Spielablauf

Grober Umriss

Um snEADy zu starten braucht ihr einfach nur run.bat auszuführen.
Ein Typischer Spielverlauf von snEADy sieht so aus:

Ganz zu Anfang muss wird das Level und die Spieler geladen. Sobald dann der RUN-Button gedrückt wird, geht's los.

Wenn keine menschlichen Spieler mit spielen läuft snEADy so schnell wie es geht. Für Menschen läuft es aber etwas gebremst.

Sobald alle Schlangen tot sind oder aus der das Level durch den Exit verlassen haben, ist das Spiel zu ende.

Ziel des Spiels

Alle anderen Schlangen zu töten und so möglichst viele Punkte erhalten.

Die Welt

Die Welt von snEADy ist zwar rechteckig, aber an den Rändern kann man von der einen auf die andere Seite wechseln. Die Schlangen können also auf der linken Seite raus fahren und kommt auf der rechten wieder ins Spielfeld rein.
Wände und andere Schlangen begrenzen die Bewegung der Schlange. Wenn sie gegen eine Wand fährt verliert sie ein Segment, wenn sie gegen eine Schlange fährt verliert sie 2 Segmente.

Während des Spiels:

Eine Schlange kann sich pro Spiel Zyklus höchstens ein Feld weit bewegen, In Wirklichkeit ist sie allerdings langsamer.

Im laufe des Spiels werden die Schlangen langsam länger, aber verlieren mit der Zeit auch an Geschwindigkeit.

Goodies

Auf dem Spielfeld werden während das Spiel läuft zufällig Goodies verteilt. Wenn eine Schlange über ein Goodie drüber fährt hat sie es gefressen.
Folgende Goodies gibt es:

- Length: (Blau-Weißen) Sie machen die Schlange länger.
- Speed: (Lila-Roten) macht die Schlange schneller.
- Points, (gelb, sehen aus wie Münzen) gibt dem Spieler Punkte
- Shorter (Schwarz-Blau) Macht die Schlange kürzer
- Slowdown (Schwarz Rot) Macht die Schlange langsamer

Goodies werden während des Spiels zufällig verteilt. Im normalen Spiel erscheinen Length, Speed- und Slowdown-Goodies in zufälligen Intervallen. Wenn eine Schlange stirbt gibt es Points Goodies und während des Sudden Death kommen die Shorter-Goodies zum Einsatz.

Eine Schlange verliert ein Segment

Wenn eine Schlange ein Segment verliert ist meistens eine andere Schlange daran Schuld. Damit falls nicht nur eine Schlange daran beteiligt ist, bekommen alle Schlangen in der Umgebung Punkte, je näher sie dran sind, desto mehr gibt es.
Wenn die Schlange in eine andere Schlange rein gefahren ist, so bekommt die andere Schlange zusätzlich zu den Punkten Umgebungspunkten auch noch Bonuspunkte. Diese erhält sie allerdings nur wenn die andere Schlange auch wirklich in sie rein gefahren ist.
Zusätzlich überträgt die Schlange die ein Segment verliert noch einen Teil ihrer Länge auf die andere Schlange. Achtung! nachdem wir einige Testmuster von Schlangen bekommen haben, haben wir festgestellt, dass dieser Punkt ungünstig ist. Es wird im Turnier keine Längenübertragung geben!
In dem Spezialfall, dass sich eine Schlange selber beißt, bekommt jede Schlange einen Punkt, die sie Berührt. Damit wollen wir den Fall belohnen, dass es ein Spieler schaffen kann, eine andere Schlange einzukringeln, auch wenn er nicht in der Lage ist sie zu töten.

Tod einer Schlange

Eine Schlange stirbt wenn sie nur noch ein Segment lang ist. In dem Fall gibt es noch mal extra viele Punkte und auch der Radius ist größer als wenn sie nur ein Segment verliert.
Auch hier gibt es wieder Bonuspunkte, und zwar eine ganze Menge

Der Exit

Nach einer gewissen Zeit geht der Exit auf. Vorher wird er behandelt wie eine Wand, sobald er allerdings offen ist, können die Schlangen das Level durch den Exit verlassen. Dafür gibt es auch noch mal einen Bonus fürs überleben. Allerdings kann es auch sein, dass mehrere Schlangen überleben. In diesem Fall hängt die Anzahl der Punkte von der Reihenfolge ab, in der die Schlangen ins Exit gehen. Wenn die erste Schlange 50 Punkte bekommt, dann bekommt die zweite 100, die dritte 150 usw...
Sollte nur noch eine Schlange übrig sein, so wird der Exit sofort auf gemacht.

