Pac-Man als Browsergame

David Rosenbusch (m21898)

Friedrichstr. 54
david.rosenbusch@googlemail.com

3. Fachsemester – Medieninformatik

Hochschule Harz

Fachbereich: Automatisierung und Informatik
Vorlesung: Webprogrammierung
Dozent: Prof. Dr. Henrik J. Paul

Abgabe: Wernigerode, 29.02.2016

Inhalt

1 Einleitung

1.1 Das Semesterprojekt

Über das dritte Fachsemester in der Vorlesung Webprogrammierung bekamen die Studierenden die Aufgabe, allein oder in Zweiergruppen den Spieleklassiker Pac-Man als eine Webanwendung zu programmieren.

Aufbauend auf dem Lehrstoff aus den Vorlesungen und Tutorien sollten die gelehrten Technologien eingesetzt werden, um die Spiellogik, Nutzerinteraktion und Datenverwaltung zu implementieren.

Als Ergebnis wird das fertige Spiel als lauffähige Webanwendung mit kommentiertem Quellcode und der Dokumentation abgegeben.

1.2 Die Zielsetzung: „Open Pac-Man“

Da die Aufgabenstellung einen gewissen Interpretationsspielraum in der Umsetzung und dem Spielstil zulässt, habe ich die Zielsetzung für dieses Projekt ein wenig spezialisiert und es „Open Pac-Man“ getauft.

Open Pac-Man soll sich durch Transparenz und einfache Modifizierbarkeit auszeichnen.

Der Nutzer soll nicht nur Spielstände laden und speichern, sondern auch eigene Level bauen und hochladen, sodass andere Nutzer sich daran erproben können. Der Nutzer soll die Funktionsweise der Datenstruktur sehen und die Daten in ihrer Reinform „anfassen“ können, um das Spielerlebnis nach eigenem Ermessen anzupassen und selbst kreativ zu werden.

Wie genau das funktioniert wird in Abschnitt 2.2 im Detail erläutert.

1.3 Aufbau der Arbeit

In den drei folgenden Kapiteln werden Konzept und Planung des Projektes, die dem Programm zugrundeliegende Logik sowie deren Implementierung erläutert und zum Ergebnis ein Fazit gezogen.

Das Kapitel Konzept befasst sich, losgelöst von der konkreten Implementierung, mit der Grundidee des Spiels, der Funktionsweise der Spielelemente (des Pac-Man, der Geister, Punkte und Level), der Bedienung und der Datenverwaltung. Das gesamte Projekt wird gewissermaßen in der Theorie beschrieben.

In Kapitel 3, Implementierung, wird die konkrete Umsetzung des Konzeptes im Quellcode grob beschrieben, also das Zusammenspiel der client- und serverseitigen Daten und die Intelligenz der Geister.

Abschließend wird im Kapitel Fazit das Endergebnis erläutert und ein Ausblick mit weiteren Verbesserungs- und Erweiterungsmöglichkeiten gegeben, sowie Hinweise zum Starten der Anwendung via XAMPP.

2 Konzept

2.1 Das Spielprinzip

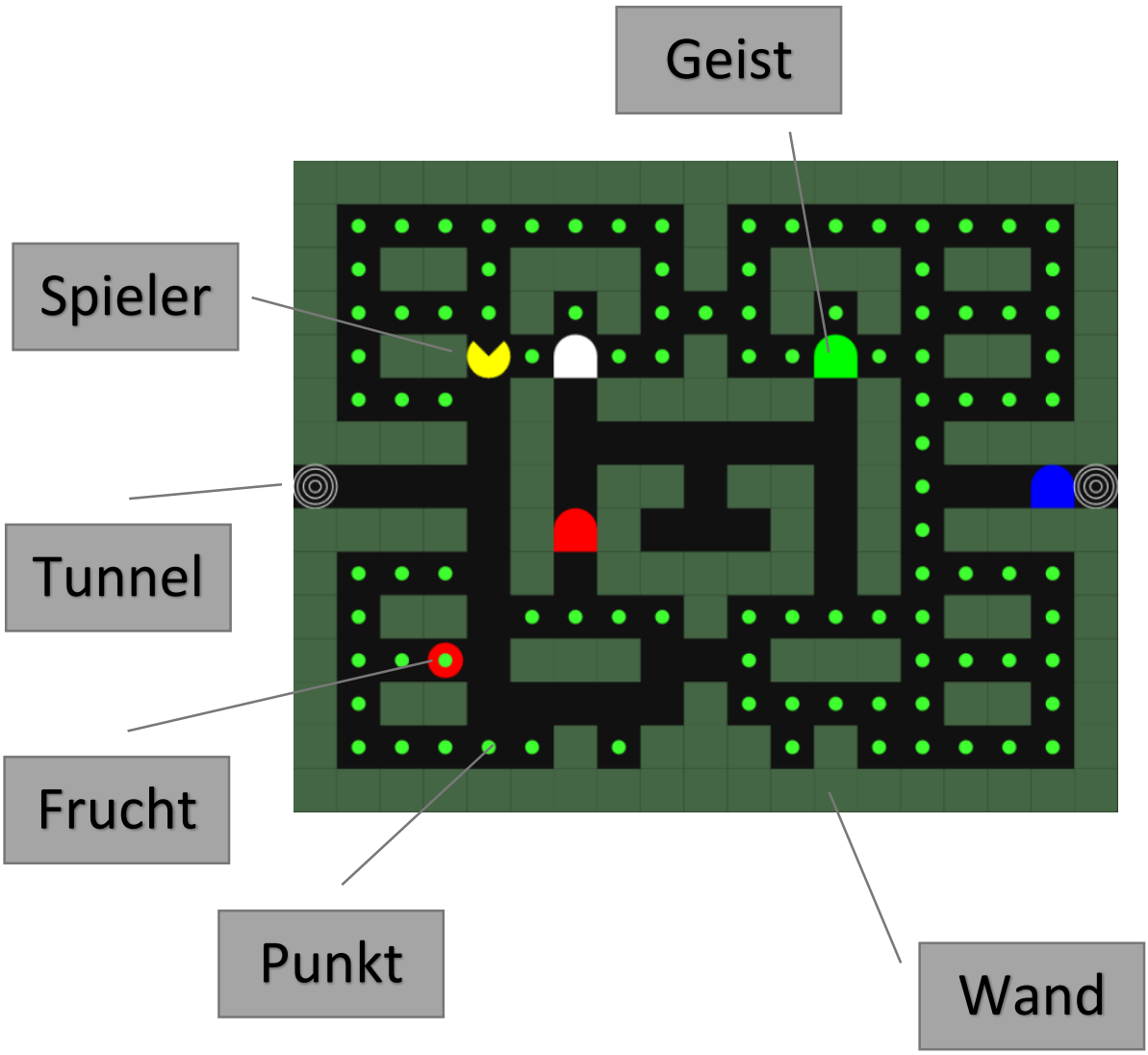
Im Spiel Pac-Man wird der Spieler auf einem 2D-Spielfeld positioniert und muss, ohne von einem der maximal vier Geister gefressen zu werden, alle über das Feld verteilten Punkte einsammeln. Die Geister bewegen sich voneinander unabhängig über das Spielfeld und fressen den Pac-Man bei Kontakt, wodurch der Spieler eines seiner drei Leben einbüßt und das Level neu starten muss. Sind alle Leben aufgebraucht, ist das Spiel zu Ende.

