Hovedopgave Datamatiker

Morten Bendiksen | Erhvervsakademi Sjælland (Campus Næstved) | 11/7 2016 - 9/1 2017

_____

Morten Bendiksen

Indholdsfortegnelse

Indledning

EXFU er et online turbaseret rollespil, der foregår i en fremtidig verden, og handler om at udforske og bekæmpe diverse livsformer, på forskellige planeter i galaksen, for at finde den mytiske civilisation EXFU.

Idéen

Skab et turbaseret spil med fokus på op til fem forskellige roller[1] der fungere som en enhed, til at bekæmpe modstanderen. Mellem kampene rejser gruppen fra sted til sted for at opløse en gennemgående gåde.
Spillet udvides løbende og mere af historien kommer frem i lyset, hvor størrer trusler hele tiden viger gruppen fra sit oprindelige mål.

Historien

Spillet foregår i et alternativt univers, hvor teknologien er langt forud.
Interplanetariske rejser er hverdag for toppen og flere civilisationer har bosat sig i den samme galakse.

Spilleren er en højere magt[2] der besidder en evne til at give enkelte skabninger i universet "umenneskelige" kræfter.
Planeten spilleren har affektioner til er under angreb, så en skabning benådes med en særlig kraft der kan besejre truslen.
Skabningen anses som en helt og får tildelt nye missioner for planetens hær.
Opgaver bliver sværere og sværere for skabningen og det går op for spilleren at det kræver en gruppe af stærke skabninger, for at besejre de nye trusler.
Spilleren følger skabningerne og guider dem til sejre.

Skabningerne sættes på en ældgammel og hidtil uopklaret mission: Lokalisér EXFU.

EXFU er en ukendt højere magt der er i stand til at udslette hele galakser, man ved endnu ikke hvor den kommer fra eller præcist hvad det er. Men hvis man fandt en måde at kontrollere den, ville man herske universet.

Under søgningen møder gruppen hele tiden nye trusler og folk i nød. Dette får dem til at vige fra deres hovedmission og bruge deres nyfundne kræfter til at hjælpe.

---

[1] Eksempler på roller i rollespil kunne være en til at tage imod skade fra modstanderen, en til at heale denne skade, samt en til at gøre skade på modstanderen.
[2] En gud, videnskabeligt eksperiment eller en mytisk ukendt sky mfl.

Faglige overvejelser

De forskellige dele af spillet bygges som blokke, der hver har sin egen status i udviklingen.
Når alle blokke har nået en vis status, opnår spillet som helhed denne status.
De tre overordnede statusser er Alpha, Beta og Live.
Kravet for Alpha er at de basale funktioner for blokken fungerer.
Beta kræver yderligere at blokken har alle funktioner, samt den grafiske del er implementeret.
Her tages hensyn til eventuelt feedback.
For at få blokken klar til at gå Live kræves yderligere at blokken er testet for fejl, samt at der er tjekket for eventuelle sikkerhedshuller og måder at bruge blokken utilsigtet.

Implementering

Spillet er udviklet med PHP som backend engine, denne sørger for at holde en database opdateret hver gang en spiller udfører en handling.

Kampe er sat op som et XML-feed, med informationer om spilleren og modstanderens status ved kampens start. Hver handling fra enten spilleren eller fjenden producere en række i XML-feed'et. Dette muliggøre, at flere spillere kan spille sammen og bruge det samme feed, samt at man senere kan gense kampen. Et feed gør det også muligt at lave en PC-version som ikke nødvendigvis behøver en webbrowser, for at køre.

Spillets initiale frontend interface er en HTML side, sat op i HTML5.
Alt design er lavet med CSS3, og animation er en kombination af CSS, javascript og jQuery[3].

Problemformulering

Hvordan udvikles et web-baseret spil med feedback fra brugerne?

Hvordan bruges ren HTML og CSS som basis for et grafisk brugergrænseflade til et spil?

Hvordan sammenkobles forskellige grafiske brugergrænseflader til en PHP backend?

[3] jQuery er et platformsuafhængigt JavaScript-bibliotek der har til formål at gøre det nemmere at lave programmer til webbrowsere.

Tekniske overvejelser

Som udgangspunkt ville jeg have en browser som brugergrænseflade, da det let kunne overføres til andre platforme, samt gav databasen en større sikkerhed.

Fordele ved at bruge en browser er minimal installation, spillet i sig selv kræver ingen. Tvunget opdatering, ændringer i spillet sker på en server, så der er kun én version distribueret.

Valget af backend programmeringssprog blev PHP, da det er det jeg er mest bekendt med.

Selve spillets backend er sat op om et XML-feed med informationer om deltagerne ved start, samt enhver handling der er gjort i spillet.
På den måde kan man overføre feed'et til ethvert interface, som blot skal fremvise informationerne på en måde der er læseligt for mennesker.

Animationer er forsøgt udført i CSS hvor muligt og ellers via javascript gennem jQuery.

Da spillet er udviklet i Google Chrome er det også optimeret til denne.
Spillet er testet acceptabelt i browseren Edge, og da Safari er bygget på webkit-engine som Chrome, skulle den også virke acceptabelt her.

Gameplay

Al beskrivelse antager at Demoen er installeret korrekt, samt at databasen er kørt.
Vejledning til dette findes på disken.

Demo

Dette er en beskrivelse af, hvordan man spiller Demoen.
Demoen indeholder helt basale elementer af spillet og giver et konceptisk indblik i hvilken spillestil det færdige spil kommer til at læne sig imod.

Start med at oprette en ny bruger
Klik på "Sign up" ved demoens startside.
Indtast 'Display name'. Navnet skal være ledigt i databasen.
Indtast 'E-mail'. Denne skal opfylde kravet [x@x.x] og skal være ledigt i databasen.
Indtast 'Password' og 'Retype Password'. Disse skal være ens.
Vælg sprog. *Guiden her antager at 'English' er valgt.*
Vælg 'Security-Level'. Hvis 'AntiKeyLog' tilvælges skal et 3-cifret nummer også vælges.
Klik på "Sign up" under formularen.

Log ind
Klik eventuelt på "Log in" i toppen, hvis login-skærmen ikke vises automatisk.
Indtast den valgte e-mailadresse og password.
Vælg eventuel den 3-cifrede AntiKeyLog kode, hvis tilvalgt under oprettelse.
Klik på "Log in" under formularen.
Afvent at 'Log in'-knappen skifter farve til grøn.
Bliver den i stedet rød er der sandsynligvis sket en fejl i indtastningen.

Start spillet
Klik på "Start Game".

Opret en ny Gruppe
Klik på "Create new Group" for at starte et nyt spil.
Vælg hvilken rolle spil-figuren skal opfylde.
Klik "Continue".
Vælg hvilke kræfter spil-figuren skal besidde.
Der skal vælges i alt to kræft-sæt, én fra midten og én fra enten toppen eller bunden.
Klik "Continue".
Indtast spil-figurens navn. Dette bliver vist under spillet.
Klik "Continue".

Vælg Gruppe
Klik på den Gruppe der skal spilles med.
Klik på "Enter World".

Start Kamp
Klik på "Battle-Simulator".

Kampens HUD[4]
Brugergrænsefladen omkring kampens 'bane' indeholder en top- og bund bar samt højre og venstre status-barre.
Top-baren bruges til at vise spillets handlinger, samt hvem der har tur næste gang.
Højre- og venstre barre viser kampens deltagere, ved klik på disse markeres denne som Target[5].
Venstre bar viser spillerens Gruppe.
Højre bar viser modstanderens Gruppe.
Bund-baren fyldes med de kræfter der er mulige pt.

'Banen'
På banen ses omkring den nuværende figur, hvis denne er spillerens, et grønt felt.
Ved klik på denne, flytter figuren sig i den valgte retning.
Ved klik på figures 'fødder' vælges de som Target.

Kampens Regler
Spilleren og modstanderen skiftes til at lave en handling.
Når et gruppemedlems livspoint (den røde bar under medlemmets navn i sidebaren) rammer 0, betragtes denne som død, og får ikke tur igen i kampen.
Gruppen der først mister alle sine medlemmer, har tabt.

Sådan laver man en handling
Når det er spillerens tur (vises med en gul firkant omkring en af spillerens gruppemedlemmer), starter spilleren med at vurdere om medlemmet skal rykke.
Positionering er endnu ikke implementeret på demoen og har derfor ingen effekt på kampens udfald.
Herefter vælges et Target.
Klik på en af de to mulige kræfter i bund-baren.

Vind eller Tab
Fortsæt med at lave handlinger indtil en af gruppernes medlemmer alle er faldet.
Modtag herefter en besked om kampens udfald, samt hvor mange handlinger der blev spillet.
Klik på knappen i bunden af beskeden, for enten at modtage en belønning, eller tage tilbage til skibet afhængigt af kampens udfald.

[4] Head-Up-Display er den grafiske brugergrænseflade. Ordet stammer fra teknologien brugt i moderne flyvevåben.
[5] Et Target er i denne sammenhæng den modstander der bliver angrebet fremover.

Kortet over Universet
Uden for kamp kan Universet udforskes.
Klik på "Map" i toppen.
Venstreklik på Solsystemer og Planeter for at zoome ind på denne.
Højreklik for at zoome ud igen.
På siden med et Solsystem drejer Planeterne langsomt omkring stjernen.
Klik på "Map" igen for at lukke kortet.

Spiller-status
Spillerens status kan ses ved at gå tilbage til startsiden.
Klik på "Huset" ved siden af 'Map'.
Fra startsiden klikkes på Brugernavnet øverst til højre.
Her vises spillerens grupper samt medlemmer af disse.

