser Adresse ist gesperrt!";
```

2.4.2 Schwierigkeiten und Lösungen

Anhand einiger Beispiele versuche ich im folgenden Abschnitt darzustellen, welche Probleme bei der Programmierung auftraten und wie ich sie löste.

2.4.2.1 Die Aktualisierung von Ereignissen

Eine der grössten Schwierigkeiten bei der Programmierung eines Browsergames ist die Aktualisierung von Ereignissen. Wenn man im Spiel z.B. irgend etwas bauen will, muss dieses Objekt zu einer bestimmten Zeit als fertiggestellt

gekennzeichnet werden. Das Spiel basiert aber nur auf Skript-Seiten, die vom Benutzer via Browser aufgerufen werden; es läuft keine spezielle Anwendung im Hintergrund, die Bauprozesse o.ä. überwachen würden. Man muss also in den Skriptseiten, die der Benutzer aufruft, immer am Anfang überprüfen, ob das Objekt bereits gebaut ist. Ich löste dieses Problem damit, dass bei einem Bauauftrag ein neuer Datensatz in die Datenbank geschrieben wird, mit Baustart- und Bauendzeit des Objekts. Beim Laden einer neuen Seite durch den Benutzer wird eine Funktion aufgerufen, die alle Gebäude, deren Bauendzeit kleiner als die aktuelle Zeit ist ladet, und sie als gebaut markiert. Da es viele verschiedene Objekte gibt, die gebaut werden können, musste ich aufgrund der Rechenkapazität darauf achten, diese Überprüfungsfunktion nur auf Seiten aufzurufen, wo sie relevant ist. Beim Beispiel Gebäudebau wird der Status auf folgenden Seiten überprüft: Bauhof, Technikbaum (Technische Voraussetzungen), Hauptübersicht, Schiffsbau, Verteidigungsbau, Forschung. Ebenfalls wird der Bau überprüft, wenn man von einem anderen Spieler angegriffen oder ausspioniert wird. Dasselbe gilt natürlich auch für das Berechnen der Rohstoffe und für die Flotten, die unterwegs sind.

2.4.2.2 Die Punkteaktualisierung

Es gab einen Bereich, wo ich die oben beschriebene Technik nicht anwenden konnte, und das war die Aktualisierung der Spielerpunkte. Im Spiel bekommt jeder Mitspieler Punkte für gebaute Objekte (Gebäude, Schiffe, Forschung etc). Es gibt eine Statistikseite, auf der man dann die Rangliste der Spieler sehen kann. Wenn man nun bei jedem Neuladen einer Seite oder beim Neuladen des Spielers die Punkte aktualisiert, funktioniert dass zwar für diesen Spieler, wenn man dann aber die Rangliste anschaut, stimmen die Punkte der anderen Spieler natürlich nicht mehr, da vielleicht ein Spieler mal zwei Tage nicht drin war und die Punkte nicht mehr aktuell sind. Es würde wohl den Server zu viel belasten, wenn man beim Aufrufen der Statistikseite die Punkte jedes Spielers neu berechnen müsste, und die Seite hätte eine extrem lange Ladezeit.
Ich löste das Problem mit Hilfe eines sog. Cron-Jobs. Der Cron-Dienst auf einem Linux-Webserver ist ein Programm, das andere Programme zeitgesteuert aufrufen kann. Die Programmiersprache PHP ist so flexibel, dass man damit auch

ein Skript machen kann, dass als „normales" Kommandozeilen-Programm aufgerufen werden kann. Der Cron-Daemon ruft nun jede Stunde ein Punkteaktualisierungs-Skript auf, welches unabhängig vom Benutzer alle Punkte berechnet und aktualisiert.

Codebeispiel Punkteupdate (Lange Zeilen werden hier auf mehrere Zeilen verteilt)

```
#!/usr/local/bin/php -q
<?PHP
  //
      // Diese Datei aktuallisiert die Punkte alle Benutzer
      // Die Datei wird auf einer Shell aufgerufen (via Cron-Job realisiert)
      // Sie wird jede Stunde aufgerufen
      //

      include("conf.inc.php");
      include("functions.php");
      dbconnect();
      $conf = get_all_config();

      //
      // Punkteberechnung
      //

      $ures = dbquery("SELECT user_id FROM ".$db_table['users'].";");
      while ($uarr=mysql_fetch_array($ures))
      {
            $user_id = $uarr['user_id'];
            $points = 0;
            $res_amount_per_point = $conf['points_update']['p1'];

            // Punkte für Schiffe (auf Planeten)
            $res = dbquery("SELECT
s.ship_costs_metal,s.ship_costs_crystal,s.ship_costs_fuel,s.ship_costs_plast
ic,s.ship_costs_food,sl.shiplist_count FROM ".$db_table['ships']." AS
s,".$db_table['shiplist']." AS sl WHERE s.ship_id=sl.shiplist_ship_id AND
sl.shiplist_user_id='".$user_id."';");
            while ($arr = mysql_fetch_array($res))
            {
            $points += $arr['shiplist_count'] * ($arr['ship_costs_metal'] +
$arr['ship_costs_crystal'] + $arr['ship_costs_fuel'] +
$arr['ship_costs_plastic'] + $arr['ship_costs_food']);
            }

            if ($points>0 && $res_amount_per_point)
                  $points = floor($points/$res_amount_per_point);
            dbquery("UPDATE ".$db_table['users']." SET user_points=$points
WHERE user_id='".$user_id."';");
      }

      dbquery ("UPDATE ".$db_table['config']." SET config_value=".time()."
WHERE config_name='statsupdate';");

      dbclose();

?>
```

2.4.2.3 Das Kampfsystem

Die grösste Herausforderung war die Programmierung eines Kampfsystems. Das Kampfsystem in meinem Spiel berechnet Kämpfe zwischen Flotten, die einen Planeten angreifen, und Flotten und Verteidigungsanlagen auf einem Planeten. Die Anforderungen an das Kampfsystem sind einerseits eine möglichst genaue und realistische Berechnung der Stärkeverhältnisse, andererseits aber eine möglichst kleine Belastung des Servers. Diese zwei Anforderungen sind schwer in Einklang zu bringen, darum musste ich das Kampfsystem nach den ersten Tests nochmals komplett überarbeiten, weil die Belastung des Prozessors bei Kämpfen mit vielen Schiffen einfach zu gross war.

Jedes Schiff und jede Verteidigungsanlage im Spiel haben fest definierte Werte für Angriffskraft und Verteidigungskraft (im Spiel „Struktur" & „Schutzschild" genannt). Das erste Kampfsystem lud alle Schiffsdaten aus der Datenbank, füllte ein sog. Array (Variable mit einer Reihe von gleichen Elementen) mit Daten zu jedem einzelnen Schiff, mischte das Array zufällig und brachte immer ein Element des Angreifers mit einem Element des Verteidigers zusammen. Danach wurden die Werte verglichen und je nach dem ein Objekt als zerstört markiert. Dies ging über mehrere Runden, bis alle Objekte am Kampf teilgenommen hatten. Es ist klar, dass z.B. bei Kämpfen mit mehreren tausend Einheiten die Performance des Servers darunter leidet. Dies hatte ich anfänglich zu wenig beachtet und schrieb darum ein einfacheres Kampfsystem: Im neuen Kampfsystem werden die Stärken aller Objekte zusammengerechnet und voneinander subtrahiert; die Anzahl der verbleibenden Objekte wird dann proportional zur anfänglichen Anzahl berechnet. Dieses System ist viel schneller, der Nachteil daran ist, dass der Zufallsfaktor keine Rolle mehr spielt, was das ganze zwar besser berechenbar macht, aber unrealistischer ist.

2.4.3 Ressourcenschonendes Programmieren

Besonders bei einem Online-Spiel, welches häufig auf einem gemieteten Server läuft und tausende von Mitspielern haben kann, ist es wichtig dass es möglichst wenig Rechnerressourcen braucht. Mit Rechnerressourcen ist die Belastung des Betriebssystems, des Prozessors und der Datenbank gemeint. Diese wird natürlich immer grösser, je mehr Spieler mitspielen und gleichzeitig auf den Server

zugreifen wollen. Daher muss der Programmcode so programmiert sein, dass er möglichst einfach verarbeitet werden kann. Es ist oft eine Art Gratwanderung zwischen Ressourcen sparen und möglichst viele Features ins Spiel einbauen. So musste ich schlussendlich auch auf einige von Mitspielern gewünschte Features verzichten, weil sie einfach zuviel Rechenkapazität verlangt hätten.
Einige wichtige Punkte zum ressourcenschonenden Programmieren sind mir wichtig geworden (diese Punkte sind sehr spezifisch für Webdesign/Serverskriptsprachen):

- *Datenbankabfragen optimieren*

  Datenbankabfragen kann man optimieren, wenn man z.B. genau angibt, welche Daten man braucht. Häufig hat eine Tabelle sehr viele Felder und man braucht nur die Daten aus einigen wenigen. Man muss zwar beim Programmieren mehr Aufwand betreiben, um diese genau anzugeben, aber die Datenbank wird weniger belastet.
- *Cookies verwenden*

  Cookies sind temporäre Daten, die vom Browser auf dem Rechner des Benutzers abgelegt werden. Ich arbeite in meinem Spiel sehr viel mit Cookies. Ich brauche sie nicht nur zur Authentifizierung des Benutzers, sondern auch zum Abspeichern der Daten des aktuell ausgewählten Planeten, der aktuellen Allianz des Benutzers und anderer benutzerspezifischer Daten, die sonst bei jedem Seitenaufruf neu geladen werden müssten.

2.5 Testphase

2.5.1 Eigene Tests

Währen dem Programmieren überprüft man logischerweise immer wieder das gerade bearbeitete Objekt auf grobe Fehler. Häufig ist es aber so, dass man vor allem Überlegungsfehler nicht bemerkt. Darum testete ich das Spiel nach der Hauptarbeit des Programmierens ganz durch. In diesem Status lag das Spiel nun als Alpha-Version vor.

