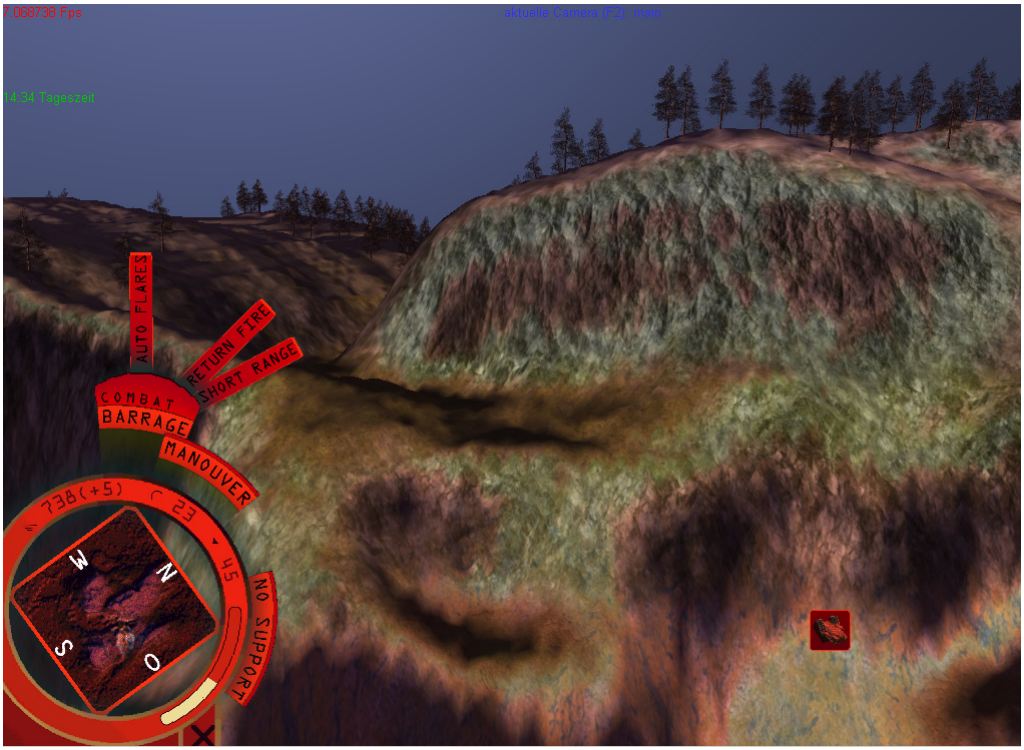
GRAPHICAL USER INTERFACE

for

Planet X – the Alliance

a science fiction real-time strategy game



Max Krichenbauer
mtd07038

HINTERGRUND

Planet-X und Titan-Works

Das Titan-Works Game Develpement Team ist ein unabhängiges Entwicklerstudio, bestehend hauptsächlich aus Studenten, Schülern und Hobbyisten. Organisiert über das Internet arbeiten die oft wechselnden Mitglieder des Teams zusammen an dem PC-Spiel: „Planet-X – the Alliance".
Es handelt sich dabei um ein 3D-Echtzeitstrategiespiel im Sciencefiction-Szenario. Die Handlung dreht sich um den Planeten Tisanis-10 (kurz: Planet-X), auf dem 3 Fraktionen um ihr Überleben und die Vorherrschaft über den Planeten kämpfen: die naturverbundenen Tisaner, die terrestrischen Menschen, und die kriegerischen Hydras.
Seit 2006 arbeite ich als Graphiker an diesem Projekt mit.

Meine Tätigkeit beschränkte sich bisher jedoch auf Concept-Art und Modeling.
Da noch sehr wenig über das Design oder die Implementierung der Benutzeroberfläche feststand, war diese EWG-Projektarbeit der optimale Anlass tiefer in die Technik und das Game-Design einzusteigen. Da mir von den anderen Teammitgliedern freie Hand bei Konzeption und Umsetzung gelassen wurde war Planet-X das ideale Test-Gelände für meine Interface-Ideen.

Game Design

Obwohl von der technischen Seite des Spiels bisher noch sehr wenig fertiggestellt wurde, existieren bereits zahlreiche Game-Design Konzepte und Vorgaben. Das User Interface musste sich nun teils an dem bestehenden Konzept orientieren, es teilweise jedoch auch mitgestalten. Denn die Benutzeroberfläche ist der Zugang zum Spiel; sozusagen das Tor zur Spielmechanik. Beide Seiten müssen sich also möglichst entsprechen.

Ein zentraler Punkt des Gamesplays ist das weitreichende Micro-Management: Die Komplexen Industriekreisläufe und Truppenbewegungen sollen sich bis ins kleinste Detail hin steuern lassen.
Diese Detailentscheidungen des Spielers sollen starken Einfluss auf den Verlauf der Spielpartien und ihren Ausgang haben. Das Können eines Spielers soll also auch durch seine Fähigkeiten in der Feinsteuerung der Spielmechanik gemessen werden. Zusätzlich zu seinem taktischen und strategischen Denkvermögen.
Daraus ergeben sich auch ganz spezielle Anforderungen an die Bedienoberfläche des Spiels: sowohl makroökonomische als auch mikrostrategische Entscheidungsmöglichkeiten müssen dem Spieler in einfacher, intuitiver Form ermöglicht werden. Da oft wenige Sekunden über Sieg oder Niederlage entscheiden können, darf es den Spieler nur wenige Mausklicks kosten, seine Strategien und Pläne umzusetzen.
Außerdem muss das ganze auch noch gut aussehen, in zeitgemäßer Graphik präsentiert werden und angenehm zu bedienen sein.