Dem Spieler bleibt meist nur die Flucht, wobei ihm sogenannte Teleporter (auch Tunnel genannt) helfen, die sich links und rechts am Rand des Levels befinden können. Tritt der Pac-Man auf einen Tunnel, erscheint er sofort auf der anderen Seite und kann so etwaige Verfolger abschütteln.

Wehren kann sich der Spieler nur mithilfe der Frucht, die sich in manchen Leveln finden lässt. Frisst der Pac-Man die Frucht, wendet sich für kurze Zeit das Blatt und der Pac-Man frisst bei Kontakt die Geister.

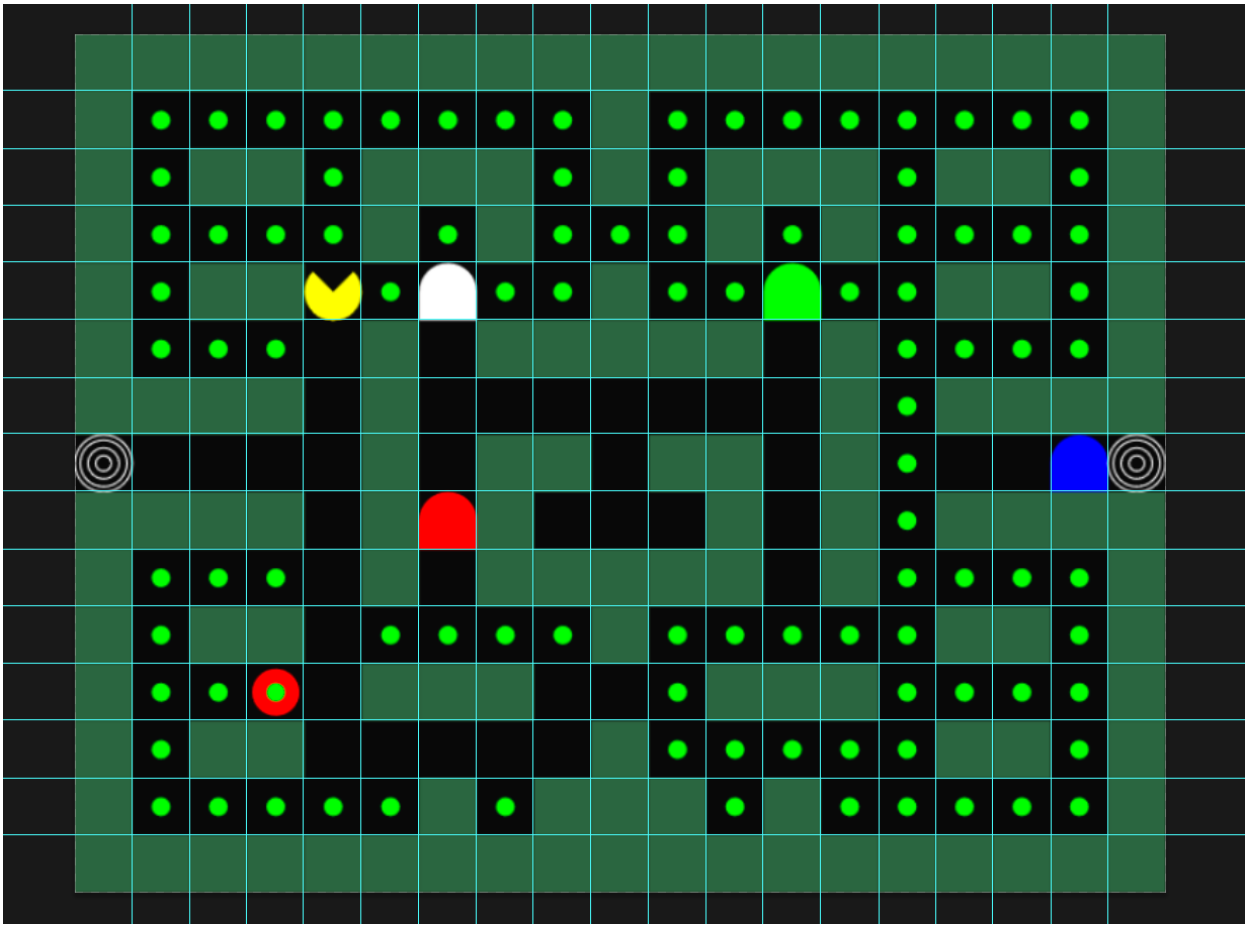
Der Spieler gewinnt ein Spiel, indem er alle Punkte auf dem Spielfeld einsammelt.