> *Die erste lauffähige Version eines Computerprogramms wird gerne Alpha-Version genannt. Der Begriff ist nicht exakt definiert, in der Regel enthält eine Alpha-Version bereits einige wichtige Bestandteile eines Softwareprodukts - es ist aber wahrscheinlich,*

dass in späteren Versionen der Funktionsumfang noch erweitert wird. Insbesondere enthält eine Alpha-Version mit großer Sicherheit viele Bugs.
(vgl. http://www.wikipedia.de/wiki/Alpha-Version, 24.10.04)

Diese Alpha-Version testete ich auf einem Webserver auf meinem lokalen Computer. Ich überprüfte alle relevanten Module auf Fehler und überzeugte mich, dass keine Grafiken fehlten. Solche Test reichen aber meistens nicht aus, da es als Programmierer immer schwierig ist, sich in die Situation des Benutzers zu versetzen, der noch kein Vorwissen über das Programm hat. Gerade bei meinem Spiel, welches ja darauf basiert, dass viele menschliche Spieler miteinander spielen, können während den eigenen Tests viele Sachen übersehen werden.

2.5.2 Der Beta-Test

2.5.2.1 Was ist ein Beta-Test?

Ein Beta-Test ist ein Test eines Programms durch externe Benutzter. Ein Beta-Test kann entweder öffentlich sein oder nur mit „eingeladenen" Benutzern durchgeführt werden. Das Programm nennt man in diesem Zustand Beta-Version.

Häufig sind Beta-Versionen die ersten Versionen eines Programms, die vom Hersteller zu Testzwecken veröffentlicht werden. Der Begriff ist nicht exakt definiert, als Faustregel zur Abgrenzung einer Beta-Version von anderen Versionen gilt in der Regel, dass das Programm vollständig ist (im Sinne, dass der Funktionsumfang in nächster Zeit nicht mehr erweitert wird), aber vermutlich noch Fehler enthält. Dann wird die Software manchmal gern Release Candidate (RC) genannt.
(vgl. http://www.wikipedia.de/wiki/Beta-Version, 24.10.04)

Ich führte mit meinem Spiel einen öffentlichen Beta-Test durch. Der Beginn des Beta-Tests war für mich eines der spannendsten Erlebnisse dieser Arbeit, da sich nun zeigen sollte, ob das Spiel gut programmiert ist und das Konzept die Spieler anspricht. Ich definierte einen genauen Zeitpunkt, ab wann das Spiel starten würde und schaltete die Anmeldung bereits ein paar Tage vor Spielbeginn ein, so dass beim Start bereits viele Spieler angemeldet waren und sofort loslegen konnten.

2.5.2.2 Wie macht man das Spiel bekannt?

Es stellte sich mir die Frage, wie man denn überhaupt Spieler für einen Beta-Test gewinnt. Ich probierte verschiedene Arten des „Werbung-machens" aus und hoffte, so möglichst viele Leute anzusprechen:

- *Rundmail an Bekannte*
  Ich schickte allen Leuten, die ich kannte und wusste, dass sie sich für so ein Spiel interessieren würden, ein E-Mail mit einer kurzen Beschreibung des Spiels und versuchte, sie für's Mitmachen zu begeistern
- *Werbung auf Webseiten*
  Bezahlte Werbung auf einer Webseite konnte ich mir nicht leisten. Es gibt aber viele Browsergame-Fanseiten im Internet, wo man sein Spiel vorstellen und bewerten lassen kann. Ich meldete das Spiel, sobald der Beta-Test am laufen war, bei solchen Seiten an, um auch ausserhalb vom Kollegenkreis darauf aufmerksam zu machen.
- *Werbung in Online-Foren*
  Was auch immer sehr gut ist, ist die Platzierung eines Links in einem gut besuchten Szenen-Internetforum. So kann man die Zielgruppe sehr gut ansprechen und kann auf eventuell auftretende Fragen zum Spiel direkt antworten.
- *Mund-zu-Mund Propaganda.*
  Ich stellte einmal mehr fest, dass Mund-zu-Mund Propaganda die beste Werbemöglichkeit ist. Viele Leute meldeten sich beim Spiel an, weil sie von einem Kollegen den Tipp bekommen haben. Es ist darum sehr wichtig, dass der Ersteindruck des Spieles gut ankommt und die Spieler nicht mit Fehlermeldungen überhäuft werden, sonst gibt es bald mal schlechte Kritik und weniger Leute, die mitspielen.

2.5.2.3 Beurteilung des Beta-Tests

Bereits nach einer Woche konnte ich sagen, dass sich der Beta-Test überaus gelohnt hat. Man bekommt einerseits ein Feedback zum Spiel, findet und korrigiert Fehler, die einem selber nie aufgefallen wären (da manche Spieler extrem kreativ beim Ausprobieren und Fehlersuchen sind) und bekommt auch viele Verbesserungs- und Erweiterungsvorschläge. Wichtig ist, dass man den

Spielern eine Möglichkeit gibt, sich auszutauschen und Vorschläge zu diskutieren. Aus diesem Grund richtete ich ein Online-Forum zum Spiel ein. Was mich enorm erstaunte war, wie viele Spieler sich für das Spiel begeisterten. Selbst Kollegen, von denen ich gedacht hätte, dass sie nie solch ein Spiel spielen würden, meldeten sich an und spielten eifrig mit. Knapp zwei Wochen nach Beginn des Beta-Tests hatte ich über 270 Mitspieler. Der Beta-Test zeigt auch auf, wie fest die Ressourcen des Servers beansprucht werden. Alleine hat man ja nicht die Möglichkeit, das zu testen und kann höchstens Hochrechnungen anstellen. In Absprache mit dem Webspace-Anbieter legte ich eine Mitspieler-Limite ein, damit der Server auf alle Fälle nicht plötzlich aufgrund der vielen gleichzeitigen Zugriffe überlastet wird.

2.6 Das fertige Spiel

Ich frage mich, ob man eine Software je „fertig" nennen kann. Man kann zwar schon einen Punkt setzen und sagen: „So, dass ist das fertige Produkt", aber mit der Zeit hat man immer wieder neue Ideen, was man noch alles hinzufügen und verbessern könnte. Vor allem bei einem solchen Spiel, wo die Gemeinschaft der Mitspieler derart involviert ist, gibt es immer wieder neue Vorschläge. Die Herausforderung dabei ist, aus der Fülle von Ideen die richtige auszuwählen, damit sowohl starke als auch schwache Spieler nicht benachteiligt werden.
Ich möchte hier nun einen kurzen Gesamtüberblick über das Spiel geben, auch wenn ein Text viel weniger aussagt als wenn man das Spiel selbst spielt:

2.6.1 Spielablauf

Das Spiel läuft rund um die Uhr und alle Spieler spielen miteinander. Man meldet sich auf der Startseite für das Spiel an. Dabei wählt man einen Nicknamen (ein Pseudonym, unter dem man im Spiel identifiziert wird) und entschiedet sich für ein Volk. Jedes Volk hat gewisse Vorteile, z.B. einen erhöhten Abbau von Rohstoffen oder kürzere Bauzeiten. Wenn man sich das erste Mal anmeldet, hat man bereits einen Planeten mit Rohstoffen und kann anfangen, die ersten Gebäude zu bauen. Man muss Minen und Fabriken bauen, um verschiedene Rohstoffe zu gewinnen, und Technologiegebäude, um z.B. Schiffe zu bauen oder Technologien zu erforschen. Wenn man dann Schiffe hat, kann man mit

anderen Mitspielern Waren handeln, neue Planeten besiedeln, um sein Imperium zu vergrössern oder gegen anderen Spielern Krieg führen. Man kann sich mit anderen Spielern auch zu Allianzen zusammenschliessen, um einander zu unterstützen. Allianzen entwickeln oft eine Dynamik, sie schliessen Bündnisse mit anderen Allianzen oder führen als komplette Allianz Krieg gegen andere, sie bauen eigene Kommunikationskanäle auf und sprechen Aktionen untereinander ab. Dadurch, dass man sehr viel erforschen und bauen kann und die Kommunikation untereinander sehr wichtig ist, wird das Spiel für viele lange interessant bleiben.

2.6.2 Features des Spiels

- Bau von Gebäuden, Schiffe, Verteidigungsanlagen
- Erforschung von Technologien
- 2-Dimensionale Raumkarte
- Technologiebaum
- Produktionsübersicht
- Nachrichtensystem aufgeteilt in verschiedene Kategorien
- Allianzsystem mit Infoseite, Rängen, Massenmail und Allianzlogoanzeige
- Spieler- und Allianzstatistiken
- Infoseite für jeden Spieler
- Einbindung eines Bildpakets, damit eigene Bilder anstelle der Originalbilder im Spiel angezeigt werden können
- Urlaubsmodus
- Hilfefunktion mit Infotabellen zu allen Objekten
- Diskussionsforum

2.6.3 Wie geht es weiter?

Das Spiel läuft nun seit Beginn der BETA-Phase unter der Internetadresse ***www.etoa.ch***. Es wird auf Wunsch der Mitspieler sicher bis auf weiteres am Laufen bleiben. Ich werde aber abklären, ob das Bedürfnis nach einem Neustart des Spieles vorhanden ist, damit alle fairere Anfangsbedingungen haben.

3 Zusammenfassung

Dieses Projekt hat mir, neben viel Arbeit, auch sehr viel Spass gemacht und ich konnte viel dabei lernen. Einerseits konnte ich sicher meine Kenntnisse über Datenbanken und über die Skriptsprache PHP erweitern, da dies mein erstes Projekt in einem so grossen Umfang ist. Zum anderen merkte ich auch, dass man den Zeitaufwand für das Programmieren oft unterschätzt, und dass es ziemlich viel Disziplin und Konzentration braucht, um dranzubleiben.