Technik

Das Spiel verwendet seine eigene Grafikengine; bzw. Multimedia-Engine, denn Sound und Text sind auch bereits integriert. Diese Engine (MME genannt) wurde in C++ geschrieben und verwendet Microsoft Direct-X als Schnittstelle zum Betriebssystem (Windows). Die MME besteht aus mehreren spezialisierten Modulen wie „Input“ oder „Math“ die von einem Kernmodul („Core“) verwaltet werden. Daher war es nötig das User Interface den Regeln für solche Module entsprechend zu konzeptionieren und gezielt in die Engine einzubinden, ohne deren Funktionsweise zu stören.

In der weiteren Spielentwicklung muss dieses Modul dann vom eigentlichen Spiel durch den Engine-Core hindurch erreichbar und programmierbar sein. Schließlich weiß nur der Game Developer welche Einheiten sich gerade auf dem Schlachtfeld befinden und welche Panzer die angewählte Fabrik bauen kann. Das User Interface wird damit modular und universell einsetzbar und kann auch bei einem starken Kurswechsel im Game-Design angepasst werden.

MENSCH–MASCHINE INTERFACE

Anforderungen

Eine Schnittstelle ist eine Verbindung für zwei oder mehr Kommunikationspartner, die nicht direkt miteinander kommunizieren können. Ein gutes Interface ist also eine gute Verbindung – für beide (d.h. alle) Seiten. Es macht die Kommunikation einfach und effizient.

Für jeden Kommunikationspartner stellt das Interface den Gesprächspartner dar. Er spricht also mit einer „Back Box“ - der ominösen „Maschine“. Und dass soll möglichst angenehm sein – für alle Beteiligten (außen dem Interface selbst natürlich).

Also Identifiziert man zunächst alle beteiligten Parteien und lernt sie kennen – denn für gewöhnlich wollen sich die Parteien gegenseitig nicht kennen lernen. Wer sind also diese „User“. Und was wollen sie eigentlich?

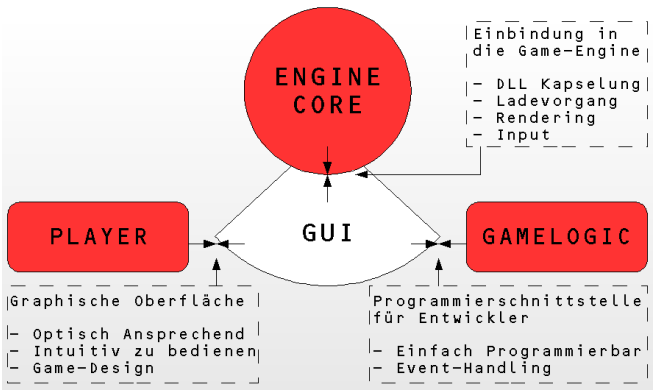
Schnittstellen

Der Spieler:
Ein Spieler war für Planet-X von Anfang an fest eingeplant. Da er gleichzeitig unsere Haupt-Zielgruppe darstellt (neben potentiellen Investoren), fokussiert sich das gesamte Projekt auf diese Person. Und da die Graphik für ein Computerspiel den heißesten Draht zum Konsumenten darstellt, konzentrieren sich unsere Design-Bemühungen auf diese Verbindung. Es ist also die einzige graphische Schnittstelle.

Die anderen beiden User stehen nur dabei, weil auch hier Schnittstellen entworfen werden mussten – nur eben Programmierschnittstellen (APIs).

Der Game Developer:
Irgendjemand muss den Spieler unterhalten. Wir hoffen für solche Zwecke ein Spiel entwickeln zu können. Leider muss das auch jemand machen. Der Game Developer ist daher Hauptkommunikationspartner des Spielers. Er ist Storyteller, Mitspieler, Antagonist und Lehrer in einer Person. (Ja, wir hoffen wenigstens eine Person für den Job zu bekommen).
Da ich das (glücklicherweise) nicht sein werde, muss das User Interface auf seiner Seite leicht zu programmieren sein – er wird es brauchen wenn er den Spieler unterhalten will.

Der Engine Designer:
Das User Interface ist natürlich nur eines der vielen Komponenten, aus denen ein Spiel besteht. (Knöpfe und Hebel allein unterhalten nur kurzzeitig). Das Projekt muss sich also in ein bestehendes Computerprogramm einbauen lassen.
Getragen wird das Spiel von der Game-Engine, die sich um Graphik, Benutzereingaben, Ton, Netzwerkverbindungen und vieles andere kümmert.
Das User Interface muss also mit den Komponenten der Engine verknüpft werden, damit das Zusammenspiel lückenlos funktioniert. Es muss graphische Ausgaben auf dem Bildschirm erzeugen können und die Benutzereingaben empfangen und an die Spiellogik weitergeben können, ohne die Funktionsweise der Engine zu behindern.
Dies ist die technischste und daher für Menschen unverständlichste Schnittstelle. Was kein weiteres Problem darstellte, da Engine-Entwickler sowieso nur noch zu geringen Teilen echte Menschen sind.