Daraufhin wird eine Punktzahl berechnet (unter Berücksichtigung der benötigten Zeit, der Anzahl an Punkten und der Zahl der im Fruchtmodus gefressenen Geister) und der Spieler in eine Bestenliste (Highscore-Liste) eingetragen.



2.2 Levels

Die Level sind als Raster angelegt. Jedes Feld hat eine x- und y-Koordinate und kann mehrere Elemente (Geister, den Spieler, eine Wand, einen Punkt, die Frucht) enthalten.

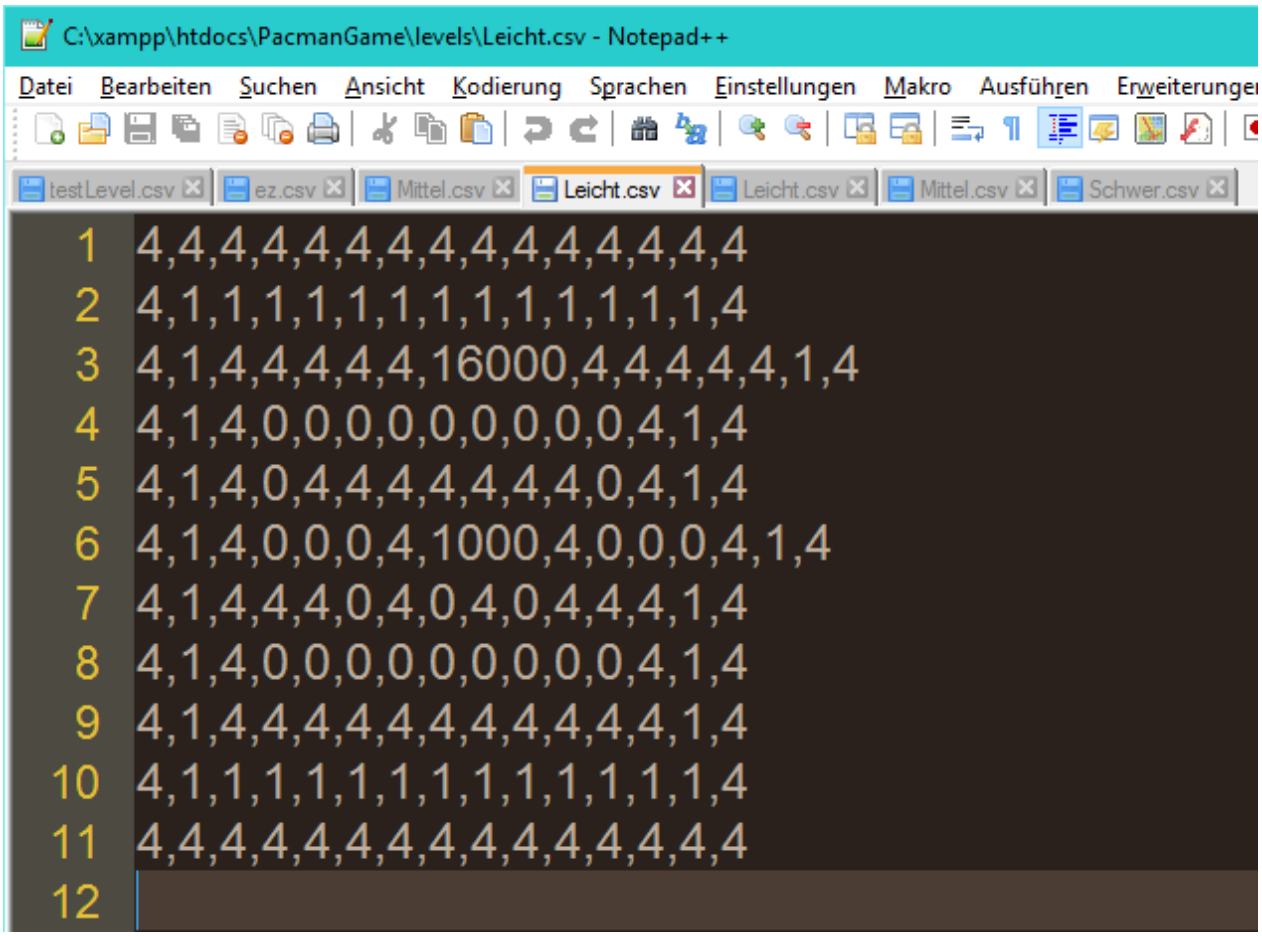


Die Datenstruktur, der das Spiel zugrunde liegt, beinhaltet für jedes Level eine Liste aus Zahlenwerten, jeder Wert repräsentiert ein Feld im Spiel.

Die Spiellogik selbst findet also von der graphischen Darstellung unabhängig im Hintergrund statt, das Bild wird lediglich immer wieder aktualisiert.