Ich habe mich durch diese Arbeit auch viel besser in die Lage von „professionelleren" Programmierern einfühlen können: als Benutzer eines Programms sieht man oft nur die schlechten Seiten, hat überall etwas auszusetzen und denkt sich, dass die Entwickler vorwärtsmachen sollten mit der Behebung von Fehlern; ich merkte dann ziemlich schnell, was für ein stressiger und nicht immer einfacher Job das ist. Vor allem während den ersten Wochen des Beta-Tests stand ich ziemlich unter Druck, da einerseits Fehler behoben werden mussten, und andererseits von den Spielern immer neue Funktionen gewünscht wurden.

Auch bei der Umsetzung des Detailkonzepts konnte ich viel lernen. Wie man sieht, konnte ich nicht alles, was im Detailkonzept vorgesehen war, umsetzen weil es zu kompliziert war, ich konnte aber wiederum andere Funktionen einbauen, die nicht im Konzept standen.

Ich habe festgestellt, dass die Meinung und Mitarbeit der Community, also der Mitspieler, sehr wichtig ist und oft unterschätzt wird.

Zusammengefasst kann ich sagen, dass die letzten Monate eine sehr lehrreiche Zeit für mich waren, mit vielen Herausforderungen, aber auch vielen guten Erfahrungen.

Ich hoffe, dass Sie, liebe Leserin, lieber Leser, nun ein wenig mehr über die Entwicklung eines Computerspiels und die damit verbundenen Schwierigkeiten und Erfahrungen wissen .

4 QUELLEN

Ich verwendete bei diesem Projekt ausnahmslos Internetquellen für die Beschaffung von Informationen. Sämtliche im Bericht verwendeten Screenshots sind von mir selbst erstellt worden.

4.1 Internetquellen

4.1.1 Informationen

Wikipedia – Die freie Enzyklopädie	http://www.wikipedia.de

4.1.2 Programmierhilfen

SelfHTML	http://selfhtml.teamone.org
SelfPHP	http://www.selfphp.de
PHP-Manual	http://www.php.net/manual/de
MySQL-Dokumentation	http://www.mysql.com/doc

4.1.3 Verwendete Software

UltraEdit 32 TextEditor	http://www.ultraedit.com
Adobe Photoshop	http://www.adobe.com/photoshop
LunarCell & SolarCell Plugin für Photoshop	http://www.flamingpear.com
Mozilla Firefox Browser	http://www.mozilla.org
Apache Httpd Server	http://httpd.apache.org
PHP	http://www.php.net
MySQL Server	http://www.mysql.com
PHPMyAdmin	http://phpmyadmin.sourceforge.net

5 ANHANG

5.1 Über Datenbanken und Programmiersprachen

5.1.1 Was ist eine Serverskriptsprache? Wie funktioniert sie?

Serverskriptsprachen erfreuen sich einer immer grösseren Beliebtheit. Man verwendet sie bei der Gestaltung von Internetangeboten wie Webseiten, Online-Shops, Foren und anderen Webapplikationen. Diese Skriptsprachen werden auf einem Webserver ausgeführt und machen, dass eine Website dynamischen Inhalt haben kann. Ein einfaches Beispiel ist ein Gästebuch auf einer Homepage: Man erstellt einen Beitrag, und dieser wird entweder in einer Textdatei oder in einer Datenbank auf dem Server gespeichert. Beim Aufrufen des Gästebuchs lädt das Skript auf dem Server diese Daten und generiert daraus z.B. eine reine HTML-Seite, die an den Browser gesendet wird. Skriptsprachen können aber noch viel mehr: Sie können Programme auf dem Server aufrufen, komplizierte Berechnungen durchführen, Benutzereingaben überprüfen, Mails senden, etc. Die am meisten verwendeten Serverskriptsprache ist PHP[7] (Personal Home Page Tools - Hypertext Preprocessor), eine Open-Source[8] Programmiersprache, deren Syntax an C und Perl angelehnt ist. PHP ist relativ einfach zu erlernen und bietet eine Fülle von unzähligen Funktionen für fast jeden Verwendungszweck. Der grosse Vorteil einer solchen Sprache ist, dass beim Client (meistens ein Browser) keine speziellen Fähigkeiten erforderlich sind und auch keine Inkompatibilitäten auftreten können, wie es z.B. bei Javascript (aufgrund Nichteinhalten der Standards bei den verschiedenen Browsern) der Fall ist. Der Quelltext bleibt ausserdem auf dem Server, nur der generierte Code ist für den Besucher einsehbar; dies ist vor allem für sicherheitsrelevante Dienste von grosser Wichtigkeit. Gleiches gilt für andere Ressourcen wie z.B. Datenbanken, die daher auch keine direkte Verbindung zum Client benötigen.

---

[7] www.php.net

[8] Software, deren Quelltext frei erhältlich ist, dieser darf auch selbst verändert und weitergegeben werden.

Ein Nachteil ist, dass jede Aktion des Benutzers erst bei einem erneuten Aufruf der Seite erkannt wird; ebenfalls wird bei jedem Aufruf die Seite neu generiert, auch wenn der Inhalt derselbe wäre; dies führt zu höherer Serverbelastung. Weitere Serverskriptsprachen sind ASP von Microsoft, die auf Visual Basic basiert, und CGI/Perl, eine Sprache die vor allem auf Unix/Linux Systemen verbreitet ist.

5.1.2 Funktion und Aufbau einer Datenbank

Eine Datenbank ist eine geordnete Datensammlung. Die Daten sind in verschiedenen Tabellen gespeichert. Man kann sich eine Datenbank als einen elektronischen Karteikasten vorstellen. Das Grundelement jeder Datenbank ist der Datensatz (er entspricht einer Karteikarte). Mehrere Datensätze bilden zusammen eine Tabelle. Die Daten einer Datenbank sind nach bestimmten Merkmalen und Regeln erfasst, geordnet und abgelegt. Hier noch ein einfaches Beispiel dazu:
Eine Datenbank in einem kleinen Geschäft könnte folgende Tabellen haben:

- Kunden
- Produkte
- Produkt-Kategorien
- Mitarbeiter

Die Tabelle hat meistens mehrere sog. Felder. Bei der Produkt-Tabelle könnten es z.B. folgende Felder sein:

- ID
- Name
- Preis
- Beschreibung
- Kategorie ID

Die meisten Tabellen in einer Datenbank haben ein ID-Feld, mit dem der jeweilige Datensatz angesprochen werden kann.
In diesem Beispiel hat die Tabelle Produkte noch ein Feld „Kategorie ID"; dies zeigt eine weitere Eigenschaft vieler Datenbanken: Tabellen können miteinander verknüpft werden und man kann mit gezielten Abfragen Daten auf effiziente Art herauszulesen.

Viele Datenbanken haben auch ein Benutzermanagement mit Rechtvergabe, so dass es z.B. einem Benutzer nur gestattet ist, Daten zu lesen, einem anderen ist es erlaubt, die Daten auch zu ändern.

Bei meinem Projekt verwendete ich das Datenbanksystem MySQL[9]. Es ist im Bereich des Internets eines der am weitesten verbreiteten Systeme, da es u.a. auch Open-Source ist. Die Datenbank läuft hierbei auf einem Server, welcher mit entsprechenden Client-Programmen oder mit einer Programmiersprache angesprochen werden kann.

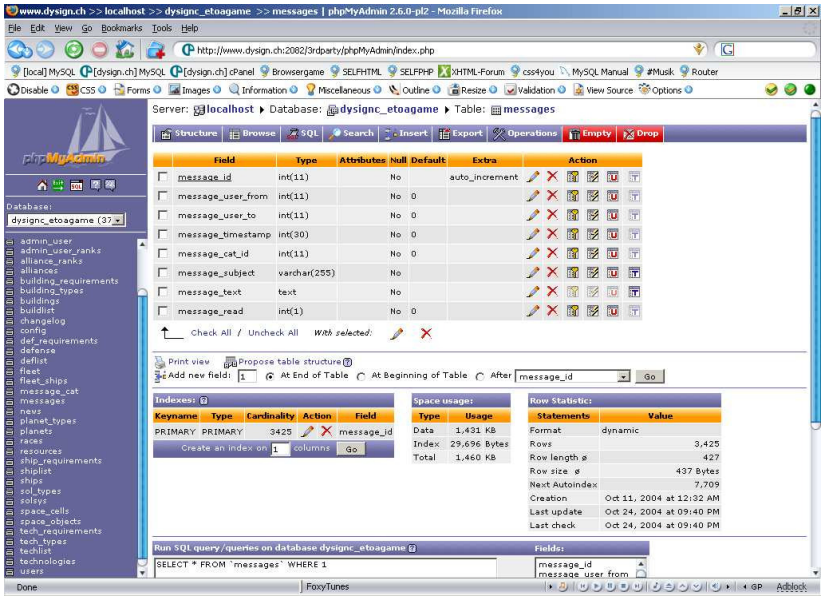

Abb. 2:
PHPMyAdmin, ein sehr populäres Verwaltungsprogramm für MySQL Datenbanken

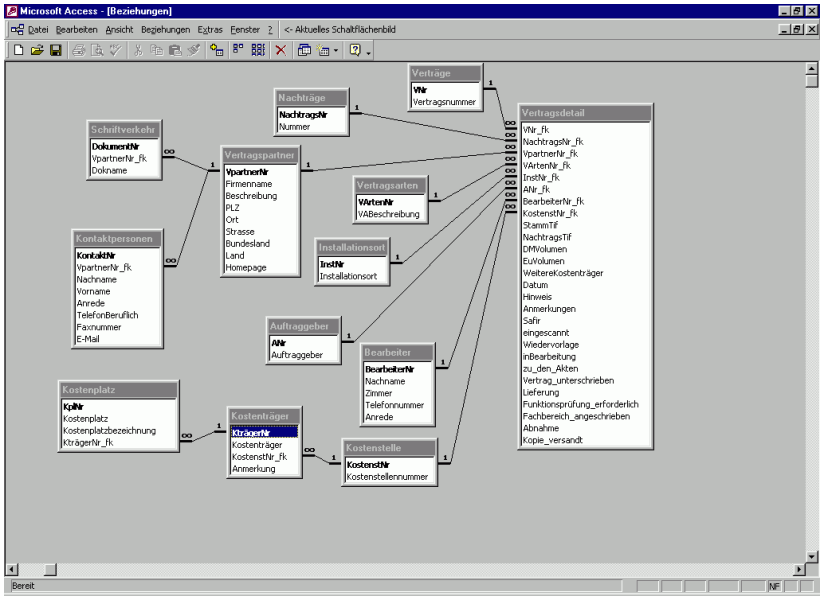

Abb. 3:
Datenbanklayout und Beziehungen der Tabellen in einer Access-Datenbank.

