##### 

**Vorlage für die DIPLOMARBEIT**

**an der HTL Anichstraße**

**Version: 1.4**

**Gültig ab: 31.07.2021**

**Verteilermethode: schulintern – moodle**

**Dokumentenstatus: freigegeben**

**Es darf nur mit aktuellen Originaldokumenten gearbeitet werden.**

**Diese Seite und die folgenden 2 Seiten dienen der Dokumentenlenkung und müssen in der Diplomarbeit gelöscht werden.**

**Änderungshistorie:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Datum** | **Änderungsgrund** | | **Änderungsstand** | | **Verantwortliche** |
|  | **28.11.18** | **Rechtschreib- und Formatierungsfehler** | | **V1.1** | | **YH** |
|  | **11.03.19** | **Inhaltsverzeichnis an V1.0 angepasst** | | **V1.2** | | **YH** |
|  | **21.03.20** | **Seite 4 angepasst Kopfzeile** | | **V1.3** | | **YH** |
|  | **15.07.21** | **Neues Logo** | | **V1.4** | | **YH** |
| Erstellt: | | | geprüft | | Freigegeben: | |
| 15.07.21, YH | | | 15.07.21, YH | | 15.07.21, YH | |
| Datum, Ersteller | | | Datum, Qualitätsbeauftragte | | Datum, Direktion | |

Information zur nachfolgenden Dokumentation der Diplomarbeit

Sehr geehrte Diplomandinnen, sehr geehrte Diplomanten,

alle Abteilungen der HTL Anichstraße haben sich entschlossen, eine einheitliche Dokumentation der Diplomarbeiten einzuführen.

Dafür wurde nachfolgende Vorlage erarbeitet, die am Moodle Server zur Verfügung gestellt wird.

Die blau geschriebenen Texte sind zur Unterstützung gedacht und müssen in der Diplomarbeit auf die Standardtextfarbe (schwarz) umgestellt werden. Das Inhaltsverzeichnis ist bindend, kann aber natürlich mit Unterpunkten beliebig erweitert werden. Optionale Teile können entfallen.

Die Vorlage ist verpflichtend zu verwenden!

**Zusatzinformationen:**

**Wie zitiere ist?**

Es gibt unterschiedliche Vorgaben für ein Zitat, die HTL Anichstraße bezieht sich auf die Regeln der Diplomarbeitswebseite des Bildungsministeriums und diese Zitierregeln sind anzuwenden:

<http://www.diplomarbeiten-bbs.at/hinweise-zum-wissenschaftlichen-arbeiten/zitation-plagiate>

**FAQ**

Auf der vom Ministerium ausgearbeiteten Seite sind sehr viele Fragen zur und über die Diplomarbeit sehr gut beantwortet. Bei Unklarheiten wenden Sie sich aber gerne an die Betreuerin, den Betreuer oder Abteilungsvorstand.

<http://www.diplomarbeiten-bbs.at/faq>

**Gendern**

Aus Gründen der besseren Lesbarkeit wird in dieser Diplomarbeit die Sprachform des generischen Maskulinums angewendet. Es wird darauf hingewiesen, dass die ausschließliche Verwendung der männlichen Form geschlechtsunabhängig verstanden werden soll.

**Sperrvermerk**

Der Sperrvermerk wird dann ins Dokument eingebunden, wenn die Auftraggeberin, der Auftraggeber dies aus verständlichen Gründen vorgibt. WENN ein Sperrvermerk notwendig ist, dann werden die Seiten, die es betrifft in der Bibliotheksversion NICHT in die DA eingebunden (Vorgabe der HTL Anichstraße), die Seiten bleiben LEER, damit beugen wir vor, dass ein gesperrtes Exemplar versehentlich öffentlich ausgegeben werden kann. Die Betreuerin, der Betreuer benötigen für die Notenfindung ein vollständiges Dokument.

Wir möchten darauf hinweisen, dass die Präsentation trotz Sperrvermerk ÖFFENTLICH ist!

**DIPLOMARBEIT**

**Titel**

**Höhere Technische Bundeslehr- und Versuchsanstalt Anichstraße**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Abteilung**

**Vollständige Bezeichnung der Abteilung**

|  |  |  |
| --- | --- | --- |
| Ausgeführt im Schuljahr 2021/ von: |  | Betreuer/Betreuerin: |
| Max Lumen 5AHxx  Maria Kunststoff 5AHxx |  | Dipl.-Ing. Robert Metall Dipl.-Ing.in Sandra Volt Dipl.-Ing. Hans Wirtschaft |

Projektpartner: Vollständige Bezeichnung der Firma, Innsbruck

Innsbruck, am TT.MM.JJJJ

Abgabevermerk: Betreuer/in:

Datum:

**SPERRVERMERK**

Auf Wunsch der Firma Vollständige Bezeichnung der Firma ist die vorliegende Diplomarbeit

für die Dauer von drei / fünf / sieben Jahren

für die öffentliche Nutzung zu sperren.

Veröffentlichung, Vervielfältigung und Einsichtnahme sind ohne ausdrückliche Genehmigung der Firma \*\*\* und der Verfasser

bis zum TT.MM.JJJJ nicht gestattet.

Innsbruck, TT.MM.JJJJ

Verfasser:

Vor- und Zuname Unterschrift

Vor- und Zuname Unterschrift

Firma:

Firmenstempel

## Kurzfassung /Abstract

Eine Kurzfassung ist in deutscher sowie ein Abstract in englischer Sprache mit je maximal einer A4-Seite zu erstellen. Die Beschreibung sollte wesentliche Aspekte des Projektes in technischer Hinsicht beschreiben. Die Zielgruppe der Kurzbeschreibung sind auch Nicht-Techniker! Viele Leser lesen oft nur diese Seite.

Das Ziel dieser Diplomarbeit ist es LEDs verschieden Muster ausgeben zu lassen, und, nach gewissen Regeln, Eingaben eines Nutzers zu verarbeiten und anzupassen und in einen Tisch zu integrieren. So soll es bestimmte Spiele auf der Oberfläche des Tisches anzuzeigen und zu spielen. … ist die reale Darstellung von Software, und die Aufzeigen, der Abbau des Stigmas bezüglich der kälte, mit der die Programmierung angesehen wird. So soll es möglich seinen ein Heißgetränk seiner Wahl auf den Tisch zu stellen und während der Zeit, in der es auf eine angenehme Wärme fällt auf demnselben ein Spiel aus der Auswahl zu genießen.

Das Ziel dieser Diplomarbeit ist das Entwickeln verschiedener Programme, die dann mithilfe von LEDs auf einer Tischoberfläche angezeigt werden können und mit der ein Nutzer interagieren kann.

Beispiel für ein Abstract (DE und EN)

Zielsetzung Kurzbeschreibung Ausgangspunkt Thema

Die vorliegende Diplomarbeit beschäftigt sich mit verschiedenen Fragen des Lernens Erwachsener – mit dem Ziel, Lernkulturen zu beschreiben, die die Umsetzung des Konzeptes des Lebensbegleitenden Lernens (LBL) unterstützen. Die Lernfähigkeit Erwachsener und die unterschiedlichen Motive, die Erwachsene zum Lernen veranlassen, bilden den Ausgangspunkt dieser Arbeit. Die anschließende Auseinandersetzung mit Selbstgesteuertem Lernen, sowie den daraus resultierenden neuen Rollenzuschreibungen und Aufgaben, die sich bei dieser Form des Lernens für Lernende, Lehrende und Institutionen der Erwachsenenbildung ergeben, soll eine erste Möglichkeit aufzeigen, die zur Umsetzung dieses Konzeptes des LBL beiträgt. Darüber hinaus wird im Zusammenhang mit selbstgesteuerten Lernprozessen Erwachsener die Rolle der Informations- und Kommunikationstechnologien im Rahmen des LBL näher erläutert, denn die Eröffnung neuer Wege zur orts- und zeitunabhängiger Kommunikation und Kooperation der Lernenden untereinander sowie zwischen Lernenden und Lernberatern gewinnt immer mehr an Bedeutung. Abschließend wird das Thema der Sichtbarmachung, Bewertung und Anerkennung des informellen und nicht-formalen Lernens aufgegriffen und deren Beitrag zum LBL erörtert. Diese Arbeit soll einerseits einen Beitrag zur besseren Verbreitung der verschiedenen Lernkulturen leisten und andererseits einen Reflexionsprozess bei Erwachsenen, die sich lebensbegleitend weiterbilden, in Gang setzen und sie somit dabei unterstützen, eine für sie geeignete Lernkultur zu finden.

This thesis deals with the various questions concerning learning for adults – with the aim to describe learning cultures which support the concept of live-long learning (LLL). The learning ability of adults and the various motives which lead to adults learning are the starting point of this thesis. The following analysis on self-directed learning as well as the resulting new attribution of roles and tasks which arise for learners, trainers and institutions in adult education, shall demonstrate first possibilities to contribute to the implementation of the concept of LLL. In addition, the role of information and communication technologies in the framework of LLL will be closer described in context of self-directed learning processes of adults as the opening of new forms of communication and co-operation independent of location and time between learners as well as between learners and tutors gains more importance. Finally the topic of visualisation, validation and recognition of informal and non-formal learning and their contribution to LLL is discussed.

On the one hand this thesis shall assist the dissemination of different learning cultures and on the other hand set off a reflection process among adults, who are in the process of live long learning and therefore support them to find a suitable learning culture.

## Projektergebnis

Allgemeine Beschreibung, was vom Projektziel umgesetzt wurde, in einigen kurzen Sätzen. Optional Hinweise auf Erweiterungen. Gut machen sich in diesem Kapitel auch Bilder vom Gerät (HW) bzw. Screenshots (SW).

Liste aller im Pflichtenheft aufgeführten Anforderungen, die nur teilweise oder gar nicht umgesetzt wurden (mit Begründungen).

**Inhaltsverzeichnis**

[Gendererklärung i](#_Toc531171703)

[Kurzfassung /Abstract ii](#_Toc531171704)

[Projektergebnis iv](#_Toc531171705)

[1 Einleitung 1](#_Toc531171706)

[2 Vertiefende Aufgabenstellung 1](#_Toc531171707)

[2.1 Schülername 1 1](#_Toc531171708)

[2.2 Schülerinnenname 2 1](#_Toc531171709)

[3 Dokumentation der Arbeit 2](#_Toc531171710)

[4 Erklärung der Eigenständigkeit der Arbeit 3](#_Toc531171715)

[I. Abbildungsverzeichnis I](#_Toc531171716)

[II. Tabellenverzeichnis I](#_Toc531171717)

[III. Literaturverzeichnis I](#_Toc531171718)

[IV. Abkürzungs- und Symbolverzeichnis III](#_Toc531171719)

[Anhang IV](#_Toc531171720)

[A1 Pflichtenheft (OPTIONAL) IV](#_Toc531171721)

[A2 Schlussfolgerung / Projekterfahrung IV](#_Toc531171722)

[A3 Projektterminplanung IV](#_Toc531171723)

[A4 Arbeitsnachweis Diplomarbeit IV](#_Toc531171724)

[A5 Datenblätter (OPTIONAL) V](#_Toc531171725)

[A6 Technische Zeichnungen (OPTIONAL) V](#_Toc531171726)

Einleitung

In der Einleitung wird erklärt, wieso man sich für dieses Thema entschieden hat. (Zielsetzung und Aufgabenstellung des Gesamtprojekts, fachliches und wirtschaftliches Umfeld)

Vertiefende Aufgabenstellung

## Christoph Schwarzenauer

• Programmierung des Spiels  
• Planung der Spiellogik

Vertiefende Aufgabenstellung laut Antrag.