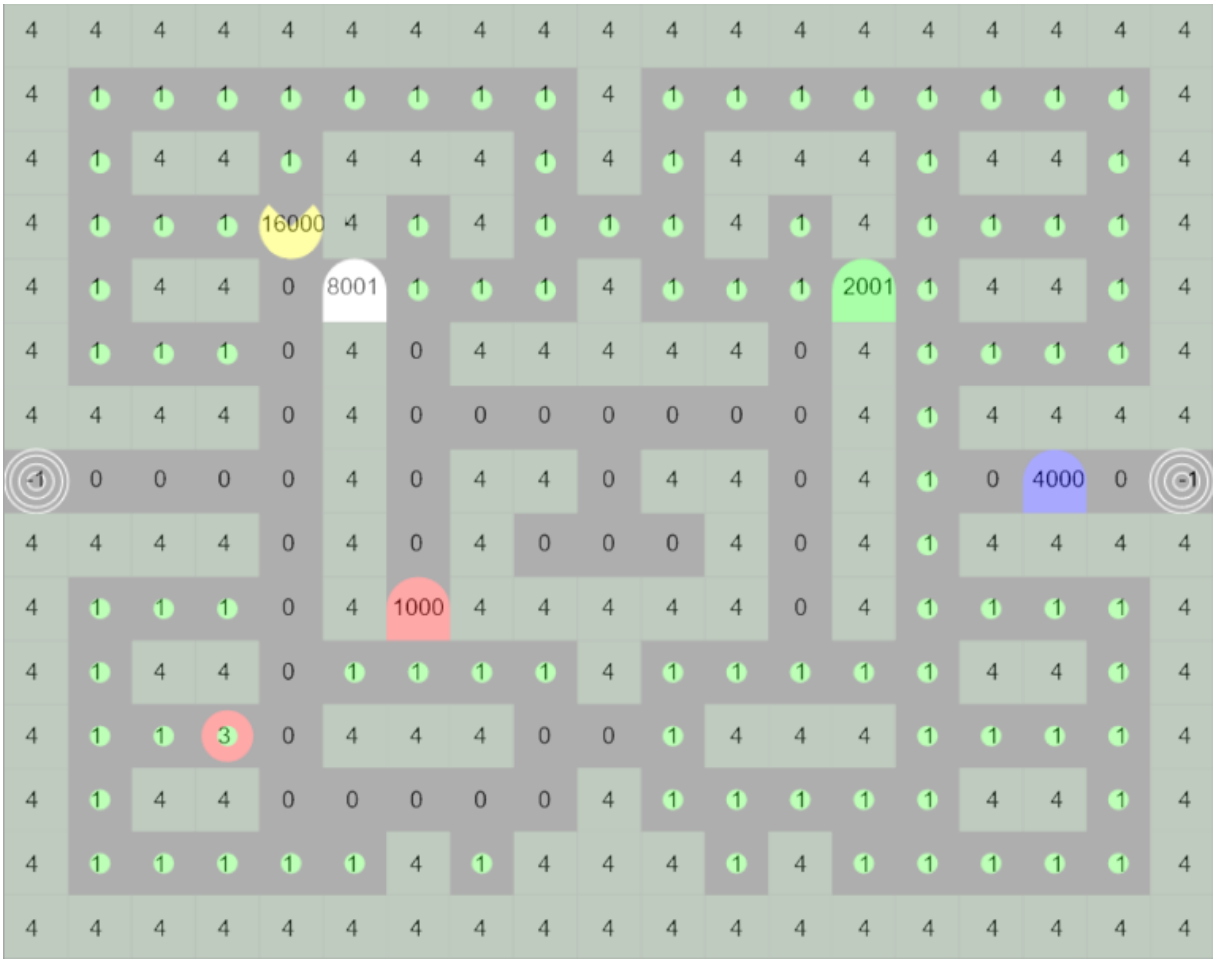
Level werden einfach durch Textdateien eingelesen, die eine Liste aus Zahlen enthalten, aus denen das spielbare Level konstruiert wird. Der sich daraus ergebende Vorteil ist, dass sich das Spiel sehr leicht um viele weitere Level erweitern lässt. Sie müssen nicht hart in den Code geschrieben werden, alles, was man zum Levelbau benötigt, ist ein Texteditor, der Umbrüche erkennen kann (denn jede Zeile Text steht für eine Zeile im Raster), wie Notepad++ (der standard Texteditor unter Windows hat Probleme mit Umbrüchen).

In der folgenden Graphik sieht man das Level *Leicht* im Text-Editor Notepad++ geöffnet, wo es direkt bearbeitet werden kann.



Folgende Werte sind für Spielelemente fest definiert:

-1	=	Tunnel
0	=	Leeres Feld
1	=	Feld mit Punkt
2	=	Frucht
4	=	Wand
1 000	=	Roter Geist
2 000	=	Grüner Geist
4 000	=	Blauer Geist
8 000	=	Weißer Geist
16 000	=	Pac-Man



Diese zahlenorientierte Art der Datenstruktur hat sowohl Vor- als auch Nachteile.

Der wichtigste Vorteil ist, dass sich bei übereinanderliegenden Elementen, durch geschickte Wertevergabe, aus einem einzelnen Zahlenwert alle Elemente rekonstruieren lassen.

Beispiel: 11 003

Der Wert 11 003 kann nur mithilfe des weißen Geistes gebildet werden (der hat den Wert 8 000), alle anderen Werte zusammengenommen, die unter 11 003 liegen, können ihn nicht erreichen.

Es ist also klar, dass auf einem Feld mit dem Wert 11 003 der weiße Geist liegt. Das bedeutet, dass die übrigen Elemente auf dem Feld zusammengenommen den Wert 3 003 haben
(11 003 – 8 000 = 3 003).

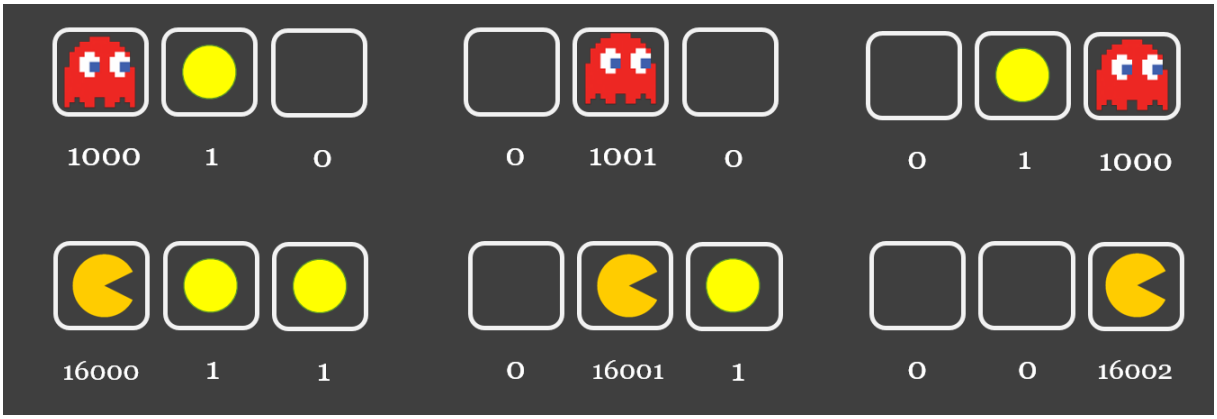
Der Wert 3 003 kann nur mithilfe des grünen Geistes gebildet werden (der hat den Wert 2 000), alle anderen Werte zusammengenommen, die unter 3 000 liegen, können ihn nicht erreichen.

