`dungeon`

Ein Miniatur-Dungeon

VWA-Frankfurt

Sommersemester 2004

Bernd Ulmann

3. Mai 2004

Kapitel 1

Einführung

Ziel des im Rahmen der Veranstaltung Software-Entwicklung-3 durchzuführenden Kleinprojektes ist es, ein einfaches und gut erweiterbares, in Perl geschriebenes Dungeon zu entwickeln. Hierunter ist eine Art webbasiertes Adventure zu verstehen, in welchem eine – in einer Datenbank hinterlegte – Welt über einen Webbrowser als Frontend zu erkunden ist.

Als Grundstock hierfür dient das Programm `d1.pl` zusammen mit der Datenbank `dungeon`, für welche eine Miniwelt (konkret ein Rundgang durch meine Wohnung :-)) bereits implementiert wurde. `d1.pl` erhebt keinerlei Anspruch auf übermäßige Eleganz und soll lediglich zur Verdeutlichung der grundlegenden Mechanismen und als Quelle für hoffentlich brauchbare Programmfragmente und Routinen dienen.

Die folgenden Abschnitte beschreiben zunächst den Aufbau der zugrundeliegenden Datenbank, um sich im Anschluß hieran dem Perl-Skript `d1.pl` sowie der Installation des Beispieldungeons zu widmen.

Kapitel 2

Die Datenbank dungeon

Die zentrale Datenbank, in welcher die Verbindungsstruktur der Orte, aus welchen sich die `dungeon`-Welt zusammensetzt, abgelegt ist, wurde in `MySQL` implementiert, so daß sie leicht auf andere Systeme als den momentanten Solaris-Rechner portierbar ist.

2.1 Die Tabellen

Nach dem Anlegen der Datenbank an sich (siehe Abschnitt 3.2), sind die benötigten (drei) Tabellen zu erzeugen und mit entsprechenden Daten zu füllen. Bei den Tabellen handelt es sich im einzelnen um:

`locations`: Diese Tabelle enthält für jeden Ort des Dungeons einen Eintrag und wurde folgendermaßen spezifiziert:

```
CREATE TABLE locations (
  lnr int NOT NULL auto_increment,
  name varchar(80) NOT NULL,
  description text NOT NULL,
  picture varchar(80) default NULL,
  PRIMARY KEY (lnr)
);
```

Einem jeden Ort ist genau eine `lnr` als Primärschlüssel zugeordnet, der mit dem Parameter `auto_increment` spezifiziert wurde, was das Einfügen neuer Orte deutlich erleichtert. Alle weiteren notwendigen Informationen werden in den Feldern `name` (enthält den Namen, d.h. die Bezeichnung des jeweiligen Ortes), `description` (enthält eine ausführliche Beschreibung des Ortes) sowie `picture` (enthält, falls vorhanden, den Namen des dem jeweiligen Ort zugeordneten Bildes) abgelegt.

`directions`: Von einem Ort zu einem anderen gelangt man, indem man sich in eine bestimmte Richtung bewegt. Alle möglichen Richtungen, in welche Fortbewegung zulässig ist, sind in der Tabelle `directions` abgelegt, die mit Hilfe des folgenden Statements erzeugt wird:

```
CREATE TABLE directions (
  dnr int NOT NULL auto_increment,
  name varchar(80) NOT NULL,
```

```
  PRIMARY KEY (dnr)
);
```

`connections`: Die eigentlichen Verbindungen zwischen verschiedenen Orten innerhalb des Dungeons sind in einer eigenen Tabelle `connections` abgelegt, die folgendermaßen erzeugt wurde:

```
CREATE TABLE connections (
  from_lnr int NOT NULL,
  from_dnr int NOT NULL,
  to_lnr int NOT NULL,
  to_dnr int NOT NULL,
  PRIMARY KEY (from_lnr, from_dnr, to_lnr, to_dnr)
);
```

Der aus allen vier Spalten zusammengesetzte Primärschlüssel stellt sicher, daß nicht aus Versehen mehrfache, identische Einträge in die Tabelle aufgenommen werden können, welche für Verwirrung bei der Darstellung möglicher Richtungen, in welche ein Ort verlassen werden kann, sorgen würden.

Für jede einzelne Verbindung zwischen zwei Räumen und jede Richtung, in welcher diese Verbindung passiert werden kann, existiert in `connections` Tabelle ein Eintrag[1].