[9] www.mysql.com

5.2 Detailkonzept

5.2.1 Über die Idee des Spiels

5.2.1.1 Geschichte

Im Jahre 10789 nach Erdzeit-Rechnung sucht eine schreckliche Katastrophe die gesamte Milchstrassen-Galaxie heim: Durch das gleichzeitige Kollabieren vieler Sterne im Zentrum der Galaxie entsteht ein riesiges schwarzes Loch, welches die Stabilität der ganzen Galaxis durcheinanderbringt und viele Sternensysteme zerstört. Die Menschheit, und auch viele andere Rassen, die die Galaxie bevölkern und viele Planeten kolonialisiert haben, entscheiden sich für ein waghalsiges Projekt: Sie wollen alle ihre Bewohner, deren Planeten noch nicht zerstört oder vom schwarzen Loch aufgesogen worden sind, in die Nachbargalaxie Andromeda evakuieren und dort eine neue Zivilisation aufbauen. Die Flüge bis dorthin dauern viele Lichtjahre, und noch fast nie war ein Wesen unserer Galaxie so weit weg geflogen. So machen sich nun viele verschiedene Gruppen auf den Weg, um in der Galaxie Andromeda eine neue, fruchtbare Heimat zu finden und dort ein neues Imperium aufzubauen.
An dieser Stelle steigen die Spieler ins Spiel ein. Und es beginnen auch schon die Probleme: denn es hat nicht unendlich viel Platz in Andromeda, und jede aufsteigende Zivilisation benötigt immer mehr Ressourcen. Es wird schon bald zu Krieg kommen zwischen den verschiedenen Spielern und ganzen Bündnissen, und die Zukunft wird zeigen, wer genug stark ist, sich die Herrschaft über Andromeda zu sichern.

5.2.1.2 Ablauf des Spiels

Am Anfang des Spiels hat jeder Spieler eine kleine Flotte, die sich im leeren Raum befindet. Er kann sich nun einen freien Planeten in einem Sonnensystem in seiner Nähe aussuchen, die Flotte dort hinschicken und seine erste Basis aufbauen. Danach produziert er Rohstoffe, baut Flotten, erforscht neue Technologien, kolonialisiert neue Planeten, und handelt oder führt Krieg mit / gegen andere Spieler.

5.2.1.2.1 Rohstoffe

Tab.1: Rohstoffarten

Name	Typ	Beschreibung
Titan	Metall	Titan wird zum Bau von Strukturen für Raumschiffe und Gebäude benötigt.
Silizium	Kristall	Silizum wird zum Bau von elektronischen Steuerungen verwendet.
PVC	Plastik	PVC ist ein Basisstoff für den Bau von Raumschiffen und Gebäuden.
Tritium	Treibstoff	Tritium ist ein Isotop des Wasserstoffes und wird zum Antrieb von Raumschiffen verwendet.
Nahrung	Nahrung	Nahrung erhält die Crew und die Arbeiter am Leben.

5.2.1.2.2 Allianzen

Allianzen sollen das Zusammenspiel unter den Mitspielern fördern. Mitglieder einer Allianz können sich nicht angreifen. Alle Punkte, die ein Spieler macht, werden durch 1000 geteilt und werden der Allianz als Allianzpunkte in der Statistik zugerechnet.

Eine Allianz kann mit anderen Allianzen Kriege führen, einen Waffenstillstand, ein Bündnis oder ein Handelsabkommen abschliessen. Ebenfalls kann sich die Allianz intern organisieren, ein Forum zum Austausch untereinander betreiben und ein internes Rangsystem aufbauen. Allianzmitglieder können z.B. auch einem schwachen Spieler beim Aufbau helfen. Alle diese Features werden nicht in das Spiel implementiert, sondern werden von den Allianzen individuell umgesetzt.

5.2.1.3 Ziel des Spiels

Über das Spielziel bin ich mir noch nicht ganz im Klaren, ich sehe da verschiedene Möglichkeiten:

- Unendlich; das Spiel läuft immer weiter, Ziel eines jeden Spielers ist es, in der Statistik möglichst gut zu werden (läuft bei vielen Browsergames so ab)
- Herrschaft über die Galaxie; die Allianz, die eine gewissen Prozentsatz an Planeten (z.B. mind. 30 % aller existierenden und 60 % aller bewohnten Planeten) in der Galaxie besitzt, gewinnt das Spiel und es beginnt eine neue Runde.

- Master-Spieler besiegen. Es gibt extrem starken Spieler (vom System gesteuert), der besiegt werden muss. Wenn er besiegt ist, fängt eine neue Runde an.

5.2.2 Funktionsweise des Programms

Generell funktioniert das Programm so, dass alle änderbaren Daten in einer Datenbank gespeichert sind (MySQL Datenbank) und diese Daten werden ausschliesslich durch Aktionen des Spielers geändert. Durch Aufrufen einer Seite im Browser startet diese ein Skript auf dem Server, welches Daten in die Datenbank schreibt oder ausliest und anschliessend die Seite generiert und an den Browser sendet. Da ich keinen eigenen, im Internet erreichbaren Webserver habe, den ich individuell konfigurieren kann und deshalb direkt keine Programme auf diesem laufen lassen kann, muss ich das ganze Projekt mit eben diesen serverbasierten Skripts realisieren. Als Skriptsprache verwende ich PHP.

5.2.2.1 Gebäude, Forschung, Schiffbau etc.

In einer Tabelle sind alle Gebäudedaten gespeichert. In einer zweiten Tabelle werden alle Gebäude, welche die Spieler bauen bzw. gebaut haben, registriert. Es wird auch gespeichert, auf welcher Stufe / Anzahl das Gebäude ist. Wenn das Gebäude im Bau ist, wird die Anfangs- und die errechnete Endzeit gespeichert. Wenn nun ein Spieler die Gebäudebau-Seite abruft, wird aus diesen Daten errechnet, welche Gebäude gebaut sind und wenn eines im Bau ist, wie lange es noch dauert. Wenn die Endzeit überschritten ist, wird das Gebäude als gebaut gekennzeichnet, indem die Start- und Endzeit gelöscht wird und die Stufe bzw. Anzahl um 1 erhöht wird. Die Bauzeit der Gebäude wird durch eine Grundzeit für jede verbaute Tonne Material, einem universalen Geschwindigkeitsfaktor und Faktoren von Sonne, Planeten und Rasse berechnet. Je höher die Ausbaustufe der Gebäude, desto mehr kosten sie und desto länger geht der Bau.

5.2.2.2 Flottenflug

Für Schiffe gibt es genau gleich wie für Gebäude eine Tabelle mit der Anzahl Schiffe eines Typs des entsprechenden Spielers pro Planet. Wenn nun eine Flotte von einem Planet zu einem anderen geschickt wird, wird die Abflugzeit gespeichert und die Ankunftszeit ausgerechnet. Die Geschwindigkeit der Schiffe hängt von der Art des Antriebs ab. Je höher der Spieler den Antrieb entwickelt hat, desto schneller fliegen die Schiffe.

5.2.2.2.1 Basisgeschwindigkeiten

Linke Spalte: Antriebstyp

Oberste Zeile: Galaxie-Zellentyp (Eine Zelle hat eine Länge / Breite von 300 AE)

Die Zahl bedeutet AE / Stunde

Verhältnisse der Antriebe: Ionen ist ¾ mal so schnell wie Rückstoss, Warp ist 1.5 mal so schnell wie Rückstoss. Mit der Weiterentwicklung der Technologie wird die Geschwindigkeit immer um 10% gesteigert.

Tab.2: Grundgeschwindigkeiten der Antriebe abhängig vom Zellentyp

Antrieb	Sonnensystem	Nebel	Asteroidenfeld	Leerer Raum
Rückstossantrieb	1000	1500	800	3000
Ionenantrieb	750	1125	600	2250
Warpantrieb	1500	7500	1600	6000

5.2.2.2.2 Basisverbrauch

Die folgende Tabelle beschreibt den Basisverbrauch des Treibstoffs Tritium pro AE. Mit der Weiterentwicklung des Antriebes und der daraus resultierenden höheren Geschwindigkeit nimmt der Verbrauch an Tritium auch zu. Es wird aber die Möglichkeit geben, Flotten nicht mit der vollen Geschwindigkeit fliegen zu lassen um Treibstoff zu sparen.

Tab.3: Grundkosten des Antriebs

Antrieb	Tritium / 100 AE
Rückstossantrieb	10
Ionenantrieb	8
Warpantrieb	15

Für jede Flotte wird eine ID vergeben; und da jeder Schifftyp ein eigener Datensatz ist, sind dann alle Schiffe mit der selben ID eine Flotte. Schiffe desselben Typs, die auf dem Planeten verbleiben, bleiben im Datensatz gespeichert, für Schiffe die fliegen, wird ein neuer Datensatz erstellt.