## Philipp Bock

• Aufbau des Tisches  
• Ansteuerbarkeit der LEDs

Vertiefende Aufgabenstellung laut Antrag.

## Kürsat Salim

Programmierung des Spiels  
• Software zur Einbindung der Benutzersteuerung

# Dokumentation der Arbeit

3.1. Da zwei der drei Mitglieder ausgestiegen sind geht diese Arbeit allleine auf Christoph Schwarzenauer zurück, bis auf jene Teile, die extra gekennzeichnet sind.

3.1 Python

Python ist eine „high-level“ Programmiersprache, die vorallem durch ihre große Standardbibliothek und durch ihre Vielseitigkeit in ihren Einsatzgebieten besticht. Eine „high-level“ Programmiersprache weist, aufgrund der Höhe der Abstraktion eine erhöhte Ähnlichkeit mit einer natürlichen Sprache auf, wodurch sie für den Nutzer, als angenehmer zu lesen und schreiben gilt. Wobei der Komparativ hier wichtig ist, denn obwohl sie um ein vielfaches intuitiver, als Beispielsweise C oder Pascal gilt, ist sie dennoch größtenteils fremd für das menschliche Auge und bringt immer noch ein hohes Zeitinvestment mit sich.

Ins Leben gerufen wurde Python von Guido van Rossum und erhielt ihren Namen aufgrund der Tatsache, dass dieser in zu dieser Zeit gerade Skripte der Serie „Monty Python’s Flying Circus“ las.

Python ist sozusagen das Fundament dieser Diplomarbeit, auf dem alles aufbaut. Diese Programmiersprache wurde gewählt, da sie für das Neopixel Modul und der Betriebssystemsunabhängigkeit ideal für das Ansteuern der LEDs mittels eines Raspberry Pi’s ist und im Team generell ein Interesse an dieser Sprache vorhanden war, wodurch nach ein paar Prototypen in der Programmiersprache C gemeinschaftlich entschieden wurde, zu Python zu wechseln. Es existieren auch eine Unzahl von Anleitungen zum erlernen der Sprache, die man ohne Probleme, mit seinem Internetanschluss, ohne Zusatzkosten abrufen kann.

LED Streifen

Als visuelle Repräsentation der Programme wurde hier ein LED-Streifen verwendet. Bei kleineren Maßen wäre aber auch eine LED-Matrix denkbar. Die Grundidee ist einem Bildschirm gleich, nur in einer viel geringeren Auflösung. So kann die LED entweder leuchten oder nicht, und einen Farbzustand, für die Unterscheidung der verschiedenen Elemente aufweisen. Man hat sich hierbei für einen LED-Streifen der Bezeichnung WS-2812b entschieden.

Netzteil

(Ampere kalkulartion)

Raspberry Pi

Beschreibung

Einsatz

Umsetzung

Einstellungen

Installation

Git repo install

Ausführen

Etc

Code

Komplexität abwärts

Zuerst die zwei einfachen, dannach Snake und Schach

Chronologisch durch, wobei jeder logische Schritt beschrieben wird

Für die einfachheit der Erklärung wird zuerst ein Prototyp ausgegeben

6?.1 Inklusiver Bau

Aus Gründen der chronologischen Programmierung wurde zuerst ein Prototyp erstellt. Dieser ist vollfunktionsfähig, jedoch wird dieser über das Terminal ausgegeben. Dieser wird für Einfachheit der Erklärung zuerst beschrieben. Da dieser aber beinahe unverändert übernommen werden kann gehen wir später auf die Änderungen ein, die gemacht wurden, sodass die Ausgabe nicht mehr auf dem Bilschirm, sondern auf den Pixeln des Tisches erfolgt. Weiters muss gesagt werden, dass die Reihenfolge der beschriebenen Programme nicht der chronologischen Reihenfolge der Programmierung, sondern der Einfachheit des Programms entspricht. So fangen wir beim einfachsten Programm an und arbeiten uns zu den komplizierten und zum Schluss fassen wir alle in die Spiellogik ein. So ist es möglich, dass Funktionen, der Programme, die ähnlich sind bei den komplexeren Programmen weniger beschrieben werden und somit ein größerer Fokus auf die Nachvollziehbarkeit der Programmlogik ist.

Auch muss vorab noch gesagt werden, dass der Autor dieser Programme ein Programmieranfänger war und sich dies sowohl auf die Zeit, die diese in Anspruch nahmen und auch auf die Qualität des Codes auswirken, jedoch nicht auf das Ergebnis der Programme, die alle wie konzipiert funktionieren.

6. 1Guess the Square

„Guess the Square“ ist ein eigens für diese Diplomarbeit entwickeltes Spiel mit einer simplen Prämisse: „Merke dir alle Punkte einer immer länger werdenden Linie und Versuche sie nachzuzeichnen“. Man startet mit einem zwei Felder langen Weg, auf einem der Felder „steht“ man sozusagen schon, somit muss man eine Entscheidung treffen, „geht“ man nach links, rechts, oben, oder nach unten. Zuerst ist dies noch relativ einfach, aber da man nach der i+1 (Quelle) Loigik immer alle vorherigen Felder mitzeichnen muss stößt man früher oder später ans Ende seiner Merffähigkeit. Der Name bezieht sich auf die Momente, die spät im Spiel auftreten, an dem der Spieler das folgende Feld vergessen hat und man sozusagen das nächste Feld raten musss.

6.2 Libraries

Bei einem Import handelt es sich um Header Dateien oder Moduele (Quelle??? Korrekt??), die in das Programm einnbezogen werden. Im Unterschied zu Beispielsweise C muss die Standard-Library nicht importiert werden. Aber dennoch ist ein Beispiel aus der Standardlibrary hilfreich für das Verständnis.

Nehmen wir die Genesis eines jeden Programmieres:

Print(„Hello World“) Print ist eine Funktion, die Argumente nimmt. In diesem Fall ist das Argument des Typs String und hat den Inhalt(??) „Hello World“. Die Anführungszeichen kennzeichnen, dass es sich hier um einen String handelt.

So kompilieren(???) und führen wir nun das Pramm aus erhalten wir im Termin die Nachricht:

Hello World

Dies ist eine reduktive Erklärung von Funktionen, die aber dem Verständnis des weiteren ungemein helfen wird.

import keyboard  
import random  
import os  
import time  
import numbers

Es wurden Module und Header(???) mit der Bezeichnung keyboard, random, os, time, numbers importiert. Bis auf das letzte Element handelt es sich um open-source(wirklich? Quelle!) librarys (???), das letzte wurde speziell für den Gebrauch von allen Programmen erstellt. Auf dieses werden wir später eingehen, wenn der Einsatz gegeben wird.

Die Libraries, die installiert werden müssen sind: keyboard und os und werden mit dem pip install X installiert, wobei X in diesem Fall für### os##### und keyboard steht.

Nun zu den Verwendungen der Libraries:

Die Library „keyboard“ wird für das Erfassen vom Drücken von Tasten verwenden. Hauptsächlich werden hier „w“,“a“,“s“,“d“ verwendet, diese Dienen zur allgemeinen Steuerung, aber auch „enter“ (Auswahl) und „backspace“ abbruch haben hier Verwendung. Für alle Programme wurde nur die Funktion keyboard.read\_key() verwendet. Diese Funktion (???) macht es möglich zu erkennen, ob eine Taste gedrückt und wurde und speichert den Namen der Taste auf eine Variable.

So würde die Variable X, wenn X=keyboard.read\_key(), beim Tastesndruck „O“, O, des Typs „Character“ zugewiesen bekommen.

Somit würde, wenn man X in die Funktion print() einspeißt, ein „O“ im Terminal ausgegeben werden.

Diese Logik stellt das A und O allem nachfolgenden dar, wenn ohne Interaktion mit dem Computer, in diesem Fall dem Raspberry, kann es kein Spiel, sondern nur ein Video geben.

Es wären aber auch andere Funktionen, wie z.B. key\_\_\_\_\_ für dieses Programm denkbar, wurden jedoch aufgrund von einigen Bugs nicht in den Code eingearbeitet.

Die Library random, ########################

Die Library os wird verwendet, da es sich bei den Maschinen in der Entwicklung sowohl für Linux, als auch Windows Betriebssdystemen gehandetl hat. So ist es möglich das Programm direkt zu übertragen, Dateipfade anzupasse, wie das im Detail aussieht sehen wir später.

Import time ####

Import numbers ####

6.3 Feld und Konstanten

Aus Gründen der Übersichtilichkeit und der Einfeichheit, mit der der Code angepasst werden kann, wurden direkt unter den Imports globale Variablen und Konstanten sowohl deklariert, als auch definiert.

SIZE = 242  
COLUMNS = 22  
STARTING\_POSITION = 120  
global field  
field = [0]\*SIZE

SIZE bezieht sich auf die Größe des Feldes, also auf die Auflösung, mit der die Ausgabe Arbeitet, in diesem Fall ist die Aufläsung die Anzahl der Pixel am Tisch und somit 242 (11\*22). Es wäre aber ohne weiteres möglich den Code für andere Auflösungen zu verwenden, man muss einfach die gewünschte Zahl stattdessen einfügen. Das gleiche gilt auch für die Konstante „COLUMNS“, was übersetzt „Spalten“ heißt, weicht somit die Anzahl der Spalten, in diesem Fall eine Reihe an Pixeln, von den hier definierten 22 ab, muss nu angepasst werden.

Die Variable field hat den Typ einer Liste und represäntiert die logische Struktur der Pixel. Hier wurde sich für eine globale Variable entschieden, da die meisten Funktionen auf diese Zugreifen und Verändern, was einen Problematiken in der Rückgabe geben könnte.

Als nächstes wird das Feld mit 242 Nullen befüllt, ist in allen Programmen immer der Stellverträter der Aussage „Pixel leuchtet nicht“, alles weitere kann sich je nach Definition auf simples eingeschalten sein, oder auf eine Farbunterscheidung beziehen. Wäre hätte Beispielsweise eine Variable den Wert 1 und eine andere den Wert 2, so leuchten beide LEDs an diesen Stellen, aber mit anderen Farben, was bedeuted, dass es sich um zwei verschiedene Objekte handelt. Wir werden später auf viele solcher Objekte treffen.