Die Spalten `from_lnr` und `to_lnr` beschreiben nun, von welchem Ort man durch Passieren der jeweiligen Verbindung, welcher der Eintrag zugeordnet ist, zu welchem anderen Ort gelangt. Entsprechend enthalten die beiden verbleibenden Spalten `from_dnr` beziehungsweise `to_dnr` die Nummern der zugehörigen Richtungen, in welche man einen Ort verläßt, beziehungsweise aus welcher kommend man an einem Ort anlangt.

2.2 Beispieldaten

Als Beispiel für die Struktur der Tabelleneinträge möge im folgenden ein Miniaturdungeon mit drei Orten dienen, wie es in Abbildung 2.1 dargestellt ist. Verläßt man Ort 1 in Richtung Osten, so gelangt man nach Ort 2 (im Westen). Entsprechendes gilt für die anderen Richtungen – verläßt man Ort 1 nach Süden, langt man in Ort 3 in der Richtung Nort-Osten an. Verläßt man 3 nach Nord-Osten, gelangt man wieder nach 1 (Süden), etc.

Zur Abbildung dieses Dungeons müßten die folgenden Tabelleneinträge erzeugt werden:

locations			
lnr	name	description	picture
1	Ort 1	Der erste Ort	ort_1.jpg
2	Ort 2	Der zweite Ort	ort_2.jpg
3	Ort 3	Der dritte und letzte Ort	ort_3.jpg

[1] D.h., daß eine Verbindung, die keine Einbahnstraße darstellt, wie dies beispielsweise Falltüren, etc. tun, durch zwei Einträge repräsentiert wird.

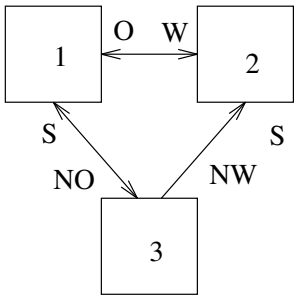

Abbildung 2.1: Beispieldungeon mit drei Orten

directions	
dnr	name
1	Osten
2	Westen
3	Sueden
4	Nord-Osten
5	Nord-Westen

connections			
from_lnr	from_dnr	to_lnr	to_dnr
1	1	2	2
2	2	1	1
1	3	3	4
3	4	1	3
3	5	2	3

Kapitel 3

Installation des Beispieldungeons

Das ZIP-File `dungeon.zip` enthält alle nötigen Files, um ein kleines Beispieldungeon zu installieren. Vorausgesetzt wird hierbei das Vorhandensein eines Webservers, der userspezifische CGI-Skripten zuläßt (idealerweise ein Apache-Server mit mod-Perl). Steht kein mod-Perl zur Verfügung, wird ein Standalone-Perl-Interpreter benötigt. Weiterhin werden ein `MySQL`-Datenbanksystem sowie das Perl-DBI-Modul für den Zugriff auf `MySQL`-Datenbanken benötigt.

3.1 Auspacken des ZIP-Files

Unter der Annahme, das der Webserver des Zielsystems dergestalt konfiguriert ist, daß er Userverzeichnisse unter `public_html` und CGI-Skripten hierunter unter `cgi-bin` erwartet, ist das File `dungeon.zip` am besten in `public_html/cgi-bin` abzulegen und mit

```
unzip dungeon.zip
```

zu entpacken. Dies erzeugt ein Unterverzeichnis `dungeon` unter `public_html/cgi-bin` mit folgendem Inhalt:

```
-rw-r--r--    1   52045 May  3 07:39 arbeitszimmer_a.jpg
-rw-r--r--    1   56430 May  3 07:39 arbeitszimmer_b.jpg
-rw-r--r--    1   32820 May  3 07:39 bad.jpg
-rwxr-xr-x    1    5320 May  3 07:39 d1.pl
-rw-r--r--    1    5873 May  3 07:41 dungeon.sql
-rw-r--r--    1  117864 May  3 07:39 garten.jpg
-rw-r--r--    1  110856 May  3 07:39 huette.jpg
-rw-r--r--    1   37714 May  3 07:39 kueche.jpg
-rw-r--r--    1   31588 May  3 07:39 rechenkeller.jpg
-rw-r--r--    1   35532 May  3 07:39 schlafzimmer.jpg
-rw-r--r--    1   56702 May  3 07:39 strasse.jpg
-rw-r--r--    1   44813 May  3 07:39 treppenhaus.jpg
-rw-r--r--    1   44439 May  3 07:39 werkstatt.jpg
-rw-r--r--    1   49073 May  3 07:39 wohnzimmer.jpg
```

Hier finden sich das eigentliche Perl-Skript, welches das Dungeon darstellt und die Navigation in ihm erlaubt, `d1.pl`, die den einzelnen Orten durch entsprechende

Tabelleneinträge in der Datenbank zugeordneten Bilder in Form von `jpg`-Files sowie ein Dump der gesamten Datenbank, wie sie für das Beispieldungeon benötigt wird.

