ASP5 – Projekt – Dokumentation

Fh hagenberg, mtd00

Christoph Atteneder
Matthias Bauer
Peter Hutterer
Daniel Kuales
Paul Lanzerstorfer
Martin Wachtler

1. Inhaltsverzeichnis

2. Die Lifeengine

Überblick

Die Life-Engine stellt das Herz des Projektes dar. Sie kümmert sich um die Verwaltung der Welt, stellt die notwendigen Klassen und Algorithmen zur Verfügung, um Kreaturen und Pflanzen zu erzeugen und diese (über)leben zu lassen. Ziel der Life-Engine war es, eine möglichst realitätsnahe Evolution zu ermöglichen, die nach Darwin das Überleben des Stärkeren fördert, indem sie die individuellen Eigenschaften der Lebewesen hervorhebt. Je nach Aussehen der Welt soll sich ein natürliches Gleichgewicht bilden, in dem sich Kreaturen und Pflanzen gegenseitig die Waage halten, um das Überleben des Lebens an sich zu garantieren.

Die Life-Engine besteht aus folgenden Bestandteilen

World
Diese Klasse stellt die Welt, den Ansatzpunkt für ein Spiel zur Verfügung.

Cell
Eine Zelle ist ein Bestandteil der Welt und spezifiziert die Klimazonen sowie die Landschaften.

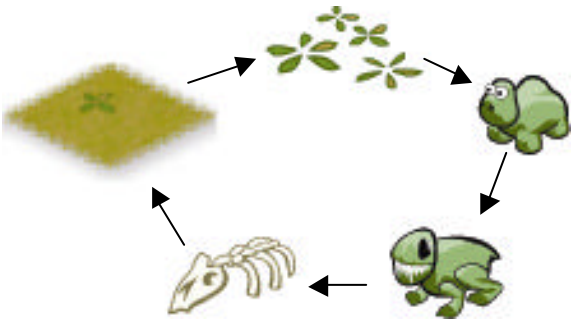
Plant
Eine Pflanze ist das Grundelement des Lebens, da sie das Grundnahrungsmittel für die Kreaturen darstellt. Pflanzen sind zellengebunden, sie gehören immer einer Zelle an.

Creature
Kreaturen sind der Hauptbestandteil des Spieles. Sie greifen auf die notwendige DNA zurück, um die Evolution zu ermöglichen

DNA
Die DNA, die die Eigenschaften einer Kreatur definiert. Sie enthält die notwendigen Algorithmen für die Evolution.

Target
Eine Kreatur hat im Optimalfall ständig ein Ziel, entweder Fressen oder Paarung. Target stellt die Verwaltung dieses Zieles zur Verfügung.

Der Nährstoffkreislauf



Einer der Hauptaufgaben bei der Entwicklung war das Aufrechterhalten eines Nährstoffkreislaufes. Nährstoffe werden zu Beginn über der Welt verstreut, die Pflanzen wandeln die auf den Zellen befindlichen Nährstoffe in Energie um. Diese Energie wird von den Pflanzenfressern aufgenommen und kann durch sie von den Fleischfressern aufgenommen werden. Bei ihrem Tod geben Kreaturen die Energie in Form von Nährstoffen wieder an die Zelle zurück. Dadurch können anfangs nährstoffarme Gebiete attraktiv für Pflanzen werden und umgekehrt.
Weiters gibt jede Kreatur pro Runde ihren Energieverbrauch an die Zelle ab, auf der sie sich gerade befindet.

Die Zelle

Grundlegender Bestandteil der Welt ist die Zelle. Sie ermöglicht die Aufteilung in Klimazonen, Landschaften und erlaubt das Wachsen der Pflanzen. Die Welt selbst ist nur ein abstraktes Gebilde und stellt nur die Verwaltungsfunktionen zur Verfügung dar. Die eigentliche, sichtbare Welt ist eine quadratische Ansammlung an Zellen, wobei die Größe variabel ist.

Aufbau

Die Zelle besitzt die Eigenschaften Temperatur sowie Wasserstand. Aus der Mischung der beiden ergeben sich (grafisch) 7 Klimazonen:
Wüste ... sehr heiß, sehr wenig Wasser
Eiswüste ... sehr kalt, sehr wenig Wasser
Tundra ... kalt, wenig Wasser oder sehr kalt, durchschnittlich viel Wasser
Prairie ... heiß, wenig Wasser
Wiese ... gemäßigt
Sumpf ... viel Wasser
Wasser ... sehr viel Wasser

Die Klimazonen sind intern feiner abgestuft, so gibt es für die Temperatur 100 verschiedene Werte (die etwa mit dem irdischen -40°C bis 60°C verglichen werden könnten), ebenso gilt dies für die Wasserwerte (0% bis 100% Wasseranteil).