SPIELEOBERFLÄCHE

Zielsetzung

Im Gegensatz zu Action-Spielen, die hauptsächlich auf Reaktionsfähigkeit beruhen, ist für ein Strategiespiel eine ergonomische Benutzeroberfläche unverzichtbar. Der Spieler muss ständig komplexe und weitreichende Entscheidungen treffen, die den Ausgang der Partie bestimmen können.
Dazu muss er zu jeder Zeit mit einer großen Menge an Informationen versorgt werden. Ohne ständigen Überblick über Ressourcen, Wirtschaftsabläufe, Truppenbewegungen und Kampfgeschehen kann er seine Einheiten nicht zum Sieg führen. Er darf aber auch nicht mit Informationen überschwemmt und damit verwirrt werden.
Umgekehrt muss er aber auch in der Lage sein, seine getroffenen Entscheidungen und Pläne möglichst unkompliziert und schnell umsetzen zu können. Das Spiel soll ihn dabei so gut wie möglich unterstützen.

Das User Interface dient also als 2Wege Kommunikationskanal.

Der Spieler soll die Rolle eines großen Heerführers einnehmen. Das heißt ihm obliegen die großen und wichtigen Entscheidungen. Er soll so wenig wie möglich mit Kleinkram belästigt werden.

Als entsprechendes Vergleichsbild sei der Kapitän eines Schiffes genannt. Der Kurs wird zwar von der Story des Spiels vorgegeben, der Kapitän behält in jedem Augenblick jedoch die volle Kontrolle über das Schiff und wird dabei von der Crew unterstützt. Dadurch kann er seine Entscheidungen zentral von der Brücke aus treffen und immer so viele Details regulieren, wie er gerade für nötig hält. In zeitkritischen Momenten mag ein grobes „hart Steuerbord“ für den Augenblick völlig ausreichen. Feinere Kurskorrekturen können später jederzeit vorgenommen werden – wenn der Kapitän denn Lust und Zeit hat.

Konzept

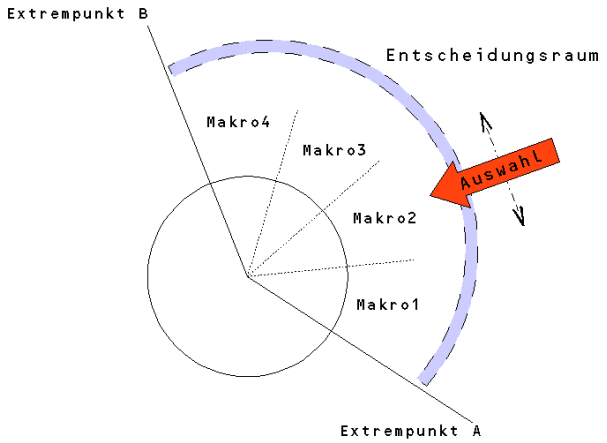
Das Vorbild für das User-Interface war daher ein zentrales Kontrollinstrument aus der Schifffahrt: der Maschinentelegraph. Über diesen Apparat werden Befehle und Meldungen über die Geschwindigkeit des Schiffes zwischen Brücke und Maschinenraum übertragen.

Hier kann der Spieler sowohl die aktuell gültigen Befehle ablesen, als auch neue Befehle geben.

Die abgestufte, sichtbar räumlich angeordnete Auswahl ist für den Benutzer intuitiv verständlich – sie findet sich überall im Alltag; etwa Drehregler für Herdplatten oder Gangschaltungen im Auto.

Diese Abstufungen werden dabei durch Makro-Instruktionen umgesetzt: jeweils ganze Pakete von Details werden zu mehr oder minder sinnvollen Gesamtkonzepten zusammengefasst – wie etwa „halbe Kraft Voraus“. Das Makro wird dann von der Spiellogik wieder in Einzelentscheidungen zerlegt; hier also in konkrete Befehle für die Maschinisten.

Legt man zunächst zwei Extrempunkte in den Entscheidungsmöglichkeiten des Spielers fest (wie „Volle Kraft Voraus“ und „Volle Kraft Zurück“), so lassen sich die entworfenen Makros räumlich dazwischen anordnen. Durch die Sortierung wird das System intuitiv bedienbar: „Stopp“ liegt logischerweise genau zwischen „Volle Kraft Voraus“ und „Volle Kraft Zurück“.



Um dem Spieler nun wieder die Kontrolle über die Details zu ermöglichen, kann er die Makro-Entscheidung in entsprechende Sub-Entscheidungen aufspalten lassen, und diese durch Wahl eines Sub-Makros einzeln Treffen.
Im genannten Beispiel würde sich die Gesamt-Maschinenleistung etwa in die Einzel-Leistungen für die linke und die rechte Maschine aufspalten.