An dieser Stelle ist lediglich darauf zu achten, daß für WORLD Lesezugriff auf alle Bilder sowie Lese- und Ausführzugriff für das Perl-Skript `d1.pl` gewährleistet ist. Weiterhin müssen Lese- und Ausführrechte für WORLD auf das Verzeichnis `dungeon` unter `public_html/cgi-bin` gesetzt sein, um das dort befindliche Perl-Skript auch starten zu können.

3.2 Anlegen und Füllen der Datenbank

Vor dem Erzeugen der Tabellen und dem Füllen selbiger mit allen notwendigen Daten, ist zunächst eine leere Datenbank zu erzeugen sowie Rechte auf dieser Datenbank für einen bestimmten Benutzer zu vergeben:

```
ulmann@saruman: mysql -u root -p mysql
Enter password:

mysql> create database dungeon;
Query OK, 1 rows affected (0.43 sec)

mysql> grant all on dungeon.*
    -> to dungeon@localhost
    -> identified by 'keeper';
Query OK, 0 rows affected (0.31 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.10 sec)

mysql> quit
ulmann@saruman:
```

Hierdurch wird dem Benutzer `dungeon`[1] Vollzugriff auf die eben erzeugte Datenbank `dungeon` eingeräumt. Bei der Verbindung mit dem Datenbanksystem muß sich dieser User mit dem Passwort `keeper` identifizieren.

Nun können die Daten des Dumpfiles `dungeon.sql` in die eben angelegte, leere Datenbank eingespielt werden:

```
ulmann@saruman: mysql -u dungeon -p dungeon < dungeon.sql
Enter password:
ulmann@saruman:
```

Durch die Eingabeumlenkung `<` werden die in `dungeon.sql` abgelegten Kommandos, mit deren Hilfe die Tabellenstrukturen erzeugt und die Daten innerhalb der Tabellen abgelegt werden, in der noch leeren Datenbank `dungeon` ausgeführt.

[1] Hierbei handelt es sich lediglich um einen Datenbankuser, d.h. dieser User muß nicht in `/etc/passwd` vorhanden sein!

Kapitel 4

Das Programm `d1.pl`

Der vollständige Quellcode von `d1.pl` ist in Anhang B angegeben – hier soll lediglich kurz der prinzipielle Aufbau des Skriptes angerissen werden – ein Blick in die Sourcen ist, wie immer, lohnenswert[1].

`d1.pl` besteht aus vier Subroutinen sowie einem Hauptprogramm, die im folgenden kurz behandelt werden:

4.1 Das Hauptprogramm

Das Hauptprogramm definiert und belegt zunächst eine Reihe von Variablen, die in der Hauptsache für die Verbindung zur Datenbank benötigt werden. Im Anschluß hieran wird die übliche Präambel für den Webbrowser erzeugt, um diesem das Format der sich anschließenden Übertragung mitzuteilen.

Der nächste Schritt besteht darin, eventuelle Parameter, die an das Programm über den GET- oder POST-Mechanismus übergeben wurden, auszulesen und in einen Hash `%FORM` zu schreiben. Wurde mindestens ein Parameter übergeben, handelt es sich nicht um den ersten Aufruf des Programmes, so daß die defaultmäßig auf 19[2] gesetzte Locationnumber `$lnr` neu bestimmt werden muß[3].

Nachdem nun auf die eine (Initialwert bei Erstaufruf) oder andere Weise (Bestimmung der nächten Locationnumber durch Aufruf von `get_lnr`) die `$lnr` des aktuellen Ortes bekannt ist, werden durch Aufruf der Subroutine `display_location` alle Informationen zum Ort an sich, d.h. Name, Beschreibung sowie ein eventuell vorhandenes Bild, angezeigt.