Es ergibt sich also, dass neben dem weißen auch der grüne Geist sich auf dem entsprechenden Feld befindet, wodurch ein Restwert von 1 003 bleibt (3 003 – 2 000 = 1 003).

Dieser wiederum lässt sich nur durch den roten Geist erreichen (vom Wert 1 000), wodurch ein Wert von 3 übrigbleibt, der sich nur aus der Kombination eines Punktes und einer Frucht ergeben kann.

Somit lässt sich herausfinden, dass sich auf einem Feld mit dem Wert 11 003 ein weißer Geist, ein grüner Geist, ein roter Geist, eine Frucht und ein Punkt befindet.

Ein weiterer Vorteil ist, dass sich beim Hinzufügen und Entfernen von Elementen auf Feldern sich kein vorheriger Wert gemerkt werden muss. Betritt ein Geist ein Feld, fügt er seinen eigenen Wert hinzu; verlässt er das Feld, zieht er ihn wieder ab und hinterlässt das Feld automatisch in seinem ursprünglichen Zustand, beispielsweise mit einem Punkt. Der Pac-Man hingegen sammelt Punkte ein und fügt diese einfach seinem eigenen Wert hinzu. Er betritt ein Feld mit einem Punkt (Wert 1), fügt seinen eigenen Wert hinzu (Wert 16 000), erhöht seinen eigenen Wert um 1 und verlässt das Feld wieder (zieht also 16 001 vom Feld ab). Dadurch hinterlässt er automatisch ein leeres Feld.



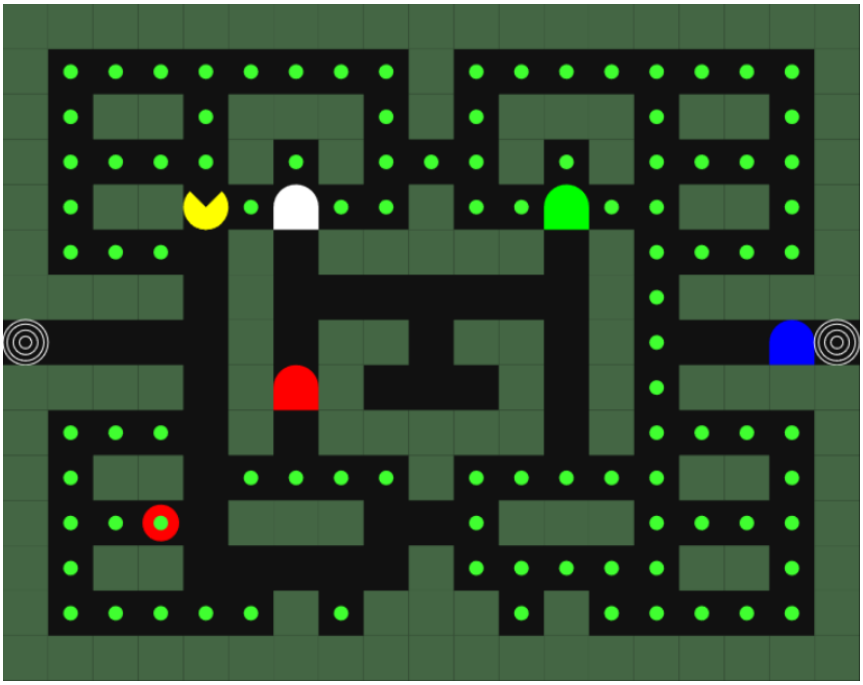
Der Nachteil an diesem System ist, dass für sein Funktionieren bestimmte Regeln befolgt werden müssen:

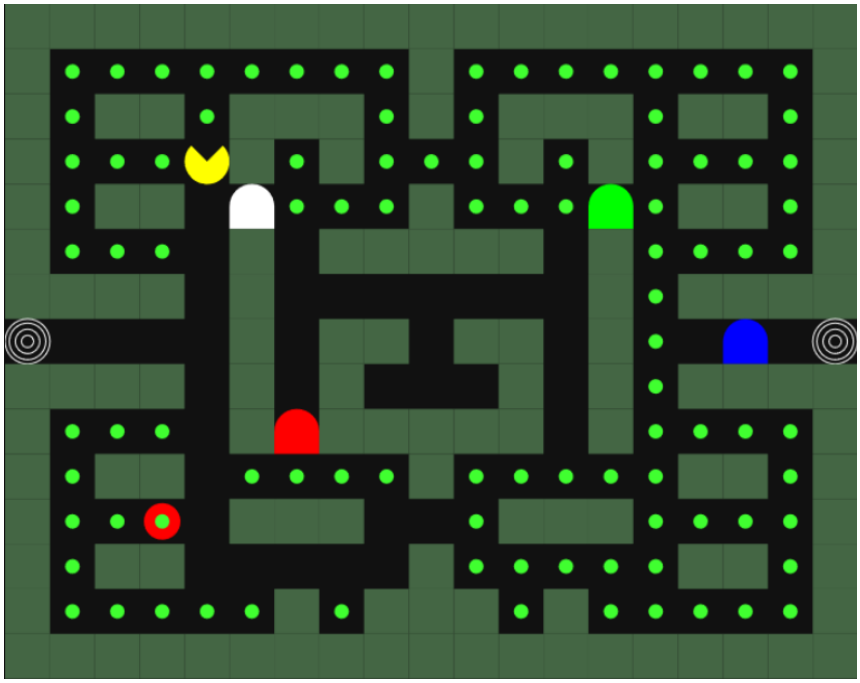
- Es darf maximal ein Punkt auf einem Feld liegen (sonst ließe sich jeder Wert durch eine Menge an Punkten bilden).
- Auf dem Tunnelfeld darf sich weder ein Punkt noch die Frucht befinden.
- Es darf nur ein Geist pro Sorte vorhanden sein (sonst könnte bspw. der Wert 8 000 für 8 rote Geister, 4 grüne Geister oder einen blauen Geist stehen).

