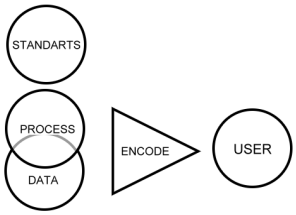
Rollenspiel mit KI-Elementen



Belegarbeit im Fach Webengineering
Wintersemester 2015/16

Inhaltsverzeichnis

1 Sinn und Zweck des Spiels

Wie in jeden Rollenspiel geht es darum, denn Charakter hochzuleveln, sich mit Rüst- und Kampfgegenständen auszurüsten und gegen den Endgegner anzutreten. Die Grundlegenden Dinge wurden entsprechende umgesetzt(ausgeklügeltes Shop-/Inventar-/Rüstsystem, verschiedene Monsterrassen, etc.) und an einen existierendes Spiel names Diablo angepasst. Fast alle Grafikelemente sind diesem Spiel entnommen und vom Author an die entsprechenden Umstände angepasst. Dabei wurde keine einheitliche Grafik benutz, sondern aus verschiedenen Teilen zugestellt.

Hochleveln:
Für ein im Kampf besiegtes Monster bekommt man Erfahrungspunkte. Man benötigt sie, um einen Stufenanstieg durchzuführen und auf eine höheres Level zu steigen. Mit den einem Stufenanstieg freigesetzten Punkte, kann der Spieler nach belieben die Charakterwerte erhöhen und so einen individuellen Charakter schaffen.

Rüst- und Kampfgegenstände:
Knapp 80 Rüst- und Kampfgegenstände stehen den Spieler über ein Inventar- und Shopsystem zu Verfügung. In gewissen Gebieten der Karte kann man Gegenstände in Truhen finden, die sich frei verwalten, anlegen bzw. verkaufen lassen.

Endgegner:
In den tiefen des nördlichen Turmes wartet der Endgegner darauf deine Charakter in einen Kampf zu verwickeln.

Link zum Spiel:
`http://new-lv.de/Diablo.htm`
Link zur PDF:
`http://new-lv.de/DiabloP.pdf`

2 Verwendete Technologien

2.1 Javascript und HTML

Im ganzen Spiel kommen zwei Technologien bzw. Programmiersprache zur Anwendung.
Zum einen HTML(Hypertext Markup Language) eine Auszeichnungssprache, die die Metabefehle als Dokumenttypdefinition (DTD) speichert und Tags verwendet und zum anderen Javascript bzw. ECMA-Script, um dynamisch die Tags zu verwalten, aktualisieren, Userdaten auswertet, etc..

2.1.1 HTMLTags

Tags sind die aus Dokumenttypdefinition geladen Befehle, die ein HTMLDokument ausmachen.
Hier sind einige der grundlegendes Befehle aufgeführt, die benötigt werde, ein HTMLDokument sinnvoll darzustellen.

Text
Traditionell kann Text im body, table und divcontainer dargestellt werden. Man kann ihn mittels Tags bzw. CSS bearbeiten.

```
<font size=''></font>
<span class=''></span>
```

Bild
Bilder kann man mittels Imagetag darstellen. Dieser Tag kann recht komplex anmuten und mehr als 20 Parameter in sich bürgen. Er besitz die besondere Eigenschaft: gibt kein Tag zum schließen.

```
<img src='pathToImage.jpg' width='100' height='100' border='0' id='Img1' style='position:absolute;' />
```

Link
Ein Link ist profaner Text, oder Bild der mit den Tag: 'a href' versehen wurde. Die Parameter und Befehle aus dem Text- bzw. Bildtag gelten auch für den Link.
Über eine Link können auch Funktion aus dem Javascript aktiviert werden.

```
<a href='url' target='_parent'>t</a>
<a href='' onMouseOver='proc1()' onClick='proc2()'></a>
```

Table
Tabelle in denen sich Text/Link/Bilder/ect. abbilden lassen. Ermöglicht browserabhängig Positionierung der Elemente.

```
 <table width='100' height='100' cellspacing='5' cellpadding='3'>
  <tr>
   <td class='tableHead'></td>
  </tr>
 </table>
```

Divcontainer
Spezielle Ebenen(Layer) in denen sich Text/Link/Bilder/ect. abbilden lassen. Ermöglicht browserunabhängig exakte Positionierung der Elemente. Auch sehr Parameterlastig.

```
 <div id='Layer1' style='position:absolute; width:100px; height:100px; z-index:1; left:0px; top:0px;
   visibility:visible; opacity:0.3; border:1px none;'></div>
```