Im Anschluß hieran wird mit Hilfe der Subroutine `display_directions` eine Liste aus Submit-Buttons aufgebaut, wobei jeder Richtung, in welche vom aktuellen Ort, gekennzeichnet durch seine `$lnr`, gegangen werden kann, ein Button zugeordnet ist.

4.2 Die Subroutine `conn`

Diese Routine baut eine Verbindung mit der Datenbank auf, wobei die zuvor im Hauptprogramm deklarierten und initialisierten Variablen als Verbindungsparameter herangezogen werden. Gelingt der Verbindungsaufbau aus irgendwelchen Gründen

[1] Im schlimmsten Fall sind Fehler zu finden, aus denen sich lernen läßt! :-)

[2] Dies ist das Resultat einer etwas späten Änderung einiger Tabelleneinträge. :-) Es wäre zugegebenermaßen intuitiver, wenn der erste Raum die Nummer 1 wäre.

[3] Zuvor jedoch wurde eine Verbindung mit der Datenbank durch Aufruf der Subroutine `conn` hergestellt.

nicht, wird das Programm, wie auch an allen anderen Stellen, an welchen Datenbankzugriffe nicht erfolgreich verlaufen, mit Hilfe einer `die`-Anweisung unter Ausgabe einer entsprechenden Fehlermeldung abgebrochen.

Rückgabewert der Funktion `conn` ist das aus dem Verbindungsaufbau resultierende Datenbankhandle `$dbh`, mit welchem im folgenden alle Datenbankzugriffe abgearbeitet werden.

4.3 Die Subroutine `get_lnr`

Die Bestimmung einer neuen Locationnumber `$lnr` geschieht durch Aufruf der Routine `get_lnr` – diese erwartet die Nummer des aktuellen Ortes sowie die Bezeichnung der Richtung, in welcher dieser Ort verlassen wird.

Die neue Locationnumber wird durch das SQL-Statement

```
select c.to_lnr
from connections c, directions d
where c.from_dnr = d.dnr
and d.name = \"$direction\"
and c.from_lnr = $lnr
```

bestimmt. Liefert diese Abfrage kein Ergebnis zurück, so deutet dies auf eine inkonsistente Datenbank hin und führt zu einem Programmabbruch.

4.4 Die Subroutine `display_location`

Mit Hilfe des SQL-Statements

```
select name, description, picture
from locations
where lnr = $lnr
```

werden zu einer gegebenen `$lnr` alle Ortsinformationen aus der Datenbank ausgelesen und im weiteren Verlauf der Routine zur Anzeige gebracht.

4.5 Die Subroutine `display_direction`

```
select d.name
from connections c, directions d
where c.from_lnr = $lne
and c.from_dnr = d.dnr
```

generiert eine Liste aller Richtungen, in welche der aktuelle Ort verlassen werden kann. Für jede Richtung wird innerhalb einer Form, welche nach einem Submit wieder das vorliegende Perl-Skript aufruft, und diesem die Richtung, deren Wahl zu dem Submit führte, sowie die bisherige `$lnr` mitteilt, ein Submit-Button erzeugt.

Die Übergabe der aktuellen Ortsnummer `$lnr` erfolgt mit Hilfe eines verdeckten Eingabefeldes[4].

[4] `hidden`

4.6 Debugging

Um das Debugging des Programmes zu vereinfachen, wurde eine globale Variable[5] `$debug` eingeführt. Wird diese mit einem Wert ungleich `0` beziehungsweise `undef` initialisiert, werden an verschiedenen Stellen des Programmes Debuginformationen ausgegeben – so wird beispielsweise der Eintritt in Subroutinen dokumentiert, etc.

[5] Einer der wenigen quasi abgesegneten Verwendungszwecke globaler Variablen!