6.4 Printfield

def printfield():  
 os.system('cls' if os.name == 'nt' else 'clear')  
 for i in range(SIZE):  
 if i%COLUMNS==0:  
 print("")  
 if field[i]==1:  
 print("[X]",end="")  
 else:  
 print("[ ]",end="")

Die printfield Funktion ist erste des Programms, da sie die einzige ist, die sich von der virtuellen (Terminal) zur realen (LED-Strefen) unterscheidet. Um vom Terminal zu den LEDs zu wechseln bedarf nur wenig aufwand. Wir werden später sehen, wie sich dies gestaltet, aber für das allgemeine Verständnis bleiben wir zunächst bei dem Terminal beispiel.

os.system('cls' if os.name == 'nt' else 'clear')

Zunächst befreien wir das Terminal von allem, was zuvor Geschehen ist, sodass wir uns zum einen auf das Spiel konzentrieren können, aber auch, da im Spiel diese Funktion mehrmals pro Sekunde aufgerufen werden kann. Man spricht von fps(Quelle) oder „Frames per Second“, also Bilder pro Sekunde.

Nun schauen wir in das Feld, gehen Element für Element (Pixel für Pixel) alle Einträge der Liste durch und geben dies nun am Terminal aus. Fall sich eine 1 an einer Stelle befindet, stellen wir dies im Terminal durch ein „[X]“, falls dies nicht der Fall ist,(in diesem Programm gibt es nur Einsen und Nullen) repräsentieren wir es mit einem „[ ]“. Beides dient der Überschaubarkeit. Es wäre auch denkbar, dass man direkt eine 1 und eine 0 ausgibt´. Und da eine lagne eindimensionale Liste fürs menschliche Auge nicht sehr angenehm ist und wir auch generell mit den LED- Streifen eine Matrix erstellen wollen, werden die Ausgaben der Einsen und Nullen ohne Zeilenumbruch gestaltet und stattdessen mithilfe eines normalen print’s nach jeder Vollen Reihe ein Zeilenumbruch erstellt.

Man kann also die printfield Funktion mit „Visualisierung der inneren Logik“ beschreiben.

6.5

def start():  
 path = []  
 path.append(STARTING\_POSITION)  
 score = 0  
 pos=True  
 while pos!=False: #as long as the endscreen did'nt happen yet  
 score+=1  
 path = randomsquare(path)  
 path\_to\_field(path,True,score)  
 pos = STARTING\_POSITION  
 n=1  
 previous\_pos = STARTING\_POSITION  
  
 while n<=score and pos!=False:  
 path\_to\_field(path,False,n)  
 printfield()  
 temporary\_pos=pos  
 pos=choice(previous\_pos,path,pos,score)  
 previous\_pos=temporary\_pos  
 print(n)  
 n+=1  
 time.sleep(0.2) #because it sometimes registers a keypress twice

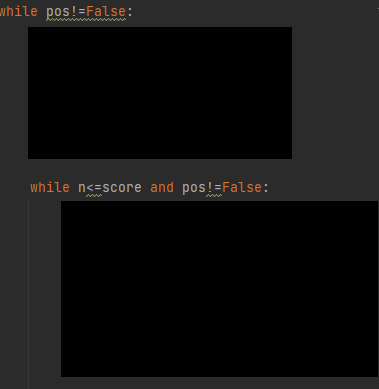
Nun kommen wir zum Herz des Spiels. Dies ist die abfoige aller Funktionen, nach einem bestimmten Schema und nennt sich „Start()“. Wir werden nun im Einzelnen auf alle ´Funktionen und auf den Algorithmus im Allgemeinen eingehen. Man zuerst alle Funktionen abarbeiten und den Zusammenhang erklären, aber da diese Dokumentation auf den Grundsatz der Verständlichkeit mit sich führt werden wir den Algorithmus chronologisch betrachten.

path = [STARTING\_POSITION]  
score = 0  
pos=STARTING\_POSITION

Als aller erst definieren wir die Variablen, die wir brauchen werden. Die Variable Path beinhaltet alle Positionnen des gesuchten Weges hat als erste Position die Startposition des Spielers. Mit Score=0 wird der Anzahl der Punkte, die man erreicht hat auf 0 gesetzt, da der Spieler an dieser Stelle noch keinen Weg korrekt nachgezeichnet hat. Der Score ist dazu dar, dass man später seine eigene Leistung mit der Höchstleistung (HIghscore) vergleichen kann.

Die Variable „pos“ beinhaltet die derzeitige Position des Spielers, womit der Algorithmus überprüfen kann, ob der Spieler richtig gelegen ist, oder nicht. Da der Spieler noch nichts gemacht hat, ist die derzeitige Postition die Startposition.

6.4.1? Die beiden while Schleifen:



Sehen wir uns nun die beiden while Schleifen an und betrachten den Innenaufbau zuerst als Blackbox. Die Bedingung der ersten While-Schleife ist, dass die aktuelle Spielerposition nicht „False“ sein darf. Im Normalfall ist Position des Spielers immer ein Integer, also eine ganze Zahl, diese „False“ soll somit bedeute, dass sich der Spieler auf keiner Position mehr befindet, also verloren hat, und somit soll das Spiel abgebrochen werden.

Die zweite While-Schleife hat die Bedingungen, dass n kleiner oder gleich groß, wie der aktuelle Score sein muss und, dass die pos wieder nicht False sein darf. Man beachte, dass bei mit einer „logischen Und-Verknüpfung“ verbunden sind, was soviel heißt, dass immer beide Bedingungen erfüllt sein müssen, sobald die Schleife von neuem beginnt, ansonsten wird sie abgebrochen. N ist hierbei, wie in jedem anderen Programm dieser Arbeit, die Rundenanzahl. Somit ist die Aussage: „Wenn die Rundenanzahl(in diesem Fall ist jede einzelne Entscheidung eine Runde) größer als der Score ist, ist die Bedingung nicht gegeben“.

Nehmen wir als einfaches Beispiel den Start. Der Score beträgt hier 1(Der eigentliche Score beträgt 0, dies wird bei der Ausgabe des Scores gemacht), da der Spieler noch keinen richtigen Weg gezeichnet hat, aber um den Weg zu Zeichnen muss er einen Schritt machen, somit spring n auf 2 sobald er gezeichnet hat und die while-schleife bricht ab und man erhält einen Score von 1 und die Rundenanzahl ist wieder 1. Nun gib der Computer 2 zu zeichnende Felder vor, momentan hat der Spieler einen Score von 1 und nach den zwei gezeichneten Feldern ist n 2, somit tritt am erneut aus der while Schleife aus und so weiter, bis man einen falschen Schritt getan hat, dann wird die Position false und man tritt aus beiden Schleifen aus und das Programm wird beendet.

6.22.??? Die erste While Schleife

Die erste while Schleife wurde folgend realisiert:

while pos!=False: #as long as the endscreen did'nt happen yet  
 score+=1  
 path = randomsquare(path)  
 path\_to\_field(path,True,score)  
 pos = STARTING\_POSITION  
 n=1  
 previous\_pos = STARTING\_POSITION

Es muss noch einmal gesagt werden, dass nach jedem richtigen Zeichnen diese Schleife durchgegangen wird. Man startet nach jedem positiv abgeschlossenen Versuch wieder bei der Startposition und immer bei der ersten Runde.

Die Variable previous\_pos beinhaltet das zuletzt besuchte Feld, in diesem Fall war der Spieler noch bei keinem Feld, weswegen diese gleich der Startposition ist.

6.4.1?.2 randomsquare

def randomsquare(path):  
 way=[COLUMNS,-COLUMNS,1,-1]  
 while True:  
 r=random.choice(way)  
 pos=r+path[-1]  
 if field[pos]!=1 and 0<=pos<SIZE:  
 path.append(pos)  
 return path

Die „randomsquare“ Funktion ist, gibt, wie der Name schon sagt, ein Zufälliges Feld aus, wobei das Zufällige Feld von der aktuellen Position weggeht und nicht in ein vorheriges Feld tappt. Dies wird mit einer while Schleife gelöst, so beschrieibt man die Variable pos immer wieder, bis auf dem zufälligen Feld keine 1 mehr vorhanden und es nicht über das Feld hinausgeht. Man könnte hier noch eine Absicherung einbauen, denn das Spiel würde in Endlosschleife laufen, wenn kein Platz mehr für den neuen Weg frei ist. Dies wurde aber nicht implementiert, da dieses Szeneario so unwahrscheinlich ist, dass man sehr wohl betrogen haben müsste, um dies zu erreichen.

Wenn dann ein valider Block gefunden wurde wird, wird der Gesamtweg (Path) mit diesem erweitert und zurückgegeben.

6.473246327478 path\_to\_field

def path\_to\_field(path,new,n):  
 if new==True: #if the path gets longer  
 for p in path:  
 field[p]=1  
 printfield()  
 time.sleep(1)  
 for x in range(SIZE):  
 field[x]=0  
 else: #in game  
 for x in range(n):  
 field[path[x]]=1  
 printfield()

Falls das „new“ Argument True ist gehen wir direkt in eine for -schleife. In dieser beschreiben wir das Feld mit der path-liste und visualisieren dann das Feld. So ist dann der Gesamte neue Pfad zu sehen. Die time.sleep(X) Funktion stellt hier die Zeit in Sekunden dar, die der Spieler den Pfad sich merken darf. Denn mit der nächsten For-Schleife wird das Feld wieder mit 0en befüllt.

Falls „new“ jedoch „False“ ist, so wird je die n. Stelle des Pfades und alle vorherigen mit 1en befüllt und angezeigt. Dies ist der Spieler input, Falls dieser richtig liegt, dann wird der Pfad bis zu diesem Punk angezeigt.

6.2232323??? Die zweite While-Schleife

Nun kommen wir zum zweiten Teil der Start() Funktion.

while n<=score and pos!=False:  
 path\_to\_field(path,False,n)  
 temporary\_pos=pos  
 pos=move(previous\_pos,path,pos,score)  
 previous\_pos=temporary\_pos  
 n+=1  
 time.sleep(0.2)

Hier wird also zuerst der schon erreichte Pfad angezeigt (z.B. Runde 1, die erste Stelle, also die Startposition), dann wird die derzeitige Position auf die temporary\_pos variable gespeichert. Dann kann der Spieler mit der move() Funktion seinen Zug machen. Hierfür benötigt man die Argumente previous\_pos, path, pos\_score,

Falls der richtige Zug gewählt wurde erhöht man n. Das time.sleep(0.2) ist für eine Bugvermeideung da, denn es kann passieren, dass der Computer zu schnell ist und der Spieler immer noch eine Taste gedrückt hat.