Nach seinem Befehl „halbe Kraft voraus“ präzisiert der Spieler nun „rechte Maschine: große Fahrt“ und „linke Maschine: kleine Fahrt voraus“. Nach demselben Prinzip lässt sich auch das Sub-Makro wieder zerlegen, bis jedes Detail wieder einzeln festgelegt werden kann. Da der Spieler stets den Überblick über viele Aufgaben behalten muss, unterstützt ihn dieses gestaffelte System bei der Umsetzung seiner Pläne. Zum einen erlaubt es ihm schnelle Entscheidungen zu treffen, zum anderen schützt es ihn davor in der Hektik aus versehen unsinnige Befehle zu geben oder sich in Detailfragen zu verstricken.
Dadurch erleichtert es auch Einsteigern den Zugang zum Spiel, da sie nicht mit zu vielen Informationen und Möglichkeiten erschlagen werden. Das Spiel übernimmt einfach so lange sie Details für sie, bis sie in der Lage sind, ein besseres Mikromanagement zu führen als die vorgefertigten Makros.

Bei der enormen Freiheit im Mikro-management von Planet-X kommen schnell enorme Mengen an Makros zusammen. Es wäre unsinnig zu versuchen für Jedes ein Icon zu finden. Zum einen sind Makros eher abstrakte Konzepte und selten wirklich treffend umzusetzen, zum anderen müsste der Spieler erst die Bedeutung jedes Icons auswendig lernen.

Die Verwendung von beschreibenden Schlagworten erschien daher geeigneter. Texte sind jedoch erst ab einer gewissen Größe gut lesbar. Ein beliebig tiefes optisches Verschachteln von Makros und Sub-Makros ist daher praktisch nicht mehr möglich. Das Stört jedoch das Konzept nicht besonders, da sich bereits in drei Hierarchieebenen sehr viel Mikromanagement unterbringen lässt.

Beispiel Einheitenkontrolle

In Planet-X soll der Spieler detaillierte Kontrolle über seine Truppen gegeben werden. Im speziellen einstellbar sein, wie sich die Einheiten verhalten, wenn vom Spieler keine konkreten Befehle gegeben werden.

Werden die Truppen etwa überraschend Angegriffen, könnten sie versuchen ihre Stellungen zu halten oder sich zurückziehen. Je nach Situation kann das eine oder andere Verhalten besser ins Gesamtkonzept des Spielers passen: das Gebiet zu verteidigen oder die Truppen zu schonen.

Noch komplexer wird das Problem, wenn Munitions- und Treibstoffverbrauch berücksichtigt werden müssen oder einige Einheiten über Spezialfähigkeiten verfügen – etwa andere Einheiten zu reparieren.

Wann sollen die Fahrzeuge beginnen Treibstoff zu sparen?
Sollen Spezialfähigkeiten automatisch eingesetzt werden oder nur auf ausdrücklichen Befehl des Spielers?

Die meisten Strategiespiele bieten keine Möglichkeit dieses Truppenverhalten zu beeinflussen. Ein mehr oder minder sinnvolles Verhalten ist fest einprogrammiert und kann nicht verändert werden. Es wird darauf gebaut, dass der Spieler bei unvorhergesehenen Ereignissen selbst die Kontrolle übernimmt und seine Einheiten manuell befehligt.

Im Echtzeitstrategiespiel „Supreme Commander“ (2007) wird dieses Konzept konsequent umgesetzt:
Fahrzeuge und Schiffe bleiben ohne einen manuellen Befehl stets an Ort und Stelle stehen. Sie feuern zwar auf alle feindlichen Einheiten die sich in Reichweite befinden, Ausweichbewegungen werden jedoch nicht vorgenommen, was sie zu leichten Zielen für die gegnerische Artillerie macht.
Auch Spezialfähigkeiten müssen vom Spieler stets manuell befohlen werden.

Ein Reparaturfahrzeug repariert also nur dann andere Panzer, wenn diese einzeln angeklickt werden (Gerade bei den enormen Truppenmassen in „Supreme Commander“ eine zeitaufwändige Tätigkeit).
Abfangjäger werden bei feindlichen Luftangriffen automatisch gestartet um die Eindringlinge zu bekämpfen. Auch dann, wenn sie hoffnungslos unterlegen sind. Seine Truppen notfalls zurückzuziehen obliegt also immer der alleinigen Kompetenz des Spielers.

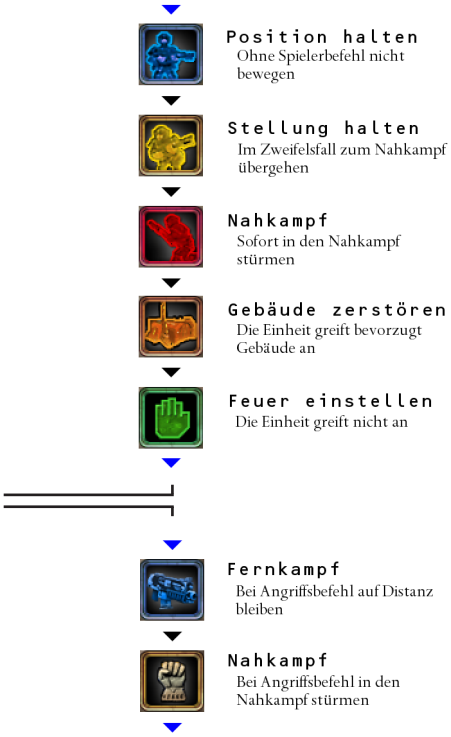
Da ein Mensch sich jedoch nur immer um eine Sache gleichzeitig kümmern kann, hat der Spieler warscheinlich gerade etwas besseres zu tun, als jedem einzelnen Infanteristen zu sagen was er tun soll. Es entspricht auch nicht gerade dem Gedanken des großen Feldherrn ihn zu niederen Aufgaben zu zwingen.