Beispiel
Einfaches HTMLDokument. Enthält die oben aufgezählten Tags.

```
<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.01 Transitional//EN'>
 <html>
  <head>
   <title>HTMLDokument</title>
   <meta http-equiv='Content-Type' content='text/html; charset=iso-8859-1'>
   <script language='JavaScript' type='text/JavaScript'>
    function proc1(x){alert(x);}
   </script>
  </head>
 <body>
  <div id='Layer1' style='position:absolute;width:100px;height:100px;z-index:1'>
   <table width='100 height='100'>
    <tr>
     <td>
      <a href='' onMouseOver='proc1(100)'><img src='pathToImage.jpg' border='0'></a>
```

```
     </td>
    </tr>
   </table>
  </div>
 </body>
</html>
```

2.1.2 Javascript: Variablen und Funktionen

Variabeln sind Objekte mit Name, Datentyp und Wert. In Javascript eine Variable zu definieren, wird der Befehl 'var' benutzt. Bsp: var x=0; Ihr wird keine sichtbarer Datentype zugeordnet, im Vergleich zu Java mit den definierten Datentypen String und int. Variabeln können global bzw. lokal verwendet werden, ist aber aus dem Namen nicht erkennbar und leitet sich vom Entstehungsort ab.
Als Funktion werden Objekte mit der Deklaration 'Funktion' bezeichnet. Es werden Werte übergeben, mit mathematischen, oder funktionalen Elementen bearbeite und möglicherweise mittels 'return' zurückgegeben. In Javascript habe alle Funktionen den gleichen Wert, d.h. eine Hierachie nach private und public existiert nicht.

2.1.3 Javascript: Datentypen

Wie bereits erwähnt kann man den Datentyp nur vom Wert bzw. Index(Array) ableiten. Der einfachste Weg ist, den Prozess bis zur Entstehung der Variable zurück zu verfolgen, um den ursprünglichen Datentyp zu erfahren. Leider kann der ursprüngliche Datentyp verloren gehen. Der Befehl 'typeof' hilft dabei den momentanen Datentyp zu ermitteln.
Nachfolgender Quellcode gibt eine kleinen Einblick in das Thema Javascript und Datentypen. Hier werden nicht nur einfache Datentypen dargestellt, sondern auch komplexe, wie Klassen und Cookies.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html><head>
  <title>HTMLDokument</title>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
  <script language="JavaScript" type="text/JavaScript">
   var int=0;
   var string="text";
   var array=[0,1];
   function proc1(x){
    var obj={get func1(){return 0;}}
    setCookie("c1",100);
    var str="<table cellpadding='5'>";
    str=str+"<tr><td width='50'>Integer</td><td width='150'>var int=0;</td><td>"+typeof int+"</td></tr>";
    str=str+"<tr><td>String</td><td>var string='text';</td><td>"+typeof string+"</td></tr>";
    str=str+"<tr><td>Array</td><td>var array=[0,1];</td><td>"+typeof array+"</td></tr>";
    str=str+"<tr><td>Objekt</td><td>var obj={get func1(){return 0;}}</td><td>"+typeof obj.func1+"</td></tr>";
    str=str+"<tr><td>Cookie</td><td>setCookie('c1',0) </td><td>"+typeof getCookie("c1")+"</td></tr></table>";
    document.getElementById("Layer1").innerHTML=""+str;
   }
   function setCookie(N,V){document.cookie=N+"="+V+";";}
   function getCookie(N){
    var n=N+"=";
    var C=document.cookie.split(';');
    for(var i=0;i<C.length;i++){
     var c=C[i];
     while(c.charAt(0)==' ')c=c.substring(1);
     if(c.indexOf(n)==0)return c.substring(n.length,c.length);
    }
    return "";
   }
  </script>
 </head>
<body>
 <div id="Layer1" style="position:absolute;width:500px;height:100px;left:15px;z-index:1">
  <table cellpadding="5">
   <tr><td width="50">Integer</td><td width="150">var int=0;</td></tr>
   <tr><td>String</td><td>var string="text";</td></tr>
   <tr><td>Array</td><td>var array=[0,1];</td></tr>
   <tr><td>Objekt</td><td>var obj={get func1(){return 0;}}</td></tr>
   <tr><td>Cookie</td><td>setCookie("c1",0) </td></tr>
   <tr><td colspan="2"><a href="" onMouseOver="proc1(100)">berechnen</a></td></tr>
  </table>
 </div>
 </body>
</html>
```

2.1.4 Javascript: Befehle

Ein Programm ohne Befehle und Anweisungen gibt es nicht und jede noch so grundlegende Aktion benötigt Befehle bzw. Befehlsstrukturen. Ob das Programm ein Packet importiert, eine Variable erstellt, bearbeitet und ausgibt, oder eine Funktion aufgerufen wird, immer sind es Befehle, die die Programmstruktur ermöglichen.
Alle Befehle für Javascript aufzulisten, wäre mit immensen Aufwand verbunden und auch auf einen Einblick in die grundlegenden Befehle, soll an dieser Stelle verzichtet werden.
Im Vordergrund stehen ganz klar die Anweisungen, die im vorgestellten Programm eine Rolle spielen, die sich als Befehle vom 'Document Object Modeling' abheben und nicht als mathematische einzustufen sind.
Eine kleine Anzahl von Befehlen entsprechen diesen Kriterien und sind als folgenden Tablelle erläutert.
Einige davon gehören zu den grundlegendes Elemente die in diesem Spiel vorkommen, z.B. 'parseInt()' mit 83 Aufrufen.

Befehl	**Bedeutung**	**Beispiel**
parseInt	String→Integer	$varint1 = parseInt(array[0]);$
replace	ersetze mit Wert	$d.style.left.replace(/px/,);$
split	splitte mit Wert	$varstring_array = string.split(" : ");$
eval	evaluiere String	$eval('varv' + i + ' = 0;');$
substring	kürze String	$string.substring(0, string_counter);$

2.1.5 Javascript: Kontrollstrukturen

Kontrollstrukturen steuern den Datenfluss anhand von Werten, Parameter und Funktionen.
Gleich den Befehlen ist ein Verzicht auf Kontrollstrukturen undenkbar.