6.36217863826 move()

def move(previous\_pos,path,pos,score):  
 while True:  
 key=keyboard.read\_key()  
 if key in ["a","d","s","w"]:  
 if key=="a":  
 if pos-1 in path and pos-1 != previous\_pos:  
 return pos-1  
 else: #bug fix  
 endscreen(score)  
 return False  
 if key=="d":  
 if pos + 1 in path and pos + 1 != previous\_pos:  
 return pos + 1  
 else: #bug fix  
 endscreen(score)  
 if key =="s":  
 if pos + COLUMNS in path and pos + COLUMNS != previous\_pos:  
 return pos + COLUMNS  
 else: #bug fix  
 endscreen(score)  
 return False  
 if key=="w":  
 if pos - COLUMNS in path and pos - COLUMNS != previous\_pos:  
 return pos - COLUMNS  
 else:  
 endscreen(score)  
 return False

Der Ablauf dieser Funktion ist recht simple, obwohl es auf den ersten Blick nicht so scheinen kann. Zuerst begibt man sich in eine while Schleife, diese wartet dann auf ein Drücken einer Taste (siehe (WWOOO????) ).

Wenn eine Taste gedrückt wurde schauen wird geschaut, ob es sich um eine valide Taste handelt.

Dies wird mit: „if key in ["a","d","s","w"]:“ gelöst. Diese Zeile liest sich ähnlich einem englischen Satz, mit schlechter Grammatik. So wird also geschaut, ob sich die gedrückt Taste in der Liste ["a","d","s","w"] wiederfindet. Falls dies der Fall ist, so gehen werden die nächsten Schritte eingeleitet.

Die nachstehenden Abfolgen sind recht einfach zusammenzufassen, wenn ein „a“ gedrückt wird, so wird von der aktuellen Position 1 abgezogen, was bedeutete, dass man nach links wandert. Bei „d“ geht man nach rechts, deshalb wird 1 hinzuaddiert. Bei „w“ wandert man ein Feld nach oben, soll heißen, man überspringt eine Reihe, so wird also die Anzahl der Spalten (COLUMS) subtrahiert. Die gleiche Logik wird bei „s“ angewendet, wodurch die Anzahl der Spalten hinzuaddiert wird.

Dies alles tritt aber nur in Kraft, falls das gewählte Feld in der path Liste vorhanden ist und man nicht rückwärts wandert.

Ansonsten kommt man in den Endscreen.

6.39874982374 Endscreen.

def endscreen(score):  
 os.system('cls' if os.name == 'nt' else 'clear')  
 visuals.score\_p()  
 visuals.num\_con(int(score-1))  
 with open("highscore\_gts.txt","r") as highscore:  
 score\_old=highscore.read()  
 if int(score\_old) < score-1:  
 with open("highscore\_gts.txt", "w") as highscore:  
 highscore.write(str(score-1))  
 print("sdfdsfdsf")  
 visuals.hscore\_p()  
 visuals.num\_con(score-1)  
 else:  
 visuals.hscore\_p()  
 visuals.num\_con(int(score\_old))

Zuerst wird die score\_p() Funktion der visuals Library aufgerufen. Was diese im Detail sehen wir uns zum Schluss an. Nur so viel sei gesagt: Sie ist für die Ausgabe der Visualiesierungen außerhalb des direkten Spiels zuständig.

Die score\_p() Funktion sieht im Terminal folgenderweise aus:

######SCREENSHOT von SCORE######

Die hscore\_p() Funktion verhält sich ähnlich, wobei das h für high steht. Das H anstatt des High’s wurde aufgrund des Platzmangels in der Ausgabe gewählt.

####Screenshot von HSCORE ######

Die num\_con() Funktion wandelt eine Zahl in die entsprechende visuelle Repräsentation um.

###Screenshot ######

Doch bevor der Highscore ausgegeben werden kann wird überprüft, ob der neue Score größer als der bislang höchste ist. Wenn dies der Fall ist, wird der alte Highscore mit dem neuen überschrieben, wenn nicht wird der alte ausgegeben.

7. Find the Invisible

Das nächste Spiel ist auch eigens für diese Diplomarbeit entwickelt worden. Das vorherige Spiel war ein Merkspiel, dieses ist ein reines Glücksspiel (wer erkennt ein Muster, nächtes ist geschick etc.). Der Titel „Find the Invisible“ drückt auch schon alles aus, was man hiermit machen muss, denn man muss das Unsichtbare finden. Zurst sieht man Blöcke in eine zufällige Richtung vorbeihuschen und dann ist man schon mit seinem Suchenden Punkt auf sich allein gestellt, denn es gilt nun herauszufinden, wo sich der Block befindet, wofür man eine gewisse Anzahl an Versuchen zur Verfügung hat.

7.1. Libraries

import keyboard  
import random  
import os  
import visuals  
import time

Die verwendeten Libraries sind diesselben, die auch schon in „Guess the Square“ verwendet wurden. Eine genaue Beschreibung findet man unter dem Punkt (6.78478327487324).

7.2.Varaiblen und Konstanten

SIZE = 242  
COLUMNS=22  
STARTING\_POSITION=120  
ATTEMPTS=3  
global field   
global attempts\_and\_score  
global block  
field = [0]\*SIZE  
attempts\_and\_score=[0]\*3  
block=[0,1,2,-ROWS+2,-ROWS+1,-ROWS,-ROWS\*2+2,-ROWS\*2+1,-ROWS\*2]

Auch die Variablen sind dem letzten Spiel ähnlich. Attempts sind die Versuche, die ein Spieler hat, den unischtbaren Block zu finden. „attempts\_and\_score“ ist eine Liste, in der sich dia Anzahl der Versuche, der Score und die Tatsache, ob noch gespielt wird, wiederfinden und sich auch immer wider aktualisiere. Die Liste Blocks hingegen beinhalten die Felder, die man finden kann. Dies wir auch am Anfang der Runde visualisert. Ist das Spiel zu schwer, kann man hier einfach neue Felder hinzufügen, ist es zu leicht, dann löscht man welche heraus. Wie wir später sehen werden beziehen sich die Felder immer auf ein „Target“-Feld, und summieren sich mit diesem.

###screeenshot###

7.3 Die schon bekannten Funktionen.

def printfield():  
 os.system('cls' if os.name == 'nt' else 'clear')  
 for i in range(SIZE):  
 if i%(ROWS)==0:  
 print("")  
 if field[i]==0:  
 print("[ ]",end="")  
 else:  
 print("[",field[i],end=" ]")  
  
 print("")  
  
  
def endscreen():  
 os.system('cls' if os.name == 'nt' else 'clear')  
 visuals.score\_p()  
 visuals.num\_con(attempts\_and\_score[1])  
 with open("Highscore\_fti.txt","r") as highscore:  
 h\_score=highscore.read()  
 if int(h\_score) < (attempts\_and\_score[1]):  
 with open("Highscore.txt", "w") as highscore:  
 highscore.write(str(attempts\_and\_score[1]))  
 visuals.hscore\_p()  
 visuals.num\_con(attempts\_and\_score[1])  
 else:  
 visuals.hscore\_p()  
 visuals.num\_con(int(h\_score))  
 attempts\_and\_score[2]=0

Sowohl printfield, als auch endscreen() wuden in den Punkten (??????) schon behandelt. Die endscreen() funktion ist nur leicht abgeändert und ihr hauptunterrschied ist, dass sie am Ende den Wert 0 an die zweite Stelle der Liste „attempts\_and\_score“ setzt, was so viel bedeute, wie „Es wird nicht mehr gespielt“.

7.3. start()

def start():  
 position=STARTING\_POSITION  
 begining=True  
 searcher=STARTING\_POSITION  
 attempts\_and\_score[0]=ATTEMPTS-1  
 attempts\_and\_score[1]=0  
 attempts\_and\_score[2]=1  
 while attempts\_and\_score[2]!=0:  
 if begining == True:  
 face(position)  
 printfield()  
 begining=False  
 target=targetposition()  
  
 x=searching(searcher,target)  
 if x==100:  
 begining=True  
 else:  
 searcher+=x  
 time.sleep(0.2)

Diese ist auch der des vorherigen ähnlich. Die Grundfunktion ist wie folgt. Zuerst werden alle Variablen angepasst, inklusive der „beginning“ Variable, die auf „True“ gesetzt wird (die Suche fängt also an). Dann wird sich in eine while Schleife begeben, die erst endet, wenn die zweite Stelle, der „attempt\_and\_score“ Liste auf 0 ist. Zur Erinnerung, dies geschieht im Endscreen, also wenn das Spiel beendet ist.

Als nächstes kommt die face() Funktion, die den Namen von „face the enemy“ und ähnlichem ableitet, was in diesem Fall heißt, dass der Spieler den zu findenden Block an sich vorbeiziehen sieht. Dann wird mit der targetposition() ein zufälliges Feld für den Block ausgewählt. Aber all dies geschieht nur in der jeweils ersten Runde, also, wenn der Spieler startet, oder den Block zuvor gefunden hat. Wir stellen sicher, dass wir nicht mehr in diese Abfrage kommen, in dem wir die Variable „beginning“ auf „False“ setzten.

Nun kommt die searching() Funktion, also die Suchfunktion, die die Aufgabe hat den Spieler durch das Feld steuern zu lassen, also nach dem Block suchen zu lassen.

Wurde der Block gefunden, sollte die x Variable den Wert 100 haben und die „beginning“ Variable wird auf „True“ gesetzt, was bedeutet, dass ein neuer Block in der nächsten Instanz der while Schleife, gefunden werden muss.

Wenn dies nicht der Fall ist, also wird „x“, also das neue Feld hinzuaddiert. (Bei falschem Raten ist x 0, da der Spieler sich nicht bewegt, sondern nur ausgewählt hat).

Das ist auch schon alles. Nun sehen wir uns an, was die genannten Funktionen im Detail machen.

7.4. face()

def face(position):  
 r=random.choice([1,-1,ROWS,-ROWS]) #directions  
 n=0  
 while n <=3:  
 for x in block:  
 field[position+x+r\*n]=1  
 printfield()  
 time.sleep(0.5)  
 for x in range(SIZE):  
 field[x]=0  
 n+=1

Das Ziel ist hier, den Block sichtbar zu machen, aber natürlich nicht auf den zu ratenden Feldern, so leicht will man es den Spieler auch wieder nicht machen. In Wirklichkeit ist das Gegenteil der Fall, denn der Block bewegt sich in eine zufällige Richtung, die aber NICHTS mit dem eigentlichen Feld, an dem er sich befindet zutun hat. Dies wurde aus Spaßgründen implementiert, denn wenn der Spieler nicht gerade diese Dokumentation gelesen hat, muss er die selbst herausfinden. Anders ausgedrückt wird eine Korrelation vorgetäuscht.