Afslutning af Demo
Klik på "Log out" øverst til højre på startsiden for at afslutte.
Klik på "X" øverst til højre på spil-siden og derefter på "Log out?" for at afslutte.
Klik på "X" igen for at fortryde afslutningen fra spil-siden.

Fremtidige udvidelser

Listen af udvidelser er kun begrænset af fantasien, og derfor beskrives kun de planlagte udvidelser til demoen.

Forside
Forsiden fyldes med nyheder.

Oprettelse og Log ind
Fejlmeddelelser implementeres.
Send formularer med 'Enter'.

Sprog
Flere sprog implementeres.
Sprog vælges fra startsidens menu og bruges som sprog ved oprettelse.
Sprog vælges ved klientens lokalitet.

Grupper
Navn på gruppe indtastes ved oprettelse.
En beskrivelse af diverse kræfter samt en overordnet beskrivelse af de enkelte kraft-sæt.
Løbende tilføjelse af op til fem gruppemedlemmer gennem gennemførelse af missioner.

Planeter
Mulighed for at udforske de enkelte planeter ved gruppen og rejse gennem rummet.

Missioner
Gruppen får missioner ved at tale med NPC'er[6] på forskellige planeter.

Kamp
Kampen fyldes med grafik og animationer.
Kræfter får en effekt, både logisk og grafisk.
Modstanderen får en mere scriptet opførelse, dvs. at kampen bliver designet så visse handlinger sker på visse tidspunkter i kampen.
Modstanderens kunstige intelligens bliver forbedret til ikke kun at vælge den største trussel.

Rumfartøj
Rumfartøjet får udfyldt tomme rum.
En Bridge implementeres, hvor diverse opdateringer for forbedringer til fartøjet kan tilkøbes.
En Engine implementeres hvor fremdriftssystemet forbedres og reducerer rejsetiden mellem destinationer.

Teleporterer
Hver storby besøgt af gruppen kan herefter tilgås via en teleporter der fjerner rejsetiden.

Implementering

Systemudviklingsmetode

Den valgte metode for udvikling bygger hovedsageligt på Scrum, med Sprints på 2 uger.
Men blander også XP ind ved prioritering af hvilke dele af projektet der er vigtigst pt.
Spillet er udviklet i blokke der alle bruger den samme model og database, men ellers fungerer som en slags stand-alone side, på den måde kan spillets funktioner ændres eller fjernes uden at påvirke resten af systemet.

Der er udviklet en "To do"-liste med spillets forskellige sider, samt en liste af Use Cases med aktører.

Under udviklingen benyttede jeg mig mest af "To do"-listen og arbejdede mig kronologisk gennem brugeroplevelsen begyndende med brugeroprettelse og afsluttende med, indtil videre at vinde eller tabe en kamp.

Der er udviklet en Tidsplan der har fungeret som delvis Sprint Backlog.

"To do", Use Cases og Tidsplan kan ses i Bilag.

Der er forinden dette projekt udviklet et "Game Design Document" der beskriver alle planer for EXFU, der dengang blev kaldt REXFUR.

---

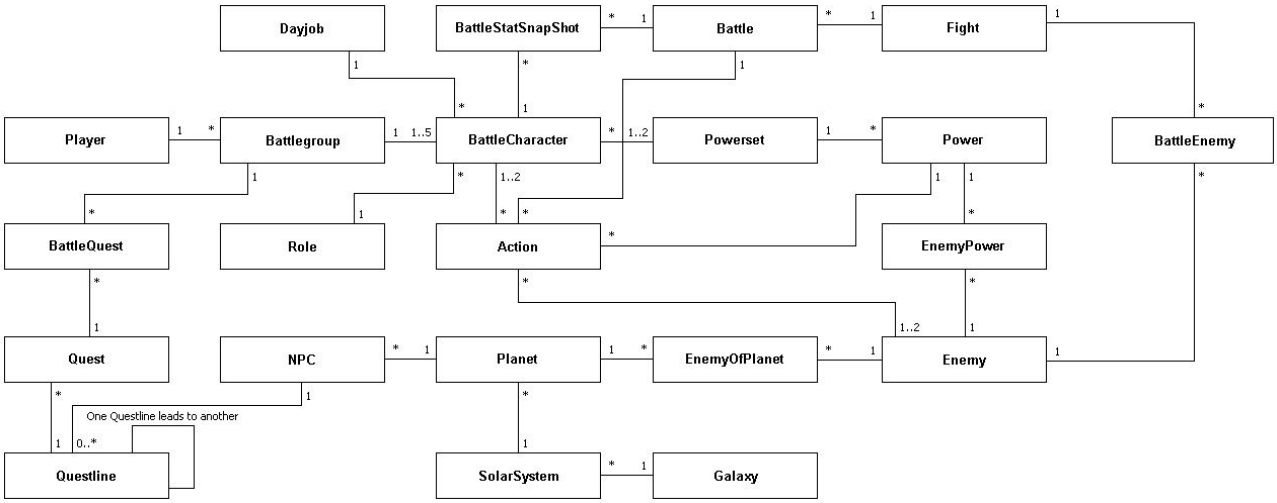
[6] Non-Playable Character, en figur i spillet som styres af systemet, men er ikke en fjende. Disse vil hovedsageligt være folk i byer og personer med opgaver til spilleren.

Her bliver spillets design beskrevet med flere detaljer.
Nogle af planerne er senere blevet ændret mens andre helt er afskrevet.
Men store dele af projektet er lavet med dette dokument i baghovedet, men med frihed til at ændre planen.

"Game Design Document" er vedlagt som Bilag.

Arkitektur

Dette er klart mit endnu største projekt, og er også grunden til at det er blevet udskudt nogle år.
Indtil videre er området nederst til venstre (Quest, Questline og NPC), samt Dayjob ikke implementeret.



Integration

Database
Databasen er udviklet i MySQL, er i 3. normal form og er pt. fyldt med test-data til brug i pre-alpha perioden. Indtil spillet går live vil data'en ændres efter behov.
Database script til oprettelse er vedlagt i Bilag.

Programmering
Modellen er integreret i PHP, grundet tidspress, er programmeringen udviklet med funktion i fokus, dvs. at det er lavet til at virke, frem for at være let at vedligeholde.
Koden er optimeret til PHP 7, men er bagudkompatible til PHP 5.2.
Inden beta'en vil koden blive gennemgået, opdateret og blive fuldt ud objekt orienteret.

Selve spillet er lige nu baseret på et XML-ark der indeholder alle informationer omkring den enkelte kamp.
Størstedelen af regler og udregninger er skubbet til klienten via javascript.
Dette mindsker processer kræften på serveren og gør spillet i stand til at køre responsivt for en større gruppe spillere ad gangen.

På et senere tidspunkt vil logikken blive ført over på server-siden i PHP, dette vil åbne op for en pc-klient der kører udenfor en browser, og kan skrives i et hvilket som helst programmeringssprog der kan hente JSON eller XML-filer fra internettet.
På den måde kan databasen også sikres mod reverse engineering[7], da forbindelsen til databasen kommer fra php-enginen'en.

HTML
HTML'en er udviklet i HTML5 og er især på kamp-siden meget XML baseret, dvs. at der bliver brugt ukendte tags der øger overskuelsen under udviklingen.

[7] Reverse Engineering, betyder omvendt konstruktion hvor en ingeniør eller anden tekniker tager et færdigt produkt, som kan være et elektronisk- eller mekanisk komponent, adskiller dette og kopierer løsninger til eget produkt.

CSS
Grundet spillet naturlige tunge brug af billeder, er sidens design meget simpelt og minimalistisk.
Der vil blive brugt CSS frem for billeder, hvor end det er muligt.
Siden er i demo-tilstanden kun delvist responsivt designet, dette vil blive rettet under beta-versionen.
Efterhånden kan CSS klare simple animationer, så med tiden vil disse blive ført herover fra javascript.

På den måde bliver animationer også en bedre integreret del af sidens øvrige design, og skulle man lave specielle designs fx for at forhindre epileptiske anfald, skal kun CSS'en ændres for denne bruger.

javascript/jQuery
Sidens effekter er lavet i jQuery.

Meget af udviklingstiden er brugt her, da hele logikken bag kampene er udviklet i en blanding af AJAX[8], jQuery og almindelig javascript.

Kampen fungerer ved først at hente status for deltagerne, og herefter fremvise disse.
Efterfølgende løbes kampens handlinger gennem.
Hver handling omhandler en angriber, en modstander, en kraft og til sidst angriberens position.
Med disse informationer beregnes og vises den nye status for involverede deltagere.

Når hele listen af handlinger er løbet igennem, går systemet i klar-mode og vurdere hvis tur det er. Hvis turen går til spilleren afventer systemet et input herfra.

[8] AJAX (en forkortelse for Asynkron JavaScript og XML) er en webudviklingsteknik til at udvikle interaktive webapplikationer. Idéen er at gøre websider mere reaktionsdygtige ved at udveksle små mængder af data mellem klienten og serveren, så hele siden ikke skal genindlæses, hver gang brugeren laver en forespørgsel. Formålet med dette er at øge websidens interaktivitet, hastighed og brugervenlighed.

Programmering af Login

Efter at have indtastet brugernavn og kode (evt. anti-keylogger) udfører systemet en verificering af informationerne.
Login-processen starter i funktionen "verify" hvor en database-forbindelse oprettes og den det indtastede brugernavn bruges til at lokalisere kodeordet (samt anti-keylogger koden)

Kodeordet er krypteret med PHP's standard bcrypt[9], og den indbyggede PHP-funktion 'password_verify' genskaber den samme algoritme for den indtastede kode, som brugt på den lagrede kode.
Hvis disse skaber det samme resultat, anses koden for at være korrekt.

Samtidig testes anti-keylogger koden, hvis brugerens sikkerhedsniveau kræver dette.