Sudden Death

Nachdem der Exit offen ist, haben die Schlangen eine gewisse Zeit ins Exit zu gelangen. Dannach fängt dann der Sudden Death an. Der sieht so aus, dass überall auf dem Spielfeld Shorter-Goodies verteilt werden. Wer denen nicht mehr ausweichen kann und zu kurz wird, stirbt. Je länger der Sudden Death dauert, desto mehr Goodies werden erscheinen, es wird also zunehmend schwerer werden überhaupt zu überleben.

Bedienung von snEADy

Das meiste in snEADy ist selbsterklärend und intuitiv. Was nicht sofort ersichtlich ist, könnt ihr nachfolgend lesen.

Die Playerliste

Die Spielliste zeit alle geladenen Spieler, sortiert nach ihrer Punkte. Wer also die meisten Punkte hat steht ganz oben.
Mit dem Halten-Verboten zeichen rechts von einem Spieler könnt ihr ihn wieder entfernen. Die nachfolgenden Spieler rutschen dann auf.

Playermenü und Levelmenü

Da könnt ihr Level oder Player laden.

Multiplayer

Um mit mehreren Menschlichen Spielern zu spielen, brauchen nur mehrere geladen zu werden. Es wird dann an einer Tastatur gespielt, die Tastenbelegungen sind dann so:

Spieler 1: Pfeiltasten
Spieler 2: WASD
Spieler 3: 8462 (Numpad)
Spieler 4: HBNM

Optionsmenü

Exittime gibt an nach wie viel Runden der Exit geöffnet werden soll.
Sudden Death Time gibt an wann die Sudden Death Phase beginnt.

Init Length und Init Speed geben an wie lang bzw. schnell die Schlangen am Anfang sein sollen.

Automatic Grow und Slowdown geben an, wie schnell die Schlangen automatisch wachsen bzw. langsamer werden.

Goody Occurence:

Max Delay: dieser Wert beschreibt wie häufig Goodies erscheinen sollen. je kleiner der Wert ist, desto öfter erscheinen sie.

Die vier Werte darunter geben an wie in welchem Verhältnis die Goodies (Length, Speed, Slowdown und Points) auftreten sollen.

Es wird also erst zufällig ausgewählt wann ein Goodie erscheinen soll, und dann welches das sein wird. Dabei kommt es nur auf das Verhältnis der Werte zueinander an.

TimeKill gibt an ob ein Spieler bei einer Zeitüberschreitung getötet werden soll, (das geht natürlich nur bei Computer Spielern)

Max Time gibt die maximale Zeit an, die die Computerspieler haben zum überlegen

und Max Memory ist der maximale Speicherplatz.

Show subcycles hat erst mal keinen deutlichen sichtbaren Effekt. Aber wenn eine sehr hohe Max. Time eingestellt ist, dann kann man eventuell schön sehen wie sich die Schlangen nach einander bewegen. Ansonsten wird das bild erst am Ende einer Runde aktualisiert.

Player Time und Player Memory geben an ob die Berechnungszeit und Speicherverbrauch der Spieler in die Konsole ausgegeben werden sollen. Die Werte stehen auch noch mal im Log, sollte es unter Activate Logging eingeschaltet sein.

Turnier

Wie das Turnier letztendlich aufgebaut ist, wissen wir erst Anfang Merz, wenn fest steht wie viele Spieler mit machen. Fest steht, dass immer 4 Schlangen in einer Arena mit einander kämpfen werden.

Der Turniermodus

Sobald wir die Teilnehmerzahl wissen, werden wir uns für einen geeigneten Tourniermodus entscheiden.

Wir haben bis jetzt (24.02.05) etwa 70 Anmeldungen. Das sind eine ganze Menge und wenn wir jeder gegen jeden spielen lassen würden, bräuchten wir Jahre um fertig zu werden.