Der Block bewegt sich mit drei Schritten (oder auch Runden) von seiner ursprünglichen Position weg, was durch die Varaible „n“ dargestellt wird. „n“ ist hier aber auch die Zahl, die mit einem gewissen Faktor „r“ multipliziert wird und zu zum Block hinzuaddiert wird. „r“ steht für random und wählt aus der Liste( 1,-1,ROWS,-ROWS] per Zufall ein Element aus, wobei alle Elemente Richtuntungen entsprechen.

Durch die Funktion time.sleep(0.5) bleibt die Feldstruktur für eine halbe Sekunde bestehen, bis sich der block weiter in eine bestimmte Richtung bewegt.

7.5. targetposition()

def targetposition():  
 while True:  
 target=random.randint(0,SIZE-2)  
 if (target+1)%22 !=0 and target%22 !=0 and target-ROWS\*2>0:#looking for errors of boarders  
 return target

Diese Funktion ist recht simple zu erklären und gleicht einer, die im vorherigen Spiel zu finden war (Punk????). Es wird eine zufällig Zahl im Feld genommen und dann überprüft, ob es valid ist, dieser Zyklus dauert so lange, bis sich eine valide Zahl gefunden hat.

7.6. searching()

def searching(searcher,target):  
 field[searcher]=2  
 printfield()  
 while True:  
 key=keyboard.read\_key()  
 if key in ["a","d","s","w","enter"]:  
 field[searcher]=0  
 if key=="a":  
 return -1  
 if key=="d":  
 return 1  
 if key =="s":  
 return ROWS  
 if key=="w":  
 return -ROWS  
 if key=="enter":  
 win=winning(target,searcher)  
 if win==True:  
 return 100  
 else:  
 return 0

Diese Funktion gleicht der vorherigen (Punkt 4893284), wodurch ich nicht genauer darauf eingehen werde. Der einzige größere Unterschied ist der Ablauf bei, bei drücken der Entertaste. Wenn dies geschieht, wird die winning() Funktion gestartet, wenn diese sozusagen mit dem boolschen Wert „True“ zurückkommt, dann gibt die searching() Funktion den Wert 100 zurück, ansonsten wird der Wer 0 zurückgegeben. Was mit diesen Werten passiert haben wird in Punkt(0938489032) besprochen.

7.7 winning()

def winning(target,searcher):  
 if searcher in [m+target for m in block]:  
 visuals.found\_p()  
 attempts\_and\_score[0]=ATTEMPTS  
 attempts\_and\_score[1]+=1 #highscore  
 return True  
  
 else:  
 visuals.X\_p()  
 if attempts\_and\_score[0]==0:  
 endscreen()  
 attempts\_and\_score[0]-=1  
 return False

In dieser Funktion wird überprüft, ob der Spieler einen Punkt ausgewählt hat, auf dem sich der Block befindet oder nicht. Dies wird mit der if-Abfrage: if searcher in [m+target for n in block]: bewerkstelligt. Kurz: es wird geschaut, ob sich der Wert von der Variable „searcher“ in der dannach erstellten Liste befindet. Die Liste ist hierbei [m+target for m in block]. Diese wird mit einer for Liste befüllt, die alternativ auch so aussehen könnte:

l=[]

for m in block:

l.append(m+target)

Dies hat das gleiche Ergebnis, benötigt aber mehr Platz. Die Grundfunktion, nämlich von der Target-Position die übrigen Elemente des Blocks hinzuaddieren, bleibt jedoch gleicht.

Falls die Position des Spielers auf dem Block ist, wird visuals.found\_p(), ansonsten visuals.X\_p(). Was diese Im Detail machen sehen wir später.

visuals.found\_p()

#####screeenshot#####

visuals.X\_p().

######screenshot#####

Falls der Versuch gescheitert ist, so kommt der Spieler, wenn er in seinem 0. Versuch ist, in den Endscreen, ansonsten wird hat er einen Versuch weniger.

8. A Worm and its hunt for food

Der Spieler versetzt sich in diesem Spiel in die „Schuhe“ eines Wurms, der nach Nahrung sucht, der Wurm ist beinahe ausgehungert und am Anfang nur mehr so groß, wie seine Nahrung, aber nach jeder ergatterten Nahrung wird dieser Wurm um die länge der Nahrung größer.

8.1.

import keyboard  
import threading  
import random  
import os  
import visuals  
import time

Die Imports sind beinahe diesselben, nur ein Eintrag hat sich geändert, und zwar Threading.   
Threading ist für die gleichzeitige Ausführung von mehreren Teilen in einem Code da. In diesem Fall sind es zwei Operationen, die gleichzeitig laufen müssen. Zum einen die Abfrage der Tasten und zum anderen die Fortbewegung des Wurms. Das Spiel wäre viel zu leicht, wenn sich der Wurm nicht mit einer konstanten „Geschwindigkeit“ bewegen würde.

8.2 Variablen und Konstanten

SIZE = 242  
COLUMNS=22  
STARTING\_POINT=120  
global field  
global mover  
field = [0]\*(SIZE\*2)  
mover=[0]

Die Variablen sind den Programmen (punkt84923894) und (324324) ähnlich, die einzige Neuerung ist die die Liste „mover“, die für die das Speichern der Richtung zuständig ist.

8.3 Die bekannten Funktionen

def printfield():  
 os.system('cls' if os.name == 'nt' else 'clear')  
 for i in range(SIZE):  
 if i%(22)==0:  
 print("")  
 if field[i]==0:  
 print("[ ]", end="")  
 else:  
 print("[X", end="]")  
  
  
 print("")

def printfield():  
 os.system('cls' if os.name == 'nt' else 'clear')  
 for i in range(SIZE):  
 if i%(22)==0:  
 print("")  
 if field[i]==0:  
 print("[ ]", end="")  
 else:  
 print("[X", end="]")  
  
  
 print("")

def moving():  
 while True:  
 key=keyboard.read\_key()  
 if key in ["a","d","s","w"]:  
 if key=="w":  
 mover[0] = -22  
 if key == "s":  
 mover[0] = 22  
 if key == "a":  
 mover[0] = -1  
 if key == "d":  
 mover[0]=1

8.3.start()

def start():  
 threading.Thread(target=moving).start()  
 for i in range(SIZE):  
 field[i]=0  
 position = [STARTING\_POINT]  
 beginning=True  
 length=0  
 going=True

Zuallererst wird durch threading.Thread(target=moving).start() ein Thread gestartet. Dann wird das Feld „bereinigt“, falls zuvor schon gespielt wurde. Die Variable „beginning“ beschreibt, ob eine neue Runde startet. Die Variable „going“ beschreibt, ob das Spiel noch am Laufen ist, falls dies nicht der Fall ist, wird das Spiel abgebrochen.

while going!=False:  
 if beginning==True:  
 food(position)  
 beginning=False  
  
 going=bordercontrol(position,length)  
 printfield()  
 time.sleep(0.3)  
 position=positionupdate(position,length)  
 if position==False:  
 break  
 if field[position[0]]==2:  
 length+=1  
 lastpos=position[length-1]  
 position.append(lastpos)  
 food(position)  
  
 for pos in position:  
 field[pos]=1

Nun sehen wir uns die while Schleife genauer an. Wenn die Variable „beginning“ den boolschen(???) Wert True beinhaltet, dann heißt dies, dass das Spiel beginnt. Mit der food() Funktion wird dann ein zufälliges Feld, auf dem der Wurm sich nicht bewfindet mit einem Nahrungsstück befüllt. Sobald dies geschehen ist, wird die „beginning“ Variable auf „False“ gesetzt.

Dannach, oder falls die „beginning“ Variable „False“ ist kommt der nächste Teil an die Reihe.

Die boardercontrol() Funktion findet heraus, ob der Wurm aus dem Spielfeld wandern würde, falls dies geschieht wird der Endscreen ausgegeben.

Die positionupdate() Funktion sieht sich aktualisiert die Positionen des Wurms und schaut, ob er in sich selbst stoßen würde (falls dies passiert, wird der Endscreen ausgegeben) und gibt einen „False“ Wert zurück, was das Spiel abbrechen lässt.

if field[position[0]]==2:  
 length+=1  
 lastpos=position[length-1]  
 position.append(lastpos)  
 food(position)

Dies if Abfrage sorgt dafür, dass falls sich der Wurm mit dem Kopf auf einem Nahrungsstück befindet sich dieser um ein Feld erweitert.

for pos in position:  
 field[pos]=1

Und zuletzt aktualisieren wir das Feld mit all den neuen Positionen, sowohl des Wurms als auch des Nahrungsstücks.

Als nächsten wollen wir uns die einzelnen Funktionen genauer ansehen.

8.4 food()

def food(position):  
 foodposition = random.randint(0, SIZE-1)  
 while foodposition in position:  
 foodposition = random.randint(0, SIZE-1)  
 field[foodposition]=2

Die food() Funktion ist den Funktionen ()dsfkpodskfölsdköfl und dlkfjlsdjfldsjf der punkte flksdjfldlsf dsjfksdjfklj ähnlich. Sie wählt ein zufälliges Feld aus und überprüft, ob dieses noch nicht vergeben ist, so lange bis es ein neues gefunden hat. Auch hier wäre wie in Punkt djsflkjkds eine Abfrage, ob es überhaupt noch freie Felder gibt möglich, aber wie auch in punkt dsujfkosdjfkjds ist es derart unwahrscheinlich, dass dieser Fall eintritt, und wurde deshalb ausgelassen.

8.5 boardercontrol()

def bordercontrol(position,length):  
 if position[0] < 0 or position[0] > SIZE - 1:  
 endscreen(length)  
 return False  
  
  
 if mover[0]==1:  
 for next in range(1,COLUMNS):  
 if position[0]==22\*next:  
 endscreen(length)  
 return False  
 if mover[0]==-1:  
 for next in range(1,COLUMNS,2):  
 if position[0]==COLUMNS\*next+COLUMNS-1:  
 endscreen(length)  
 return False

Die Erklärung hier ist recht simple, denn die erste if-Abfrage hat den Sinn herauszufinden, ob der Kopf des Wurms über dem Feld ist und die nächsten zwei if-Abfragen schauen, ob der Wurm über die rechte oder linke Grenze wandert.

8.6 positionupdate()

def positionupdate(position,length):  
 if length>0:  
 field[position[length]]=0  
 for i in reversed(range(length)):  
 position[i+1] = position[i]

Mit der ersten if-Abfrage wir geschaut, ob die länge des Wurms größer als 0 ist. Falls dies der Fall sein sollte wird die letzte Stelle des Wurms(hinweis, die Länge des wurms beginnt bei 0) auf 0 gesetzt. Und mit der for-Schleife nehmen alle Glieder, außergenommen das erste, die Positionen der der vorherkommenden Glieder ein.

else:  
 field[position[0]]=0

Falls die Länge des Wurms 0 beträgt wird die Position einfach auf 0 gesetzt.

if field[position[0]+mover[0]]==1:  
 endscreen(length)  
 return False

Wenn die nächste Position des Wurm auf ein Feld stößt, dass schon mit einer 1 besetzt ist, als schon vom Wurm selbst besetzt ist, gibt die Funktion „False“ zurück.

else:  
 position[0]=position[0]+mover[0]

Falls der Wurm aber nicht in sich selbst stößt wird die Position aktualisiert, indem die letztgewählte Richtung (mover[0]) hinzuaddiert wird.