Brugerens ID sendes retur til Login-siden der afgører om brugeren får adgang, og initiere login-funktionen hvis nødvendigt.

```
public function verify($user, $password, $antikeylog) {
    $passHash = "";
    $antiHash = "";
    $id = "";
    $this->connect();
    $userDB = $this->select("select id,intsecuritylevel,strpassword,intantikeylog from player where stremail='$user'");
    while ($row = $userDB->fetch()) {
        $passHash = $row['strpassword'];
        $antiHash = $row['intantikeylog'];
        $id = $row['id'];
        $securitylevel = $row['intsecuritylevel'];
    }
    $this->disconnect();
    if (password_verify($password, $passHash) && ($securitylevel == 0 || password_verify($antikeylog, $antiHash))) {
        $userDB = $this->select("select id,strpassword,intantikeylog from player where stremail='$user'");
        while ($row = $userDB->fetch()) {
            $passHash = $row['strpassword'];
            $antiHash = $row['intantikeylog'];
            $id = $row['id'];
        }
        $result = $id;
    } else {
        $result = 0;
    }
    return $result;
}
```

[9] bcrypt er en adgangskode hashing funktion designet af Niels Provos og David Mazières, baseret på Blowfish cipher, og præsenteret på USENIX i 1999. Udover at indarbejde en salt til at beskytte mod regnbue tabel angreb, er bcrypt en adaptiv funktion: over tid, kan iterations tælleren øges for at gøre det langsommere, så det forbliver resistent over for brute force angreb selv med stigende CPU-kraft.

Funktionene 'login' henter én bestemt bruger og bygger henholdsvis et Player-objekt og et Battlegroup-objekt.

Player-objektet indeholder basale informationer om brugeren fra databasen.

Battlegroup-objektet indeholder informationer om alle de grupper spilleren har oprettet, samt de BattleCharacter's der er tilknyttet gruppen, og alle kræfter BattleCharacter'erne besidder.

Begge objekter gemmes midlertidigt, mens spilleren er logget ind, i en session på serveren, og serialiseres[10], så de kan bruges på andre sider.

```
public function login($userID) {
    $userDB = $this->select("select * from player where id='$userID'");
    while ($row = $userDB->fetch()) {...11 lines }
    $userDB = $this->select("select battlegroup.id,battlegroup.strname,battlegroup.intincombat from player join battlegroup on battl
    while ($row = $userDB->fetch()) {...52 lines }

    $_SESSION['player'] = serialize($player);
    $_SESSION['battlegroup'] = serialize($battlegroup);
}
```

[10] I datalogi, i forbindelse med at gemme eller sende data, er serialisering metoden til at konvertere en datastruktur eller et objekt til et format, der fx kan gemmes i en fil eller sendes over et netværk og senere genskabes i samme computermiljø eller i et andet computermiljø. Ved serialisering og deserialisering kan man lave en semantisk identisk kopi af den originale datastruktur.

Programmering af Language

Language-filen starter med at kigge i objektet af brugeren, der er logget ind, for at finde ud af, hvilket sprog vedkommende har valgt.

Herefter indlæser systemet en fil der definerer alle de variable tekststrenge.

Afslufningsvis køres et tjek på om der er tekststrenge uden en definition.
Dette gøres for at sikre at alle oversættelser er fuldt ud opdaterede.
Udførelsen af tjekket foregår ved at have et array af alle variable tekststrenge, og teste om de er blevet defineret, hvis ikke de er, bliver de her defineret med en fejlbesked samt information om hvilket sprog og hvilken tekststreng der er tale om, for at kunne genkende denne i spillet.
Senere kunne en automatisk registrering ske ved fejl.

```php
if (isset($_SESSION['player'])) {
    include_once $_SERVER['DOCUMENT_ROOT'].'/model/Player.php';
    $player = unserialize($_SESSION['player']);
    $lang = $player->getStrLanguage();
} else {
    $lang = "en";
}
include_once "language/symbols.php";
include_once "language/$lang.php";

include_once "language/catcherrors.php";
```

```php
$before = "[$lang]:";
$after = "!";

$var = [    "SETTING_BTN",    "SETTING_LIGHT",    "SETT
    'MENU_BATTLE_SKIP_ANIMATION',    'SIGN_UP_DISPLAY_N
    'SIGN_UP_SECURITY_LEVEL',    'SIGN_UP_SECURITY_LEVE
    "SIGN_UP_TEXT",    "LOGIN_TEXT",    "LOGOUT_TEXT",
    "CREATE_GROUP_HEAL_DESC",    "CREATE_GROUP_CONTROL_
    'GLOBAL_HEAL_LONG',    'GLOBAL_CONTROL_LONG',    'G
    "CREATE_GROUP_SELECT_MAIN_POWERSET",    "CREATE_GRC
    "MENU_MAP"];

foreach ($var as $v) {
    if (!defined($v)) {
        define($v, "$before $v $after");
    }
}
```

Programmering af Battle

Bagsiden af en kamp er XML genereret i PHP. Variablerne $characters, $enemies og $actions indeholder informationer fra databasen sat sammen til strenge med xml-tags omkring.

```php
$xml = "<?xml version='1.0' encoding='UTF-8'?>";
$xml .= "<battle id='$battleID'>";
$xml .= "<battlegroup>$characters</battlegroup>";
$xml .= "<enemygroup>$enemies</enemygroup>";
$xml .= "<actions>$actions</actions>";
$xml .= "</battle>";
header("Content-type: text/xml");
echo $xml;
```

Programmering af Action

En handling i kampen modtages fra kamp-siden via enten et input fra spilleren, eller systemet selv ved handling fra modstanderen.

Al input valideres og renses for forsøg på SQL-injection[11].
Herefter tjekkes for hvem der laver en handling, og hvem handlingen er rettet mod.

Der tjekkes om den handlende har lov til at bruge den brugte kraft, og modtaget en fejlbesked hvis ikke.

```
$res = 0;
$battleID = filter_input(INPUT_GET, 'battle', FILTER_SANITIZE_NUMBER_INT) + 0;
$actorID = filter_input(INPUT_GET, 'actor', FILTER_SANITIZE_NUMBER_INT) + 0;
$actorTYPE = substr(filter_input(INPUT_GET, 'actor', FILTER_SANITIZE_STRING), 0,
$powerID = filter_input(INPUT_GET, 'power', FILTER_SANITIZE_NUMBER_INT) + 0;
$targetID = filter_input(INPUT_GET, 'target', FILTER_SANITIZE_NUMBER_INT) + 0;
$targetTYPE = substr(filter_input(INPUT_GET, 'target', FILTER_SANITIZE_STRING),
$moveX = filter_input(INPUT_GET, 'movex', FILTER_SANITIZE_NUMBER_INT) + 0;
$moveY = filter_input(INPUT_GET, 'movey', FILTER_SANITIZE_NUMBER_INT) + 0;
//$cost = filter_input(INPUT_GET, 'cost', FILTER_SANITIZE_NUMBER_INT) + 0;
if (!empty($battleID) && !empty($actorID) && !empty($actorTYPE) && !empty($powe
    if ($actorTYPE == "p") {
        $SQLactor = 'playeractor';
        $validateSQLactor = "battlecharacter";
        $validateSQL = "powerset ON battlecharacter.fk_powerset = powerset.id O
    } else {...5 lines }
    if ($targetTYPE == "p") {...3 lines } else {
        $SQLtarget = 'enemytarget';
    }

    $fields = ['fk_battle', $SQLactor, 'fk_power', $SQLtarget, 'intmovex', 'int
    $validate = $DB->select("SELECT $validateSQLactor.id FROM $validateSQLactor
    if ($row = $validate->fetch()) {
        $action = $DB->insert("actions", $fields, [$battleID, $actorID, $powerI
    } else {
        $res = "BATTLE_POWER_NOT_FOUND";
    }
} else {
    $res = "BATTLE_MISSING_PARAMETERS";
}
echo $res;
```

[11] SQL injection (fra engelsk at indskyde) er et angreb rettet mod databaselaget i en applikation, ved at indskyde fjendtlig SQL kode i et SQL-kald. Angrebet udnytter en sårbarhed i håndteringen af brugerinput og databasekald.

Efterfølgende udviklingsarbejde

Den fremtidige udviklingsplan er at bruge den samme mængde tid allerede brugt, på at færdiggøre de planlagte basis features, og fører spillet fra version 0.2 til 0.5, hvilket vil blive kaldt EXFU: Alpha.

På dette tidspunkt vil jeg publicere spillet på et domæne ala 'exfu-online.com' og implementere muligheden for at brugeren kan indsende feedback.

Når spillet er gennemtestet, alt grafisk er implementeret og alle fundne sikkerhedshuller er lukket føres spillet i version 0.8 EXFU: Beta.

Her anses spillet internt som færdigt, og fokus føres over primært på fejl og balance mellem de forskellige kræft sæt. Der opfordres stadig kraftigt på feedback fra brugeren.
Under denne periode kan databasen nulstilles efter behov, samt specielle databaser der starter spillerne i en bestemt tilstand, eller sted.
Her vil hele PHP-koden blive gennemgået og ændret til fuldt objekt-orienteret, designet vil blive finpudset og animation vil så vidt muligt føres over i CSS.
HTML vil blive valideret og til sidst optimeres kode og spil-data[12] til at fylde mindst muligt for at optimere siden download-hastighed uden at miste kvalitet.

Når spillet fungerer som ønsket, og alle fundne fejl er blevet rettet. Og tilbagemeldingen fra brugerne generelt er positiv, føres spillet i version 1.0 EXFU

Denne version kaldes internt 'Live' og der fokuseres primært på balance mellem de forskellige kræft sæt, samt opståede fejl.