Deswegen werden wir uns ein möglichst faires System ausdenken, so dass jeder Spieler gegen möglichst viele verschiedene Spielerkombinationen antreten kann. Dass wir den Wettkampf dann in irgend einer Weise auf Splitten müssen steht daher außer Frage. Aber wir versuchen ein knock out zu vermeiden, so dass jeder Spieler bis zum Ende mitspielen kann.

Faires Spiel!

Oberstes Gebot ist natürlich ein faires Spiel. Das bedeutet, dass ihr die zweifellos vorhandenen schwächen im Programm nicht ausnutzen werdet. Wir haben versucht alles so sicher zu machen wie es möglich uns möglich ist, aber auch wir sind nicht perfekt. Wir würden uns freuen, wenn ihr uns von Schwachstellen berichten würdet, damit wir sie beseitigen können.

Ihr dürft keine eigenen Threads benutzen.

Eine Sache gibt es, die gar nichts mit dem Spiel selbst zu tun hat. Solltet ihr euren Spieler in dem Home-Verzeichnis von eurem CS-Account abspeichern, dann sorgt bitte dafür, dass die Rechte eures Verzeichnisses so gesetzt sind, dass es nicht von anderen geöffnet werden kann. Standardmäßig sind die Sicherheitseinstellungen so, dass jeder in euer Home-Verzeichnis rein schauen kann, er kann aber nichts ausführen oder schreiben. Diese Einstellungen könnt ihr am einfachsten ändern indem ihr an den Sun-Rechnern direkt die Rechte der Ordner verändert.

Belegbedingungen

Für die Teilname am Turnier sowie die für den Beleg (Zulassung für die EAD-Klausur) muss euer Spieler folgende Bedingungen erfüllen:

- Ein faires Spiel
- Der Spieler sollte keine Exceptions verursachen.
- Der Spieler muss innerhalb der geforderten Zeit seinen Zug berechnen
- Er darf nur begrenzt viel Speicher belegen.
- Er muss die Schlangenprüfung absolvieren können. Das heißt, er sollte im Level "Beleg" einen einzelnen LukeWallwalker schlagen können. Schlagen bedeutet in dem Zusammenhang, dass Luke eindeutig durch das Einwirken eurer Schlange getötet wird. Wir werden diesen Test mehrmals laufen lassen damit wir sicher sein können, dass es

kein Zufall war. Es ist egal wie viele Punkte die Schlangen am Ende haben. Für den Beleg zählt nur, dass eure Schlange den armen Luke tötet.

Das schwierigste wird wahrscheinlich sein, Luke Wallwalker zu schlagen. Schaut euch einfach genau an was er macht, dann wird euch bestimmt eine Möglichkeit einfallen ihn zu besiegen.

Auf welcher Hardware das Spiel laufen wird

Der Wettbewerb wird auf Folgenden Rechnern ausgetragen:

Alle sind Sun Blade 1500:

Sun Pool 1

Hemrelin	141.44.21.60
Otter	141.44.21.61
Marder	141.44.21.62
Nerz	141.44.21.64
Eule	141.44.21.119
Uhu	141.44.21.121
Kauz	141.44.21.122
Dommel	141.44.21.123

Sun Pool 2

Pfirsich	141.44.21.40
Quitte	141.44.21.41
Aprikose	141.44.21.43
Pflaume	141.44.21.44
Birne	141.44.21.45

Einstellungen von snEADy für das Turnier

Für das Turnier wird das Script "tournament" ausgeführt.

Folgende Einstellungen werden für das Turnier verwendet: dies ist nur eine Auswahl der wichtigsten, wenn ihr alles genau wissen wollt, dann schaut ins tournament.script.

Berechnungszeit	100 Millisekunden
Zeit das Spielfeld zu analysieren	1 Minute

Maximaler Platzverbrauch	8 MB
Startlänge	20
Anzahl Zyklen bis die Schlange um 1 länger wird	50
Anzahl Zyklen bis die Schlange um 1 langsamer wird	1000
kleinstes Bewegunsgdelay (höchste Geschwindigkeit)	2
sart- Bewegungsdelay (Anfangsgeschwindigkeit)	4
größtes Bewegunsgdelay (niedrigste Geschwindigkeit)	6
Effekt der Length Goodies	+20 Segmente
Effekt der Points Goodies	+10 Punkte
Effekt der Shorter Goodies	-5 Segmente
Effekt der Speed/Slowdown Goodies	-/+ 1 Speed Delay
Exit Time	5000
Sudden Death Time	7000

Punkte (wie die Punkte berechnet werden steht weiter unten)

Punkteradius für den Kill eines Segments (dist = Δrow + Δline)	1
Punkteradius für den Kill einer Schlange (dist = Δrow + Δline)	10
Extrapunkte für Schlangen, die eine Schlange berühren die sich selber beißt	1

Welche Levels werden für das Tournier verwendet?