Zusätzlich hat eine Zelle noch die Eigenschaft "Nährstoffe". Diese werden zu Beginn auf der Welt verteilt, jede Zelle bekommt ihren Anteil. Die Nährstoffe sind das unterste Glied der Nahrungskette. Die Aufteilung der Nährstoffe geht nach Klimazone, so hat etwa eine Wüste weniger Nährstoffe als ein Sumpf, wobei dieser wieder weniger als eine Wiese hat.

Die Entstehung einer Welt

Die Entstehung einer Welt ist in folgende Schritte unterteilbar:

1. Anlegen der Zellen
2. Erstellung der Landschaftszonen
3. Auffüllen der restlichen Felder
4. Blur zum Abrunden von Übergängen
5. Erstellung des großen Flusses
6. Leichter Blur
7. Erstellung der mittleren und kleineren Flüsse
8. Nährstoffverteilung

Die einzelnen Parameter können mittels des bitTools, ein Konfigurations-Programm, geändert werden. So etwa ist die Anzahl der Zellen und damit Größe der Welt einstellbar. Dies gilt ebenso für die Anzahl der einzelnen Klimazonen, wobei die Größe der einzelnen Zonen abhängig von der Weltgröße ist.
Da nach der Erstellung der Landschaften Zellen frei bleiben können, werden diese mit Zufallswerten gefüllt. Der erste Blur dient dazu, die Übergänge zwischen den Landschaftszonen etwas abzuschwächen, so etwa gibt es im Umfeld von Seen eine Sumpfzone, im Umfeld von Wüsten Steppenzonen usw. Danach wird der große Fluss erstellt, wenn der User dies gewünscht hat (im Setup einstellbar). Dieser beginnt stets an einer Kante der Welt und geht, je nach Mäanderbildung bis zum gegenüberliegenden Ende der Welt oder versickert in der Welt. Um auch hier die Übergänge abzurunden, wird ein weiterer Blur-Filter, diesmal allerdings mit kleinerem Radius über die Welt gelegt.
Danach wird die vom User gewünschte Anzahl an kleinen und mittleren Flüssen erstellt. Diese entspringen inmitten der Welt, münden jedoch immer in einen Fluss oder See ein.
Die Nährstoffverteilung geschieht am Ende der Erstellung nach oben erwähnten Schema.
Danach ist die Welt fertig und das Leben nimmt seinen Lauf.

Der Ablauf einer Runde

Eine Runde besteht aus zwei Bestandteilen: Aktualisieren der Pflanzen und nachfolgendes Aktualisieren der Kreaturen. In beiden Fällen werden alle in der Welt befindlichen Pflanzen und Kreaturen durchgegangen und ihre jeweilige Update-Methode ausgeführt. Der genauen Ablauf der einzelnen Funktionen wird weiter unten beschrieben.

Lebewesen

Es gibt zwei Arten von Lebewesen: Kreaturen und Pflanzen. Der wesentliche Unterschied liegt in der Verwendung der DNA. Während die Pflanzen intern fix definiert sind und daher keinen Mutationen unterliegen, greifen die Kreaturen intern auf eine komplexe DNA zurück. Ein weiterer Unterschied besteht darin, dass Pflanzen zellengebunden sind, d.h. eine Pflanze kann stets nur auf einer Zelle sein, während Kreaturen sich auch innerhalb der Zelle bewegen können und sich auch über Zellen hinweg bewegen können.

Die Pflanzen

Die Pflanzen stellen das zweite Glied der Nahrungskette da, sie nehmen die Nährstoffe aus dem Boden (d.h. aus den Zellen) auf und wandeln sie in Energie um. Die Pflanzen haben die Eigenschaften: Wassertoleranz, Temperaturtoleranz, Nährstoffverbrauch sowie Besamungsradius und Reifezeit. Wasser- und Temperaturtoleranz spezifizieren die Bereiche, in denen die Pflanzen überlebensfähig sind, der Nährstoffverbrauch, wieviele Nährstoffe pro Pflanze verbraucht werden. Die Reifezeit gibt an, nach wievielen Runden die Pflanze eine Besamung einer Nachbarzelle durchführen kann, der Besamungsradius gibt an, wie weit diese Zelle entfernt sein kann.
Der Prozess der Besamung ist zufällig, es wird eine Zelle innerhalb des Radius gewählt und der Samen auf dieser Zelle niedergelassen. Ist der Samen überlebensfähig, entsteht dort eine neue Pflanze. Wächst dort bereits eine Pflanze, so wird die Anzahl der Pflanzen um den Nährstoffwert erhöht. Ist der Samen nicht überlebensfähig, so geht er verloren, es dauert je nach Pflanze eine gewisse Zeit, bevor sie wieder fähig ist, sich auszubreiten.
Da die Parameter der Pflanzen beliebig verstellbar sind, gibt es intern eine hohe Zahl an möglichen verschiedenen Pflanzen. Die grafische Ausgabe beschränkt sich auf 7 den Zonen entsprechende, deswegen werden auch intern nur 7 Pflanzentypen angelegt.