Diese Regeln ließen sich zwar durch ein noch geschickteres Verteilen der fest definierten Werte reduzieren, da sie sich allerdings ohnehin mit den Spielregeln decken, habe ich mir diesen Aufwand erspart.

2.3 Bewegung

Das Spiel läuft in Runden ab, wobei jede Runde 200 Millisekunden dauert. In jeder Runde können alle Geister und der Spieler sich je einen Schritt bewegen. Ein Schritt entspricht einem Feld im Raster des Spielfeldes.





In jeder Runde geht der Pac-Man einen Schritt. Er bewegt sich kontinuierlich, auch ohne Eingabe des Nutzers. Daher merkt sich der Pac-Man lediglich die Richtung, in welche er sich bewegt, bis sie sich wieder ändert. Zugleich kann jeder Geist pro Runde genau eine Aktion ausführen. Daraus ergibt sich, dass alle Spielfiguren dieselbe Geschwindigkeit haben, weswegen es sich nicht um ein Renn- sondern um ein Strategiespiel handelt.

2.4 Das Menü

Das Menü, in dem der Spieler sich in der Web-Applikation bewegt, ist sehr einfach aufgebaut.

Der Spieler kann unter *Spielen* einen Level auswählen, unter *Highscores* eine der Bestenlisten einsehen. Pro Level existiert eine Bestenliste (es sei denn, das Level wurde noch nie geschlagen).

Zusätzlich kann der Spieler ein selbst erstelltes Level hochladen, welches auf dem Server gespeichert wird und somit auch von allen anderen Spielern gespielt werden kann. Zum Testen dieser Funktion habe ich eine Level-Datei in den Abgabeordner gelegt (*Davids Lieblingslevel.csv*).

Wichtig beim Erstellen eigener Level ist, dass alle unter 2.2 genannten Regeln eingehalten werden müssen. Außerdem muss das Level von einer Wand umrandet sein, für Geister und den Spieler darf das Gebiet außerhalb des Rasters nicht erreichbar sein.

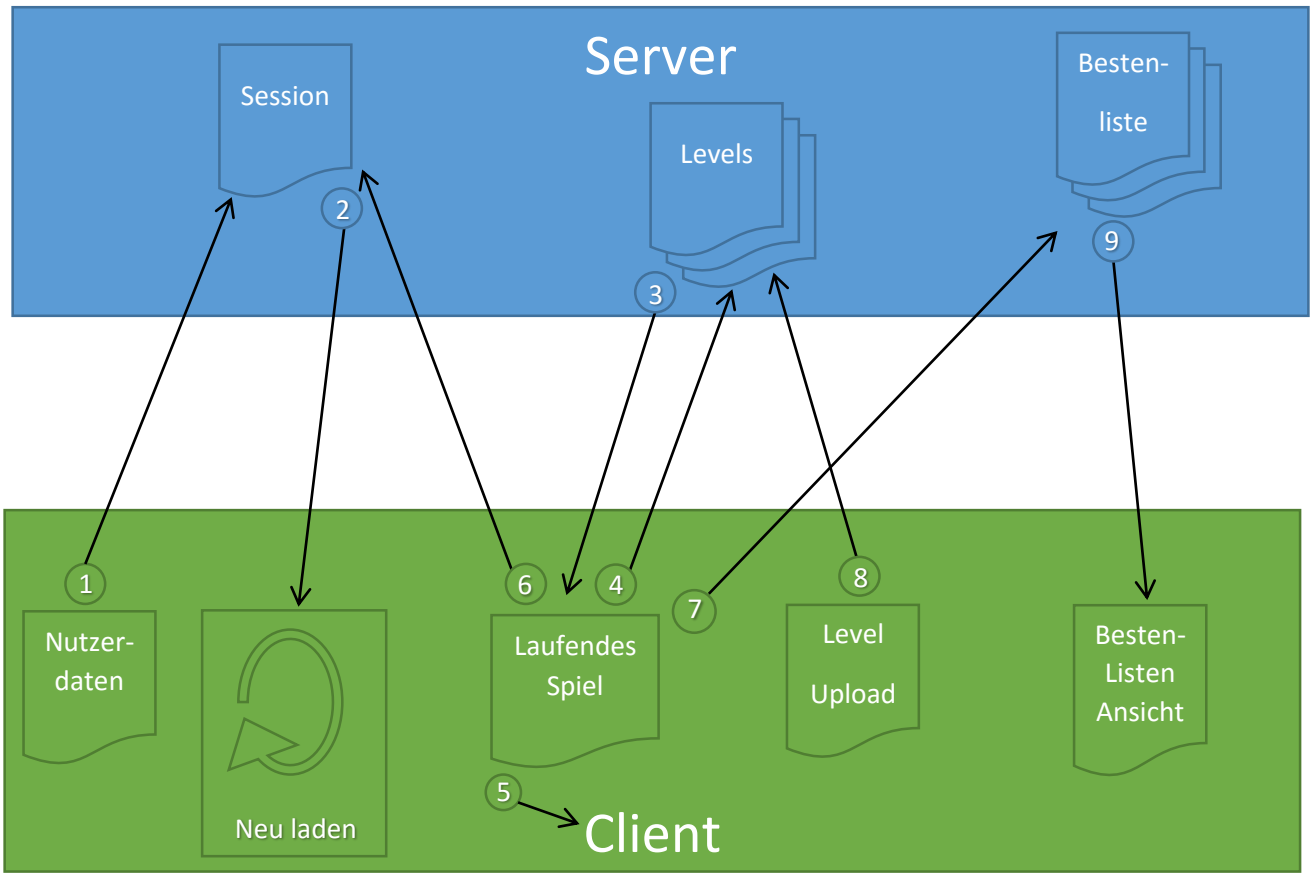
Solang der Nutzer sich in dem Menü bewegt, werden seine Nutzerdaten (sein Name, unter Umständen das aktuelle Level) in einer Session gespeichert. Klickt er auf *Logout*, wird die Session beendet.