Wir sind noch nicht so weit und haben noch keine Tournierlevels fertig gestellt. Wie viele es geben wird, hängt von der Teilnehmerzahl und der Tourniermethode ab, die wir letztendlich wählen werden.

Fest steht, dass auch unbekannte Levels benutzt werden. Vor allem gegen Ende des Tourniers, wenn es darum geht den Sieger fest zu stellen, werden auf jeden Fall unbekannte Levels benutzt.

Wenn ihr Lust habt, designd euch eigene Levels, eine Genaue Anleitung dafür wird es in Kürze geben. Wir geben euch schon zwei Testlevels mit, aber diese sind nicht repräsentativ für das Spiel. Die Turnierlevels kommen dann so bald wie möglich.

Siegbedingungen

Bei einem Fight zählen nur die Punkte. Ob die Schlangen überleben ist nicht wichtig, es zählen allein die Punkte. Die Platzierung der Spieler ist dann für das Turnier entscheidend. Die erreichten Punkte werden als Sekundärwertung gespeichert, falls zwei Spieler gleich auf sind.

Ordner im snEADy Verzeichnis

player

Im player-Ordner sind alle Spieler die geladen werden können. Beim Punkt Implementierung steht was ihr machen müsst um einen eigenen Spieler hinein zu laden.

levels

Im levels Ordner sind die Level-Dateien drin gespeichert.

scripts

Im scripts Ordner sind start-scripts drin, die beim Start des Programms ausgeführt werden können. Möchte man ein Script beim start des Programms aufrufen, so reicht der Aufruf:

```
java -jar snEADy.jar -script myScript
```

Wird beim Start von snEADy kein Skript angegeben, so wird default.script ausgeführt.
Die Turnierbedingungen können auch mit

logs

Wärend des Spiels werden eventuell viele interessante Daten gewonnen. Und auch um kontrollieren zu können wer gewonnen hat bzw. später ein Spiel nachvollziehen zu können gibt es die Logs. Diese werden im Ordner logs abgespeichert.

pics

Im Ornder pics befinden sich die Bilder für die Grafische Oberfläche.

Implementierung eines Spielers

Dateistruktur

Legt für euren Spieler bitte ein eigenes Package mit dem gleichen Namen wie euer Spieler heißt an. In dieses Package kommen alle Klassen rein, die ihr für euren Spieler braucht. Der Packagename sollte außerdem klein geschrieben sein.
Im player-Verzeichnis selbst muss dann noch eine Klasse rein, die nur zu eurer eigentlichen Spielerklasse weiter leitet.
Ein Beispiel wäre:

player\MyPlayer.java
player\MyPlayer.class
player\myplayer\MyPlayer.java
player\myplayer\MyPlayer.class
player\myplayer\Hilfsklasse1.java
player\myplayer\Hilfsklasse1.class
player\myplayer\Hilfsklasse2.java
player\myplayer\Hilfsklasse2.class
player\myplayer\MyDoku.txt

Bitte schreibt eine Dokumentation zu eurem Spieler in einem freien Format (also nicht Word) und legt sie mit in das Verzeichnis von eurem Package.

die Klasse "MyPlayer", die direkt im player-verzeichnis liegt, ist dazu da, um das Laden der Spieler im Programm zu vereinfachen. Sie leitet eigentlich nur auf eure richtige Spielerklasse weiter, die im Package liegt. Die Java-Datei "player\MyPlayer.java" sieht so aus:

```
public class MyPlayer extends myplayer.MyPlayer {}
```

Was programmiert werden soll

Eure Aufgabe ist es, eine Java-Klasse
eure Klasse von der snEADy-klasse Player abzuleiten und die Funktion calculate zu implementieren. In dieser Funktion soll euer Spieler möglichst intelligent entscheiden wohin die Schlange ziehen soll.