Die Kreaturen

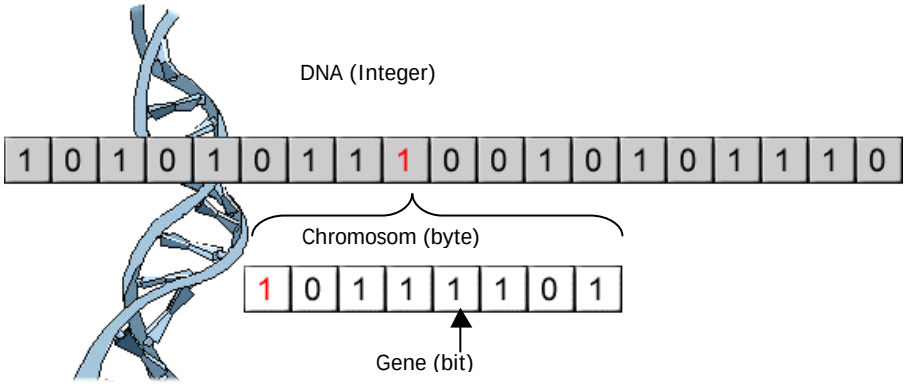
Sie stellen den Hauptbestandteil des Spieles dar und sind dementsprechend komplex aufgebaut. Jede einzelne Kreatur greift auf ihre eigene, spezifische DNA zurück, die die Eigenschaften der Kreatur bestimmt. Dazu zählen Grundeigenschaften wie Größe, Sichtweite und Geschwindigkeit, aber auch komplexere wie die Paarungsintervalle und die Anzahl an Kindern, die pro Paarung entstehen können.
Im Folgenden eine Liste der Eigenschaften einer Kreatur:

- Position auf der Zellen
- aktuelle Energie
- Alter
- Zeit bis zur nächsten Paarung
- Lebenserwartung
- Maximale Anzahl an Kindern bei der Paarung
- Geschwindigkeit
- Sichtweite
- Temperaturtoleranz
- Wassertoleranz
- Art der Nahrung (Pflanzen- oder Fleischfresser)
- Energieverbrauch pro Runde
- Paarungsintervall
- Zeit bis zur Geschlechtsreife
- Maximale Energie
- Menge an Nahrung, die pro Runde aufgenommen werden kann
- Startenergie (Energie, die die Kreatur zur Geburt benötigt)
- Name der Rasse
- Größe

All diese Eigenschaften werden durch die DNA spezifiziert und berechnet.

Die DNA

Da das Hauptziel des Spieles eine möglichst realitätsnahe Evolution war, wurde eine eigene DNA entwickelt, die der DNA der realen Welt in Grundzügen entspricht, jedoch den extremen Komplexitätsgrad auf ein überschau- und vor allem programmierbares Spektrum reduziert.
Jede DNA besteht aus einem Genom, das 19 Chromosome besitzt. Diese spezifizieren in Kombination die verschiedenen Körperteile (so etwa sind die ersten 4 Chromosome für den Körper verantwortlich). Jedes dieser Chromosome besteht aus 7 Genen (Bits), die für den Gesamtwert des Genes verantwortlich sind. Eine bestimmte Kombination aus den Genen ergibt einen bestimmten Wert (0 oder 1) für das Chromosom. Die Reihenfolge der Chromosom-Werte spezifiziert nun das Genom genau, was ausschlaggebend für die Rasse der Kreatur ist.



Diese Kombination ermöglicht es, dass es eine große Anzahl an Kreaturen einer Rasse gibt, die dennoch eine unterschiedliche interne Struktur haben. Ähnlich wie in der realen Welt, wo es eine große Zahl an verschiedenen Menschen gibt, die dennoch alle zur Rasse Mensch gehören.

Mutationen treten bei jeder Paarung auf, sind jedoch auf die Gene beschränkt. Erst wenn die Gene derart mutieren, dass sich das entsprechende Chromosom mitändert, wird die Mutation tatsächlich sichtbar. Dadurch besteht die Möglichkeit, dass sich zwei Kreaturen derselben Rasse paaren und keine Mutationen auftreten. Umgekehrt besteht die Möglichkeit, dass sich dieselben Kreaturen mit anderen paaren und Mutationen auftreten, obwohl auch die anderen Kreaturen derselben Rasse sind.
Bei der Paarung werden nun die Gene miteinander kombiniert und ergeben dadurch einen anderen Genstruktur für den Bastard. Abhängig von dieser Genstruktur errechnet sich der Wert des Chromosoms. Im Laufe der Entwicklung wurden mehrere Merging-Verfahren für die Chromosomensätze entwickelt, die aktuelle, vierte Version wirkt sich im Laborversuch mit einer Genstabilität von über 99% aus, was eine Mutationsrate von etwas mehr als 10% auf das Genom bezogen bewirkt, die für ein Spiel dieser Art optimal ist.