```
Werten:     if(x==1){}
Parameter:  if(typeof x!='undefined'){}
Funktionen:
function f1(x){return Math.floor(x/2);}
            if(f1(x)>3){}
```

Profane Kontrollstrukturen wie grösser, kleiner, gleich, positiv, negativ, etc. werden in der nächsten Tabelle ausser acht gelassen und nur spezielle ausgewählte abgebildet.

Befehl	**Bedeutung**	**Beispiel**
typeof	Typ der Variable	$if(typeofvar! =' undefined'){}$
isNaN	is not a Number	$if(isNaN(x)){}$
indexOf()	ermittelt Index	$if(navigator.userAgent.indexOf('Firefox')! = -1){}$

2.2 Javascript: Mathe

Es gibt spezielle Befehle die für die Arbeit mit Nummern.
Die hier aufgeführten Befehl sind stille Begleiter fast jeder Funktion des Spiels.
Allein Math.floor() kommt 73 mal und random() knapp 63 mal im Quellcode vor.

Befehl	**Bedeutung**	**Beispiel**
Math.floor()	rundet ab	$Math.floor(1/3);$
Math.ceil()	rundet auf	$Math.ceil(1/3);$
Math.random()	Zufallszahl	$Math.random() * 10;$
Math.abs()	Betrag	$Math.abs(-100);$

Nützlicher Link der tieferes Verständniss zur Arbeit mit diesen Befehlen vermittelt:
`http://www.w3schools.com/js/js_math.asp`

2.3 Javascript: DOM

Jedes bewegte Objekt hat seine Animation den Befehlen des Objektmodelling zu verdanken. Dieser dynamische Teil ermöglich überhaupt erst die grafische Darstellung in Bild und Text auf so hohem Niveau. In jeder Timerrunde werden die Koordinaten von 8 Charakteren(1 Charakter und 7 Monster) gesetzt und je nach Aktion bzw. Position wechselt der Pfad zum Bild, so dass eine komplette Visualisierung in Millisekundentakt errechnet und abgebildet werden kann.

Befehl	**Bedeutung**	**Beispiel**
getElementById()	Tag über ID	$document.getElementById("div1")$
getElementsByTagName()	Tagarray über Tagname	$getElementsByTagName("div");$
getElementsByTagName()[i].id	Tagarray mit Tagname: ID	$..ByTagName("div")[i].id$
getElementsByTagName().length	Tagarray: Länge	$d.length;$
style.left	X Position	$d.style.left = 100 + px";$
style.top	Y Position	$d.style.top = 100 + px";$
style.width	Breite	$d.style.width = 100 + px";$
style.height	Höhe	$d.style.height = 100 + px";$
style.opacity	Transparenz	$d.style.opacity = 1.0;$
style.visibility	Sichtbarkeit	$d.style.visibility = "visible";$
style.zIndex	z-index	$d.style.zIndex = 100;$
src	Pfad zum Bild	$d.src = "img1.jpg";$
innerHTML	Taginhalt	$d.innerHTML = " .. ";$
setAttribute	Tagattribut	$d.setAttribute("id","div1");$
onmousedown	Event bei Mausklick	$document.onmousedown = function(e)\{\}$
↪	...	$IE : e.clientX \| FF : e.pageX$
ondblclick	Event bei Doppelklick	$document.ondblclick = function(e)\{\}$
onkeydown	Event bei Tastendruck	$document.onkeydown = function(e)\{\}$
↪	...	$if(e.keyCode == 49)\{\}$

3 Verwendete Grafik

Die Grafik orientiert sich an dem Spiel Diablo, d.h. Elemente der Karte, Charaterklassen, Gegenstände, etc. wurden eben diesen Spiel entnommen. Eine einheitliche Grafik ist dennoch nicht gegeben, da Elemente aus unterschiedlichen Teilen zusammengestellt wurden. Jeder Gegenstand oder Charakterbild wurde einzeln aus sogenannten Sprite Sheets entnommen, so dass der damit verbunden Aufwand 1/5 der gesamten Entwicklungszeit ausmacht. In der folgenden Tabelle wurden einzelne GFX-Elemente aufgeliste und mit Stückzahl bzw. Größe versehen.

Name	**Anzahl**	**Größe**
Style/Map	65	12,6MB
Charaktere	473	1,4MB
Gegenstände	78	380KB
Zauber	185	400KB
Gesamt	801	14,78MB

4 Programmaufbau

Um den Programmaufbau in gänze zu erläutern, wurde die folgende Grafik erstellt.
Insgesamt sind es 59 Funktion und 3 Eventhändler zu unterschiedlichen Spielprinzipien.
Die angegebene Zahl zu den Programmabschnitten gibt die Grenze zur nächsten Funktion.

In einer weiteren Grafik wird verdeutlich, wie verschachtelt die einzelnen Funktion sind.