Manche Spiele bieten daher zumindest ein rudimentäres Kontrollsystem, mit dem sich das Verhalten der Truppen beeinflussen lässt. Für gewöhnlich handelt es sich dabei um eine Auswahl vorgefertigter Verhaltensmuster, aus denen eines Ausgewählt wird.
Teilweise wird dabei sogar zwischen dem Bewegungsverhalten (wie sich die Truppen auf dem Schlachtfeld bewegen) und dem Waffenverhalten (wie die Einheiten ihre Waffen einsetzen) unterschieden.

In „Dawn of War“ (2004) werden die Verhaltensmuster durch Icons repräsentiert und das aktuell verwendete Verhaltens-Icon wird im User Interface angezeigt (für die derzeit selektierte Einheit). Möchte der Spieler das Verhalten der Einheit ändern, so klickt er so oft auf das aktuelle Icon, bis das gewünschte Verhalten angezeigt wird. Es entsteht also eine Art „durch-zappen“ durch die bereitgestellte Auswahl.

Eine sichtbare Anordnung ist für den Spieler dabei nicht gegeben. Er klickt einfach so oft, bis das Spiel seiner Meinung ist. Er muss die Verhaltens-Icons also bereits gelernt haben um beim richtigen Icon mit dem „zappen“ aufzuhören. Gerade in Stresssituationen kommt es also häufig zum „verklicken“, wenn man erst zu spät bemerkt, das das gewünschte Icon bereits angezeigt wurde. Man muss dann „zurück-zappen“ (mittels rechter Maustaste – was wiederum nicht ersichtlich ist) oder solange „weiter-zappen“ bis das Icon wieder vorbei kommt.

Einheitenkontrolle bei "Dawn of War"



Screenshot aus dem Echtzeitstragegiespiels “Dawn of War” von Relic Entertainment (2004).

Das Verhalten der Einheiten kann über Icons am rechten unteren Bildrand feingesteuert werden.
Bei längerem Zögern blendet das Spiel eine Erläuterung des gewählten Icons ein.

Ein ähnliches Konzept verwendete bereits 1994 der Klassiker „Total Annihilation“. Statt Icons wurden Schlagworte verwendet, die das symbolisierte Verhaltensmuster umschrieben – etwa „Fire at will“ (Feuern nach eigenem Ermessen – die Einheit schießt auf alles was sich bewegt) oder „Maneuver“ (Manövrieren – die Einheit weicht feindlichem Beschuss aus).

Texte entsprechen zwar weniger den menschlichen kognitiven Fähigkeiten als Bilder, müssen dafür im Gegensatz zu Icons nicht erst auswendig gelernt werden.

Zusätzlich wurden durch Lichter am Ende des Textes die auswählbaren Verhaltensmuster in eine sichtbare Reihenfolge gebracht. Jede Auswahl wird durch eine eindeutige „Lampe“ gekennzeichnet. Das erleichtert die geistige Anordnung der Wahlmöglichkeiten im Entscheidungsraum.

Da beide Spiele nur eine begrenzte Auswahl an Verhaltensmustern bereitstellen, funktioniert dieses System sehr gut. Fügt man jedoch weitere Auswahlmöglichkeiten hinzu oder möchte detaillierte Veränderungen des Spielers an den Verhaltensmustern zulassen, so stößt dieses System schnell an seine Grenzen – genauer gesagt stößt der Spieler an seine Grenzen diesen System kontrollieren zu wollen.