Die Intelligenz:

Gleich vorweg: Etwas in der Art von Künstlicher Intelligenz ist nur sehr bedingt vorhanden, die verwendeten Algorithmen sind sehr einfach und lassen sich nicht mit der KI vergleichen, die in ähnlichen professionellen Projekten verwendet wird.
Die Kreaturen verfolgen zwei Ziele: Fressen und Fortpflanzen, als Zwischenziel gilt die Fortbewegung um eines der Hauptziele zu erreichen.
Eine Kreatur sucht jede Runde innerhalb ihres Sichtradius ein Ziel. Ist die Zeit bis zur nächsten Paarung noch nicht verstrichen oder ist die Kreatur am Verhungern, wird nach Fressen gesucht, ansonsten nach einem paarungsbereitem Partner. Wurde ein Ziel gefunden, bewegt sich die Kreatur in Richtung des Zieles (siehe Pathfinding weiter unten). Wurde kein Ziel gefunden, bewegt sich die Kreatur in eine beliebige Richtung vom aktuellen Ort weg.
Hat die Kreatur bereits in der letzten Runde ein Ziel gefunden, so wird die Gültigkeit desselben überprüft (existiert das Ziel noch, ist es außerhalb der Sichtweite, ...) und die Kreatur bewegt sich weiter auf das Ziel zu. Ist es erreicht, wird entweder gefressen oder die Paarung findet statt.
Ist das Ziel in einem Bereich, der von der Kreatur nicht betreten werden kann (siehe Wassertoleranz und Temperaturtoleranz), so muss ein anderes Ziel gesucht werden. Dies bewirkt auch, dass Kreaturen, die auf Zellen, auf denen sie nicht überlebensfähig sind, ausgesetzt oder geboren werden, bewegungsunfähig sind. Ähnlich einem Fisch, der sich in der Wüste nur schwer fortbewegen kann.

Die Paarung:

Die Paarung erfolgt nach dem bereits in der DNA besprochenen Merging-Verfahren. Zwei Kreaturen, die beide paarungsbereit sein müssen, mergen ihre DNA, wodurch die Bastarde entstehen. Um diese überlebensfähig zu machen, wird ein Teil der Energie von den beiden Eltern auf die Kinder übertragen, damit diese den Weg zur ersten Futterstelle überleben können. Die Menge an Energie, die eine Kreatur zu Beginn braucht, definiert sich durch die DNA. Da bei Mutationen diese stark differieren kann, kann es sein, dass ein Elternteil die Geburt des Kindes nicht überlebt, da die komplette Energie auf das Kind übergeht. Nach der Paarung sind beide Elternteile stark geschwächt, die Energieaufnahme ist für das Überleben notwendig, wird allerdings dadurch erschwert, dass in unmittelbarer Umgebung jetzt mehr Kreaturen vorhanden sind, die sich gegenseitig das Futter wegfressen.

Futter:

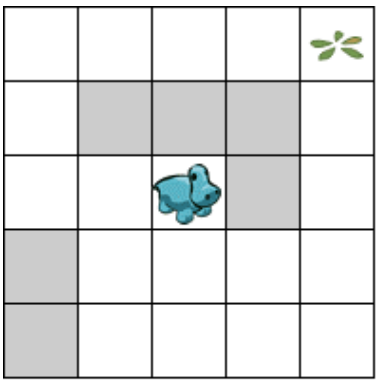
Es gibt zwei Arten von Futter: Pflanzen und andere Kreaturen. Ob eine Kreatur Fleisch- oder Pflanzenfresser ist, definiert sich aus der DNA. Pflanzenfresser funktionieren nach einem einfachen Prinzip:

1. Pflanzen suchen
2. zum Ziel bewegen
3. fressen.

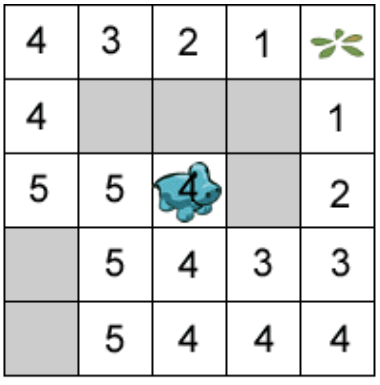
Der einzige Risikofaktor ist, dass die Pflanze von einer anderen Kreatur in der Zwischenzeit (während der Bewegung) gefressen werden kann.
Bei Fleischfressern muss überprüft werden, ob das Ziel überhaupt essbar ist. Hierbei wird eine einfache Methode angewandt, die die DNA des Zieles mit der des Fleischfressers vergleicht. Eine Variable definiert den minimalen Unterschied, der hierbei auftreten muss, um zu gewährleisten, dass eine Kreatur nicht zum Kannibalen wird.
Da sich die Pflanzenfresser fortbewegen, muss bei den Fleischfressern öfter der Pfad angepasst werden, wie sie zum Ziel kommen können. Sollte die Kreatur bemerken, dass das ausgewählte Ziel schneller ist als die Kreatur selber, wird ein anderes Ziel gesucht. Beim Fressvorgang selbst stirbt die andere Kreatur, die Energie geht zum Teil auf den Fleischfresser über (abhängig von der Nahrungsaufnahmemenge), der Rest wird an die Zelle in Form von Nährstoffen zurückgegeben.