[12] Her er data alt der ikke er programmeret: Billeder, Lyd mm.

Konklusion

Kommunikation er nøglen til succesfuld udvikling af et spil af denne genre, det er nemt at lave et spil, der er sjovt for én selv. Men når man forsøger at lave et udbredt spil er det vigtigt at høre en masse stemmer, og prøve at appellere til alle.
Dog skal man have en vis vision for spillets retning, for at holde en renere stil, da man ikke kan tilfredsstille alle.

Idéen for den tekniske del af spillet, var dels for at se om det kunne lade sig gøre, og hvorvidt det var praktisk at gøre.
Jeg havde - og har endnu ikke set det gjort før, og da det er min hjemmebane inden for programmering og design skulle det afprøves.

HTML og CSS har stærke fordele frem for andre platforme når det kommer til design, da man hurtigt kan lave brede 'selects' og påvirke lige præcis de elementer man ønsker.

At have JSON eller XML som grundelement for spillet giver det mulighed for at udvikle et interface i php, som kan fungere på næsten hvilket som helst system, i næsten alle programmeringssprog.

Med PHP som backend-engine, kan jeg gemme alle database forbindelser væk, bag en side der er blevet parset.
Sikkerheden overlader jeg i starten til professionelle, da prisen for en webserver helt klart er pengene værd, frem for at håndtere den del selv.

Design, især når man designer mens man programmere har sine ulemper, det man spare i tid mister man i frit design. Man designer ud fra hvad der er muligt, og hurtigst at lave nu.
Og inden spillet kan udgives bør jeg gå tilbage og give mig selv tid til at lave et rigtigt design til siden.

Og hele den grafiske del af spillet generelt tager lang tid. Og igen en bedre plan for hvilke mål jeg skal nå ville have hjulpet en masse, da jeg ikke havde brugt flere dage på at lave det grafiske test-data, som så endte med at blive skåret fra.

Jeg glæder mig til at nå mine delmål i projektet og se om spillet kan komme til at flyde.
Hvis istedet spillet synker, med et brag, ser jeg dette som en del af mit CV, med noget at vise frem.
Programmerings niveauet et ikke ret højt, så der har ikke været mange udfordringer der.
Den største udfordring har været at tage stilling til hvordan spillet skulle fungere.
Jeg har nu gennem de sidste, mindst ti år, forestillet mig forskellige måder spillet kunne foregå, og at skulle vælge et endeligt design, var svært da det ofte betød at man skulle fravælge flere idéer der ligeså vel kunne fungere.
Jeg håber en dag at kunne få en fod indenfor i spilbranchen.
Hvilken titel jeg ender med, ved jeg ikke.

Bilag

To do

Planlægning
#01 Tidsplan
#02 Domænemodel
#03 Usecases-liste

Rapport
#10 Hovedopgave Rapport

Design
#20 Site-design
#21 Character art
#22 Powers art
#23 World art

Database
#30 Opsætning af database
#31 Indsætning af data

HTML/CSS
#40 Opsætning af sider [HTML]
#41 Design af sider [CSS]
#42 Frontend funktioner [javascript/jQuery]
#43 Responsive

Programmering
#50 Opsætning af klasser
#51 Login
#52 Tilmelding
#53 Battlegroup
#54 Quest
#55 Battle
#56 Feed
#57 Spaceship
#58 Planets and solar systems

Usecases

#51	Login			
ID	**Navn**	**Primær aktør**	**Kompleksitet**	**Prioritet**
#1	Log ind	Player	Mellem	1
#2	Log ud	Player	Mellem	2
#3	Oprettelse af ny spiller	Player	Høj	1

#53	Gruppe			
ID	**Navn**	**Primær aktør**	**Kompleksitet**	**Prioritet**
#4	Oprettelse af ny gruppe	Player	Høj	1
#5	Slet eksisterende gruppe	Player	Mellem	3
#6	Valg af character rolle	Player	Lav	1
#7	Valg af character element	Player	Lav	1
#8	Valg af character udseende	Player	Lav	2
#9	Færdiggørelse af gruppe	Player	Høj	1

#54	Quest			
ID	**Navn**	**Primær aktør**	**Kompleksitet**	**Prioritet**
#10	Oprettelse af ny quest	Game Master	Høj	5
#11	Slet eksisterende quest	Game Master	Mellem	10
#12	Start af ny quest	Player	Lav	2
#13	Færdiggørelse af quest	Player	Mellem	2

#53	Gruppe			
ID	**Navn**	**Primær aktør**	**Kompleksitet**	**Prioritet**
#14	Oprettelse af ny battle	Game Master	Høj	5
#15	Slet eksisterende battle	Game Master	Mellem	10
#16	Start ny battle	Player	Lav	1
#17	Gør skade på fjenden	Player	Mellem	1
#18	Tag skade fra fjenden	System	Mellem	1
#19	Brug rolle definerende kræfter	Player	Mellem	1

Tidsplan

	Opstart	Uge	28-29	[11/7-24/7]
Todo ID		**Dato**	**Prioritet**	**Tidsestimering**
#01	Sæt tidsplanen op	11/7	1	15m
#01	Indsæt de første 2 uger	11/7	1	15m
#02	Opsæt basal design	11/7	1	3t
#03	Første udkast af use case-liste	11/7	2	2t
#10	Opsæt rapport	11/7	3	1t
#21	Lav basale versioner af hvert element	12/7	1	1d
#22	Lav basale versioner af hvert element	13/7	1	1d
#23	Byg et psd-dokument til dannelse af nye verdensdele	14/7	3	1d
#20	Lav simpelt design	15/7	1	1d
#02	Færdiggør designet	18/7	1	1t
#30	Lav script ud fra domænemodel	18/7	1	3t
#31	Indsæt minimal data til brug i test	18/7	2	2t
#50	Opret utility classes til database-forbindelse og formularer	18/7	1	2t
#40	Opret login side	19/7	1	2t
#41	Design Login	19/7	2	2t
#42	Login via json	19/7	1	1t
#42	Frontend finish på Login	19/7	3	1t
#51	Opret Login side	19/7	1	2t
#50	Opret model-classer	22/7	1	2t

	Game-site opstart	Uge	30-31	[25/7-7/8]
Todo ID		**Dato**	**Prioritet**	**Tidsestimering**
#20	Finpuds design	25/7	3	1t
#40	Tilmeldings side	25/7	2	1t
#41	Design Tilmeldings side	25/7	2	2t
#52	Opret tilmelding	25/7	2	2t
#42	Tilføj validering samt kald til php	25/7	3	2t
#43	Gør hele sitet, login og tilmelding responsiv	26/7	5	3t
#51	Opret en komplet bruger i session ved login	26/7	2	2t
#20	Design til game-sitet	26/7	2	2t
#40	Opret /game	26/7	1	30m
#41	Opsæt /game samt indsæt dynamisk data	27/7	1	2t
#42	Tilføj frontend funktioner	27/7	2	1t
#43	Responsiv /game-site	27/7	4	30t
#53	Opret gruppe-oprettelse	27/7	1	1t
#40	Opret valg af rolle og powerset	1/8	1	1t
#41	Design valg af rolle og powerset	1/8	1	3t
#42	Programmer funktioner for valg af rolle og powerset	1/8	2	2t
#42	Fortsat	2/8	5	3t
#53	Tilføj gruppe- og character oprettelse	3/8	1	4t
#02	Tilføj klassen Role	4/8	1	15m
#53	Opret gruppe med rolle	4/8	1	30m
#30	Tilpas database med rolle	4/8	1	15m
#31	Ændre midlertidige powerset til mere permanente. Samt giv hver powerset en kræft	4/8	2	1t
#42	Valg af gruppe samt kald til php	4/8	1	1t
#53	Valg af gruppe	4/8	2	30m
#40	Opsæt battle	5/8	1	1t
#41	Design battle	5/8	1	2t

	Spilbar Prototype	Uge	43-44	[24/10-6/11]
Todo ID		**Dato**	**Prioritet**	**Tidsestimering**
#1000	Tilføj mulighed for valg af sprog	24/10	3	1t
#1001	Forbedr design efter at se det med friske øjne, samt ændre al tekst på alle sider fra statisk til dynamisk	24/10	3	2t
#1002	Tilføj mulighed for at gemme kampe i database	24/10	1	1t
#56	Programmer basal Battle-feed med data fra database	24/10	1	2t
#1002	Tilføj Actions til Battle-feed'et	25/10	1	1t
#56	Tilføj Actions samt tilføj informationer nødvendige for GUI'en	25/10	1	3t
#42	Indlæs Battle-feed'et som xml via ajax	25/10	1	3t
#42	Udfyld og opbyg HTML af battle-side med data fra xml	27/10	1	6t
#42	Færdiggør opsætning fra xml	28/10	1	3t
#42	Lav simple animationer der viser Actions	28/10	1	2t
#42	Gør det muligt at starte Action-historien fra et vilkårligt punkt	28/10	2	2t
#55	Tilføj Action-historie til Battle frontend	29/10	3	30m
#41	Design action historie	29/10	1	1t
#42	Lad action-historien vise de seneste handlinger i forhold til tidspunktet, og tilføj dem løbende for hver handling	29/10	1	4t
#1001	Finpuds design af battle	1/11	1	2t
#55	Tilføj basale kræfter	2/11	1	30m
#42	Tilføj ture og lad kræfter kalde en ny action, samt lad fjenden bruge sine kræfter automatisk ved tur	2/11	1	5t