Kompilieren

Mit folgender Zeile könnt ihr euren Spieler aus dem player-Verzeichnis heraus zu kompilieren:

Windows:

```
javac -classpath ..\snEADy.jar;. MyPlayer.java
```

Linux/Unix:

```
javac -classpath ../snEADy.jar:. MyPlayer.java
```

Das Interface 'Serializable'

Wir benutzen die Schnittstelle 'Serializable' um zu kontrollieren wie viel Speicher euer Spieler verbraucht. Bei der Berechnung werden statische Variablen aber nicht beachtet. Wir haben nichts dagegen dass ihr statische Variablen benutzt, aber bitte speichert keine großen Datenmengen als statisch ab, das wäre unfair den anderen Spielern gegenüber.

Was ihr machen müsst ist einfach nur das Interface in jede eurer Klassen zu implementieren, etwa so:

```
import java.io.Serializable;
public class MyClass implements Serializable, MyInterface
```

Wenn ihr von einer anderen Klasse ableitet, die Serializable bereits implementiert, dann braucht ihr es in der abgeleiteten Klasse nicht noch einmal tun. Es schadet aber auch nichts.
Weiter gibt es nichts zu tun. Ihr braucht keine Funktionen von Serializable implementieren, java hat da schon einige Standardfunktionen, die wir benutzen.

Durch Serializable kann es außerdem vorkommen, dass beim kompilieren einige Warnungen auftauchen. Die könnt ihr getrost vernachlässigen. Solltet ihr unsicher sein ob die Warnungen von eurem Spieler stammen oder von dem Interface, dann entfernt es einfach mal. Sollten die Warnungen dann nicht verschwinden liegt es nicht an dem Interface.

Was muss ich für den Beleg wissen?

Schaut euch bitte die Dokumentation zu snEADy an. Sie liegt im Verzeichnis doc. Das ist eine vollständige Doku und für euch Größtenteils nicht interessant. Allerdings solltet ihr euch das Package "player" anschauen. Das könnt ihr in der linken oberen Liste auswählen.

Allgemeines zu snEADy

snEADy ist wie ein Runden basierendes Gesellschaftsspiel aufgebaut. In jeder runde im Spiel darf jeder Spieler ein mal überlegen. Damit die Schlangen aber auch unterschiedlich schnell sein können, darf sie nicht jedes mal wenn der Spieler überlegt hat ziehen. Eine genauere Beschreibung dazu findet ihr in der Dokumentation zu der Klasse OwnSnakeInfo und GameInfo.

Jeder Spieler erhält die gleichen Informationen, die der Menschliche Spieler auch hat. Er sieht das gesamte Spielfeld und alle anderen wichtigen Daten. Schaut euch einfach die Dokumentation zu dem Package 'player' an, da ist jeder Wert genau beschrieben.

Technisches

Von eurem Spieler wird beim starten des Spiels eine Instanz gebildet, die das ganze Spiel hindurch erhalten bleibt. Ihr könnt also alles was ihr berechnet habt die ganze Zeit über behalten. Damit alles fair bleibt, kontrollieren wir sowohl den Speicherverbrauch eures Spielers als auch die Zeit die der Spieler für seine Berechnung benötigt.

Grober Ablauf eines Spielzyklusses

Alle Spieler denken nach einander über ihren Zug nach. Wenn eine Schlange an der Reihe ist sich zu bewegen tut sie es direkt nachdem ihr Spieler den Zug berechnet hat. Der Spieler der danach dran ist, sieht dann schon das aktualisierte Spielfeld, so dass keine Situation entstehen kann, dass zwei Schlangen gleichzeitig auf ein freies Feld ziehen.