Pathfinding:

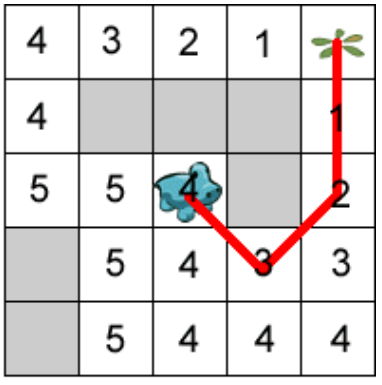
Der in bitLife angewandte Path-Finding-Algorithmus wird im Allgemeinen als A* („A Star“) bezeichnet. Bei diesem Algorithmus wird vom Zielpunkt ausgegangen und die benachbarten Zellen mit dem Abstand zum Ziel versehen. Dadurch wird der Abstand zum Startpunkt errechnet und auch gleichzeitig überprüft, ob ein Weg zum Ziel überhaupt möglich ist. Anhand dieser Nummerierung wird dann der kürzeste Weg errechnet und abgespeichert.
Bei diesen sehr einfachen System werden auch Hindernisse (in unserem Fall aufgrund von Wassergehalt und Temperatur unbetretbare Zellen) berücksichtigt.



Step 1



Step 2



Step 3

Nicht betretbare Zelle

3. Die Gameengine

Überblick

Was wäre ein Spiel ohne grafische Ausgabe? Gerade deshalb war es besonders wichtig diese Darstellung so performant wie möglich und dem Spielgenre passend zu wählen. Hierbei wurden Funktionen sowohl der DirectDraw API als auch GDI verwendet. Als Ausgangspunkt unserer Programmierung diente uns die isometrische Gameengine von Ernest Pazera (http://www.isohex.net), welche adaptiert und erweitert wurde.
Die Game-Engine kümmert sich wie schon erwähnt um die grafische Ausgabe der Vorgänge in der Life-Engine. Sie stellt die notwendigen Klassen und Algorithmen zur Verfügung, um die Oberfläche der Welt, die einzelnen Pflanzen, Kreaturen und deren Bewegung zu visualisieren. Weiters beinhaltet sie auch Funktionen zur Benutzerinteraktion.

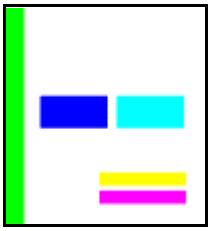
Die Game-Engine besteht aus folgenden Bestandteilen:

Game

Hier wird das Vollbild – Fenster, das DirectDraw Objekt und alle notwendigen DirectDraw – Surfaces, erstellt und die Soundengine gestartet. Weiters beinhaltet diese Klasse den Gameloop um alle Benutzerinteraktionen und Animationen kontrollieren und die grafische Darstellung erneuern zu können.
Weiters beinhaltet sie noch Funktionen und Methoden zur dynamischen grafischen Generierung von Kreaturen und deren Abspeichern, als auch die Ausgabe von Statusmessages um eine ein- und ausblendbare Textausgabe zur Überprüfung von Parametern zu ermöglichen.
Von ihr ausgehend werden alle anderen Klassen, wie die Isometrische Engine angesprochen.

InterfaceElement

Definiert die grundlegenden Eigenschaften aller Interfaceelemente wie Menüs, Abfrage von Benutzerinteraktionen und vieles mehr.
Die Abfrage der Benutzerinteraktion wurde mittels MouseMaps (isohexcore/IsoMouseMap.cpp) realisiert. Hierbei wird abgefragt welche Farbe sich unter der aktuellen Position des gedrückten Mauszeigers befindet.



MainMenu

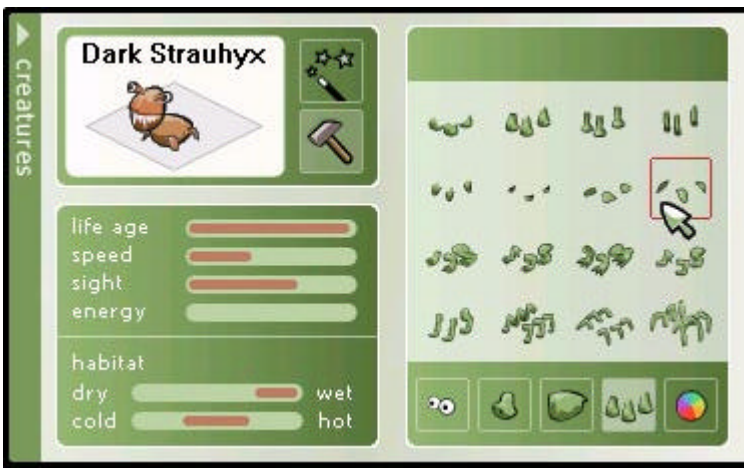
Beinhaltet alle notwendigen Spieloptionen wie Minimieren, Soundeinstellungen und Beenden des Spieles.