Anhang A

dungeon.sql

```
CREATE TABLE connections (
  from_lnr int NOT NULL,
  from_dnr int NOT NULL,
  to_lnr int NOT NULL,
  to_dnr int NOT NULL,
  PRIMARY KEY (from_lnr, from_dnr, to_lnr, to_dnr)
);

CREATE TABLE directions (
  dnr int NOT NULL auto_increment,
  name varchar(80) NOT NULL,
  PRIMARY KEY  (dnr)
);

CREATE TABLE locations (
  lnr int NOT NULL auto_increment,
  name varchar(80) NOT NULL,
  description text NOT NULL,
  picture varchar(80) default NULL,
  PRIMARY KEY  (lnr)
);

--
-- Dumping data for table `connections`
--

INSERT INTO connections VALUES (1,8,2,2);
INSERT INTO connections VALUES (2,2,1,8);
INSERT INTO connections VALUES (1,7,3,2);
INSERT INTO connections VALUES (3,2,1,7);
INSERT INTO connections VALUES (1,3,4,4);
INSERT INTO connections VALUES (4,4,1,3);
INSERT INTO connections VALUES (1,2,6,1);
INSERT INTO connections VALUES (6,1,1,2);
INSERT INTO connections VALUES (1,10,7,3);
INSERT INTO connections VALUES (7,3,1,10);
INSERT INTO connections VALUES (1,5,16,6);
INSERT INTO connections VALUES (16,6,1,5);
INSERT INTO connections VALUES (3,1,8,2);
```

```
INSERT INTO connections VALUES (4,3,5,4);
INSERT INTO connections VALUES (5,4,4,3);
INSERT INTO connections VALUES (19,2,8,1);
INSERT INTO connections VALUES (8,1,19,2);
INSERT INTO connections VALUES (8,3,1,4);
INSERT INTO connections VALUES (1,4,8,3);
INSERT INTO connections VALUES (8,2,9,1);
INSERT INTO connections VALUES (9,1,8,2);
INSERT INTO connections VALUES (16,10,15,3);
INSERT INTO connections VALUES (15,3,16,10);
INSERT INTO connections VALUES (16,2,14,1);
INSERT INTO connections VALUES (14,1,16,2);
INSERT INTO connections VALUES (16,9,13,4);
INSERT INTO connections VALUES (13,4,16,9);
INSERT INTO connections VALUES (13,2,8,1);
INSERT INTO connections VALUES (16,3,12,4);
INSERT INTO connections VALUES (12,4,16,3);
INSERT INTO connections VALUES (12,1,18,2);
INSERT INTO connections VALUES (18,2,12,1);
INSERT INTO connections VALUES (16,1,11,2);
INSERT INTO connections VALUES (11,2,16,1);
INSERT INTO connections VALUES (16,8,10,2);
INSERT INTO connections VALUES (10,2,16,8);
INSERT INTO connections VALUES (16,4,17,3);
INSERT INTO connections VALUES (17,3,16,4);

--
-- Dumping data for table `directions`
--

INSERT INTO directions VALUES (1,'Norden');
INSERT INTO directions VALUES (2,'Sueden');
INSERT INTO directions VALUES (3,'Osten');
INSERT INTO directions VALUES (4,'Westen');
INSERT INTO directions VALUES (5,'auf');
INSERT INTO directions VALUES (6,'ab');
INSERT INTO directions VALUES (7,'Nord-Ost');
INSERT INTO directions VALUES (8,'Nord-West');
INSERT INTO directions VALUES (9,'Sued-Ost');
INSERT INTO directions VALUES (10,'Sued-West');

--
-- Dumping data for table `locations`
--

INSERT INTO locations VALUES (1,'Treppenhaus_unten',
 'Vom Treppenhaus aus kann man alle Raeume eines Geschosses
  erreichen. Ausserdem ist es die Verbindung zu den anderen
  Geschossen und zur Aussenwelt. Das Treppenhaus unten hat
  zwei Tueren nach Westen, zwei Tueren nach Norden, eine Tuere