```
timerloop{
  setMapCharakter{
    Kollision{} // Kollision mit allen Elementen der Karte
    .
    .
    .
  }
  .
  .
  .
}
```

5 KI

5.1 allgemein

Künstliche Intelligenz ist ein weit gefächertes Forschungsgebiet der Informatik, welches sich vom Rest der damit verbunden Teilgebiete abhebt. Der funktionalen bzw. mathematischen Aspekt der Programmierung tritt dabei in den Hintergrund und neue Fragenstellung aus dem Bereich Psychologie und/oder Ethik kommen zum Vorschein.
Welche Bestandteile muss eine Programm aufweisen, um als KI-Programm eingestuft zu werden?

1 Möglichkeiten der Reflexion bzw. Selbstreflexion treten auf.

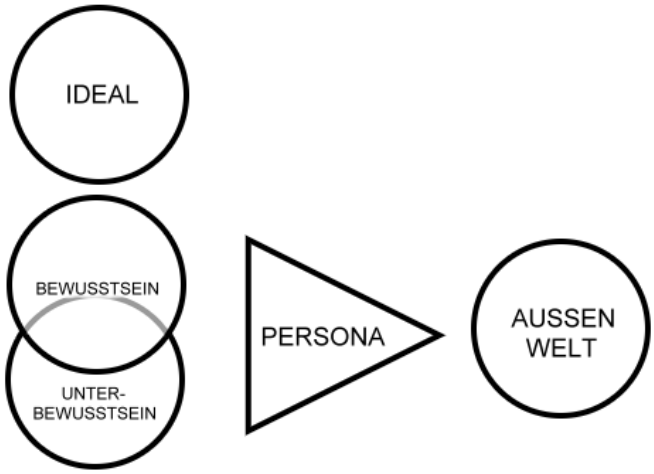
2 Lernprozesse finden statt.

3 Ein Erfahrungspool ist vorhanden.

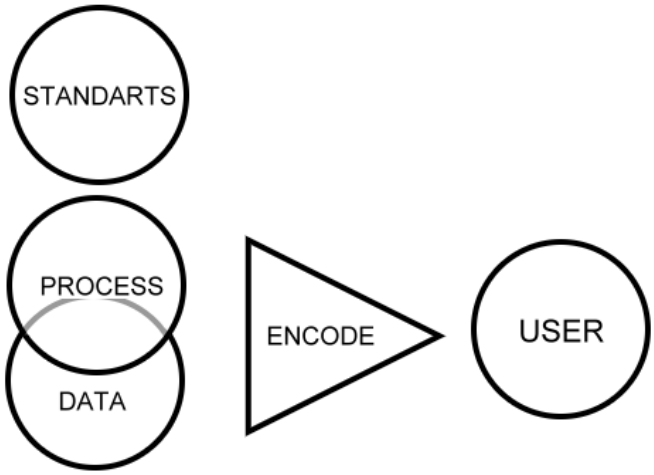
4 Strukturen von Bewusstsein werden abgebildet.

5 etc.

Nummer 4 wurden dabei als maßgeblich Strukture im Spiel verankert, d.h. Funktion und Prozesse wurden dabei nach dem Abbild des Bewusstsein formuliert. Dabei wurden verschieden Prinzipien umgesetzt, die sich leicht in Form einer Grafik erklären lassen.



Dementsprechend wurden diese Aspekte für die KI umgesetzt.



5.2 process

- Funktion wird über den Timer Runde für Runde aktiviert.
- Vergleichbar mit dem aktiven Bewusstsein, aber ohne Ruhezustand.
- In einem Startzustand werden die Basiswerte(Datentyp Cookie) erstellt.
- Es wird standartmässig die Funktion 'Data' aufgerufen.
- Spezielle Funktion läd und verrechnet Werte aus dem permanenten Speicher.

5.3 data

- 3 Speicherarten(Datentyp Cookie) vergleichbar dem Gedächtnis eines Menschen.

  - Shortterm: kurzfristiger Speicher
  - Mediumterm: mittelfristiger Speicher
  - Longterm: permanenter Speicher

- Mittels Modulo errechnet man, wann die anfallenden Userdaten in den mittelfristige bzw. permanenten Speicher übernommen werden.
- Der kurz- und mittelfristiger Speicher wird daraufhin gelöscht.

5.4 standarts

- Für jeden Wert der als Userdaten gespeichert wird, gibt es einen 'Standart'.
- Ein 'Standart' kategorisiert eine Wert in 5 Bereiche.
- Ermöglicht der KI den User einzustufen und daraufhin zu behandeln.

0 Standart0.0: Wechsel der Karte

1 Standart0.1: Siegreich aus dem Kampf

2 Standart0.2: Siegreich mit Zauber

3 Standart0.3: Kontakt mit Händler

4 Standart0.4: Truhe gefunden

5 Standart0.5: Gold aufgelesen