5.2.2.3 Kampfsystem

Das Kampfsystem wird so funktionieren, dass die Spieler Flotten bauen und einander damit angreifen können. Man kann nur Planeten angreifen. Die darauf stationierte Flotte verteidigt automatisch den Planeten. Ebenfalls können stationäre Verteidigungsanlagen gebaut werden. Jedes Objekt hat Struktur-, Schild- und Waffenpunkte, welche durch Forschung erhöht werden können. Bei einem Kampf werden Einheiten zufällig miteinander konfrontiert, danach werden je die Waffenpunkte der einen Einheit von den Schild- und Strukturpunkten der anderen Einheit abgezogen. Die Einheit, welche am Schluss noch Strukturpunkte hat, gewinnt. Der Kampf kann mehrere Runden dauern. Wenn die angreifende Flotte gewinnt, kann sie 50% der auf dem Planeten gelagerten Rohstoffe klauen, vorausgesetzt, soviel hat in den Laderäumen Platz.

5.2.3 Layout

Da das Spiel mit Hilfe eines gewöhnlichen Internet-Browsers gespielt werden kann, sind dem Layout gewisse Grenzen gesetzt. Ich stelle mir den Aufbau etwa so vor:

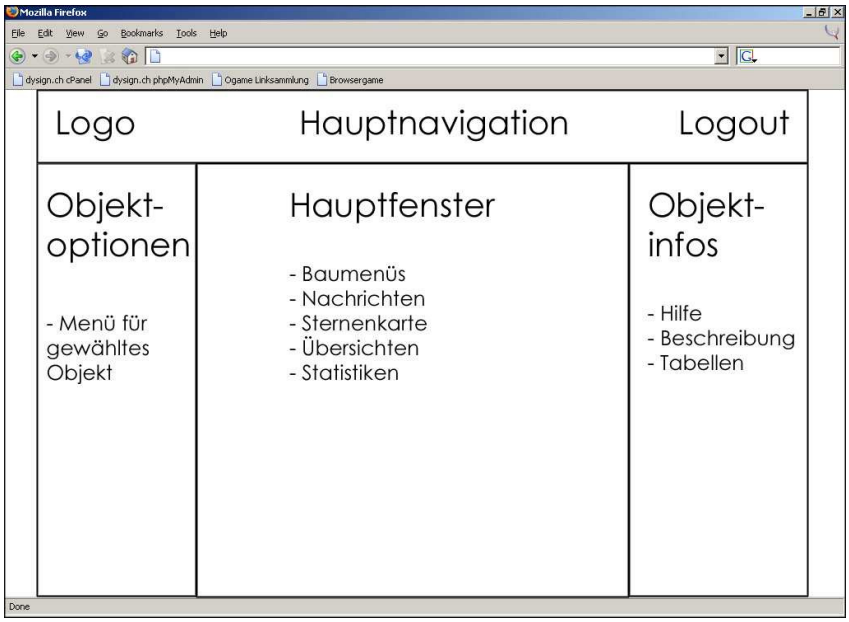

Abb. 4:
Grundlayout des Spiels

Der Seitenaufbau besteht aus einem Frameset mit einem Top, einem Left und einem Right-Panel und einem Quadratischen Inhalts-Frame. Auf der Seite sind beliebig skalierbare leere Frames.

5.2.3.1 Die genaue Anordnung der Frames

Das Spiel ist gedacht für eine Bildschirmauflösung von 1024*768 (Von mir aufgezeichnete Statistiken haben ergeben, dass 75% der Surfer (aufgezeichnet auf einer Seite speziell für Browsergame-Spieler) diese Auflösung verwenden, daher richte ich das Spiel auf diese Auflösung aus). In der Breite ist das ganze Layout 900px lang, und in der Höhe 605px; der Raum aussen herum bleibt frei skalierbar, damit Leute mit einer anderen Auflösung das Spiel immer noch zentral auf dem Bildschirm haben. Der Top-Frame mit dem Logo und der Hauptnavigation ist 100px hoch, der untere Teil damit noch 550px. Das Hauptfenster ist ebenfalls 550px breit, d.h. quadratisch, die linke Seite ist 200px breit und die rechte 150px.

5.2.3.2 Sternenkarte

Die Sternenkarte zeigt immer den aktuellen Sektor. In diesem Sektor gibt es 10 * 10 Zellen, jede Zelle ist 40px * 40px gross.

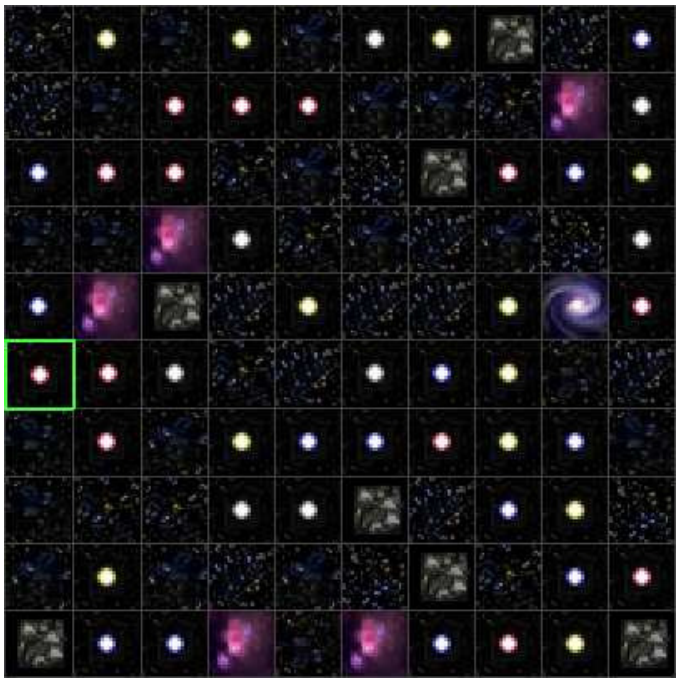

Abb. 5:
Beispiel der Sternenkarte

5.2.4 Datenbankstruktur

Tab.4: Datenbanktabellen

Tabelle	Beschreibung
alliance_ranks	Ränge innerhalb der Allianz
alliances	Allianzdaten
building_types	Gebäudetypen
buildings	Gebäude
buildling_requirements	Gebäudeabhängigkeiten
buildlist	Liste der gebauten Gebäude für jeden Spieler
changelog	Änderungsverzeichnis
config	Generelle Konfigurationseinstellungen
def_requirements	Verteidigungsabhängigkeiten
defense	Verteidigungsanlagen
deflist	Liste der gebauten Verteidigung für jeden Spieler
fleet	Flottendaten
fleet_ships	Schiffsdaten zu jeder Flotte
messages	Nachrichten
messages_cat	Nachrichtenkategorien
news	Neuigkeiten
planet_types	Planeten-Typen
planets	Planeten
races	Rassendaten
ressources	Ressourcendaten
ship_requirements	Schiffsabhängigkeiten
shiplist	Liste der existierenden Flotte für jeden Spieler
ships	Schiffe
sol_types	Sternen-Typen
space_cells	Zellen im Koordinatensystem
space_objects	Daten zu den Objekten (Asteroidenfelder, Nebel)
tech_requirements	Technologieabhängigkeiten
techlist	Liste der erforschten Technologien für jeden Spieler
technologies	Forschungen
User	Benutzerdaten

5.2.5 Programm-/Verzeichnisstruktur

Alle relevanten Seiten für das Spiel, die im Inhaltsframe aufgerufen werden, werden durch eine main-Datei (main.php) geladen. Damit muss man nicht für jede Seite den Anfang- und Schlussteil schreiben. Die Verzeichnisstruktur sieht folgendermassen aus:

Verzeichnisstruktur	
/	Hauptverzeichnis; dort sind: Index-, Config-, Functions und Layoutdateien.
/content	Alle Dateien, die im Inhaltsframe geladen werden
/images	Ordner für Bilder
/style	Alle Bilder, die direkt mit dem Spiel zu tun haben
/galaxy	Bilder für die Sternenkarte
/planets	Planetenbilder
/ships	Bilder der Schiffe
/defense	Bilder der Verteidigung
/technoligies	Bilder der Forschung
/buildings	Bilder der Gebäude
/info	Alle Dateien, die im Infoframe rechts geladen werden
/index	Dateien, die auf der Startseite verwendet werden

5.2.6 Regeln für die Namensgebung

- Variabeln werden englisch benannt und kleingeschrieben
- Datenbanktabellen und -felder werden englisch benannt und kleingeschrieben
- Standard für HTML ist XHTML 1.0, für Stylesheets CSS 1
- Konstanten werden englisch benannt und alle Zeichen grossgeschrieben

5.2.7 Objekte

5.2.7.1 Galaxie, Sektoren

Die Galaxie ist die oberste Ebene in der Raumansicht. Sie ist unterteilt in Sektoren, welche mit einem Koordinatensystem definiert werden. Jeder Sektor enthält wieder mehrere Objekte (Zellen). Dies können Sonnensysteme, interstellare Nebel, Asteroidenfelder, Wurmlöcher oder ganz einfach leerer Raum sein.

Standardmässig gehe ich von 10 * 10 Sektoren aus, jeder Sektor besteht wieder aus 10 * 10 Zellen. Jede Zelle hat im Spiel eine Seitenlänge von 300 AE (dass ist etwa der Durchmesser unseres Sonnensystems) und damit eine Fläche von 90'000 AE^2. (1AE = Astronomische Einheit = Abstand Erde-Sonne = 1,49597870691 $\times 10^{11}$ m).

Die Wahrscheinlichkeit, dass eine Zelle ein Sonnensystem ist, ist 40%, ein Asteroidenfeld ist 7%, ein Nebel 6%, und ein Wurmloch hat 2% Wahrscheinlichkeit.