  nach Osten, eine Tuere nach Sueden und eine Treppe nach oben.',
  'treppenhaus.jpg');
INSERT INTO locations VALUES (2,'Kueche','Die Kueche hat eine Tuere
  nach Sueden, ein Fenster nach Norden und eine Klappe nach Osten.',
```

```
  'kueche.jpg');
INSERT INTO locations VALUES (3,'Wohnzimmer','Das Wohnzimmer hat eine
  Tuere nach Sueden, eine Tuere nach Norden, zwei Fenster nach Norden
  und ein Fenster nach Osten.','wohnzimmer.jpg');
INSERT INTO locations VALUES (4,'Keller_a','Der Keller_a hat eine
  Tuere nach Westen und eine Tuere nach Osten. Er hat auch ein
  Fenster nach Sueden.','keller_a.jpg');
INSERT INTO locations VALUES (5,'Keller_b','Der Keller_b hat eine
  Tuere nach Westen und ein Fenster nach Osten.','keller_b.jpg');
INSERT INTO locations VALUES (6,'Gaeste-WC','Das Gaeste-WC hat eine
  Tuere nach Norden und ein Fenster nach Sueden.','gaeste_wc.jpg');
INSERT INTO locations VALUES (7,'Rechenkeller','Der Rechenkeller hat
  eine Tuere nach Osten und ein Fenster nach Westen, es ist wohl
  klar, dass weder Tuere noch Fenster ganz zu oeffnen sind.',
  'rechenkeller.jpg');
INSERT INTO locations VALUES (8,'Garten','Im Garten steht ein grosses
  Haus und eine kleine Huette, er hat eine Eingangstuere nach Norden
  und verschiedene Treppen und Wege.','garten.jpg');
INSERT INTO locations VALUES (9,'Huette','Die Huette steht im Garten
  und hat eine Tuere und ein Fenster nach Norden.','huette.jpg');
INSERT INTO locations VALUES (10,'Bad','Das Bad hat eine Tuere nach
  Sueden und ein Fenster nach Norden.','bad.jpg');
INSERT INTO locations VALUES (11,'Arbeitszimmer_a','Das Arbeitszimmer
  A hat eine Tuere nach Sueden und ein Fenster nach Norden.',
  'arbeitszimmer_a.jpg');
INSERT INTO locations VALUES (12,'Arbeitszimmer_b','Das Arbeitszimmer
  B hat eine Tuere nach Westen, eine Tuere nach Norden, ein Fenster
  nach Norden und eins nach Osten.','arbeitszimmer_b.jpg');
INSERT INTO locations VALUES (13,'Schlafzimmerr','Das Schlafzimmer
  hat eine Tuere nach Westen und eine Tuere nach Sueden. Ausserdem
  hat es ein Fenster nach Osten.','schlafzimmer.jpg');
INSERT INTO locations VALUES (14,'Klo','Das Klo hat eine Tuere nach
  Norden und ein Fenster nach Sueden','klo.jpg');
INSERT INTO locations VALUES (15,'Werkstatt','Die Werkstatt hat eine
  Tuere nach Osten und ein Fenster nach Sueden','werkstatt.jpg');
INSERT INTO locations VALUES (16,'Treppenhaus_oben','Das Treppenhaus
  hat oben zwei Tueren nach Norden, zwei Tueren nach Westen, eine
  Tuere nach Sueden und zwei Tueren nach Osten. Es gibt eine Treppe
  nach unten.','treppenhaus_oben.jpg');
INSERT INTO locations VALUES (17,'Kleiderkammer','Die Kleiderkammer
  hat nur eine Tuere nach Osten.','kleiderkammer.jpg');
INSERT INTO locations VALUES (18,'Balkon','Der Balkon hat eine Tuere
  nach Norden, von ihm aus kann man in den Garten sehen.',
  'balkon.jpg');
INSERT INTO locations VALUES (19,'Strasse','An der Strasse liegt der
  Eingang im Sueden zum Garten.','strasse.jpg');
```