```
Beispiel am Standart0.1
var victories=50;
var standart=Standart0.1;//5:30:60:80:110
var standart_array=standart.split(":");
//victories>standart_array[1]
//victories<standart_array[2]
//victories=standart_array[1]
```

5.5 encode

- Die Speicher der KI über die Userdaten werden aufgelistet.
- Der Name 'encode' läßt sich von der Möglichkeit der KI ableiten, die Daten zu verschlüsseln.
- Das entspricht der Maske der Persona.
- Mit dem Verschlüsseln der Userdaten, versucht die KI zu täuschen.

5.6 Umsetzung

9/10 wurden umgesetzt
- alle KIattribute wie: process,data,etc.
- verschlüsselt encode und decodieren mit Hotkey 'D'
- belebt Gegner
- beeinflusst Kampfschaden

1/10 unvollständig
1: 1/10 unvollständiger Handlungsspielrahmen
↪ weitere Prinzipien, wie die KI auf den User wirkt

6 Problem mit großen Programmen

In sehr große Programme(mehr als 1000 Zeilen) gibt es manchmal Probleme, die sich für den Leihen nicht sofort erschliessen. Verfälschte Ergebnisse, die sich schlecht einordnen lassen, oder komplette Abstürze, die nur durch plötzlich abgebrochene Funktion an Systementscheidenden Stellen zu erklären sind und andere Probleme treten bei großen Programmen häufiger auf.
Die folgenden Probleme sind bei diesem Spiel öfters in Erscheinung getreten.

timer
Geschwindigkeit wird durch die Komplexität der Funktionen bestimmt.

parseInt
Bei arithmetischen Befehlen wird der Wert nicht mehr als Zahl erkannt.