CreatureLab

Dient zur dynamischen Erstellung einzelner Kreaturen und deren Aussetzung.
Der Linke Bereich des Interfaces beinhaltet die aktuelle Darstellung der erstellten Kreatur, dessen Namen, Eigenschaften und Lebensraum. Ein Austauschen einzelner Körperteile bewirkt eine Änderung der Eigenschaften der Kreatur.
Folgende Körperteile stehen zur Verfügung:

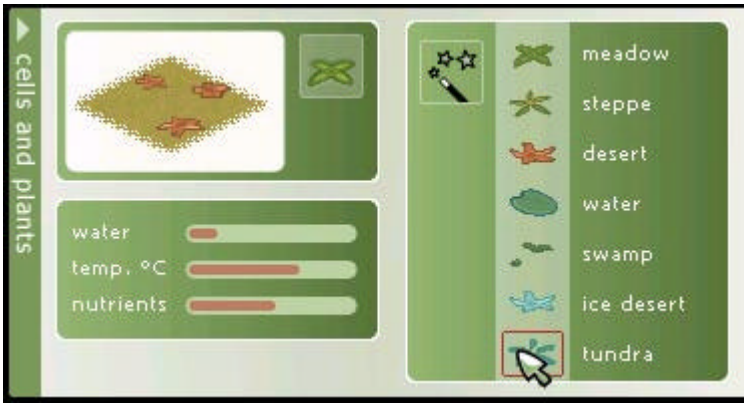
- Augen
- Kopf
- Körper
- Beine
- Farbe

Durch diese Vielfalt von Möglichkeiten kann der Benutzer 520 000 unterschiedliche Kreaturen erschaffen.
Mittels Zauberstab können diese schlussendlich in der Welt ausgesetzt werden.



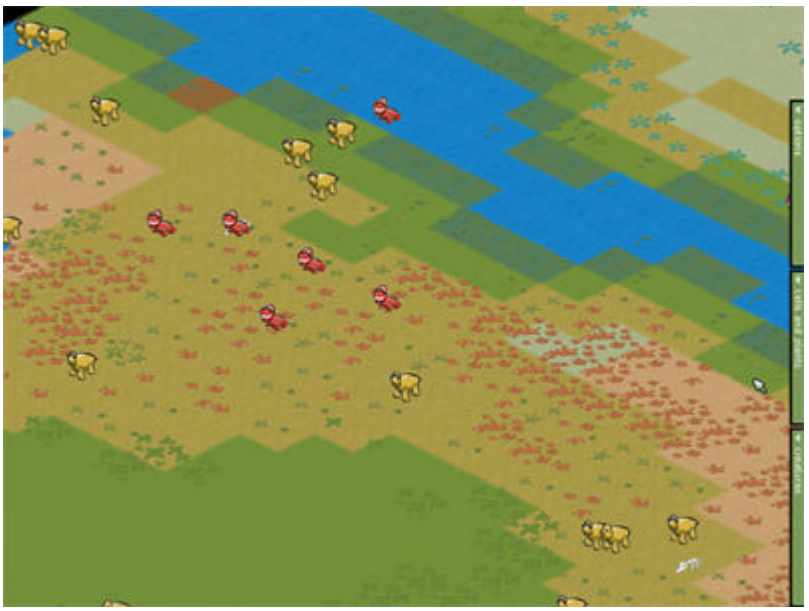
WorldLab

Mithilfe des WorldLabs lassen sich Informationen zu selektierten Zellen ausgeben und neue Pflanzen erstellt und in der Welt ausgesetzt werden. Auch im WorldLab hat jede Pflanze unterschiedlich Eigenschaften, welche sie für bestimmten Untergrund predestiniert.



Isohexcore

Die Klassen dieser isometrischen Engine ermöglichen die isometrische Darstellung, das Laden von grafischen Elementen und Scrollen. Weiters stellen sie Grundfunktionen zur Benutzerinteraktion zur Verfügung.



Einzelne Grafische Elemente werden als Tilesets geladen bzw. gespeichert. (isohexcore/TileSet.cpp)

Hierbei kann sowohl eine transparente Farbe, der sichtbare Berich der Grafik als auch Ankerpunkte zur genauen Positionierung definiert werden.
Nach dem Laden von einzelnen Tilesets kann man per Indexposition auf den jeweiligen Bereich des einzelnen Tilesets zugreifen.

4. BitLife Grafik

Die grafische Darstellung spielt bei bitLife eine sehr wichtige Rolle, soll doch die Evolution und die somit auftretenden Mutationen anschaulich visualisiert werden. Wir einigten uns sehr bald auf die klassische isometrische Ansicht, wie sie von Strategiespielen bekannt ist. Die Grundgestaltung orientiert sich am Comicstil, dafür kennzeichnend sind die schwarzen Outlines und die flächigen Farben. Hauptfarbe ist gras-gelbgrün um den Aspekt des ‚Lebens' noch hervorzuheben.