8.7 moving()

def moving():  
 while True:  
 key=keyboard.read\_key()  
 if key in ["a","d","s","w"]:  
 if key=="w":  
 mover[0] = -22  
 if key == "s":  
 mover[0] = 22  
 if key == "a":  
 mover[0] = -1  
 if key == "d":  
 mover[0]=1

Diese Funktion ähnelt den dkjfdsjfkjdskfjdsjflk dfkdsjfkljdsklfdl kfjkdsjfkl.Hat aber den Unterschied, dass sei als Thread aktiviert wird. Zur Erinnerung, ein Thread läuft parallel zum Hauptprogramm. Dies ist in diesem Fall angewendet worden, da man nicht auf die Richtungswahl des Spielers warten kann; der Wurm soll sich auch ohne zutun des Spielers in eine bestimmte Richtung begeben.

9. Chess

Schach, in diesem Fall „Chess“ (englischer Begriff für Schach) ist kfdjsfkjsdfjkldsjfdksfjdsjfkdsjfkjsdkfjkdsjfjdsjfksdjkfjdsjf

Es muss aber gesagt werden, dass es hier um ein Schachspiel vor 16?? handelt, also noch keine „en- passant(?)“ Zugmöglichkeit besitzt. Auch gibt es keine automatische Erkennung von Schach Matts(???), oder von „Schach‘s“(????). Somit muss bei einem Matt einfach das Spiel mit „backspace“ neu gestartet werden.

Das Programm besitzt jedoch automatische Erkennung von Richtigen und falschen Zügen. So kann beispielsweise kein Bauer, einen fremden Bauern im Feld vor sich „fressen“. Auch ist es nicht möglich eigene Figuren zu „fressen“ oder z.B. mit einem Turm diagonal zu gehen. Weiters besitzt das Spiel eine automatische Erkennung von der Farbe des berechtigten Zuges, so kann z.B. Schwarz nicht zwei Züge hintereinander machen und es werden auch Figuren automatisch entfernt. Auch muss noch die Funktion der Rochade hinzugefügt werden.

Die Züge werden mit einem Auswahlscursor gemacht, in dem man zuerst (mit wasd) auf das die gewünschte Figur geht und diese auswählt (Enter) und dann auf das gewünschte Feld, auf dem die Figur platziert werden soll, geht und wieder mit Enter bestätigt.

Dies hat jedoch zur Folge, dass das Programm eine weitaus erhöhte Komplexität aufweist, als es beispielsweise eine Texbasierte Eingabe (z.B E2 zu E3) hätte.

Wir befinden uns nun beim letzten Spiel. Wie in Punk(???) erklärt wurde wird hier weniger auf Details eingegangen, denn dies würde den Rahmen sprengen. Hier werden nur mehr die generellen Eigenschaften der Unterprogramme beschrieben, da die generellen Ideen in den Programmen zuvor schon geschildert wurden.

9.1 Libraries

import keyboard  
import time  
import os

Die Imports sind bei diesem Programm um einiges kürzer, da wir bei Schach keinen Zufall brauchen und es auch sonst ohne jegliche externen visuellen Einlagen auskommt.

9.2 Variablen und Konstanten

size = 242  
COLUMNS=22  
global board  
global board\_choice  
board=[0]\*64  
global Pawn\_w,Pawn\_b,Pawn\_start\_w,Pawn\_start\_b, Rook\_w, Rook\_b,Knight\_w, Knight\_b, Bishop\_w,Bishop\_b, Queen\_w, Queen\_b, King\_w, King\_b,white\_pieces,black\_pieces, mate\_and\_quit  
mate\_and\_quit=[0,0]  
Pawn\_w = [1, 9, 17, 25, 33, 41, 49, 57]  
Pawn\_b = [x + 5 for x in Pawn\_w]  
Rook\_w = [0, 56]  
Rook\_b = [7, 63]  
Knight\_w = [8, 48]  
Knight\_b = [15, 55]  
Bishop\_w = [16, 40]  
Bishop\_b = [23, 47]  
Queen\_w, Queen\_b, King\_w, King\_b = [24], [31], [32], [39] #Queens are lists because there can be more than one  
Pawn\_start\_w=list(Pawn\_w)  
Pawn\_start\_b=list(Pawn\_b)  
white\_pieces=[]  
black\_pieces=[]

In diesem Programm werden zwar die Libraries eingespart, dafür ist aber die Liste der (globalen) Variablen umso länger. Dies hat den Grund, wenngleich es auch zuerst paradox klingen mag, dass der Code um ein Vielfaches übersichtlicher ist, denn es müssen somit nicht immer alle Variablen in jedes Unterprogramm eingespeißt werden, sondern jedes Unterprogramm kann direkt auf alle Variablen zugreifen und diese verändern.

9.3. printfield

def printfield(b):  
 os.system('cls' if os.name == 'nt' else 'clear')  
 x=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0, 0, 0, 0, 0, 0, 0, b[0], b[1], b[2], b[3], b[4], b[5], b[6], b[7], 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, b[8], b[9], b[10], b[11], b[12], b[13], b[14], b[15], 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, b[16], b[17], b[18], b[19], b[20], b[21], b[22], b[23], 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, b[24], b[25], b[26], b[27], b[28], b[29], b[30], b[31], 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, b[32], b[33], b[34], b[35], b[36], b[37], b[38], b[39], 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, b[40], b[41], b[42], b[43], b[44], b[45], b[46], b[47], 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, b[48], b[49], b[50], b[51], b[52], b[53], b[54], b[55], 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, b[56], b[57], b[58], b[59], b[60], b[61], b[62], b[63], 0, 0, 0, 0, 0, 0, 0,  
 ]  
 for p in range(len(x)):  
 if p%22==0:  
 print()  
 if x[p]==0:  
 print("[ ]",end="")  
 else:  
 print("[",x[p],end=" ]")

Das printfield() Unterprogramm weist diesmal einen kleinen Unterschied auf, denn anstatt wie gewohnt direkt das Feld auszugeben muss hier zuerst das Schachfeld, dass die Dimensionen 8\*8 aufweist, auf die für den Tisch relevanten 22\*11 angepasst werden. Hierfür würde es eine Unzahl an Möglichkeiten geben, es wurde sich aber für eine Schleifenfrei Direktumwandlung entschieden, in der ein Feld zuerst mit 0en befüllt wird und dann visuell Punkt für Punkt das Schachfeld in die gwünschte Position eingetragen wird. Dannach wird, wie in Punkt kldfsjhkdslj und jdfjlkds das Feld ausgegeben.

9.4 start()

def start():  
 n=1  
 mate\_and\_quit = [0, 0]  
 Pawn\_w = [1, 9, 17, 25, 33, 41, 49, 57]  
 Pawn\_b = [x + 5 for x in Pawn\_w]  
 Rook\_w = [0, 56]  
 Rook\_b = [7, 63]  
 Knight\_w = [8, 48]  
 Knight\_b = [15, 55]  
 Bishop\_w = [16, 40]  
 Bishop\_b = [23, 47]  
 Queen\_w, Queen\_b, King\_w, King\_b = [24], [31], [32], [39]  
 while mate\_and\_quit[1]==0:  
 board\_update()  
 printfield(board)  
 move\_choice(n)  
 n+=1

Die Variablen werden hier erneut definiert, wenn dies nicht der Fall wäre, würde das Programm einfach den letzten Zustand beibehalten und nicht in ein neues leeres Spiel wechseln. Wenn das Schachspiel als einzelnen Programm existieren würden, dann würde dies nicht notwendig sein, da es sich aber um ein Spielekomplex handelt ist dies nötig.

Der Ablauf der while Schleife ist relativ kurz, da sich die ganze Komplexität in den Unterprogrammen befindet.

9.5. boardupdate

def board\_update():  
 white\_pieces.clear()  
 black\_pieces.clear()  
  
 for n in range(64):  
 board[n]=0  
  
 for piece in Pawn\_w:  
 board[piece]="1"  
 white\_pieces.append(piece)  
 for piece in Pawn\_b:  
 board[piece]="2"  
 black\_pieces.append(piece)  
 for piece in Rook\_w:  
 board[piece]="5"  
 white\_pieces.append(piece)  
 for piece in Rook\_b:  
 board[piece]="6"  
 black\_pieces.append(piece)  
 for piece in Knight\_w:  
 board[piece]="3"  
 white\_pieces.append(piece)  
 for piece in Knight\_b:  
 board[piece]="4"  
 black\_pieces.append(piece)  
 for piece in Bishop\_w:  
 board[piece]="7"  
 white\_pieces.append(piece)  
 for piece in Bishop\_b:  
 board[piece]="8"  
 black\_pieces.append(piece)  
 for piece in Queen\_w:  
 board[piece] = "Q"  
 white\_pieces.append(piece)  
 for piece in Queen\_b:  
 board[piece] = "W"  
 black\_pieces.append(piece)  
 board[King\_w[0]] = "K"  
 white\_pieces.append(King\_w[0])  
 board[King\_b[0]] = "K"  
 white\_pieces.append(King\_b[0])

Ziel dieses Unterprogramms ist es das Feld zu aktualiseren, dies wird mit Hilfe einer Armada an for-Schleifen durchgeführt. Andere Algorithmen wurden zuerst Versucht, blieben aber nich Bug-frei, so wurde sich für diese, wenn auch unschönere, Lösung entschieden.

9.6 move\_choice()

def move\_choice(n):  
 position=COLUMNS  
 selected=[]  
 correct=False  
 while True:  
 time.sleep(0.2)  
 key = keyboard.read\_key()  
 if key in ["a", "d", "s", "w","enter","backspace"]:  
 if key == "a" and position>0 and position%8!=0:  
 position=square\_selection(position-1)  
 if key == "d" and position<63 and position%8!=7:  
 position=square\_selection(position+1)  
 if key == "s" and position<56:  
 position=square\_selection(position+8)  
 if key == "w" and position>8:  
 position=square\_selection(position-8)  
 if key == "enter":  
 selected.append(position)  
 if len(selected)==2:  
 correct=check(selected,n)  
 if correct==True:  
 board\_update()  
 break  
 selected = []  
 if key=="backspace":  
 mate\_and\_quit[1]=1  
 break

Dieses Unterprogramm agiert ähnlich denen der Punkte jdfksdjf und jsdkjflkdsjf. Die einzigen Unterschiede sind zum einen die Auswahl, die beim drücken der Enter Taste ausgeführt wird und das beenden des Programmes, dass bei einem Backspace erfolgt. Wie schon in Punkt djskfjklsd angedeutet beinhaltet das Spiel keinen Mechanismus, der überprüfen würde, ob ein Schach-Matt erfolgt ist, oder nicht, weswegen ein solcher „mechanisches“ notwendig ist.