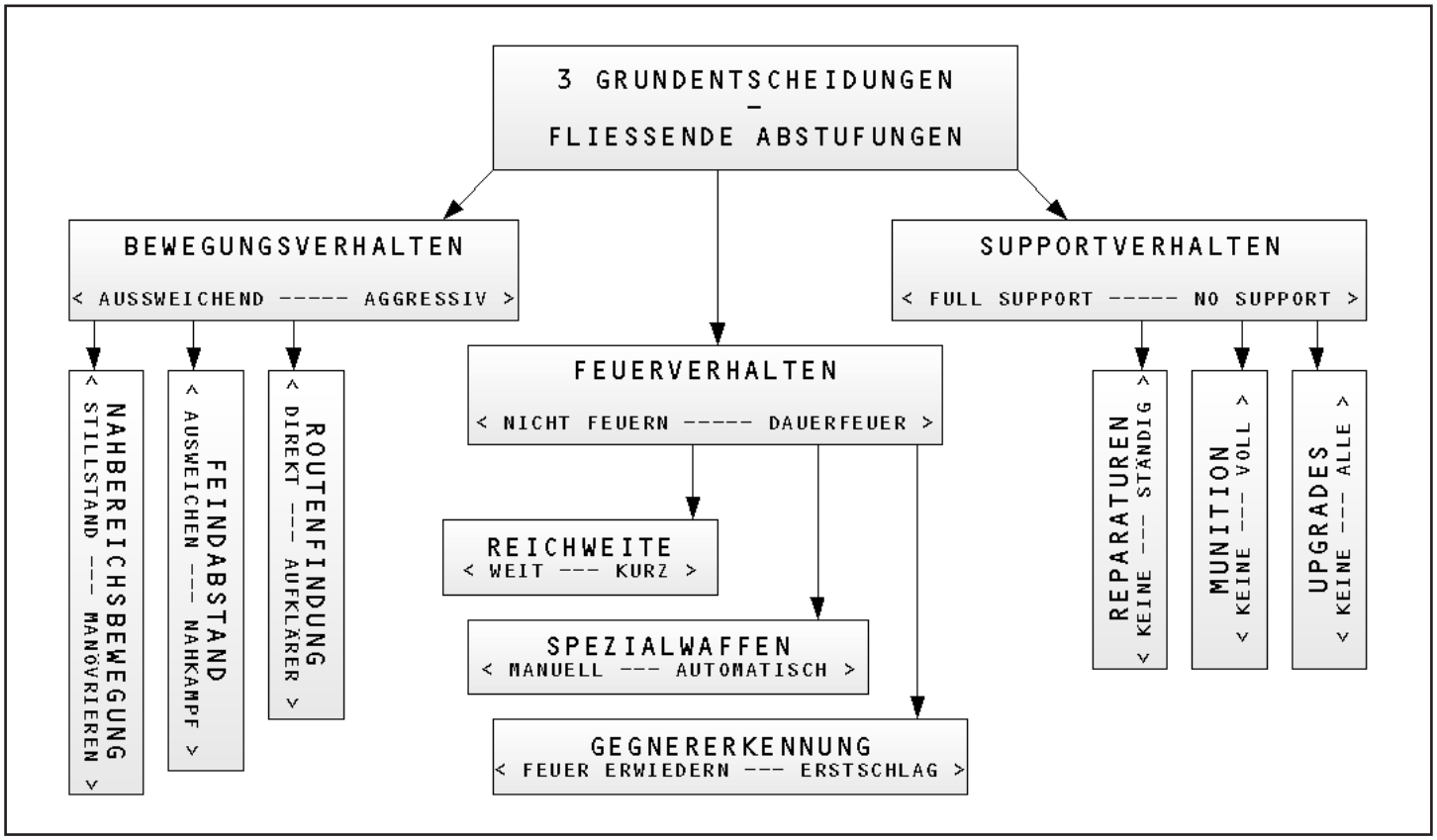
Total Annihilation
(Cavedog Entertainment - 1994)

Einheitenkontrolle in Planet-X

Für einen Strategen ist zunächst die Zentralisierung seiner Entscheidungsgewalt wichtig. Er darf nicht gezwungen werden jede Einheit einzeln anklicken zu müssen um einen Plan durchzusetzen – außer natürlich er will derart detaillierte Pläne umsetzen. Alle Fähigkeiten – sowohl aktiv als auch passiv – müssen also direkt bei der betreffenden Einheit einstellbar sein. Abhängig von dieser Einstellung können dann die einzelnen Einheiten selbstständig agieren und so den Spieler bei der Ausführung seines Konzeptes unterstützen.

Beim Bewegungs- und Waffenverhalten ist dies leicht umzusetzen, denn zwischen den Extrempunkten gibt es zahlreiche sinnvolle Verhaltenskonzepte. Vom sofortigen Rückzug über den totalen Stillstand bis hin zum verfolgen entdeckter Feinde lassen sich die gängigsten Makros definieren und gliedern. Ebenso beim Waffenverhalten, bei dem zwischen Waffenruhe, munitionssparendem Verhalten und Sperrfeuer verschiedene Zwischenschritte möglich sind.
Aber auch passive und Spezialfähigkeiten können nach diesem Prinzip reguliert werden. Reparatureinheiten – etwa Sanitäter oder Mechaniker – können also können also vom Spieler einen Handlungsfreiraum zugewiesen bekommen um selbstständig andere Einheiten zu reparieren.

Umgekehrt erhält jede Einheit einen Wert der angibt, wie oft sie sich Reparaturen anfordert – bzw welche Priorität diese Einheit bei Reparaturteams hat. Der Spieler kann so seine Wertvollen Truppen gezielt besser versorgen lassen oder verhindern, dass veraltete Truppenteile ständig Ressourcen verbrauchen.