```
var x=10;
var y=20;
var z=x+y;
\\ Lösung 30
\\ Ergebnis 1020
var z=parseInt(x)+parseInt(y);
\\ Ergebnis 30
```

Funktion
Es kann vorkommen, dass Funktion nicht richtig ausgeführt werden, weil der Benutzer zu schnell zwischen unterschiedlichen Programmelementen wechselt. An Systementscheidenden Stellen können Werte verloren gehen, oder das Programm versagt in Gänze. Ein Lösungsansatz: Kapselung der Funktionen in Klassen.

7 Erklärung über die eigenständige Erstellung der Belegarbeit

"Hiermit erkläre ich, dass ich die vorliegende Belegarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.
Die Stellen der Belegarbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht.
Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet." Quelle[1]

8 Quellen

Quelle[1]
Erklärung über die eigenständige Erstellung der Hausarbeit
https://www.wissenschaftliches-arbeiten.org/hausarbeit/aufbau/die-erklaerung.html

9 Anhang

Systemvoraussetzungen
Browser
Das Spiel wurde für 2 Browser ausgelegt.
Internetexplorer und Firefox

PC
Desktop, Laptop und Tabletts
Grösse lässt sich individuell Einstellen.
Systemvoraussetzung: unbekannt

Internet
Spiel kann mit unterschiedlicher Netzwerktechnologie genutzt werden, auch mit 56k Modem.
Etwas längere Startladezeit, da eine Prozedur die meisten Bilder vorläd.

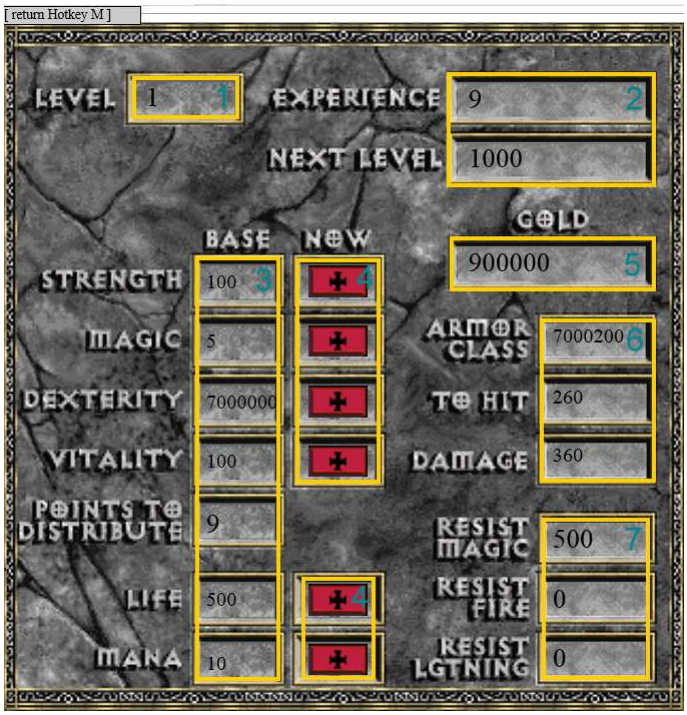
Nutzerdokumentation
Starbildschirm
Vom Startbildschirm aus kann der Benutzer zwischen zwei Charakterklassen wählen.
1. Barbar - Spezialität: Kampf
- Für den Anfänger ausgelegte, da hohen Angriffs- und Abwehrwert
2. Nercomancer- Spezialgebiet: Zauber
- Für den erfahren Heldentypen, da schwach im Angriff- bzw. Abwehr

Karte
Gleich zu Beginn kommt der Benutzer mit der Karte in Berührung.
In dieser ersten Karte werden 3 grundlegende Spielprinzipien vermittelt.
- Aufnehmen von Gold mittels darüberlaufen
- Öffnen von Truhen durch annähern
- betreten eines Ladens durch annähern
Ab der zweiten Karte die mittels betreten des Kartenrandes geladen wird, stehen den Benutzer Kampfoptionen zu Verfügung. Das heißt mit annähern an ein Monster starte automatische die Kampfsimulation, die mit dem Sieg über das Monster endet. Dem magischen Charakter stehen natürlich die Zauberkräfte, durch Doppelklick auf Monster bereit.
Es gibt mehrere Richtungen, die der Benutzer abschreiten kann, die sich durch unterschiedliche Spielprinzipien auszeichnen.
- Nord: nach längeren Laufen treten immer mehr Steine auf und die wahrscheinlich des Auftretens des Eingangs zu den Katakomben nimmt zu.
- Süd: je länger man nach Süden läuft umso grösser ist die Wahrscheinlichkeit Gold, Truhen bzw. Händler zu treffen.
- West/Ost: derzeit noch ohne Veränderung

Menu
Unter dem Kartenrand befindet sich eine Menuleiste, die verschiedene Optionen zu Verfügung stellt.