Etwas mehr Details zum Spiel

Scripting

Der Start von snEADy wird über scripte geregelt, auch alle Einstellungen und alles was man mit der grafischen Oberfläche einstellen kann geht auch über Scripts. Ich möchte hier nicht alles Haar klein erklären, wenn ihr genau wissen wollt was welche Befehle machen dann schaut einfach in die Dokumentation von M_Main.control. Wenn dann trotzdem noch Fragen sind, dann schreibt einfach etwas ins Forum.
Das Prozenteichen: '%' ist ein Kommentarzeichen. Alles was in einer Zeile hinter dem % kommt wird ignoriert. Außerdem werden alle Tab- und Leerzeichen am Anfang und am Ende einer Zeile weg geschnitten.
Aber bitte bedenkt, dass die grafische Oberfläche die Befehle ein klein wenig anders dar stellt, es mag also bei einigen Befehlen vor kommen, dass sie entweder gar kein, oder ein etwas abgewandeltes Handling haben als ihr es von der grafischen Oberfläche gewohnt seit.

Die Konsole

Es gibt in snEADy eine Eingabekonsole bei der ihr Script-Befehle eingeben könnt. Sie wird mit F12 geöffnet. Einfach die Befehle eintippen, die funktionieren genauso wie im Script.

Leveldesign

Beim Leveldesign gibt es einiges zu Beachten. zunächst ist es vielleicht nicht unwichtig zu wissen, dass die Datei zeilenweise ausgelesen wird. Deswegen werden nur Befehle, die am Anfang der Zeile stehen verarbeitet. Das Kommentarzeichen '%' funktioniert aber wieder genauso wie in den Scripts.

Zu den Befehlen:
<name> Der name des Levels. Er muss nicht zwangsläufig der selbe sein wie die Datei heißt. Es dürfen auch Leerzeichen verwendet werden.
<size> 75x75 Die größe des Levels. Passt bitte auf, dass das Spielfeld genau diese Maße hat, es wird "Höhe x Breite" gelesen
<maxPlayer> die Maximale Anzahl an Spielern die für das Level vorgesehen ist.
<comment> Hier kann ein mehrzeiliger Kommentar zum Level stehen.

<endcomment>

<autor> Der Autor des Levels
<playField> zwischen playField und endPlayfield wird das Spielfeld definiert.

Mögliche Zeichen:
'.' : freies Feld
'#': wand
'A': Exit
'0'-'9': Startpositionen der Schlangen
'a'-'e': Goodies, die bei Spielstart schon rum liegen sollen.

Beim Leveldesign gibt es einige Sachen zu beachten:
Damit die Wände im Spiel richtig dargestellt werden können, muss jede Wand in alle Richtungen mindestens 2 Felder dick sein. Da die Welt rund ist, verbinden sich auch die Wände an den Ecken. Bitte beachtet also, dass die Wände an den Rändern auch zusammen passen.
Der Exit muss immer aus einem 5x5 - Feld voll 'A's bestehen. Es kann derer beliebig viele geben, so viel halt Platz ist. Zwei Exit - Blocks sollten sich möglichst nicht berüren. Es gibt keine Garantie dass die dann noch richtig dargestellt wird.

Für die Turnierlevels gilt: sie müssen doppelt symmetrisch sein, so dass alle 4 Schlangen am Anfang die gleichen Chancen haben.
<endPlayfield>

Einbinden eigener Grafiken:
Um ein Level grafisch an die eigenen Bedürfnisse anzupassen, ist zunächst im Ordner Levels ein Unterordner anzulegen, dessen Name dem **Filenamen** des Levels entspricht.
In diesen müssen alle Grafiken die anstelle der Standardgrafiken genutzt werden sollen, reinkopiert werden.
Ein kompletter Levelskin besteht aus folgenden Grafiken:

```
floor.png
wall.png
```

`exit.png`
`walledges.png`

Die Standardgrafiken befinden sich im Ordner `pics/game/`
Ein Levelskin muss nicht alle Grafiken beinhalten. Wird eine der Grafiken nicht gefunden, so wird automatisch die Standardgrafik benutzt.

WICHTIG!
Alle Grafiken, müssen bestimmten Normen entsprechen.
Für `floor.png` und `wall.png` gilt, dass ihre Größe, sowohl in x- als auch in y-Richtung eine durch 6 teilbare Zahl ist.

Die Grafiken `exit.png` und `walledges.png` müssen exakt die Größe der Originale haben.