5.2.7.2 Sonnensysteme, Sterne

Jedes Sonnensystem besteht aus einem Stern und mehreren Planeten. Es gibt verschiedene Typen von Sternen, welche einen Einfluss auf die Produktionsfähigkeit der jeweiligen Planeten haben. Jeder Stern hat einen Namen (von existierender Liste kopiert)

Tab.5: Sternentypen

Sternentyp	Spezialität
Gelber Stern	Nahrungsproduktion * 1.5
Roter Stern	Bewohnerwachstum * 1.5
Blauer Stern	Gebäudebauzeit * 0.75
Weisser Stern	Energieproduktion * 1.5

Die Menge der Planeten pro Sonnensystem ist zufällig und reicht von 5 bis 15.

5.2.7.3 Planeten

Planeten sind die Grundsteine eines jeden Imperiums. Auf einem Planet kann der Spieler Gebäude zur Rohstoffgewinnung bauen, unbekannte Technologien erforschen und Raumschiffe bauen, um gegen andere Spieler Krieg zu führen, Rohstoffe zu transportieren oder neue Planeten einzunehmen. Es gibt unterschiedliche Planetentypen:

Tab.6: Planetentypen

Planetentyp	Spezialität
Erdähnlicher Planet	Nahrungsproduktion * 1.5
Wasserplanet	Windenergie * 2
Wüstenplanet	Siliziumproduktion * 1.5
Eisplanet	Tritiumproduktion * 1.5
Dschungelplanet	Bewohnerwachstum * 1.5
Gebirgsplanet	Titanproduktion * 1.2, Siliziumproduktion * 1.3
Gasplanet	Unbewohnbar

Ein Spieler kann Planeten mithilfe eines Kolonialisierungsschiffes kolonialisieren, oder mit einem Invasionsschiff eine bereits kolonialisierten Planeten übernehmen. Planeten können nicht zerstört werden. Da Planeten nicht gleich gross sind, besitzt jeder Planet eine zufällige Anzahl an Feldern, auf denen

gebaut werden kann, das Spektrum reicht von 400 – 900 Felder. Auch die Temperatur ist unterschiedlich, was Auswirkungen auf die Energieproduktion und Nahrungsproduktion hat. Das Spektrum reicht von –130 bis 150° Celsius.

5.2.7.4 Wurmlöcher

Wurmlöcher sind topologische Konstrukte der Raumkrümmung, die weit voneinander entfernt liegende Bereiche des Universums durch eine 'Abkürzung' verbinden. Im Spiel treten Wurmlöcher zufällig verteilt im Raum auf. Jedes Wurmloch ist mit einem anderen Verbunden. Man kann Schiffe in das Wurmloch schicken und kann so eine Flotte in sehr kurzer Zeit ans andere Ende der Galaxie befördern.

5.2.7.5 Rassen

Es gibt natürlich auch verschiedene Rassen. Jede Rasse hat ihre Vor- und Nachteile. Nachfolgend eine Liste mit den Hauptmerkmalen:

Tab.7: Rassen

Rasse	Merkmale
Andorianer	Die Andorianer sind zugleich humanoid und insektoid. Sie haben graublaue Haut und weißes Haar. Auf ihrem Kopf haben sie zwei Fühler, die ihnen zur feinfühligen sinnlichen Wahrnehmung dienen. Ihre Stärke ist die Produktion künstlicher Stoffe wie Plastik.
Centauri	Die Centauri haben die besten Wissenschaftler des Universums, darum können sie auch schneller Technologien erforschen.
Ferengi	Die Ferengi sind eine humanoide Rasse. Sie sind etwas kleinwüchsiger als Menschen und haben große Ohren. Die Stärke der Ferengi liegt beim Abbau von Metall.
Minbari	Die Minbari sind eine humanoide Rasse. Ihre Stärke ist das Gewinnen von Tritium als Treibstoff für Schiffe.
Orioner	Die Orioner sind eine humanoide Rasse aus der Nähe des Orions. Die Gesellschaft der Orioner besteht hauptsächlich aus Schmugglern und Piraten. Ihre Schiffe sind bekannt für ihre Schnelligkeit.
Rigelianer	Die Rigelianer stammen aus dem Rigel-System. Ihre Stärke liegt im Abbau von Kristallen, die für Steuereinheiten in Gebäuden und Schiffen verwendet werden.
Terraner	Eine eher jüngere Rasse, deren Vorfahren ursprünglich vom Planeten Erde kamen. Die Menschen sind besonders gut in Forschung, der Herstellung von Plastik und dem Anbau von Nahrung.
Vorgonen	Die Vorgonen sind eine Rasse, die vor allem gut Bauen kann. Sie können ihre Gebäude viel schneller fertig stellen als alle anderen.

5.2.7.6 Bewohner, Crew, Arbeiter

Ohne Menschen läuft auch in der Zukunft nichts; jedes Gebäude braucht Arbeiter, welche die Anlagen bedienen. Ebenfalls braucht jedes Schiff eine Crew, ohne Crew kann es nicht starten. Die Menschen brauchen natürlich auch Gebäude, in denen sie Leben können -> Wohnmodule. Die Menschen vermehren sich prozentual zu ihrer Anzahl. Wenn nicht genügend Wohnmodule vorhanden sind, steigt die Anzahl der Menschen nicht weiter. Bei einer Schlacht ist es natürlich so, dass Schiffe zerstört werden. Es ist aber so, dass sich immer ca. 70% – 90% der Besatzung mithilfe von Rettungskapseln retten können und wieder auf dem Heimatplaneten landen. Für die Menschen muss ebenfalls genügend Nahrung vorhanden sein, sonst können nicht alle ernährt werden. Menschen, die hungern, werden nicht arbeiten und können keine Schiffe steuern.

5.2.7.7 Gebäude

Gebäude erweitern die Basis mit neuen Funktionen oder dienen dazu, Rohstoffe zu produzieren. Gewisse Gebäude können ausgebaut, andere können mehrfach gebaut werden. Jedes Gebäude benötigt für den Betrieb elektrische Energie; der Bedarf an Energie nimmt pro Ausbaustufe zu.

5.2.7.7.1 Rohstoffgebäude

In diesen Gebäuden wird pro Stunde eine bestimmte Anzahl Rohstoffe gewonnen. Pro Ausbaustufe erhöht sich diese Menge exponentiell.

Tab.8: Rohstoffgebäude

Name	Beschreibung
Titanmine	Produziert Titan
Siliziummine	Produziert Silizium
Chemiefabrik	Produziert PVC
Tritiumsynthesizer	Produziert Tritium
Gewächshaus	Produziert Nahrung

5.2.7.7.2 Speicher

Es gibt eine natürliche Maximalgrenze für Rohstoffe auf einem Planeten. Wenn diese erreicht ist, wird kein weiterer Teil dieses Rohstoffs produziert. Speicher dienen zum Einlagern von Rohstoffen und erhöhen damit diese Maximalgrenze.

Tab.9: Speicher

Name	Beschreibung
Titanspeicher	Speichert Titan
Siliziumspeicher	Speichert Silizium
Plastikspeicher	Speichert PVC
Tritiumtank	Speichert Tritium
Nahrungsmittel-Lagerhalle	Lagert Nahrung

5.2.7.7.3 Kraftwerke

Alle Gebäude benötigen Strom für den Betrieb. Dieser wird durch Kraftwerke zur Verfügung gestellt. Je höher ein Kraftwerk ausgebaut ist, desto mehr Energie produziert es.

Tab.10: Kraftwerke

Name	Beschreibung
Windkraftwerk	Nicht sehr leistungsfähiges Kraftwerk
Solarkraftwerk	Effizientes Kraftwerk. Je näher der Planet bei der Sonne ist, desto mehr Energie wird produziert
Gezeitenkraftwerk	Effizientes Kraftwerk, welches Energie aus dem Hubunterschied von Ebbe und Flut gewinnt
Fusionskraftwerk	Produziert Energie aus der Kernfusion von Wasser- und Sauerstoff. Sehr effizient

5.2.7.7.4 Andere Gebäude

Diese Gebäude sind Voraussetzungen für andere Objekte.

Tab.11: Andere Gebäude

Name	Beschreibung
Planetenbasis	Grundgebäude jedes Planeten. Bietet Platz für Bewohner, Lagerräume für Rohstoffe und eine minimale Energieversorgung durch ein integriertes Erdwärmekraftwerk
Wohnmodul	Mit einem Wohnmodul wird die Kapazität für Bewohner erhöht
Forschungslabor	Im Labor werden neue Techniken entwickelt
Schiffswerft	In der Werft werden alle Raumschiffe gebaut
Waffenfabrik	In der Waffenfabrik werden Verteidigungsanlagen gebaut
Flottenkontrollzentrum	Muss gebaut werden, damit überhaupt Flotten starten können. Jede Ausbaustufe erhöht die Anzahl der Flotten, die gleichzeitig unterwegs sein können

5.2.7.8 Forschungen

Tab.12: Forschung

Name	Beschreibung
Schildtechnik	Schilde schützen Schiffe und Verteidigungsanlagen gegen gegnerischen Beschuss. Je höher die Schildtechnik ist, desto mehr halten die Schilde aus
Spionagetechnik	Mithilfe der Spionagetechnik können Spionagesonden gebaut werden, welche gegnerische Planeten ausspionieren. Je höher die Technologie ist, desto genauer der Bericht
Stealth-Technik	Die angreifende Flotte wird vom Gegner erst später erkannt. Ebenfalls erhält der Gegner schlechtere Spionageberichte, je höher entwickelt die Technologie ist
Solartechnik	Mithilfe der Solartechnik können Energielieferanten gebaut werden, die auf Solarzellen setzen
Lasertechnik	Durch die Entwicklung der Lasertechnik lassen sich leistungsfähige Verteidigungsanlagen bauen
Rückstosstriebwerk	Standardantrieb für einfache Raumschiffe. Basiert auf der Verbrennung von mitgeführtem Treibstoff
Ionenantrieb	Weiterentwickelter Antrieb für Schiffe. Treibstoffgas wird ionisiert, und durch Abstossung dieses Ionenstrahls wird das Schiff angetrieben. Langsamer, aber billiger Antrieb.