	Prototype forbedres	Uge	45-46	[14/11-25/11]
Todo ID		**Dato**	**Prioritet**	**Tidsestimering**
#56	Tilføj en liste med alle spillere I kampen, samt arranger dem efter tur.	7/11	2	5t
#42	Animeret livs bar, samt andre småjusteringer og optimeringer	8/11	3	6t
#1000	Ændre hard-coded test data til dynamisk data	10/11	1	4t
#1000	Tilføjelse af ny kamp	11/11	1	3t
#1001	Streamline designet af battle	14/11	3	4t
#1002	Tilføj positioner til actions i database	15/11	1	15m
#56	Brug data om positioner fra database i battle-feeded	15/11	1	15m
#41	Vis mulige ryk for den enkelte spil-karakter	15/11	2	30t

#42	Hent alle ryk fra XML samt animer bevægelse ved valg af ryk	15/11	2	4t
#42	Forhindre at flere spillere kan stå på samme felt	15/11	3	1t
#1001	Finpuds designet	16/11	3	2t
#55	Tilføjet fejlbeskeder til Battle	16/11	1	1t
#42	Forbedret Targeting, og klargjort fjendens AI til valg af target	16/11	1	4t
#55	Tilføjet Thread som fjenden bruger til Targeting	17/11	2	4t
#42	Tilføjet AI til fjenden så den altid angriber spilleren med mest Thread	18/11	1	4t
#02	Ændre Domænemodel til at inkludere en gruppe fjender	21/11	1	30m
#1002	Tilføj fjender som grupper i stedet for enkelte entiteter, samt giv diverse 0-værdier for test-data en bedre værdi	21/11	1	1t 30m
#55	Ændre PHP-skabelsen af battle-XML så den henter fjender som en gruppe	21/11	1	2t
#1000	Ændre utility til start af ny kamp	21/11	1	1t
#42	Tilføj mulighed for at vinde eller tabe	22/11	1	5t
#1001	Brugerprofil hvor man kan se simpelt statistik over brugernes bedrifter	24/11	2	4t
#41	Redesign battle user interface	25/11	2	2d

	Out-of-Battle	Uge	**47-48**	**[5/12-16/12]**
Todo ID		**Dato**	**Prioritet**	**Tidsestimering**
#58	Begynd 'Out-of-battle' - Kort over solsystemet	5/12	2	6t
#58	Kort over galakser og planeter	6/12	2	6t
#02	Tilpas Domænemodel med nødvendige attributter	7/12	1	2t
#1003	Push nuværende version som 0.2 Pre-alpha	7/12	2	15m
#1001	Design basal layout til spaceship	8/12	1	3t
#1000	Implementér basal ship	9/12	1	3t

Database script

```
drop database if exists rexfur;
create database rexfur;
use rexfur;

create table galaxy(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    primary key(id)
)engine=innodb;

create table solarsystem(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    intpositionx int(5) NOT NULL,
    intpositiony int(5) NOT NULL,
    fk_galaxy int(11) NOT NULL,
    primary key(id),
    foreign key(fk_galaxy) references galaxy(id)
)engine=innodb;

create table planet(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    inttype int(5) DEFAULT 0,
    intgravity int(5) DEFAULT 100,
    intradiation int(5) DEFAULT 0,
    intposition int(5) NOT NULL,
    introtation int(5) DEFAULT 1,
    fk_solarsystem int(11) NOT NULL,
    primary key(id),
    foreign key(fk_solarsystem) references solarsystem(id)
)engine=innodb;

create table npc(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    inttype int(5) DEFAULT 0,
    strappearancepath varchar(50),
    fk_planet int(11) NOT NULL,
    primary key(id),
    foreign key(fk_planet) references planet(id)
)engine=innodb;

create table questline(
    id int(11) NOT NULL AUTO_INCREMENT,
    strtitle varchar(20) NOT NULL,
    intformerquestline int(11) DEFAULT 0,
    fk_npc int(11) NOT NULL,
    primary key(id),
    foreign key(fk_npc) references npc(id)
)engine=innodb;

create table quest(
    id int(11) NOT NULL AUTO_INCREMENT,
    strtitle varchar(50) NOT NULL,
    intplayerxponcomplete int(11) DEFAULT 0,
    strquesttext text,
    intquestinline int(11) DEFAULT 0,
    fk_questline int(11) NOT NULL,
    primary key(id),
    foreign key(fk_questline) references questline(id)
)engine=innodb;
```

```
create table player(
    id int(11) NOT NULL AUTO_INCREMENT,
    strdisplayname varchar(20) NOT NULL,
    intlogout int(11) DEFAULT 0,
    strlanguage varchar(5),
    intsecuritylevel int(5) DEFAULT 1,
    stremail varchar(50) NOT NULL,
    strpassword varchar(255),
    intantikeylog varchar(255),
    intplayerlevel int(5) DEFAULT 1,
    intplayerxp int(11) DEFAULT 0,
    primary key(id)
)engine=innodb;

create table battlegroup(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    strshipname varchar(20),
    intincombat int(11) NOT NULL DEFAULT 0,
    colshipname varchar(11),
    intshipinternet int(5) DEFAULT 0,
    intshipcpu int(5) DEFAULT 0,
    fk_player int(11) NOT NULL,
    primary key(id),
    foreign key(fk_player) references player(id)
)engine=innodb;

create table role(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20),
    primary key(id)
)engine=innodb;

create table dayjob(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20),
    primary key(id)
)engine=innodb;

create table powerset(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    inttype int(5) DEFAULT 0,
    strimagepath varchar(50),
    straudiopath varchar(50),
    primary key(id)
)engine=innodb;

create table battlecharacter(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    intpowerlevel int(11) DEFAULT 100,
    intstrength int(5) DEFAULT 0,
    fk_role int(11) NOT NULL,
    fk_battlegroup int(11) NOT NULL,
    fk_dayjob int(11),
    fk_powerset int(11) NOT NULL,
    fk_secondarypowerset int(11) NOT NULL,
    primary key(id),
    foreign key(fk_role) references role(id),
    foreign key(fk_battlegroup) references battlegroup(id),
    foreign key(fk_dayjob) references dayjob(id),
    foreign key(fk_powerset) references powerset(id),
    foreign key(fk_secondarypowerset) references powerset(id)
)engine=innodb;
```

```
create table powers(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(40),
    inttotaloutput int(11) DEFAULT 5,
    introunds int(5) DEFAULT 0,
    intcooldown int(5) DEFAULT 1,
    intcost int(5) DEFAULT 2,
    intthread int(5) DEFAULT 2,
    inttarget int(5) DEFAULT 0,
    fk_powerset int(11) NOT NULL,
    primary key(id),
    foreign key(fk_powerset) references powerset(id)
)engine=innodb;

create table enemy(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(20) NOT NULL,
    inttype int(5) DEFAULT 0,
    intrank int(5) DEFAULT 0,
    primary key(id)
)engine=innodb;

create table battlequest(
    id int(11) NOT NULL AUTO_INCREMENT,
    fk_battlegroup int(11) NOT NULL,
    fk_quest int(11) NOT NULL,
    primary key(id),
    foreign key(fk_battlegroup) references battlegroup(id),
    foreign key(fk_quest) references quest(id)
)engine=innodb;

create table enemyofplanet(
    id int(11) NOT NULL AUTO_INCREMENT,
    fk_enemy int(11) NOT NULL,
    fk_planet int(11) NOT NULL,
    primary key(id),
    foreign key(fk_enemy) references enemy(id),
    foreign key(fk_planet) references planet(id)
)engine=innodb;

create table enemypowers(
    id int(11) NOT NULL AUTO_INCREMENT,
    fk_enemy int(11) NOT NULL,
    fk_powers int(11) NOT NULL,
    primary key(id),
    foreign key(fk_enemy) references enemy(id),
    foreign key(fk_powers) references powers(id)
)engine=innodb;

create table fight(
    id int(11) NOT NULL AUTO_INCREMENT,
    strname varchar(25) NOT NULL,
    intenemypowerlevel int(11) NOT NULL,
    primary key(id)
)engine=innodb;

create table battle(
    id int(11) NOT NULL AUTO_INCREMENT,
    fk_battlegroup int(11) NOT NULL,
    fk_fight int(11) NOT NULL,
    primary key(id),
    foreign key(fk_fight) references fight(id)
)engine=innodb;

create table battleenemy(
    id int(11) NOT NULL AUTO_INCREMENT,
    fk_fight int(11) NOT NULL,
    fk_enemy int(11) NOT NULL,
```

```
    primary key(id),
    foreign key(fk_fight) references fight(id),
    foreign key(fk_enemy) references enemy(id)
)engine=innodb;

create table battlestatsnapshot(
    fk_battle int(11) NOT NULL,
    fk_battlecharacter int(11) NOT NULL,
    intpowerlevel int(11) NOT NULL,
    inttalents int(11) NOT NULL,
    primary key(fk_battle,fk_battlecharacter),
    foreign key(fk_battle) references battle(id),
    foreign key(fk_battlecharacter) references battlecharacter(id)
)engine=innodb;

create table actions(
    id int(11) NOT NULL AUTO_INCREMENT,
    fk_battle int(11) NOT NULL,
    playeractor int(11),
    playertarget int(11),
    fk_power int(11) NOT NULL,
    enemyactor int(11),
    enemytarget int(11),
    intmovex int(5) DEFAULT 0,
    intmovey int(5) DEFAULT 0,
    intdate int(11) NOT NULL,
    primary key(id),
    unique key(playeractor,enemyactor,intdate),
    foreign key(fk_battle) references battle(id),
    foreign key(playeractor) references battlecharacter(id),
    foreign key(playertarget) references battlecharacter(id),
    foreign key(fk_power) references powers(id),
    foreign key(enemyactor) references enemy(id),
    foreign key(enemytarget) references enemy(id)
)engine=innodb;
```

Game Design Document

Spil Introduktion

REXFUR er et online turbaseret rollespil, der foregår i en fremtidig verden, og handler om at udforske og bekæmpe diverse livsformer, på forskellige planeter i galaksen, for at finde den mytiske planet REXFUR.