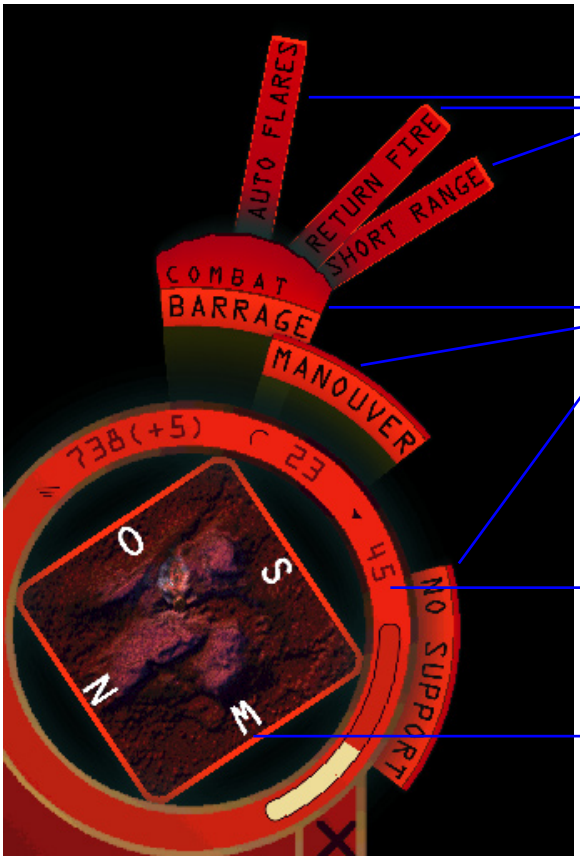
Die drei definierten Entscheidungsräume: Bewegungsverhalten, Waffenverhalten und Support-Verhalten werden nun an den Maschinentelegraphen angefügt. Eine einheitliche Verwendung der Schieberichtung erleichtert die Bedienung: nach oben werden die Makros aggressiver und verschwenderischer, nach unten hin ausweichender und sparsamer.

Dann wird jeder verwendete Entscheidungsraum aufgespalten in mögliche Teilentscheidungen.
Beim Bewegungsverhalten lässt sich etwa der bevorzugte Abstand zu möglichen Gegnern festlegen: stürmt die Einheit in den Nahkampf oder bleibt sie auf Distanz. Auch die Wegfindung oder der Manövrierbereich lassen sich beeinflussen. So kann den Truppen mehr Bewegungsfreiheit eingeräumt werden.

Das Waffen- oder Feuerverhalten kennt ähnliche Unterscheidungen. Wie nah soll man einen Gegner kommen lassen, bevor das Feuer eröffnet wird? Soll die Einheit warten bis sie selbst beschossen wird oder sofort zuschlagen? Sollen Spezialtechniken automatisch eingesetzt werden?
Das Support-Verhalten gliedert sich in die drei Arten von Unterstützung, die die Truppen in Planet-X erhalten können: Reparaturen, Munition und Treibstoff sowie technische Upgrades. So kann der Spieler getrennt Entscheiden welche Unterstützung er den ausgewählten Einheiten zukommen lassen will.

Denkbar ist auch eine dritte Untergliederungsstufe. In den meisten Fällen sind jedoch 2 Ebenen bereits ausreichend Komplex.

Funktionsweise des Maschinentelegraphen (Beispiel)



Sub-Spaken

Zur Aufteilung des Entscheidungsraumes in Sub-Makros.
Hier: Feuer erwiedern auf kurze Reichweite und automatischer Einsatz von Leuchtraketen (Spezialfähigkeit).
Sub-Sub-Spaken waren bisher nicht notwendig, sind aber möglich.

Spaken

Schieberegler zum Auswählen der Makros.
Der Name des Makros kann ebenso angezeigt werden wie die Bedeutung des Schiebers.
Hier: Sperrfeuer auf gegnerische Einheiten und Ausweichen von feindlichem Beschuss. Die Einheit wird weder repariert und betankt.
(Der Begriff Spake stammt aus der Schiffahrt)

Maschinentelegraph

Kernstück des GUI. Hier können verschiedene Informationen angezeigt werden. Z.B. vorhandene Ressourcen.

Map

Auf der Landkarte werden die Positionen der eigenen Truppen markiert. Frei drehbar.

PROGRAMMIERSCHNITTSTELLE

Aufbau

Das User Interface ist in drei Bereiche eingeteilt: das Heads-Up Display (HUD), das während des Spielens eingeblendet wird, die Menus, in denen der sich der Spieler vor und nach der Partie befindet, sowie den kontextsensitiven Mausfunktionen.

Das Heads-Up Display war das Hauptziel dieses Projekts. Der größte Teil der Arbeit floss in Konzeption, Planung, Entwicklung und Design dieser Komponenten. Da nicht sicher war ob daneben überhaupt Zeit für die anderen Bereiche bleibt, wurden diese vorsorglich ausgegliedert. Da sie jedoch an das HUD angelehnt sind, können sie später vergleichsweise einfach entwickelt und eingefügt werden.
Die Menus sind das erste was der Spieler von seinem Spiel zu sehen bekommt. Hier kann er Partien starten und Einstellungen vornehmen. Auch Splash-Screens mit Ladebalken (hier: Ringen) gehören zu den Menuelementen.

Die Maus spielt bei einem Strategiespiel eine große Rolle. Um die Bedienung so ergonomische wie möglich zumachen, soll der Mauszeiger je nach Kontext sein Aussehen ändern um dem Spieler Feedback zu geben. Leider erwies sich die Entwicklung dieser Funktion als technisch schwierig und wurde daher aus Zeitgründen weggelassen. Für das fertige Spiel ist diese Funktionalität jedoch fest eingeplant.