Hyperraumantrieb	Hypothetischer überlichtschneller Raumschiffantrieb. Der dem Warpantrieb zugrunde liegende Quanteneffekt verzichtet darauf, das Raumfahrzeug durch Beschleunigung zu bewegen, was gemäß der Relativitätstheorie infolge schließlich unendlicher Massenzunahme spätestens an der Lichtgeschwindigkeit scheitern muss, sondern verzerrt das Raum-Zeit-Gefüge derart, dass das Fahrzeug quasi in einem eigenen Mini-Universum eingesperrt mitsamt diesem sein Ziel erreicht
Fusionsenergie	Durch Gewinnung von Energie durch Kernfusion lassen sich sehr leistungsfähige Energieproduzenten bauen
EMP-Technik	Der Elektromagnetischer Impuls (EMP) bezeichnet einen kurzzeitigen, hoch energetischen, breitbandigen, elektromagnetischen Strahlungsimpuls. Dieser kann elektronische Systeme lahm legen

5.2.7.9 Schiffe

Alle Schiffe benötigen als Treibstoff Tritium. Es gibt verschiedene Antriebsarten für die Schiffe. Jedes Schiff kann eine gewisse Menge Rohstoffe transportieren, benötigt aber auch eine Mindestanzahl an Crewmitgliedern, um überhaupt starten zu können.

Tab.13: Schiffe

Name	Beschreibung
Transporter	Transportiert Rohstoffe von einem Planet zum anderen. Rückstossantrieb
Jäger	Einfaches Kampschiff. Rückstossantrieb
Kreuzer	Mittleres Kampschiff. Ionenantrieb
Schlachtschiff	Schweres Kampschiff. Warpantrieb
Invasionsschiff	Mit diesem Schiff kann man einen Planeten einnehmen, aber erst, wenn keine feindliche Verteidigung mehr existiert. Rückstossantrieb
Räumer	Sammelt Trümmer ein, die nach einer Schlacht im Orbit des Planeten kreisen. Rückstossantrieb
Sammler	Sammelt Titan von Asteroidenfeldern. Ionenantrieb
Nebelsauger	Fliegt in interstellare Nebel, saugt das Gas ab und verarbeitet es zu Treibstoff. Ionenantrieb
Spionagesonde	Spioniert einen gegnerischen Planeten aus und liefert einen Bericht seiner Gebäude, Flotte und Verteidigung. Ionenantrieb

5.2.7.10 Verteidigung

Verteidigungsanlagen schützen die Basis gegen generische Überfälle. Jede Verteidigungsanlage benötigt Strom zum Betrieb. Wenn zuwenig Strom vorhanden ist, funktionieren nicht alle Anlagen einwandfrei.

Tab.14: Verteidigungsanlagen

Name	Beschreibung
Flak	Einfaches Abwehrgeschütz
Abfangraketen	Abfangraketen verfolgen ihr Ziel und richten erheblichen Schaden an
Laserturm	Dieses Geschütz richtet einen gebündelten und extrem starken Energiestrahl auf ihr Ziel
EMP-Kanone	Die EMP-Kanone schleudert Schockwellen gegen die gegnerische Flotte und kann die elektronischen Systeme der Schiffe für eine gewisse Zeit lahm legen, die Schiffe werden damit kampfunfähig
Prismaspiegel	Durch mehrere Spiegel und Linsen wird ein Laserstrahl derart gebündelt, dass damit selbst grosse Schiffe beschädigt werden können

5.2.8 Funktionen

5.2.8.1 Nachrichtensystem

Die Spieler können sich untereinander Nachrichten senden. Ebenfalls erhält der Spieler vom System Nachrichten, z.B. wenn ein Transport angekommen ist oder es einen Kampf gab. Systemnachrichten und Spielernachrichten werden optisch getrennt.