9.8 check()

def check(selected,n):  
  
 if n%2!=0 and friendly\_fire(selected[1],n)==False:  
  
 if selected[0] in Pawn\_w:  
 if Pawn(selected,Pawn\_w,Pawn\_start\_w,1,n)==True:  
 return True  
 elif selected[0] in Rook\_w:  
 if Rook(Rook\_w,selected,n)==True:  
 return True  
 elif selected[0] in Bishop\_w:  
 if Bishop(Bishop\_w,selected,n)==True:  
 return True  
 elif selected[0] in Knight\_w:  
 if Knight(Knight\_w,selected,n)==True:  
 return True  
 elif selected[0] in Queen\_w: #because a Queen is just a combination of Rook and a Bishop  
 if Bishop(Queen\_w,selected,n)==True:  
 return True  
 elif Rook(Queen\_w,selected,n)==True:  
 return True  
  
 elif selected[0] == King\_w[0]:  
 if selected[1] == 0 and 0 in Rook\_w:  
 i=Rook.index(0)  
 Rook[i]=24  
 King\_w[0]=16  
 if Capture(King\_w,selected,n)==True:  
 return True  
 else:  
 King\_w[0]=selected[1]  
 return True  
 else:  
  
 return False  
  
 if n%2 ==0 and friendly\_fire(selected,n)==False: #move black  
 if Pawn(selected, Pawn\_b, Pawn\_start\_b,-1,n)==True:  
 return True  
 elif selected[0] in Rook\_b:  
 if Rook(Rook\_b,selected,n)==True:  
 return True  
 elif selected[0] in Bishop\_b:  
 if Bishop(Bishop\_b,selected,n)==True:  
 return True  
 elif selected[0] in Knight\_b:  
 if Knight(Knight\_b, selected, n) == True:  
 return True  
 elif selected[0] in Queen\_b: #because a Queen is just a combination of Rook and a Bishop  
 if Bishop(Queen\_b,selected,n)==True:  
 return True  
 elif Rook(Queen\_b,selected,n)==True:  
 return True  
 elif selected[0] == King\_b[0]:  
 if Capture(King\_b,selected,n)==True:  
 return True  
 else:  
 King\_b[0]=selected[1]  
 return True  
  
 else:  
 return False

9.9 friendly\_fire()

def friendly\_fire(square,n):  
 if n%2!=0:  
 if square in white\_pieces:  
 return True  
 else:  
 return False  
 elif n%2==0:  
 if square in black\_pieces:  
 return True  
 else:  
 return False

Das friendly fire Unterprogramm schaut, ob es sich bei dem ausgewählten Feld um eines mit einer farbgleichen Figur besetzten handelt. Friendly fire ist ein Begriff aus dem englischen, der dafür verwendet wird zu sagen, dass ein Teammitglied angegriffen werden kann, in diesem Fall wurde er ironisch verwendet.

9.9 Die Figuren

def Pawn(selected,pawn,pawn\_start,factor,n):  
 if selected[1]==selected[0]+factor and board[selected[1]]==0:  
 try: n=pawn.index(selected[0])  
 except: return False  
 pawn[n] = selected[1]  
 return True  
 elif selected[1] == selected[0] + 2\*factor and selected[0] in pawn\_start and board[selected[1]]==0:  
 x = pawn.index(selected[0])  
 pawn[x] = selected[1]  
 return True  
 elif (selected[1]==selected[0]+9\*factor or selected[1]==selected[0]-7\*factor):  
 if n%2!=0:  
 if Capture(Pawn\_w, selected, n) == True:  
 return True  
 else:  
 return False  
 else:  
 if Capture(Pawn\_b,selected,n)==True:  
 return True   
 else:  
 return False  
 else:  
 return False  
  
def Rook(rook,selected,n):  
 legal\_move=False  
 if (-7<=(selected[1]-selected[0])<=7) and selected[1]//10==selected[0]//10: #is it a straight row and are they in the same row?  
  
 for x in range(selected[0]+1,selected[1]): #is there a piece in the path?  
 if board[x] != 0:  
  
 return False  
 legal\_move=True  
  
 elif (selected[1]-selected[0])%8==0:  
  
 y=(selected[1]-selected[0])//8 +1  
  
 for x in range(1,y): #is there a piece in the path?  
 if board[selected[0]+x\*8]!=0:  
  
 return False  
 legal\_move=True  
  
 if n%2!=0:  
 if Capture(Rook\_w,selected,n)==True:  
 return True  
 elif n%2==0:  
 if Capture(Rook\_b, selected, n) == True:  
 return True  
 if legal\_move==True:  
 x = rook.index(selected[0])  
 crook=rook.copy()  
 rook[x] = selected[1]  
 if Check\_and\_Mate(1)==False:  
 return True  
 else:  
 rook=crook.copy()  
 return True  
 return False  
  
  
def Bishop(bishop, selected,n):  
 legal\_move=False  
 if (selected[1] - selected[0]) % 7 == 0:  
  
 s=selected[0]  
 for x in range(1,(selected[1]-selected[0])//7):  
 s+=7  
 print(s)  
 if board[s]!= 0:  
 return False  
  
 legal\_move=True  
  
 elif (selected[1] - selected[0]) % 9 == 0:  
 s=selected[0]  
 for x in range(1,(selected[1]-selected[0])//9):  
 s+=9  
 if board[s]!= 0:  
 return False  
 legal\_move=True  
 if n % 2 != 0:  
 if Capture(Bishop\_w, selected, n) == True:  
 return True  
 elif n % 2 == 0:  
 if Capture(Bishop\_b, selected, n) == True:  
 return True  
 if legal\_move==True:  
 x = bishop.index(selected[0])  
 bishop[x] = selected[1]  
 return True  
  
 return False  
  
  
def Knight(knight,selected,n):  
 if selected[1]==selected[0]+10 or selected[1]==selected[0]-10 or selected[1]==selected[0]+6 or selected[1]==selected[0]+6 or selected[1]==selected[0]+17 or selected[1]==selected[0]-17 or selected[1]==selected[0]+15 or selected[1]==selected[0]-15:  
 if n % 2 != 0:  
 if Capture(Knight\_w, selected, n) == True:  
 return True  
 elif n % 2 == 0:  
 if Capture(Knight\_b, selected, n) == True:  
 return True  
 x = knight.index(selected[0])  
 knight[x] = selected[1]  
 return True  
 return False

Die Unterprogramme der Figuren beinhalten all möglichen Züge, die diese Fgur machen kann. Hierbei wird auch geschaut, ob in den Feldern dazwischen auch keine Figur sich befindet (der Springer ist ausgenommen).

capture()

def Capture(piece,selected,n):  
 if n%2!=0: #capture the black piece, ... if there is one  
 if selected[1] in Pawn\_b:  
 Pawn\_b.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Rook\_b:  
 Rook\_b.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Knight\_b:  
 Knight\_b.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Bishop\_b:  
 Bishop\_b.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Queen\_b:  
 Queen\_b.remove(selected[1])  
 x = piece.index(selected[0])  
 piece[x] = selected[1]  
 return True  
  
 else:  
 return False  
  
 if n%2==0: #capture the white piece, ... if there is one  
 if selected[1] in Pawn\_w:  
 Pawn\_w.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Rook\_w:  
 Rook\_w.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Knight\_w:  
 Knight\_w.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Bishop\_w:  
 Bishop\_w.remove(selected[1])  
 x=piece.index(selected[0])  
 piece[x]=selected[1]  
 return True  
 elif selected[1] in Queen\_w:  
 Queen\_w.remove(selected[1])  
 x = piece.index(selected[0])  
 piece[x] = selected[1]  
 return True  
 else:  
 return False

Das Capture Unterprogramm sieht sich a, ob sich eine Figur auf dem gewünschten Feld befindet, entfernt diese dann.

10.1 visuals

Das visuals Modul wurde eigens für diese Diplomarbeit entwickelt und beinhaltet alle visuellen Elemente, die extern hinzugefügt werden. Ein Beispiel, wäre hierfür die Zeit, die dann im Hauptprogramm standardgemäß ausgegeben wird. Sie beinhalten aber auch alle visuellen Repräsentationen von den Zahlen 0-9, aus denen sich alle anderen Zahlen erstellen lassen, aber auch die Logos der Spiele und die Score-Anzeigen.

10.2 Listen

import os  
import time  
  
zero=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
one=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
two=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
three=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
  
  
four=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
  
five=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
six=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
  
seven=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
  
eight=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
nine=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
score=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,1,1,1,0,1,1,0,1,1,1,0,1,1,1,0,1,1,1,0,0,  
 0,0,0,0,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,0,0,0,  
 0,0,0,1,0,0,1,0,0,1,0,1,0,1,1,1,0,1,1,1,0,0,  
 0,0,1,0,0,0,1,0,0,1,0,1,0,1,1,0,0,1,0,0,0,0,  
 0,0,1,1,1,0,1,1,0,1,1,1,0,1,0,1,0,1,1,1,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
hscore=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 1,0,1,0,1,1,1,0,1,1,0,1,1,1,0,1,1,1,0,1,1,1,  
 1,0,1,0,0,0,1,0,1,0,0,1,0,1,0,1,0,1,0,1,0,0,  
 1,1,1,0,0,1,0,0,1,0,0,1,0,1,0,1,1,1,0,1,1,1,  
 1,0,1,0,1,0,0,0,1,0,0,1,0,1,0,1,1,0,0,1,0,0,  
 1,0,1,0,1,1,1,0,1,1,0,1,1,1,0,1,0,1,1,1,1,1,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
chess=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
  
gts=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,  
 0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,  
 0,0,0,0,0,0,1,1,1,1,0,0,1,1,1,1,1,0,0,0,0,0,  
 0,0,0,0,0,0,1,1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,1,1,1,1,0,0,1,1,1,1,1,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
fti=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
awaihff=[0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,  
 0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,  
 0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,  
 0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
  
X=[0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,1,0,  
 0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,  
 0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,  
 0,0,0,0,1,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,  
 0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,  
 0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,  
 0,0,0,0,1,0,1,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,  
 0,0,0,1,0,0,1,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,  
 0,0,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1,0,0,  
 0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]  
  
found=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,  
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

All diese Listen beinhalten visuelle Repräsentation von Zahlen, Logos und Bestätigungen. Zero bis nine sind die Repräsentationen der Zahlen 0 bis 9, die sowohl für die Scores, als auch für die Zeitausgabe. Als nächstes sind die Logos, für die Spiele Chess, awaihff, gts, fti an der Reihe, wobei die letzten vier die Kürzel der zuvor dokumentierten Spiele sind. X ist die Ausgabe für das nicht treffen des Blocks im Spiel „Find the Invisible“, und „found“ die Bestätiogung über den gefundenen Block im selben Spiel.