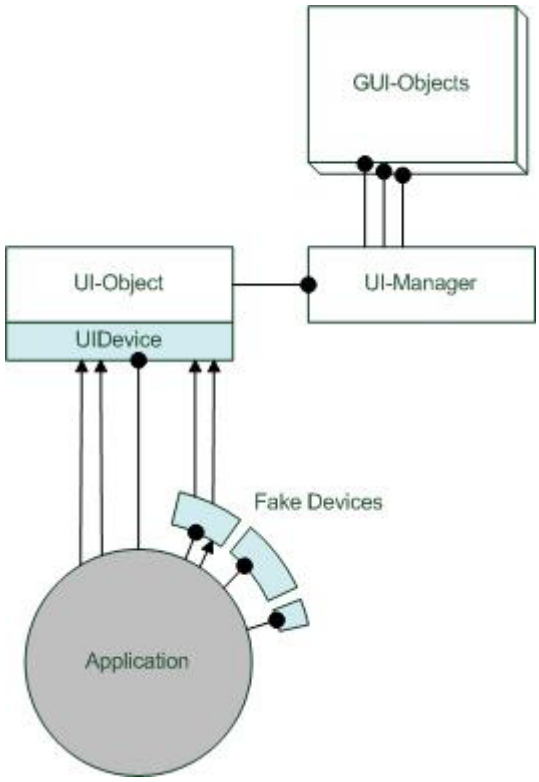
Jeder dieser drei Bereiche funktioniert durch die Verwendung von Interface-Objekten, wobei jedes Objekt im Programmcode einem graphischen Bedienelement auf dem Bildschirm entspricht. Zur einfacheren Bedienung werden die Objekte in logischen Baumstrukturen verwaltet.

Bedienung

Um diese Elemente nutzen zu können, müssen sie vom Spiel – also dem eigentlichen Programm – erzeugt, verändert und gegebenenfalls wieder gelöscht werden können.
Zunächst ist dazu ein direkter Zugriff auf das User Interface nötig. Der Spieleentwickler muss alle Kontrollelemente, Anzeigen, Beschriftungen und Werte jederzeit verändern können.

Dies bedroht jedoch die interne Funktionsweise der Engine. Bei Bedienungsfehlern könnte so das ganze Spiel abstürzen – etwa wenn die Spiellogik ein Kontrollelement entfernt ohne es zunächst aus dem Register zu entfernen.
Gibt man dem Game Developer jedoch nur Nummern und Funktionen statt den eigentlichen Objekten, so wird die Programmierung sehr kompliziert und unergonomisch für ihn.

Als optimaler Kompromiss stellte sich daher die Verwendung von Pseudo-Objekten – so genannten Fake Devices dar. Jedes dieser Fake Devices repräsentiert ein tatsächliches Interface-Element, ohne mit ihm verbunden zu sein. Der Programmierer kann es jederzeit so verwenden als ob es das echte Objekt wäre. Intern werden jedoch alle Aufrufe und Manipulationen an eine Kontrollstruktur



im User-Interface weitergereicht, wo sie – wenn gültig – auf das tatsächliche Objekt angewandt werden. Als eindeutiges Identifikationsmerkmal zwischen Interface-Objekt und Fake Device werden dazu ID-Nummern verwendet, welche bei jeder Aktion mit gereicht werden.

Auf demselben weg ist auch ein abfragen der derzeitigen Werte möglich. Es kann also festgestellt werden, ob der Spieler etwas an den Kontrollelementen verändert – also Eingaben getätigt hat.
Da es jedoch sehr ineffizient ist, ständig zu überprüfen ob der Spieler gerade dieses oder jenes Interface-Element angeklickt hat, wurde zusätzlich ein Event-System eingebaut. Der Game Developer kann Programmteile als Zuhörer („event listener“) für ein Interface-Element registrieren. Wann immer der Spieler nun dieses Element anklickt oder per „drag-and-drop“ verändert, wird der Zuhörer vom User Interface automatisch benachrichtigt. So können etwa Kontrollelemente für die Einheiten KI direkt mit den entsprechenden Routinen verknüpft werden.

ENGINE-ARCHITEKTUR

Die Multimedia-Engine besteht aus mehreren Einzelmodulen, die als dynamische Bibliotheken geladen werden können. Dieses Konzept hat den Vorteil, dass verschiedene Programme die Engine verwenden können, ohne sie selbst zu implementieren. Neben dem Spiel an sich benötigen etwa auch unsere Level-Editoren oder Test-Programme dieselbe Engine. Beziehungsweise die Teile davon die sich brauchen, den jeder Teilbereich kann einzeln geladen werden.

Für das User Interface ist vor allem das Zusammenspiel mit den Input- und Graphik-Komponenten wichtig, sowie die Fähigkeit selbst geladen und verwaltet werden zu können. Dazu musste es in den Rendering-Zyklus eimplementiert werden und eine Schnittstelle für die Aufnahme von Benutzereingaben durch Maus und Tastatur bereitstellen.