5.2.8.2 Statistik

Über einen entsprechenden Button ist eine Statistik abrufbar, welche die Stärke der Spieler in folgenden verschiedenen Kategorien bewerten:

- Gebäudepunkte (pro 1000 verbaute Tonnen Ressourcen 1 Punkt)
- Flottenpunkte (pro 1000 verbaute Tonnen Ressourcen 1 Punkt)
- Verteidigungspunkte (pro 1000 verbaute Tonnen Ressourcen 1 Punkt)
- Forschungspunkte (1 Forschungspunkt pro Forschung, pro 1000 verbaute Tonnen Ressourcen 1 Punkt)
- Anzahl Planeten / Gebietgrösse (Pro Planet 1000 Punkte)
- Schlachtpunkte
- Gesamtwertung

Die Punkte sind in der User-Tabelle gespeichert und werden jedes Mal aktualisiert, wenn ein Spieler etwas baut bzw. kolonialisiert oder forscht.

5.2.8.3 Spieleroptionen

Folgende Optionen kann ein Spieler einstellen :

- Nickname ändern (pro 14 Tage 1 mal)
- Passwort ändern
- E-Mail ändern
- Urlaubsmodus (Alle Produktionen stehen still, es fliegt keine Flotte, der Spieler kann nicht angegriffen werden)
- Account löschen

5.2.9 Prioritätenliste

Mir ist klar, dass nicht alle in diesem Konzept genannten Sachen auf einmal programmiert werden können. Hier folgt somit nun eine zeitlich sortierte Liste, was ich in welcher Reihenfolge implementieren will (Die meiner Ansicht nach am schwierigsten umsetzbaren Punkte sind *kursiv* formatiert):

- Datenbankstruktur
- Grunddesign / -aufbau der Oberfläche
- Raumkarte (Galaxie, Sektoren, Planeten)
- Benutzer (Registrierung, Login, Optionen)
- Gebäudebaumenü
- Rohstoff- / Energieübersicht
- Forschungsmenü
- Schiffbaumenü
- *Flug*
- Verteidigungsbaumenü
- *Kampfsystem*
- Statistik
- Nachrichtensystem
- Rassen
- Wurmlöcher
- Grafisches Design

5.2.10 Quellen

Im folgenden eine Liste von Quellen, die mir als Inspiration und Hilfestellung für dieses Konzept gedient haben:

Wikipedia – Die freie Enzykopädie http://de.wikipedia.org

ESA – Europäische Raumfahrtsgesellschaft http://www.esa.int

NASA http://www.nasa.gov

Browsergames.net http://www.browsergames.net

Galaxy-News http://www.galaxy-news.de

Formeln+Tafeln, DMK/DPK, Orell Füssli Verlag AG, Zürich, 2001

5.3 Screenshots – Bilder des Spiels

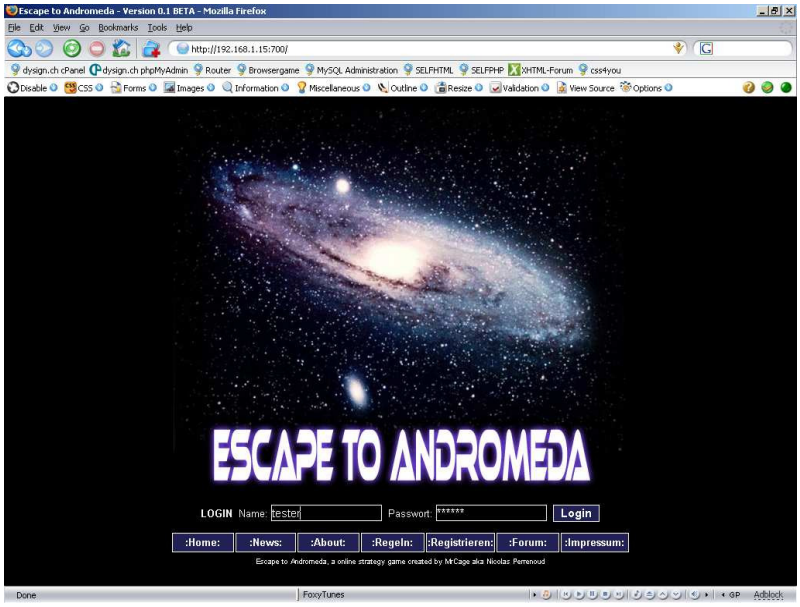

Abb. 6:

Erster Entwurf einer Startseite

Abb. 7:

Die finale Startseite

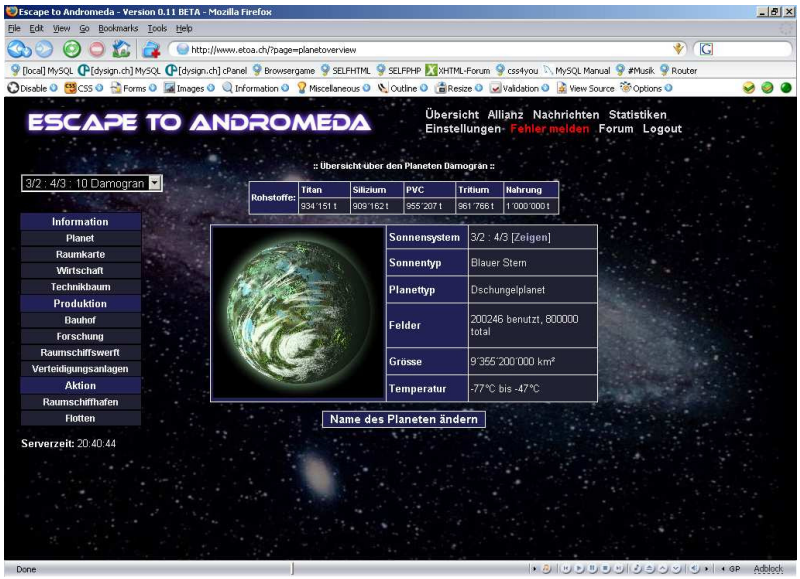

Abb. 8:

Erster Entwurf einer Startseite

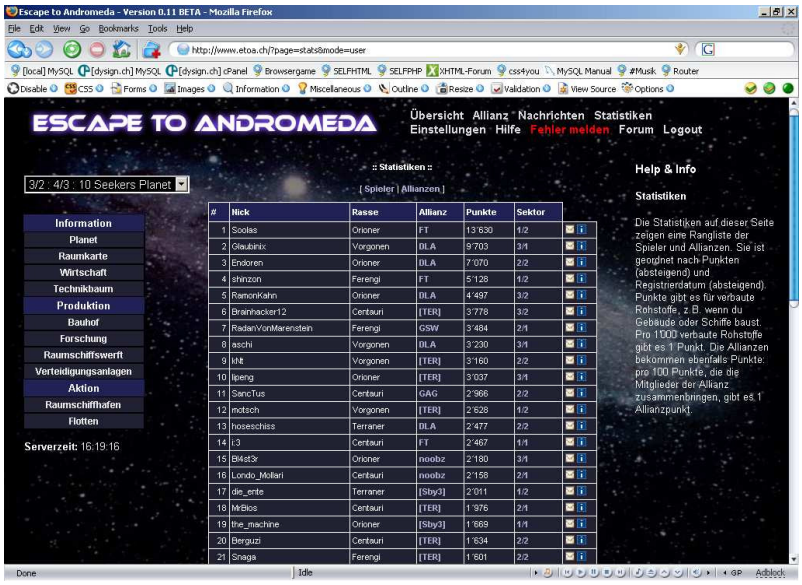

Abb. 9:
Statistiken

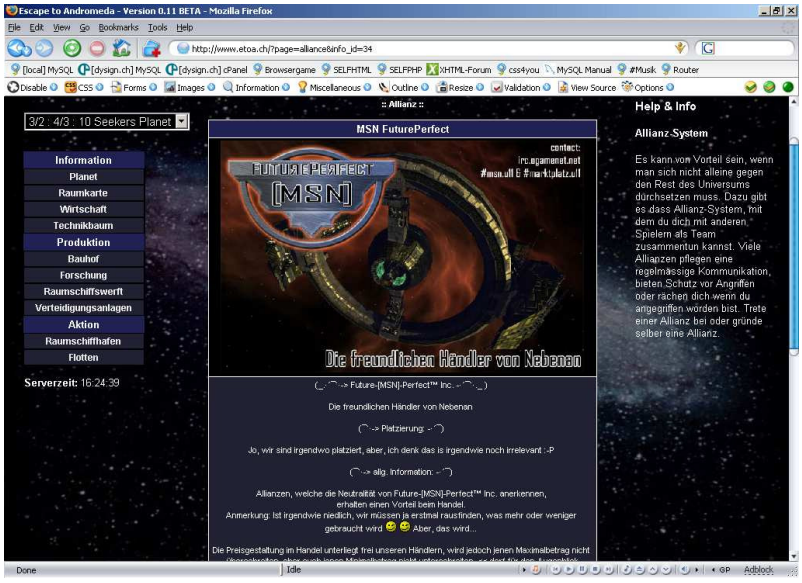

Abb. 10:
Infoseite einer Allianz

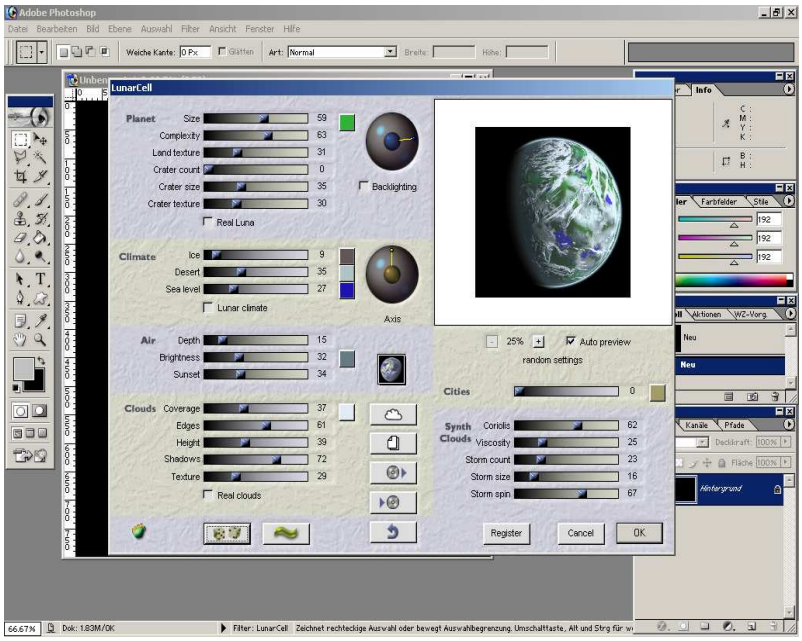

Abb. 11:
Planetenbild-Software LunarCell

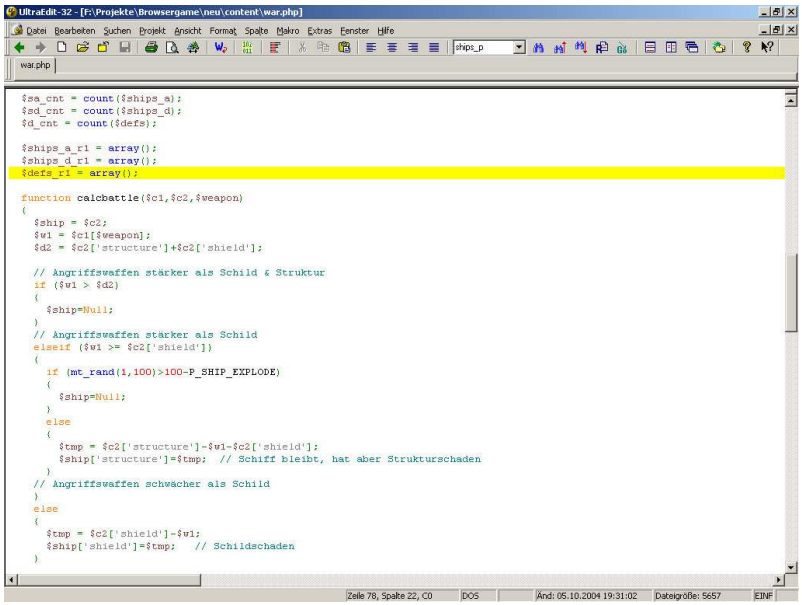

Abb. 12:

Code des ersten Kampfsystems im UltraEdit Texteditor

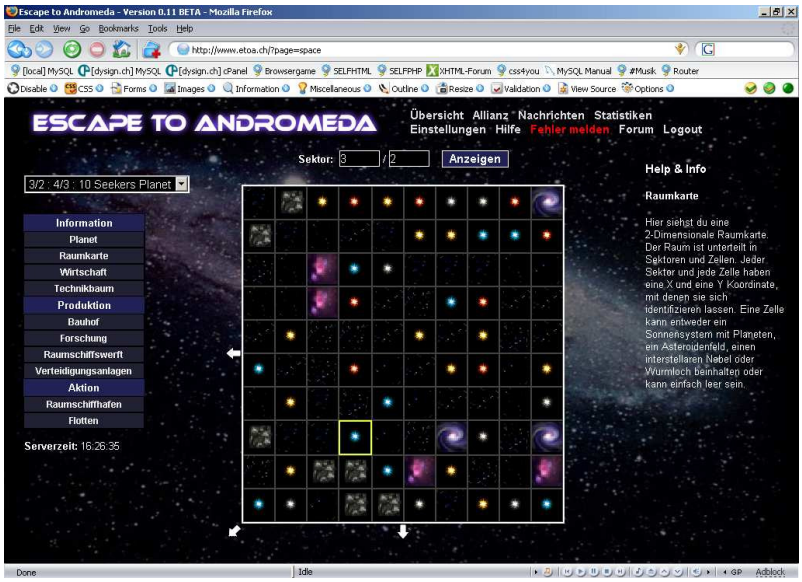

Abb. 13:

2-D Raumkarte

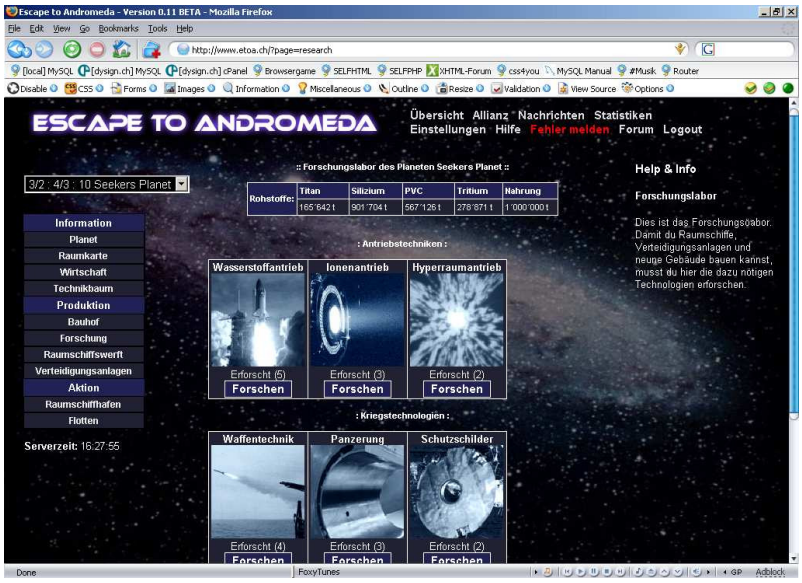

Abb. 14:

Forschungslabor