`exit.png`	`60x30`
`walledges.png`	`72x6`

Dabei setzen diese Grafiken sich aus Einzelelementen zusammen. Beispielsweise beim Exit gibt es den geschlossenen (0<=x<30) und den offenen (30<=x<60).
Bei den Walledges handelt es sich um die Grafiken für die Wandkannten. Es ist nicht ganz einfach solch eine Grafik zu erstellen. Wem das zu kompliziert ist, lässt die Grafik einfach standard. Ich mein so hässlich sind die Standardwandgrafiken ja auch nicht. Für alle die es trotzdem versuchen wollen, gibt's hier einige kurze Bemerkungen:
Jedes Einzelteil (z.B. ‚Nord') ist 6x6 Pixel groß (Die Größe eines Spielfeldelements) und beschreibt die Himmelsrichtung, in die die Wand ‚schaut'. Dann gibt es noch konkave und konvexe Ecken. Die Reihenfolge, in der die Einzelelemente in der Grafik abgespeichert werden ist wie folgt:

Nord West konkav,
Nord West konvex,
Nord ,
Nord Ost konkav,
Nord Ost konvex,
Ost ,
Süd Ost konkav,
Süd Ost konvex,

Süd ,
Süd West konkav,
Süd West konvex,
West ,

Bei fragen schreibt eine Email an lars@sneady.de

Interpretation der Logdateien

Die log-Dateien enthalten Angaben über Aktionen im und um den Spielablauf.
Schaut sie euch an, sie sollten eigentlich selbsterklärend sein.
In control.log werden alle Befehle gespeichert, die in irgend einer Weise eingegeben wurden. Also per GUI, Script oder Konsole.
In player.log werden alle Aktionen der Player gespeichert. In result.log wird das Ergebnis am Ende des Spiels gespeichert.

Weitere Details zur Implementierung

Was es mit den Goodies auf sich hat

Goodies werden auf dem Spielfeld zufällig verteilt. Die Wahrscheinlichkeit, dass ein Goodie erscheint ist gleich verteilt zwischen 0 und Goodies Occurence: MaxDelay. Welches Goodie erscheint ist wiederum gleich verteilt, über die Summe der Probabilities. Das bedeutet, die Wahrscheinlichkeit dass ein Length Goody erscheint ist Len/(Len + Spd + Slwdn + Pts). Gleiches gilt natürlich für die anderen auch.

Wenn ein Goodie platziert wird, dann wird zufällig eine Position berechnet, und wenn das Feld frei ist, wird das Goodie an die Stelle gesetzt. Sollte das Feld nicht frei sein, so wird maximal 100 mal probiert es zu setzen, sollte dann immer noch kein freies Feld gefunden worden sein, so wird es nicht gesetzt.
Genauso wird auch verfahren, wenn die Point Goodies bei dem tot einer Schlange verteilt werden. Sollte kein freier platz gefunden werden, so wird das Goodie nicht gesetzt. Es kann also sein, dass weniger als die normalerweise 5 Goodies erscheinen.

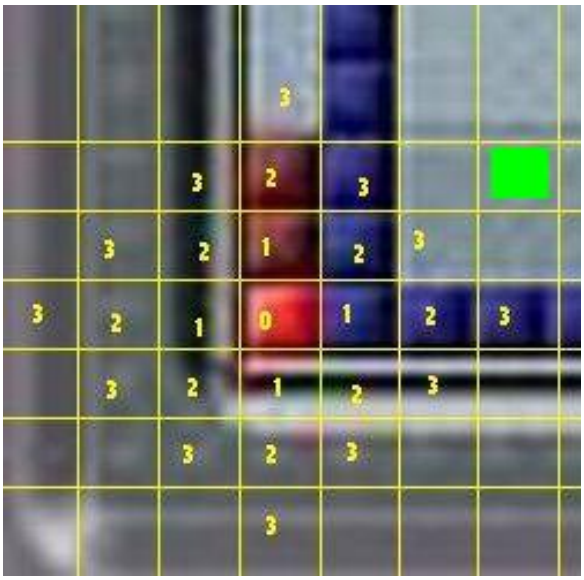
Während des Sudden Death werden nur noch Shorter Goodies erzeugt. Davon wird in jedem Spiel Zyklus eins gesetzt. Auch hier gilt: wenn nach 100 Versuchen kein platz gefunden wurde, wird kein Goodie gesetzt.