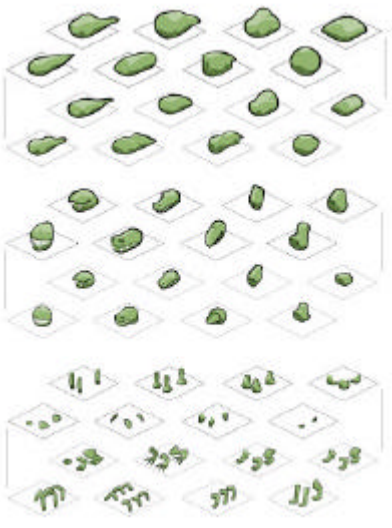
Der Boden

Die sieben verschiedenen Klimazonen (Wüste, Steppe, Graslandschaft, Sumpf, Wasser, Tundra, Eiswüste) sind farblich entsprechend gestaltet. Die einzelnen isometrischen Tiles sind 100 x 50 px groß und werden zu einer Landschaft zusammengefügt. Damit die Übergänge zweier Tiles weniger scharf sind, besitzen die Tiles nicht eine gerade Kante, sondern fransen pixelig aus, fügen sich aber genau in nebenliegende Tiles. Sie besitzen wenig Struktur und sind kontrastarm. Dadurch können Muster, die durch große Flächen gleicher zusammengesetzter Tiles entstehen würden, möglichst vermieden werden.

Pflanzen

Für die einzelnen Klimazonen gibt es auch dazugehörige Pflanzen. Sie sind allesamt so gestaltet, dass sie flach am Boden wachsen. Somit kann man auf die Überprüfung, ob ein Tier vor oder hinter einer Pflanze steht, verzichten.

Tiere



Wir mussten einen Weg finden, um Mutation und somit die Unterschiede der Tiere bestmöglich zu veranschaulichen und gleichzeitig durch eine DNA zu beschreiben. Die Tiere werden in die Körperteile Augen, Kopf, Körper und Beine aufgeteilt. Zusätzlich erhalten sie noch eine Farbe. Die Anzahl der einzelnen Teile wurde fix definiert.

Alle diese Einzelteile wurden als separate Grafiken erzeugt, damit später Tiere aus den Einzelteilen zusammensetzt werden können.

Alle Grafiken entstanden in Freehand und wurden in Photoshop in Pixelgrafiken umgewandelt und nachbearbeitet.

Menü

Wichtig war es, die vordefinierten Interaktionsmöglichkeiten möglichst übersichtlich und einfach zu gestalten. Die Menüs sind am rechten Rand angeordnet und können bei Bedarf aus- und eingefahren werden. Sie sind passend zum Stil des Spiels gehalten.

5. SoundEngine

Im wesentlichen besteht die SoundEngine von bitLife aus drei Teilen die im Nachfolgenden näher beschrieben werden. Für die Wiedergabe der Geräusche wurden Funktionen der DirectSound-Bibliothek verwendet. Die Wiedergabe der Musik wurde mit DirectShow realisiert.

Tiergeräusche

Da in Bitlife über 500.000 Tiere vorkommen können, ist es etwas schwierig, für jede einzelne Tierart ein eigenes Geräusch zu kreieren. Wir beschränkten uns daher auf derzeit 16 verschiedene Tierkategorien, wobei auf jede Kategorie zwei verschiedene Laute kommen. Einen, der beim Auswählen abgespielt wird, einen weitern, den das Tier während des Spiels selbständig wiedergibt. Welche Kategorie welchem Tier zugewiesen wird, hängt im wesentlichen vom Kopf des Tieres ab, aber zu einem kleinen Teil auch vom Körper, da dies auch in der Natur die wesentlichen Einflussfaktoren auf die Stimmen sind.

Effektgeräusche

Verschiedenen Funktionen im Spiel sind natürlich auch Geräusche zugewiesen, wie zum Beispiel einem Mausklick auf einen Button, oder das Aufrufen eines Menüs.

Hintergrundmusik

Die Hintergrundmusik im Spiel ist im wesentlichen ein einfacher Mp3-Player. Es wird zu Beginn des Programms das Unterverzeichnis ‚mp3\' nach Mp3-Dateien gescannt. Alle gefundenen Dateien werden dann in zufälliger Reihenfolge wiedergegeben. Werden keine Dateien gefunden, wird natürlich auch keine Musik wiedergegeben.
Zu Projektbeginn wurde auch eine dynamische Lösung für die Musik angedacht. Die Musik sollte auf bestimmte Faktoren im Spiel reagieren, wie zum Beispiel die Menge der lebenden Tiere. Je mehr Tiere im Spiel, desto stressiger, aktiver sollte die Musik sein. Die Musik sollte aus einzelnen kurzen Parts dynamisch zusammengesetzt werden, jedoch stellten sich zu Beginn massive Probleme mit der Synchronität der einzelnen Loops ein. Leider konnten bis zum Schluss nicht alle Probleme beseitigt werden. Es stellte sich auch heraus, dass es nicht unbedingt einfach ist, dynamisch eine Art Melodie zu kreieren. Daher entschieden wir uns, nur den Mp3-Player für die Hintergrundmusik zu implementieren.