Genre

REXFUR er et online turbaseret multiplayer fantasy rollespil.

Gameplay

I kamp

Spilleren styrer op til fem karakterer, af forskellige roller, hver rolle har et sæt kræfter, som spilleren strategisk skal sammensætte, for at nedkæmpe fjenden.

Mellem planeter

På rumskibet, kan spilleren opgradere ting som raketter, til at reducere tiden mellem planeter, skjold til at reducere skader på rumskibet og øge dets hårdførhed, karakternes rumdragt, så de kan udforske nye farligere planeter.

På rumskibet er der et træningsrum, hvor spilleren kan øve sine kræfter, og styrke karaktererne.

Målgruppe & platforme

Spillet egner sig til folk, som interesserer sig for fantasy genren, der savner et turbaseret spil.

Spillet optimeres til de største webbrowsere, men vil blive udviklet, med senere henblik på udvidelse, til et client-baseret interface til PC.

Look and Feel

Spillet foregår i rummet, med en futuristisk atmosfære og arkitektur i storbyerne.

Udseendet variere fra planet til planet, nogle planeter har mindre ressourcer til rådighed, hvilket giver den et mere faldefærdigt udseende, mens andre har en rigere og mere eksklusiv stil.

Spillet føles stratetisk udfordrende, og efter en kamp mod en svær fjende, vil man sidde tilbage, med en følelse af, at man har gennemført en kompliceret labyrint.

Historie

Den første karakter, spilleren laver, vågner op på et laboratorie uden hukommelse om hvem den er.

Karakteren starter sin færd med at snakke med en forsker, der forklarer at et eksperiment er succesfuldt, og at karakteren nu er soldat for planetens hær.

Efterfølgende bliver karakteren sendt videre til en sergent, der mestre den rolle og element spilleren har valgt. Sergenten lærer karakteren sine første basale kræfter, før han sender karakteren ud på sin første mission.

Planeten er under angreb fra en nærliggende planet, der forsøger at udvinde ressourcer.

Krigen sender karakteren ud på et langt eventyr, hvor spilleren kommer til at styre op til fem karaktere, med forskellige roller og kræfter der tilmelder sig kampen.

Snart bliver det klart at planetens hær behøver et stærkt våben, for at vende slaget, og spillerens gruppe får til opgave at finde beviser på, at den mytiske planet REXFUR eksisterer.

REXFUR er en ældgammel planet, som ikke er blevet sporet, af de galaktiske radarer, i mange tusinde år, og mange mener at planeten er en myte, planeten skulle have været hjemsted for nogle yderest intelligente og stærke krigsguder, med våben teknologi der langt overstiger, hvad der er tilgængeligt, i resten af galaksen.

Spor fundet af spillerens gruppe, peger i en ny retning, der ikke er blevet udforsket, og sender dem nær og fjernt, men REXFUR forbliver gemt indtil videre.

How to Play

Oprettelse

Spilleren starter spillet med, at vælge sin første karakter.
Denne karakter er valgfri, i forhold til hvilken rolle karakteren skal udfylder, i gruppen, og hvilket element, den skal have kontrol over.

Tutorial

Startfasen inden den første rigtige mission, viser spilleren, hvordan den valgte rolle, skal spilles, samt viser, hvad den senere bliver i stand til, at udrette.
De basale kræfter bliver øvet, indtil spilleren føler sig selvsikker på, at den vil kunne, udføre kræfterne med succes.

Spilleren skal kæmpe mod sergenten, der er væsentligt stærkere, men som ikke gør synderligt modstand, alle kræfter tilgængeligt skal afprøves, før tutorialen kan gennemføres første gang.

Kamp

Kampe fungerer, ved at karakteren og fjenden skiftevis vælger en kræft at bruge, denne kræft kan gøre forskellige ting, afhængigt af rolle og fjende type.

Eksempler på kræfter kan være en der sørger for, at karakteren er den mest truende i kampen, og derved får fjendens fulde opmærksomhed, en kræft der gør hele gruppen stærkere de næste par runder, en der beskytter en karakter i et skjold, der mindsker det næste angreb mod karakteren, en der låser fjenden fast, så den ikke kan gøre noget i en runde, eller en kræft der får karakteren til at springe i luften i en stor eksplosion, der gør stor skade på alle fjender, men som gør at karakteren ikke kan gøre noget de næste par runder.

Kampen udkæmpes på et gitter-mønstret rektangel, hvor både karakterer og fjender kan bevæge sig til de omkringliggende felter.

Formålet er så for spilleren, at udtænkte en strategisk plan, og derved bekæmpe fjenden.

Når fjenden falder, er der mulighed for at spilleren kan se statistik over kampen, og måske derved forbedre sig til næste gang.
Samtidig stiger karakterernes styrke, og med tiden lærer de flere kræfter.

Første mission

Den første mission går ud på at skaffe sergenten nogle ting, som udføres ved at snakke med folk i nærheden.
Dette er for at vise at missioner, kan være andet end at bekæmpe fjender.

Første planet

På den første planet går spillet ud på, at samle en gruppe på i alt fem, bestående af en til at tage imod skade, en til at heale skader, og tre til at uddele skade.
Spilleren følger en linje af missioner, der sender karakteren i kontakt med de forskellige gruppemedlemmer.

Nye karakterer tilbydes i sæt af tre, rollen og elementet karaktererne kontrollere, er tilfældigt, så man ikke bare kan udregne og vælge den mest optimale kombination af karakterer.
Spilleren skal så vælge én af de tre, og fravælger derved også de to tilbageværende.

Der er en del tid mellem nye gruppemedlemmer, så spilleren langsomt kan lære at styre flere karakterer samtidigt.

Når spilleren har samlet en gruppe bestående af fem karakterer, starter en afsluttende linje af missioner der sender gruppen i én af flere retninger.

Jagten på REXFUR

Spilleren kæmper sig vej fra spor til spor, hver kamp gør karaktererne stærkere og stærkere, og de lærer nye kræfter i kampen mod, forskellige livsformer på planeterne sporene fører dem til.

Rumskibet

Mellem planeterne kan spilleren træne i en simulator, der på en sikker måde kan vise kampe, der kan justeres i sværhedsgrad, uden at risikere en lang restitutionstid.
Spilleren kan også, til forskel fra rigtige kampe, afslutte disse kampe før tid.

Hvis en eller flere karakterer falder i kamp, bliver deres livspoint sat til 1 og en restitutionstid begynder, karakteren bliver låst til rumskibet, indtil dens livs point er på 100%.

Spilleren kan starte opgraderinger af skibet, der reducere tiden på forskellige dele af spillet.

Teknisk

Spillet bliver udviklet med PHP som backend engine, denne sørger for at holde en database opdateret hver gang, en spiller udfører en handling.

Kampe vil blive sat op som et feed, hvor hver række er en handling fra enten spilleren eller fjenden.
Dette muliggør, at flere spillere kan spille sammen og bruge det samme feed, samt at man senere kan gense kampen.
Et feed gør det også muligt at lave en PC-version som ikke nødvendigvis behøver en webbrowser, for at køre.

Spillets initiale frontend interface vil være en HTML side, sat op i HTML5, alt design vil blive lavet med CSS3, og animation vil blive lavet i en kombination af CSS, javascript og jQuery.

Lyd på siden afspilles af browserens egen afspiller, og bliver styret af jQuery og evt. CSS.

Siden skal optimeres til de største browsere der understøtter HTML5 og CSS3, men bør stadig virke i ældre browsere, dog uden lyd og animation.
Med et feed og en backend engine i php, kan GUI'en udvides til næsten et hvert system, som blot vil fungere, som interface til engine'en.

Animation af karaktererne og kræfterne brugt i kamp, bliver lavet af en sprite af formatet PNG24, der muliggør halvgennemsigtighed, og giver det bedste grafiske resultat.

Spriten skal udvikles som en multianimation, hvor den kan loopes på x-aksen, og karaktererens forskellige positurer ligger på y-aksen.

Systemkrav

Spillet kræver en internetforbindelse for at fungere, samt en browser.
Det er et krav at spilleren har tilsluttet en mus, for at udnytte alle funktioner i spillet, et tastatur kan tilsluttes for at gøre visse ting lettere, men dette er ikke et krav.

For at få den fulde oplevelse af spillet, kræves det at den valgte browser supporterer HTML5 og CSS3, samt at javascript er aktiveret.

Spil Arkitektur

Der vil blive udviklet en domæne-model der beskriver arkitekturen for det fulde spil.

På nuværende tidspunkt kan nævnes, de basale ting som:

Grupper

En spiller kan have flere grupper, men kan kun styre én af gangen.
Hver gruppe består af op til fem karakterer.
Ved oprettelse af en ny gruppe starter spilleren et nyt spil fra start.
Hver gruppe har sit eget rumskib.

Kræfter

Hver kræft er en del af et set af kræfter.
En karakter får tildelt op til to sæt kræfter, hvor de kan trække kræfter fra.
Fjender får tildelt kræfter individuelt, og har derfor hver, et fastsat antal kræfter.

Solsystemer

Planeter er alle en del af et solsystem, og har en fast position og hastighed for rotation, omkring stjernen i midten.
Hvert solsystem er en del af en galakse, og har en fast position heri.
Med tiden kan flere galakser tilføjes som udvidelser, men spillets første version har kun en enkel galakse.

Missioner

Hver mission har en missions type der forklare, om missionen handler om at samle ting, tale med en person eller bekæmpe nogle fjender mm.
Alle missioner er en del at en linje af missioner.