Optional = erste aktiv, wenn das Objekt genutzt wird, es zu einen Kampf kommt, etc.
D.h. im allgemeinen nicht sichtbar.
1. Erste Kästchen mit Werten:
1.1 Charakter HP/MP
1.2 optional: Kampf mit Monster = ID und Schaden
1.3 optional: benutzter Gegenstand
1.4 Himmelrichtungen
1.5 Objekte der Kollision = Monster,Zauber,Stein,Truhe,Händler und Gold
1.6 optional: Objekte der KI = Kartenwechsel,Kämpfe,Zauber,Truhen,Händler und Gold
2. HP/MP Kugeln: zeigt die Lebenkraft und Manavorrat bzw. die verwendeten Schriftrollen an
3. Gegenstände die mittels Hotkey 1-5 aktiviert werden
4. kleines Menu mit unterschiedlichen SSpielräumen”

Inventar
Im Inventar kann man sich mit unterschiedlichen Gegenständen ausrüsten. Dafür reich ein Mausklick auf das entsprechende Fach. Weiter unter findet man Einzelne Slots nebeneinander und untereinander. Sie dienen kleinen Gegenständen als Position und die ersten 5 können von der Karte aus aktiviert werden.
1. Helm
2. Amulett
3. Waffe
4. Rüstung
5. Schild/Zweitwaffe
6. Ringe
7. Schriftrollen
8. Gegenstände die von der Karte aus genutzt werden können
9. Slots für einzelne Gegenstände

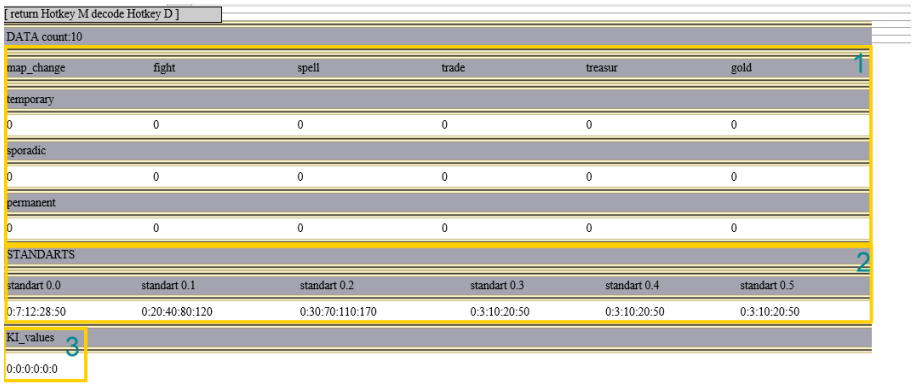
Charakterwerte
Für jede Charakterklasse gibt es typische Werte, die in eine speziellen Raum des Spiels abgebildet werden. Es gibt Standartwerte und Werte sich sich anhand dieser Standartwerte berechnen.
Für jeden Levelanstieg, der sich über die Erfahrungspunkte ermitteln lassen, werden Punkte zu verteilen vergeben.
1. Level
2. Erfahrungspunkte und nächstes Level
3. Standartwerte
4. Buttons und die Punkte auf die Werte zu verteilen
5. Werte sich sich anhand dieser Standartwerte berechnen
6. Resistenzen(magisch)



Schmiede
Jeden Gegenstand mit Slots kann über die Schmiede verbessert werde.
Sie werden einzeln mit den entsprechenden Fächern/Slots abgebildet.
Einfacher Mausklick auf das Fach listet alle Gegenstände auf, die dafür geeignet sind.
1. Gegenstand
2. Slots
3. geeigneter Gegenstand

KI
Die KI soll das Spiel interessanter und schwieriger machen, z.B.: indem sie erlegte Monster wiederbelebt.
Jeden einzelnen Wert kann man sich anzeigen und falls nötig mit dem Hotkey 'D' decodieren lassen.
Leider noch nicht ganz umgesetzt, ist sie dennoch mit Basisaktion versehen und kann es dem Spiel schwer machen.
1. 3 Speicherarten

2. Standarts
3. ermittelte Userdaten



Hotkeys
Hotkeys erleichtern die Bedienung ungemein und ermöglichen schnellen Wechsl zwischen den Spielattributen.

Name	**Hotkey**
Map	M
Inventar	I
Charakterwerte	V
Schmiede	B
KI	K
KI decodieren	D
Gegenstand auswählen	1-5
Zauber auswählen	7-9

Endgegner
Ein riesiges Monster wartet in den Tiefen der nördlich Katakomben.
Rüste dich also gut aus und stelle dich dieser Herausforderung.