Umsetzung

Die SoundEngine wurde mit drei Klassen realisiert. Die Klasse WAVLoader wurde aus der Isometric-Engine von Ernest Pazera übernommen und kapselt des Laden der Wav-Files in einen Buffer. Die Haupteinheit der Engine stellt die Klasse SoundEngine dar. Diese übernimmt die Initialisierung von DirectSound und stellt Schnittstellen zum Abspielen der Tier- und Effektgeräusche, zum Aktivieren und Deaktivieren der Musik und der Geräusche und ebenfalls zum Setzen der Lautstärke von Musik und Geräuschen zur Verfügung.
Letztlich gibt es noch die Klasse Music, die für die Wiedergabe der Hintergrundmusik sorgt. Die Wiedergabe wurde in eine eigene Klasse gekapselt, um sie leicht austauschen zu können. Da geplant war, die Musik dynamisch zu gestalten, aber von Anfang an klar war, dass dies zu Problemen führen könnte, wurde darauf geachtet, die Wiedergabe leicht austauschbar zu halten. So kann zu einem späteren Zeitpunkt der jetzige Mp3-Player einfach durch eine andere Klasse mit den gleichen Schnittstellen ersetzt werden und der Rest der SoundEngine bleibt unverändert.

6. Nicht implementierte Funktionen/Ideen

Im folgenden eine Liste von Ideen, die im Laufe des Designs und der Entwicklung aufgekommen sind, jedoch aus verschiedenen Gründen nicht implementiert wurden:

Herdentrieb
Kreaturen mit einem Herdentrieb sollten sich bevorzugt in Gebieten aufhalten, die mit anderen Kreaturen der gleichen Rasse besiedelt sind. Kreaturen ohne Herdentrieb sollten diese Gegenden meiden. Die Implementierungszeit war hierfür nicht mehr vorhanden, auch die Befürchtung, dass dies die CPU leicht in die Knie zwingen könnte, hinderte uns an der Umsetzung.

Gift
Kreaturen sollten eine Eigenschaft namens "Gift" haben. Ebenso eine Eigenschaft, die die Toleranz gegenüber Giften angibt. Dadurch wären Kreaturen durch das Fressen einer anderen Rasse gestorben. Angedacht wurde hier auch noch die Lernfähigkeit einer Rasse: Wenn einmal eine Kreatur an Gift gestorben ist, sollten Kreaturen dieser Rasse die giftige Kreatur meiden. Aus Zeitmangel wurde diese Idee nicht verwirklicht.

Intelligenz
Angedacht wurde die Möglichkeit abstufbarer Intelligenz. So sollten Kreaturen in der Grundstufe nur Futter bzw. Partner suchen (wie dzt. implementiert), Kreaturen mit höherer Intelligenz dagegen sollten auch aufpassen, ob Feinde in der Nähe sein könnten. Diese Idee wurde nur angedacht, die genauen Stufen der Intelligenz wurden nicht ausgefeilt.

Laden/Speichern
Bereits teilweise implementiert wurde die Möglichkeit, eine Welt speichern und auch wieder laden zu können. Da jedoch vor allem das Laden ein großer Quell von Bugs ist, wurde die Version nicht offiziell in das Release übernommen. Die entsprechenden Funktionen sind vorhanden, werden jedoch nicht von der Oberfläche zur Verfügung gestellt.

Tagging
Um Kreaturen und deren Auswirkung auf die Welt über mehrere Generationen verfolgen zu können, wurde das System des Taggings angedacht. Hier wird eine Kreatur auf Wunsch mit einer zursätzlichen ID versehen, die über Generationen hinweg vererbt wird. Das System ist zum Teil implementiert, jedoch nicht in Verwendung.

Wasserkreislauf
Ähnlich wie sich die Nährstoffe durch Aufnahme der Kreaturen in Form von Energie der Pflanzen und spätere Rückführung durch den Tod auf eine andere Zelle im Laufe des Spiels immer wieder neu verteilt, verteilt sich auch das Wasser. Wasser wird aus mehreren Zellen gesogen, bewegt sich ein paar Zellen und wird nach einer gewissen Strecke wieder dem Boden rückgeführt.

Vier Ansichten
Je nach Richtung in der sich die Kreaturen fortbewegen, sieht man sie aus den verschiedenen Perspektiven.

Dynamische Hintergrundmusik
Die Musik im Hintergrund passt sich dynamisch der Spielsituation an. Wenn zum Beispiel sehr viele Kreaturen am Bildschirm zu sehen sind, steigert sich die Musik.