Brugergrænseflade (brugerkontrol)

Oprettelse

Under oprettelsen af en gruppe, får spilleren præsenteret en skærm, hvor den kan vælge hvilken karakter den vil starte ud med.
Dette gøres ved brug af en mus.
Spilleren vil aldrig være i tvivl om hvad der skal gøres, for at komme videre i processen.

Uden for kamp

Uden for kamp bevæger spilleren sin første karakter rundt i verdenen, på samme måde som i kamp, i et gitter-mønstret rektangel, hvor man kan rykke til de omkringliggende felter, dog kan man også bevæge sig diagonalt i denne tilstand.
Hvert felt har en forudbestemt chance for at starte en kamp, dog kan spilleren altid starte en kamp på det nuværende felt karakteren står på, hvis spilleren sådan ønsker.

I kamp

I kamp tømmes skærmen for alt, der ikke har med kamp at gøre.
Det er ikke muligt at afbryde en kamp før tid.
Hvis siden lukkes under kampen, fortsætter den næste gang spilleren logger ind på den pågældende gruppe, og det skabte feed, bruges til at opbygge kampen hvor spilleren slap.

Toppen af skærmen viser en liste over hvem får tur hvornår, og roterer gennem denne liste.

I venstre side kan spilleren se alle karaktere i gruppen, samt deres liv og energi point mm.

Højre side er det samme for fjenden.
Hvis der er flere end én fjende, og den ene bliver besejret, fjernes denne fra listen.
Flere fjender kan løbende komme til kampen med tiden.

Bunden af skærmen viser de tilgængelige kræfter, for den aktive karakter, hvis tur det er.
Når det er fjendens tur til at angribe, fjernes denne bar i bunden.

Rumskib

Rumskibet består af to rum, hver med sit eget formål.

Hoveddækket viser information om rejsen forud, her kan man se præcis hvornår man ankommer til destinationen.
Det er også her man kan se hvor længe hver karakter skal bruge, for at blive klar til kamp, hvis denne er faldet i kamp for nyligt.
Opgraderinger på skibet styres også herfra.

Det andet rum er trænings-simulatoren, hvor spilleren sikkert kan skabe sine egne designede kampe, fx kan en særlig simulation der tester gruppens udholdenhed, ved løbende og konstant at øge hvor meget skade der bliver udsendt, spilleren skal så se hvor mange runder den kan overleve.
Før eller siden vil en sådan kamp tabes, da det er den eneste måde at afslutte kampen, men da det blot er en simulation, er der ingen ventetid efter et sådan tab.

Artwork (design)

Designet på siden skal være meget simpelt, uden en masse forstyrrende elementer.
Dette tema er en gennemgående faktor for al visuelt, der ikke omhandler karakterer, fjender og de kræfter der bruges.

Oprettelse

Ved oprettelse af en ny gruppe vises et gitter, med alle roller og elementer der kan vælges.
Når spilleren klikker på et felt, vises den valgte karakter i højre side, samt en række informationer om, hvilke fordele og ulemper, der er ved denne karakter.
Hvis karakteren godkendes, skal spilleren vælge hvilken sekundær rolle og element karakteren skal have.

Uden for kamp

Når spilleren er uden for kamp, skal det være muligt at åbne et kort hvor spilleren kan zoome ud til Galaksen, og zoome ind på andre solsystemer og planeter.

Øverst i højre hjørne skal der være et link til en menu, samt hurtige genvejsknapper til at slukke for lyd mm.

Spilleren skal have mulighed for at åbne et index over alle fjender i spillet, men kan kun se informationer om de fjender der er bekæmpede.

Spilleren kan også se informationer om alle karaktererne i gruppen, fx hvor stærke de hver er, hvilke kræfter de har mm.

I storbyer skal en chatbox poppe op, nederst i venstre hjørne.
Denne chatbox kan bruges af alle i en storby, eller ombord på deres rumskibe.

I kamp

Når en kamp startes, vises en indlæsningsskærm, der bagved fjerner alle knapper fra menuen, på nær lyd kontrol, den fjerner også kortet, og chatboxen hvis denne er aktiv.

I venstre side vises en liste over gruppens karakterer, med information om liv og energi, samt hvilke ting i kampen der påvirker deres styrke mm.
Liv og energi vises med et rektangel, der bliver kortere mod venstre som værdien falder.
Ting der ændre karakterernes styrke, vises med en firkant til højre for karakteren, gode ting vises med en blå kant, og dårlige ting vises med en rød kant.

Det samme er gældende for fjendens gruppe i højre side.

I toppen vises alle deltagere i kampen i små firkanter, med deres protræt samt en blå eller rød kant, afhængeligt af om det er fjende, eller karakter fra gruppen.
Hver gang en deltager angriber, skubbes deltageren ned af listen, afhængigt af, hvor stort et angreb der blev brugt.
Små ubetydelige kræfter skubber mindre, og giver mulighed for at angribe med denne karakter igen hurtigere, mens stærke kræfter, giver en længere ventetid mellem angreb.

I bunden vises en række af firkanter, der hver har et billede af den kræft, der refereres til.
Et lille vindue med en beskrivelse af kræften, vises når spilleren fører musen over en kræft.
Når en kræft vælges, trækkes hele bundbaren ned udenfor skærmen og deaktiveres, indtil det er spillerens tur igen.

Samtidig afspilles en animation, der viser den grafiske repræsentation, af den valgte kræft.
Dette foregår i midten af skærmen, på "spillepladen", der består af et rektangel udfyldt med en masse felter i et gitter-mønster.
Disse felter er hver en plads, hvor karaktere og fjender kan stå.
Det er ikke muligt, at være flere på et enkelt felt.

Rumskib

Designet af GUI'en ombord på rumskibet, skal være futuristisk, det skal være et mørkt design med "lysende" kanter. Spilleren skal kunne vælge farve på rumskibet, dette påvirker hvilken farve kanterne på designet "lyser".

Da kamp-simulatoren er netop dette, en simulator, skal rummet kampen holdes i, klædes i et net af turkise streger.
Resten af elementer fra kamp-skærmen, skal også klædes i et simulations-design.

Tekniske krav for grafik

Måden animationer laves på, er ved at have et sprite, for hver karakter og fjende.
Et sprite er et billede, med en masse små billeder i.
På spritens x-akse, skal der være en levende animation der kan loopes, det vil sige, at det sidste billede på en række, hænger sammen med det første på rækken.
På y-aksen er en række forudbestemte positurer.
Fx er første række altid en afslappet normal holdning, denne animation kører når karakteren eller fjenden, bare står uden at gøre noget.
Andre positurer er ting som:
Angreb fremad, angreb omkring, ramt af et svagt angreb, ramt af et stærkt angreb, fastholdt, døende og bekæmpet.

Dvs. at valg af, hvilken animation skal afspilles, kan generaliseres til et tal, i stedet for en række i en database for hver karakter eller fjende.

Det samme er gældende for hvert sæt kræfter, alle har samme antal kræfter, så det kan laves på samme måde.

For at undgå skarpe kanter, der ikke passer ind i baggrunden, skal alle billeder være i formatet PNG24, med gennemsigtighed.
Fordelen ved dette format er, at der er 256 toner af alpha farve, altså hvor gennemsigtig den enkelte pixel kn være.
Ulempen ved formatet er størrelse, i tilbage da formatet blev lavet, var den gennemsnitlige download-hastighed meget lavere end i dag, så det kunne tage flere minutter at indlæse et enkelt sprite.
Men i dag er dette problem ikke eksisterende, og selv indlæsning af flere sprites i en enkel kamp, klares på sekunder.

Karakterer

På nuværende tidspunkt er der ikke en fast liste over hvilke, eller hvor mange karakterer spilleren kommer i kontakt med.

Der bliver udviklet en formular, for skabelsen af nye karakterer.
Denne formular indeholder alt omkring denne karakter, med navn, alder, hjemplanet mm.
Samt en baggrundshistorie og ønsker for karakteren.
Dette vises ikke nødvendigvis i spillet, da hemmelige planer ellers kunne blive afsløret for spilleren.

Der er ingen baggrundshistorie for de karakterer spilleren kontrollerer, her fortælles deres historie gennem spillet.

Tank

Tanken er gruppens skjold, hvis primære opgave er, at tage alt opmærksomheden fra fjenderne, så største delen af indgående skade lander her.
Dette gør det nemmere for resten af gruppen, da spilleren kan planlægge mod store angreb, ved at bruge de overlevelseskræfter, tanken er propfyldt med.

Tanken angriber fjender på tæt hold, og har en tung rusting med et stærkt skjold.

Tanken kan vælge mellem Damage eller Support som den sekundære opgave.

Vælges Damage, yder Tanken nu også til gruppens totale skade uddelt, og fjender falder hurtigere, Damage hjælper tanken med at være den største trussel i kampen, og på den måde kan den nemmere holde fjendernes opmærksomhed.

Vælges Support, får Tanken i stedet flere kræfter til at overleve, og kan nemmere sænke indgående skade på andre gruppemedlemmer.

Support

Support rollen hjælper hele gruppen, ved at kunne vælge hvilke ting, der skal være stærkere på et givent tidspunkt.
Dette hjælper gruppen ved fx, at gøre healing stærkere de næste tre runder, i en periode hvor hele gruppen tager høj skade, hvor efter Supporten kan øge hvor meget skade gruppen uddeler, i en periode hvor fjenden tager øget skade.

Supporten angriber på fjender på tæt hold, mens den kan styrke gruppen på afstand.
Rustningen er den stærkeste blandt alle roller, men uden et skjold.

Supporten kan vælge mellem Tank eller Healer som den sekundære opgave.

Vælges Tank, kan Supporten nu trække fjender væk fra tanken i et par runder, samt overleve disse runder.
Supporten kan ikke holde sig i live for altid.