3 Implementierung

3.1 Datenströme und Generierung

Um die Datenstruktur des Spieles und die Nutzerdaten zwischen dem Server und dem Client auszutauschen, ist ein Zusammenspiel der entsprechenden Datenströme notwendig.

Dieses Zusammenspiel wird im folgenden Diagramm veranschaulicht:



1: Der Nutzer meldet sich an und startet die Session.

2: Die Session-Daten werden beim Laden auf eine neue Seite übertragen.

3: Der Nutzer lädt ein Level vom Server und startet ein Spiel.

4: Der Nutzer speichert einen Zwischenstand des Spiels auf dem Server ab.

5: Der Nutzer speichert einen Zwischenstand des Spiels lokal auf seinem Rechner.

6: Der Endstand des Spiels wird auf den Server übertragen und die Gesamtpunktzahl wird ausgerechnet. Der Server teilt dem Nutzer beim Laden einer neuen Seite seinen Score (Punktestand) mit.

7: Der Score (die Endpunktzahl beim Sieg) wird in die entsprechende Bestenliste eingetragen.

8: Der Nutzer lädt einen selbst geschriebenen Level auf den Server.

9: Der Server überträgt die Bestenliste an den Client, damit dieser sie einsehen kann.

3.2 Die Spiellogik

Dieser Abschnitt beschreibt, wie der Code für das lauffähige Spiel aufgebaut ist.

Wann immer ein Spiel geladen werden soll, geschieht das über die Datei *game.php*, der eine Anzahl von Leben übermittelt wird. Wann immer ein Geist den Spieler erwischt, wird die Seite mit einem Leben weniger neu geladen. Liegt die Anzahl Leben bei null, wird dem Spieler der Game-Over-Screen angezeigt, mit einem Link zum Hauptmenü.

Ansonsten wird das aktuelle Level aus der entsprechenden Level-Datei auf dem Server geladen. Daraufhin wird das Spielskript zusammengebaut, welches aus mehreren JavaScript-Dateien und vom Level abhängigen Code besteht. So wird über PHP in JavaScript die Anzahl der Leben festgelegt.

Der lauffähige Code besteht aus:

- dem Player-Verhalten (*player.js*)
- dem Geister-Verhalten (*ghosts.js*)
- der Spiellogik (*logic.js*)
- dem 2D-Array, welches von PHP auf JavaScript übertragen wird und das Spielfeld bildet.

Der interessante Teil des Zusammenspiels aus JavaScript und PHP liegt im Zusammenbau des Levels. Dies geschieht im PHP-Code der inkludierten Datei *graphFunc.php* mit ihrer Funktion *build()*.

```php
/*
* Baut ein PHP-Array in ein JS-Array um und liest alle noetigen Variablen aus
* Parameter: Das 2D-Array mit den Felddaten
*/
function build($labyrinth)
{
    $code = ""; // Variable fuer den JavaScript Code, der zurueckgeliefert und in die Website eingebunden wird
    $y = 0; // Aktuelle Zeile im 2D-Array
    foreach($labyrinth as $row) // Array wird Zeilenweise ausgelesen
    {
        $code = $code.buildLine($row, $y); // JS-Code zum Bauen der Zeile wird dem Code angefuegt
        $y++;
    }
    return $code; // Fertigen Code zurueckliefern
}
```

Diese Funktion liefert den nötigen JavaScript-Code zurück, durch den das Spiel erst lauffähig gemacht wird. Sie nimmt das 2D-Array entgegen, welches vom Server aus der Level-Datei (csv) gelesen wurde und liest dieses Zeile für Zeile aus (in der Funktion *buildLine()*).

Die Funktion *buildLine()* nimmt eine Zeile des Arrays entgegen und bearbeitet sie Spalte für Spalte:

- Der Zellenwert wird in die entsprechende Zelle in das 2D-Array in JavaScript kopiert.
- Der Zellenwert wird interpretiert (siehe Seite 6) und entsprechende Elemente (Punkt, Frucht, Geister und/oder Pac-Man) werden initialisiert.
- Der JavaScript-Code, der aus der Zeile generiert wurde, wird angehängt.

Dieser Ablauf wird für jede Zeile und jede Spalte im Array ausgeführt, wonach am Ende der JavaScript-Code zurückgegeben wird, der das lauffähige Spiel komplettiert.

Das Spiel wird also gewissermaßen erst zur Laufzeit programmiert. Die dadurch entstehenden Möglichkeiten werden im Abschnitt *Ausblick* erläutert.