10.1 Unterprogramme

Es werden die Projektergebnisse dokumentiert.

* Grundkonzept
* Theoretische Grundlagen
* Praktische Umsetzung
* Lösungsweg
* Alternativer Lösungsweg
* Ergebnisse inkl. Interpretation

Weitere Anregungen:

* Fertigungsunterlagen
* Testfälle (Messergebnisse…)
* Benutzerdokumentation
* Verwendete Technologien und Entwicklungswerkzeuge

Beispiel zu oberer Definition:

## Grundkonzept

## Theoretische Grundlagen

### ….

## Praktische Umsetzung

….

# Erklärung der Eigenständigkeit der Arbeit

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe. Meine Arbeit darf öffentlich zugänglich gemacht werden, wenn kein Sperrvermerk vorliegt.

|  |  |  |
| --- | --- | --- |
| Ort, Datum |  | Verfasser 1 |

|  |  |  |
| --- | --- | --- |
| Ort, Datum |  | Verfasserin 2 |

…

# Abbildungsverzeichnis

[Abbildung 1: XYZ vi](#_Toc415201720)

[Abbildung 2:ABC VI**.**](#_Toc415201721)

# Tabellenverzeichnis

[Tabelle 1: Arbeitsaufstellung IV](#_Toc415201490)

# Literaturverzeichnis

(in alphabetischer Reihenfolge geordnet nach Nachname)

https://www.python.org/doc/essays/blurb/

<https://www.geeksforgeeks.org/python-ways-to-flatten-a-2d-list/> (2D liste zu 1D liste)

https://docs.python.org/3/library/random.html

Beispiele:

(Übernommen aus dem Leitfaden des BMBF Reife- und Diplomprüfungen März 2014)

1. Werke eines Autors:

Nachname, Vorname: Titel. Untertitel. - Verlagsort: Verlag, Jahr.

Nachname, Vorname: Titel. Untertitel. Auflage - Verlagsort: Verlag, Jahr.

Beispiele:

Sandgruber, Roman: Bittersüße Genüsse. Kulturgeschichte der Genußmittel. – Wien: Böhlau, 1986.

Messmer, Hans-Peter: PC-Hardwarebuch. Aufbau, Funktionsweise, Programmierung. Ein Handbuch nicht nur für Profis. 2. Aufl. - Bonn: Addison-Wesley, 1993.

1. Werke mehrerer Autoren:

Nachname, Vorname; Nachname, Vorname; Nachname, Vorname: Titel. Untertitel. Auflage - Verlagsort: Verlag, Jahr.

Beispiel:

Bauer, Leonhard; Matis, Herbert: Geburt der Neuzeit. Vom Feudalsystem zur Marktgesellschaft. - München: Deutscher Taschenbuch Verlag, 1988.

1. Sammelwerke, Anthologien, CD-ROM mit Herausgeber:

Nachname, Vorname (Herausgeber): Titel. Untertitel. Auflage - Verlagsort: Verlag, Jahr. Nachname, Vorname: Titel. Untertitel. In: Nachname, Vorname (Herausgeber): Titel. Untertitel. Auflage - Verlagsort: Verlag, Jahr.

Beispiele:

Popp, Georg (Hg.): Die Großen der Welt. Von Echnaton bis Gutenberg. 3. Aufl. - Würzburg: Arena, 1979.

Killik, John R.: Die industrielle Revolution in den Vereinigten Staaten. In: Adams, Willi Paul (Hg.): Die Vereinigten Staaten von Amerika. Fischer Weltgeschichte Bd. 30. - Frankfurt am Main: Fischer Taschenbuch Verlag, 1977. Killy, Walther (Hg.): Literatur Lexikon. Autoren u. Werke deutscher Sprache. – München: Bertelsmann, 1999. (Digitale Bibliothek, 2)

1. Mehrbändige Werke:

Nachname, Vorname: Titel. Bd. 3 - Verlagsort: Verlag, Jahr.

Beispiel:

Zenk, Andreas: Leitfaden für Novell NetWare. Grundlagen und Installation. Bd. 1 - Bonn: Addison Wesley, 1990.

1. Beiträge in Fachzeitschriften, Zeitungen:

Nachname, Vorname des Autors des bearbeiteten Artikels: Titel des Artikels. In: Titel der Zeitschrift, Heftnummer, Jahrgang, Seite (eventuell: Verlagsort, Verlag).

Beispiel:

Beck, Josef: Vorbild Gehirn. Neuronale Netze in der Anwendung. In: Chip, Nr. 7, 1993, Seite 26. - Würzburg: Vogel Verlag.

6. CD-ROM-Lexika

Beispiel:

Encarta 2000 - Microsoft 1999.

1. Internet:

Nachname, Vorname des Autors: Titel. Online in Internet: URL: www-Adresse, Datum. (Autor und Titel wenn vorhanden, Online in Internet: URL: www-Adresse, Datum auf jeden Fall)

Beispiel:

Ben Salah, Soia: Religiöser Fundamentalismus in Algerien. Online im Internet: URL: »http://www.hausarbeiten.de/cgi-bin/superRD.pl«, 22.11.2000. Der Weg zur Doppelmonarchie. Online in Internet: URL: http://www.parlinkom.gv.at/pd/doep/d-k1-2. htm, 22.11.2000.

1. Firmenbroschüren, CD-ROM:

Werden Inhalte von Firmenunterlagen verwendet, dann ist ebenfalls die Quelle anzugeben.

Beispiel:

Digitale Turbinenregler. Broschüre der Firma VOITH-HYDRO GmbH, 2012.

9. Abbildungen, Pläne Werden Abbildungen aus einer fremden Quelle [z.B. Download, Scannen] in die Diplomarbeit eingefügt, so ist unmittelbar darunter die Quelle anzugeben:

Beispiel:

Abb. 1: Digitaler Turbinenregler [ANDRITZ HYDRO]

10. Persönliche Mitteilungen:

Beispiel:

Persönliche Mitteilung durch: König, Manfred: Kössler GmbH Turbinenbau am 8. März 2013.

# Abkürzungs- und Symbolverzeichnis

Anhang

# A1 Pflichtenheft (OPTIONAL)

Zur Umsetzung des Projektzieles werden messbare Kriterien formuliert.

A2 Schlussfolgerung / Projekterfahrung

A3 Projektterminplanung

Screenshots der MS Project-Datei. Die Ausgabe muss lesbar sein (eventuell auf mehrere Bilder verteilen). Insbesondere ist darauf zu achten, dass die Zeitachse und die Vorgangsachse auf jedem Bild sichtbar sind! Es muss nicht MS-Project verwendet werden!

Projektbalkenplan (Gantt-Diagramm)

Excel

## A4 Arbeitsnachweis Diplomarbeit

Dieser erfolgt durch ständige Aufzeichnungen der Schüler im Projekttagebuch.

Für jeden Projektmitarbeiter wird eine Tabelle gemäß Muster ausgefüllt. In dieser Aufzeichnung werden auch die Unterrichtsprojektanteile, die in die Arbeit eingeflossen sind aufgezeigt.

Tabelle 1: Arbeitsaufstellung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | | | | |
| **Datum** | **Uhrzeit** | **Stunden**  **nn:nn** | **Beschreibung** | **Betreuer** |
| 01.11.2020 | 08:00–11:30 | 2 | T eambesprechung |  |
|  |  | 3 | Teambesprechung |  |
|  |  | 2 | Teambesprechung |  |
|  |  | 2 | Planung eines Tetris-ähnlichen Progamms (in C) |  |
|  |  | 2 | Beginn des Tetris-ähnlichen Programms (in C) |  |
|  |  | 2 | Teambesprechung |  |
|  |  | 3 | Programmierung des Tetris-ähnlichen Programms |  |
|  |  | 6 | Programmierung des Tetris-ähnlichen Programms |  |
|  |  | 2 | Recherche Threads |  |
|  |  | 2 | Programmierung des Tetris-ähnlichen Programms |  |
|  |  |  |  |  |
|  |  | 2 | Testen und Bugfixen des Tetrisähnlichen Programms |  |
|  |  | 3 | Beginn Tetrisähnliches Programm (Python) |  |
|  |  | 4 | Programmierung Tetrisähnlichnesdfdf |  |
|  |  | 4 | Programmierung Tetris-ähnlliches Programm |  |
|  |  | 2 | Teambesprechung |  |
|  |  | 2 | Testenb Tetrisähnliches Programm |  |
|  |  | 1 | Keyboard-Modul Recherche |  |
|  |  | 1 | Überlegen und Skizzieren von weiteren Programmen |  |
|  |  | 5 | Programmierung von Can\_you\_find\_the\_way? |  |
|  |  | 3 | Programmierung von Can\_you\_find\_the\_way? |  |
|  |  | 5 | Programmierung von Can\_you\_find\_the\_way? |  |
|  |  | 2 | Teambesprechuchung |  |
|  |  | 2 | Testen Can\_you\_\_ |  |
|  |  | 4 | Bugfixing |  |
|  |  | 3 | Snake |  |
|  |  | 2 | Snake |  |
|  |  | 2 | Teambesprechung |  |
|  |  | 2 | Teambesprechung |  |
|  |  | 5 | Snake |  |
|  |  | 6 | Snake |  |
|  |  | 2 | Recherche Neopixel Modul |  |
|  |  | 8 | Einrichtung Raspberry Pi |  |
|  |  | 4 | Testen an 5 LEDS |  |
|  |  | 3 | Realumsetzung Snake |  |
|  |  | 2 | Realumsetzung Snake |  |
|  |  | 3 | Planung Lösung des Tisches |  |
|  |  | 6 | Blocks |  |
|  |  | 2 | Blocks |  |
|  |  | 1 | Blocks |  |
|  |  | 1 | Nummern zu Matrixindex |  |
|  |  | 3 | Überarbeitung Can you |  |
|  |  | 2 | Zeitanzeige |  |
|  |  | 1 | Zeitanzeige Testen |  |
|  |  | 1 | Zeitanzeige real |  |
|  |  | 2 | Programmieren von Schach |  |
|  |  | 4 | Schach |  |
|  |  | 5 | Schach |  |
|  |  | **SUMME** |  |  |

A5 Datenblätter (OPTIONAL)

Meist sind die Datenblätter sehr umfangreich, daher werden im Anhang nur die notwendigen Bereiche dargestellt, auf der CD wird das vollständige Datenblatt gespeichert

A6 Technische Zeichnungen (OPTIONAL)

***Abzugeben sind:***

1 gebundene Dokumentationen mit Deckblatt (Format: A4) – für die Bibliothek

(die HTL Bindung ist zu verwenden!!)

1 Korrekturversion (Form und Aussehen mit Projektbetreuer vereinbaren)

1 Version für die Firma (Optional – mit Betreuer vereinbaren)

2 CDs mit allen Unterlagen (Word, Bilder, Code…)