Vælges Healer, får Supporten nu mulighed for, at forhindre skade på gruppemedlemmer, og kan hjælpe healeren med små heals her og der.

Healer

Healerens primære opgave er, at holde gruppen i live.
Det hjælper også spilleren med fx, at lade en Damage stå stille og blive ramt af ting på jorden længere, hvis healeren kan holde den i live, dette gør at gruppens totale uddelte skade, ikke går ned pga. gruppens behov for at flytte sig.

Hvis alle har fuldt liv, og healeren ikke har noget at heale, kan den angribe fjender på afstand ligesom al healing også er på afstand.
Healeren har en let rustning.

Healeren kan vælge mellem Support eller Control som den sekundære opgave.

Vælges Support, får healeren mulighed for, at styrke gruppen med mindre effektive, men permanente kræfter.

Vælges Controller, kan healeren nu hjælpe med, at gøre fjenden svagere, og derved sænke indgående skade.

Controller

Controllerens opgave er, at kontrollere fjenderne på flere måde, enten ved at stoppe dem i at angribe, eller gøre deres angreb svagere.
Samtidig gør Controlleren væsentligt skade på fjenderne.

Controlleren angriber fjender på afstand, og har den letteste rusting i gruppen.

Controlleren kan vælge mellem Healer eller Damage som den sekundære opgave.

Vælges Healer, får Controlleren mulighed for, at hjælpe healeren med større heals af og til.

Vælges Damage, ændres controllerens basale kræfter til, også at yde større skade.

Damage

Opgaven for Damage'n er meget simple, gør skade til fjenden.
Kompleksiteten kommer i, at vælge de rigtige kræfter for, at gøre så meget skade som muligt.

Damage'n angriber fjender på afstand, og har en mellem tung rusting.

Damage'n kan vælge mellem Controller eller Tank som den sekundære opgave.

Vælges Controller, kan Damage'n nu også bidrage med små hold her og der.

Vælges Tank, får Damage'n mulighed for, at trække opmærksomhed fra fjender, dette gøres på afstand da Damage'n ikke så let kan holde sig i live, hvis fjenden indhenter Damage'n.
Udover gives mulighed for at heale sig selv.

Level Design

Kampe

Designet af kampenes "spilleplade", afhænger af den planet man er på.
Nogle vil have naturlige områder på pladen, der er gode eller dårlige.

Nogle vil have vand eller lava, der gavner og skader forskellige karakterer, afhængigt af hvilket element de kontrollere.

Nogle har klipper og vægge, der gør det umuligt at gøre skade fra afstand.

Fjenderne kan variere, i hvor mange fjender der er fra start, hvor mange der er i løbet af en kamp, og hvor stærke de hver er og hvilke kræfter de har.

Missioner

Missioner varierer mellem, nogle hvor man skal skaffe en ting til en kontakt, nogle hvor man blot skal tale med en ny kontakt, nogle hvor man skal bekæmpe en bestemt fjende, nogle hvor man skal bekæmpe flere fjender, af en bestemt type eller på en bestemt planet.

Nogle er en enkel mission på en linje, andre har flere i træk.

Globale grafiske elementer

Uden for kamp kan spilleren bevæge sin karakter rundt på "spillepladen", neutrale ting som træer, bygninger mm. låser feltet og spilleren må gå udenom, ligeledes kan man ikke bevæge sig over bjerge.

Der er kun bestemte steder på planeter, man kan lande sit rumskib.

Karakterer

Når man interagere med en karakter, kommer som minimum en kort besked frem fra dem.
Kan man intet med denne karakter, får man blot en kort kommentar.

Hvis karakteren har muligheder for at samtale, vises disse i bunden af beskeden.

Har karakteren en mission på sig, vil dette vise sig i form af, en farvet diamant over deres hoved.
Farven på diamanden, afhænger af missionens status.

"Spillepladen"

Både i og uden for kamp, kan felter individuelt låses, for at skabe en følelse af, at det grafiske der ses, virkeligt er der.

Samtidig giver det i kamp, mulighed for, at skabe lidt gameplay, i form af små labyrinter eller forhindringer, der gør, at en kamp vil føles anderledes hver gang.

Når man kanten af rektanglet, uden for kamp, kan man gå ind, på det ved siden af liggende kort.
Når man kanten i kamp, kan man kun følge kanden rundt, eller vende om.

Solsystemer

Galaksen er enorm, hvis man brugte realistiske rejsetider, ville selv den nærmeste planet tage flere måneder at besøge, så et realistisk spil ville ikke være sjovt, når det ikke er det primære gameplay.

Rejsen mellem planeter bliver skåret ned til minutter, mens solsystemer kan tage fra timer til enkelte dage afhængigt af afstanden.

For at skære ned på tid, brugt på at rejse frem og tilbage, har hvert solsystem en planet med en storby.
I hver storby står en teleporter, der kan teleportere karakterens rumskib, til andre storbyer. Kravet for at kunne teleportere til en ny destination, er at man har rejst væk fra den.
Dvs. at hver gang man har opdaget en ny storby, og skal forlade denne, vil den hurtigste måde være, at rejse tilbage til en anden storby.

For at undgå spørgsmål omkring, hvorfor man ikke bare kan lande sit rumskib, lige ved siden af det sted, man skal besøge, er det kun muligt, at lande på forudbestemte steder, dette forklares ved, at der er et globalt sikkerhedssystem, i hele galaksen, der beskytter mod uvedkommende besøg og ting som komet- og meteornedslag.

Udvikling

Beta

Inden spillet bliver udgivet i version 1.0, vil spillet være i en beta version 0.x.
Udviklingen vil foregå ved, at der er faste datoer, hvor en ny version, skal vises til spillerne.
På disse datoer, skal spillet være i en fungerende tilstand, og ting der ikke er klar bliver rykket til næste version.

Der skal kommunikeres med spillerbasen, omkring hvad der er kommet af nye ting, samt hvad der er test-fokus for denne version.

Samtidig med, at der bliver udviklet mere af spillet, for hver version, bliver der også kigget på feedback, for de allerede testede systemer.
Så hver nye version, vil have rettelser af både systemer og fejl, samt introducere et nyt system for spillerne at teste.

Når størstedelen af spillet, er blevet udviklet vil alle systemer fortsat rotere i en test, feedback, fix loop.

Vedligeholdelse

Når spillet bliver udgivet i version 1.0, vil der stadig blive kigget på feedback løbende, eftersom der gerne skulle komme nye øjne til.
Især ting som, balance mellem de forskellige spilbare karakteres kræfter ændres her, hvis nogle svinger for meget i både den gode eller dårlige retning.

Rapporterede og fundne fejl rettes også løbende.

Videreudvikling

Så snart spillet udgives i version 1.0, starter processen for næste version, der tilføjer mere til historien, giver spillerne mere styrke og introducere nye eller forbedrede systemer.

Udviklingen af de nye systemer, vil ske sideløbende på en anden server, så der ikke interfereres med den forrige version.

Audio & lyd effekter

Lyd i spillet kommer til, at være samme stil som gamle 8-bit konsoller tilbage i 80'erne.

Kvaliteten vil være bedre, men længden og evnen til at loope vil forblive.

Musik genren i spillet vil være instrumental og spænder fra rock til klassisk af forskellige stile og tempoer, der er passende i forhold til, hvad der sker på skærmen og omgivelserne, samt hvad spilleren har til opgave.

Musik

Hver planet har en liste, af hvilke stykker musik der kan spilles.

Da musikken kan loope, kan den afspilles så længe man vil.

Omgivelse

Hver planet har en lydfil af omgivelser.

Volumen på omgivelses lyde, vil som standard være lavere end alt andet.
Lyden kan indeholde ting som vand, vind og andet vejr, insekter og andre dyr der kunne lure i baggrunden.

Tale

For at gøre spillet lettere, at udgive på forskellige sprog, vil al tale og lyde fra fjender, og andre karakterer, være en serie af biplyde i stil med morsekode, dog med en mindre kontrast mellem lyd og ingen lyd.

Lydens længde afhænger af sætningen der bliver "sagt", dog til et vis punkt.

Kræfter

Kræfternes lyde vil være forskellige eksplosioner og "swoosh"-lyde.

Ligesom en sprite, laves lydene af kræfter på en linje, med faste intervaller mellem hver kræft.

Tekniske krav for lyd

Tale skal laves som en lang række af morsekode, der skal ikke være gentagelser i løbet at lydklippet, men slutningen skal passe sammen med starten.

Forskellige stemmer laves med forskellige toner brugt i lydklippet.

Kræfter skal have en bestemt længde, med plads til fx 10 kræfter af 5 sekunder hver.
Positionen af kræfterne er vigtig, da fx kræft nr. 2 altid vil starte på 5 sekunder mærket og kører indtil 10 sekunder mærket.
På den måde spares en masse data på database siden.

Hver type lyd skal kører i sin egen afspiller, og spilleren skal kunne kontrollere volumen for musik, omgivelse og kræfter.

Globale auditive elementer

Hver planet har en række musik stykker, samt en omgivelses-fil der kan afspilles, flere planeter kan have den samme stykke musik eller omgivelses-fil.

Alle karakterer og fjender, har en tale-fil.

Fil-type og kvalitet er ubestemt på nuværende tidspunkt.

Menu Fx

Grundet platformen, vil menuer ikke indeholde lyd ved navigation, da man ikke bevæger sig ét menupunkt af gangen.

En lyd vil blive afspillet ved ændring af lydstyrke.

Marketing

Key Features

Styr en gruppe på op til 5. Styr dit eget rumskib og opgrader det.

Ny visuel teknologi. Udforsk en galakse.

Kontroller elementerne. Besejr de største fjender.