Der Spielablauf selbst wird in *logic.js* geregelt. *logic.js* definiert zunächst feste Feldwerte, wie die der Geister, des Spielers, der Wände etc., damit diese wiederverwendet und ohne großen Aufwand abgeändert werden können. Folgende Funktionen sind in *logic.js* enthalten:

- gameToText()
  Das aktuelle Level wird in Klartext umgewandelt, der sowohl auf dem Server, als auch auf dem Client als Level-Datei abgespeichert werden kann.
  Parameter: ein String, der an Stelle jedes Zeilenumbruchs gesetzt wird. Dieser Parameter ist nötig, weil Umbrüche innerhalb einer URL (%0D%0A) anders definiert sind, als in Textdateien (\n). Je nachdem, wie der resultierende Text weitergeleitet wird, muss ein anderer newLine-Parameter übergeben werden.
- action()
  Je nach durch den Nutzer gedrückter Taste wird das Spiel gestartet, pausiert, gespeichert, abgebrochen oder der Pac-Man bewegt
- update()
  Das Spielfeld wird neu gezeichnet. Dies geschieht, indem das 2D-Array, welches den Spielstand definiert, Zelle für Zelle ausgelesen wird. Es wird weiterhin geprüft, ob sich der Pac-Man und ein Geist auf demselben Feld befinden.
  Zudem werden als Gimmick sämtliche Felddaten weiter unten auf der Seite aufgelistet, was dem Spieler einen Einblick in die Datenstruktur ermöglicht (und vor allem beim Debugging eine große Hilfe ist).
- paintGhost()
  Zeichnet die Form eines Geistes an eine gesetzte Koordinate.
- timer()
  Diese übergeordnete Funktion ruft die anderen bei jeder neuen Runde auf.
  Der Zeitzähler wird inkrementiert, jeder Geist darf einmal agieren, der Spieler bewegt sich und die Sieg- und Niederlagebedingung werden überprüft.

3.3 Die Logik der Geister

Es gibt drei Geistertypen:

- Rot (der Unberechenbare)
  Der rote Geist ist am einfachsten gestrickt. Er geht von Kreuzung zu Kreuzung und wählt stets zufällig eine neue Richtung (nicht ganz zufällig, da die Wahrscheinlichkeit dafür, dass er eine 180° - Drehung vollzieht, sehr gering gehalten wird).
- Grün und Blau (die Jäger)
  Diese Geister halten stets horizontal und vertikal nach dem Pac-Man Ausschau. Sehen sie ihn, halten sie direkt auf ihn zu. Verlieren sie ihn aus dem Blickfeld, begeben sie sich zur letzten Stelle, an der sie ihn gesehen hatten. Sehen sie, dass der Pac-Man unter dem Einfluss der Frucht steht, ergreifen sie die Flucht. Sehen sie ihn nicht und kennen sie auch keinen vorherigen Aufenthaltsort, fallen sie auf das Verhalten des roten Geistes zurück.
- Weiß (der Schummler)
  Der weiße Geist kann den Spieler in einem gewissen Radius durch Wände hindurch spüren. Tut er dies, bewegt er sich direkt auf den Spieler zu. Er ist allerdings nicht besonders klug, läuft dabei gegen Wände und begreift auch nicht immer, dass er fliehen sollte, wenn der Spieler die Frucht gegessen hat.
  Kann er den Spieler nicht spüren, fällt auf das Verhalten der Jäger zurück.

4 Fazit

4.1 Hinweis zum Starten der Webanwendung

Die Webanwendung sollte im Prinzip in jedem modernen Browser laufen, da keine exotischen Bibliotheken verwendet werden, sondern nur grundlegende Techniken (JavaScript, PHP, HTML 5). Getestet wurde die Anwendung auf den neuesten Versionen von Firefox und Google Chrome.

Zum Starten der Anwendung kopieren Sie einfach den gesamten Abgabeordner (mit allen Leveln, Quelldateien und Highscore-Listen) in Ihren htdocs-Ordner.

Sollte der Level-Upload nicht funktionieren, stellen Sie bitte sicher, dass Sie das Hochladen von Dateien in Ihrer php.ini (falls Sie XAMPP verwenden) nicht deaktiviert haben.

Die php.ini finden Sie unter xampp/php/ .

Schauen Sie in der Nähe der Zeile 913 nach folgender Einstellung und stellen Sie sie auf *On*:
file_uploads=On

4.2 Gelungene und weniger gelungene Aspekte

Grundsätzlich bin ich mit dem Endprodukt sehr zufrieden, vor allem mit der Modifizierbarkeit des Spielerlebnisses durch den Nutzer.

Dieser kann nicht nur durch das Erstellen eigener Level, sondern durch das direkte Eingreifen in die Datenstruktur über die URL Änderungen vornehmen. Dem Namen Open Pac-Man wird alle Ehre gemacht. Ein resultierender Nachteil ist, dass das Schummeln ohne weiteres möglich ist.

So lässt sich die Seite des Spiels einfach neu laden, kurz bevor man vom Geist gefressen wird, oder man verändert ganz einfach die verbleibende Anzahl an Leben in der Adresszeile:

… *game.php?level=levels/Schwer.csv&lives=100000*

Ein weiteres Problem ist, dass das Spiel bei diesem Eingriff abstürzen kann, wenn die Anzahl an Leben in der URL mehr als einstellig ist. Nochmals zu erwähnen sind die Level-Regeln (siehe Seite 7), bei deren Verletzung die Datenstruktur des Spielfeldes nicht mehr eindeutig ist.

Außerdem lässt sich das Verhalten der Geister verwirren, wenn das Spielfeld nicht von Wänden umrandet ist und sie quasi über den Weltenrand „hinausspähen“ können.

4.3 Ausblick

Gern hätte ich einen rudimentären Level-Editor gebaut, damit der Nutzer sich seine eigenen Spielfelder nicht in einem Texteditor zusammenbauen muss. Dafür fehlte leider die Zeit.

Zudem hätte ich mir gerne die Tatsache zunutze gemacht, dass das Spiel gewissermaßen erst zur Laufzeit programmiert wird, schließlich werden verschiedene Skripte bausteinartig erst beim Spielstart zusammengefügt. So ließe sich die Modifizierbarkeit signifikant erhöhen, indem ich beispielsweise die Möglichkeit böte, eine eigene Geister-KI als Teil des Levels ins Spiel zu laden.

Insgesamt werte ich das transparente und modifizierbare „Open Pac-Man“ als Erfolg.