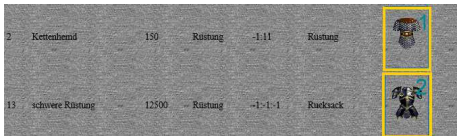
Bedienbeispiele Startbildschirm
Wähle eine Klasse mittels Klick auf das Bild.
Links: Barbar
Recht: Necromancer

Karte wird betreten.
1. Charakter
2. Truhe
3. Gold
4. Händler und NPC

Charakter nähert sich der Truhe.
1. Charakter + Truhe
2. Meldung welcher Gegenstand gefunden wurde
↪ in dem Fall Rüstung

Inventar wird betreten und auf das Rüstungsfach geklickt.

Vorhandene Rüstungen werden aufgelistet bzw. können mit Klick auf das Bild ausgewählt werden.



Im Hauptraum des Inventars ist nun die neue Rüstung im Rüstungsfach abgebildet.

Mittels Hotkey 'M' kann nun die Karte betreten werde.
Doch vorsicht – ihr seid nun nicht mehr auf der Ausgangskarte, sondern mitten im Kampfgeschehen.
Ihr solltet nur voll ausgerüstet die Karte betreten, ansonsten kommt es ganz schnell zu einem Game Over.
Mitten im Kampfgeschehen, reicht ein einfacher Klick auf einen Gegner, der Rest wird vollständig animiert.
1. Charakter
2. besiegte Gegner
3. sich nähernde Gegner
4. frei stehende Gegner

Ist der Kampf beendet und es sollten noch einige Gegner stehen, müsst ihr aktiv eingreifen und euch den Monster mittels klick nähern.
1. Charakter
2. besiegte Gegner
3. frei stehende Gegner

Jetzt liegt es an euch, ob ihr Richtung Norden geht, um euch den Endgegner vorzuknöpfen, oder Richtung Süden, damit ihr euch mit Gegenständen ausrüsten könnt. Für erfahren Rollenspieler stellt sich die Frage natürlich gar nicht erst, da ein Kampf gegen übermächtige Monster nur gut ausgerüstet zu gewinnen ist.

Hier noch einige wichtige Bedienbeispiele:

1. Shop
Näher dich einen Händler. Sei schnell, denn der Gegner kann den Händler ausschalten und du kann den Laden nicht mehr betreten.

Im Laden stehen dir die Optionen kaufen und verkaufen zu Verfügung.
1.1 Laden
1.2 eigenes Inventar
1.3 Option kaufen
1.4 Option verkaufen

Willkommen Reisender Gold: 900000 1

D	Name	Stärke	Preis	Bild	Klasse	Option 3
1	Schwert	70	500		Waffe	kaufen
2	leichtes Kettenhemd	50	700		Rüstung	kaufen

Eigenes Inventar 2

ID	Name	Stärke	Verkaufspreise	Bild	Klasse	Option 4
0	Holzfälleraxt	100	1090		Waffe	verkaufen
1	Helm	50	90		Helm	verkaufen

2. Charakterwerte
Wenn der Charakter mehr als 20 Punkte zum verteilen hat, reicht ein 'onMouseOver', um den Wert zu erhöhen, ansonsten ein 'onClick'.
Die Werte sollten sinnvoll zugeordnet werde. Bsp: Barbar und Magiewert.

3. Schmiede
Wie bereits erwähnt kann jeder Gegenstand mit Slots, verbessert werden.
Betrete die Schmiede und wählt das Objekt aus.

Daraufhin öffnet sich eine Auswahl aller dafür in Frage kommender Gegenstände.

Nun erscheint der Gegenstand im Fach.

Wichtig: Nachdem das Objekt geschmiedet wurde, muss es im Inventar **neu** angelegt werden.
Die Alternative heißt: Karte betreten und sterben.