Anhang B

d1.pl

```
#! /usr/bin/perl

#
# Test dungeon for SE-3, VWA, SoSe 2004
# B. Ulmann fecit, 02-MAY-2004
#

# Be sure to warn on undeclared variables, etc.
use strict;
#use warnings;
use DBI;

# $debug = 1 results in some debug output - this is a global variable! :-)
my $debug = 0;

# Connect to database
sub conn
{
    my ($driver, $database, $hostname, $port, $user, $password) = @_;

    if ($debug)
    {
        print "DEBUG: conn ()<br>\n";
    }

    my $dsn = "DBI:$driver:database=$database;host=$hostname;port=$port";
    my $dbh = DBI -> connect ($dsn, $user, $password); # Connect to database

    die "<b>conn (): Could not connect to database!" unless $dbh; # Did not work
    die "<b>conn (): Connection error: ", $dbh -> errstr if $dbh -> err;

    print "<hr>\n" if $debug;

    return $dbh;
}

# Print all information about a location specified by a location number $lnr
sub display_location
{
    my ($dbh, $lnr, $picture_prefix) = @_;

    if ($debug)
```

```
    {
        print "DEBUG: display_location ()<br><ul><li>lnr = $lnr<li>picture_prefix: $picture_prefix</ul>
    }

    my $statement = "select name, description, picture from locations where lnr = $lnr";
    my $sth = $dbh -> prepare ($statement);
    $sth -> execute;

    die "<b>display_location (): Error: ", $sth -> errstr if ($sth -> err);

    my ($name, $description, $picture);
    $sth -> bind_columns (\$name, \$description, \$picture);
    die "<b>display_location (): Nothing to fetch!" if !($sth -> fetch);
    $sth -> finish;

    $picture = $picture_prefix . $picture if $picture;

    print "<center><h1>$name</h1></center><hr>\n";
    print "<center><img src=\"$picture\"></center><hr>\n" if $picture;
    print "$description<hr>\n";
}

# Create a button for each possible direction to go to
sub display_directions
{
    my ($dbh, $lnr, $cgi) = @_;

    if ($debug)
    {
        print "DEBUG: display_directions ()<br><ul><li>lnr = $lnr</ul>\n";
    }

    my $statement = "select d.name
from connections c, directions d
where c.from_lnr = $lnr
and c.from_dnr = d.dnr";
    my $sth = $dbh -> prepare ($statement);
    $sth -> execute;

    die "<b>display_directions (): Error: ", $sth -> errstr if ($sth -> err);

    my $direction;
    $sth -> bind_columns (\$direction);

    print "<form action=\"$cgi\" method=\"POST\"><center><table><tr>\n";
    while ($sth -> fetch ())
    {
        print "<td><input name=\"direction\" type=\"submit\" value=\"$direction\"></td>\n";
    }
    $sth -> finish;

    print "<td><input name=\"lnr\" type=\"hidden\" value=\"$lnr\"></td>\n";
    print "</tr></table></center></form>\n";

    print "<hr>\n" if $debug;
}

#  Get next location number. This is determined from the current location number
```

```
# and the name of the direction to go to.
sub get_lnr
{
    my ($dbh, $lnr, $direction) = @_;

    if ($debug)
    {
        print "DEBUG: get_lnr ()<br><ul><li>lnr = $lnr<li>direction = $direction</ul>\n";
    }

    my $statement = "select c.to_lnr
from connections c, directions d
where c.from_dnr = d.dnr
and d.name = \"$direction\"
and c.from_lnr = $lnr";
    my $sth = $dbh -> prepare ($statement);
    $sth -> execute;

    die "<b>get_lnr (): Error: ", $sth -> errstr if ($sth -> err);

    my $next_location;
    $sth -> bind_columns (\$next_location);
    die "<b>display_location (): Nothing to fetch!" if !($sth -> fetch);
    $sth -> finish;

    return $next_location;
}

##############################################################################################
#
# main program
#
##############################################################################################

# Set access parameters:
my $driver = "mysql";
my $database = "dungeon";
my $hostname = "localhost";
my $port = 3306;
my $user = "dungeon";
my $password = "keeper";
my $picture_prefix = '';
my $cgi = "d1.pl"; # Name of the cgi to be called

# Some more variables
my $lnr = 19;      # Initial location number

# first_run = 1 denotes that the script was called for the first time
my $first_run = 1;

# Prepare the output of HTML code:
print "Content-type: text/html

<html>
 <body>
";

if ($debug) # We are running in debug mode!
```

```
{
    print "<center><b>Running in debug mode!</b></center><hr>\n";
}

# Read parameters from GET or POST access:
my ($buffer, @pairs, $pair, $name, $value, %FORM);

# Read in text
$ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
if ($ENV{'REQUEST_METHOD'} eq "POST")
{
    read (STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
}
else # Not POST -> GET :-)
{
    $buffer = $ENV{'QUERY_STRING'};
}

# Split information into name/value pairs
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)
{
    ($name, $value) = split(/=/, $pair);
    $value =~ tr/+/ /;
    $value =~ s/%(..)/pack("C", hex($1))/eg;

    $FORM{$name} = $value;
    $first_run = 0;
}

# Dump parameters:
if ($debug)
{
    print "DEBUG: Parameters:<br><ul><li>first_run = $first_run\n";
    foreach my $key (keys %FORM)
    {
        print "<li>$key = $FORM{$key}\n";
    }
    print "</ul>\n";
}

# Connect to database
my $dbh = conn ($driver, $database, $hostname, $port, $user, $password);

if (!$first_run) # Not the first run - so determine the actual room number
{
    $lnr = get_lnr ($dbh, $FORM{'lnr'}, $FORM{'direction'});
}

display_location ($dbh, $lnr, $picture_prefix);
display_directions ($dbh, $lnr, $cgi);

print " </body>
</html>
";

$dbh -> disconnect;
```