Das Punktesystem

Abstandsberechnung:

Letztendlich geht es ja darum Punkte zu bekommen. Hier wird nun erklärt wie es funktioniert. Zunächst ist wichtig zu wissen wie Abstand in snEADy definiert ist.

Weil die Schlangen sich nur wagerecht oder senkrecht bewegen können wird eine Entfernung auch genau so definiert. Also ist der Abstand D zwischen zwei Punkten A und B immer D = |A.r - B.r| + |A.l - B.l|. Also immer die Addition der Reihen (r = row) und Spaltendifferenz (l = line).



In diesem Bild ist verdeutlicht wie die Entfernung berechnet wird. Da wo der rote Kopf ist, ist die Entfernung natürlich 0. In den 4 angrenzenden Feldern ist sie 1, an die angrenzenden Felder 2 usw... Die Distanz ist also die Anzahl der Züge, die eine Schlange bräuchte um auf direktem weg dort hin zu gelangen.

Punkteberechnung:

Wenn eine Schlange ein Segment verliert oder gar stirbt, bekommen Schlangen in der Nähe Punkte. Die Anzahl der punkte hängt von der Entfernung ab. Im oberen Bild verliert die Rote Schlange im nächsten Zug ein Segment und im darauf folgenden stirbt sie.

Die Punktevergabe wird für beide Fälle grundsätzlich gleich berechnet. nur dass es für den Kill wesentlich mehr Punkte gibt als für ein Segment.

Für beide gibt es einen Radius: Segment Radius und Kill Radius. (Im Script heißen sie etwas anders). Im Turnier ist der SR = 1 und KR = 10. Für die Berechnung der Punkte gilt:

Der Kopf der Schlange die ein Segment verliert oder getötet wird ist das Zentrum. Wenn eine Schlange nun ein Segment innerhalb des Radius hat, bekommt sie 2^(Radius-Dist) Punkte. Dabei ist die Entfernung von einer Schlange zum Zentrum immer die kürzeste Entfernung aller Segmente dieser Schlange zum Zentrum.

Angenommen im Oberen Bild stirbt die rote Schlange, dann gibt es für die Blaue 2^(10-1) = 512 Punkte, und für eine weitere Schlange die bei dem Grünen Feld ein Segment hätte 2^(10-5) = 32 Punkte. gleiches gilt auch wenn die rote Schlange nur ein Segment verliert, aber da der SR 1 ist, bekommt nur die blaue Schlange 2^(1-1) = 1 Punkt.

Wenn eine Schlange in eine andere Schlange rein fährt, dann bekommt diese die doppelte Anzahl an Punkte.

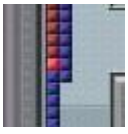
Außerdem gilt: wenn eine Schlange in eine Wand fährt verliert sie ein Segment, fährt sie gegen eine Schlange (egal ob sie es selbst oder eine andere Schlange ist) verliert sie 2 Segmente.

Arten der Punktevergabe:

Im letzten Abschnitt ist zwar schon alles beschrieben, aber hier noch einmal visualisiert, für ein besseres Verständnis.
Um ein bisschen Platz zu sparen ist die rote Schlange einfach R, und die blaue entsprechend B. Und damit wir ein bisschen mit Zahlen arbeiten können, nehme ich die Turniereinstellungen als Beispiel für die Punkteberechnung.

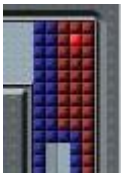
R verliert ein Segment an der Wand, und B bekommt 2^(1-1) = 2^0 = 1 punkte.

R wird gleich sterben und B wird 2^(10-1) = 2^9 = 512 Punkte bekommen.



R verliert zwei Segmente weil er in eine Schlange rein fährt und B bekommt also 2 * 2^(1-1) = 2 * 2^0 = 2 Punkte.

R wird gleich sterben und B wird 2 * 2^(10-1) = 2 * 2^9 = 1024 Punkte bekommen.



R verliert zwei Segmente weil er gegen sich selbst fährt, und B bekommt einen Punkt, weil er R berührt. (dass die rote Schlange hier ganz eingeringelt ist, ist Zufall, das muss nicht sein. Es genügt wenn sie sich an einem Feld berühren.)

Nur für die, die gern alles wissen möchten...

coming soon..

Genauer Ablauf eines Spielzyklusses