Snake

Dokumentation

Inhalt

[Informieren 4](#_Toc86156270)

[Planen 7](#_Toc86156271)

[Entscheiden 8](#_Toc86156272)

[Realisieren 9](#_Toc86156273)

[Kontrollieren 10](#_Toc86156274)

[Auswerten 11](#_Toc86156275)

# Informieren

Noch bevor wir uns über unser Projekt und dessen Ablauf Informieren konnten, mussten wir uns natürlich entscheiden, was wir überhaupt machen sollen. Wir haben uns zusammengesetzt und zuerst darüber nachgedacht, in welche Richtung unser Projekt gehen soll. Schnell haben wir uns dafür entschieden, dass unser Projekt etwas mit Programmieren zu tun haben sollte. Zu dieser Zeit, war uns bereits auch klar, dass unser Informatikprojekt ein Spiel werden soll. Wir haben uns ein paar simple Spiele herausgesucht und getestet. Dabei ist herausgekommen, dass das Snake Spiel das wohl spannendste zum Programmieren ist.

Zum Umfang des Projekts sollte nicht nur das Programmieren, sondern auch das Planen, das Finden der richtigen Technik und das Sammeln von Infos und Ressourcen gehören.

* **Das Spiel selber soll eine mit C# programmierte Desktopapplikation werden, welche das eher bekannte Snake Game beinhaltet. Also ein Spielfeld, auf welchem man eine Schlange herumnavigieren kann und sie beim Einsammeln von «Punkten» immer länger wird. Das Ziel ist es nicht mit sich selber zu kollidieren, was je länger die Schlange wächst immer schwieriger wird.**

Möglich soll dabei das Einstellen von drei verschiedenen Schwierigkeitsstufen, sowie das navigieren der Schlange per Pfeiltasten sein. Je nach Schwierigkeit werden die Geschwindigkeit und die Hindernisse auf dem Feld angepasst. Um uns aber einen gewissen Zeitpuffer zu verschaffen, sollten die Hindernisse und die Levels nur optional sein.

Mit dem Projektauftrag waren nun die meisten Fragen zum Projekt selber Geklärt. Jedoch verblieben noch ein paar wenige organisatorische Fragen, welche gerade für ein Projekt im Team sehr entscheidend sein können:

* ***Wie soll alles verwaltet werden? [A]*** -notwendig um Zeitplan einzuhalten-



-Da wir zu zweit an einem Projekt arbeiten, müssen wir unsere Dateien und Fortschritte richtig abspeichern. So, dass wir beide darauf Zugriff haben und uns nicht gegenseitig in den Weg kommen, während wir am Arbeiten sind.

* ***Wer kann was? [B]*** -vereinfacht den Prozess sehr-

-Bei unserer Arbeit, wollen wir beide so viel wie möglich mitarbeiten. So können wir auch beide etwas Lernen. Doch wer kann schon was und was können beide nicht und muss noch nachgeschlagen werden?

* ***Wie organisieren wir uns? [B]*** -vereinfacht den Prozess sehr-

-Nachdem wir geklärt haben wer welche Fähigkeiten hat, können wir unter uns bestimmte und massgeschneiderte Aufgaben verteilen. So kommen wir am besten voran, da jeder das macht was er am besten kann.

Um die Wichtigkeit dieser Fragen zu bestimmen, haben wir sie, wie man es schon an den Buchstaben in den eckigen Klammern sehen kann, nach dem bekannten Eisenhower-Prinzip geordnet. So konnten wir deren Wichtigkeit gut bewerten und um das ganze noch ein bisschen logischer zu machen, haben wir uns noch kleine Kommentare zu den Fragen ausgedacht, welche aufzeigen, warum die Fragen wichtig sind.

Um nun diese Fragen zu klären, haben wir uns verschiedenste Lösungen ausgedacht, und diese dann mit einer simplen Entscheidungsmatrix verglichen. Herausgekommen sind dabei die folgenden Antworten:

***Wie soll alles verwaltet werden? [A] «P1»***

|  |  |  |  |
| --- | --- | --- | --- |
| Kriterien: | Gewichtung: | GitHub | Auf einem Pc |
| Komplexität | 5 | 2 | 7 |
| Verschiedene Geräte | 8 | 9 | 1 |
| Gesamtwert | 13 | 11 | 8 |
| Score |  | 0.8 | 0.6 |

~Dateimanagement über GitHub

***Wer kann was? [B] «P2»***

|  |  |  |  |
| --- | --- | --- | --- |
| Kriterien: | Gewichtung: | Gioele | Gian-Luca |
| VS | 8 | 5 | 6 |
| C# | 10 | 2 | 5 |
| GUI | 6 | 1 | 5 |
| Gesamtwert | 24 | 8 | 16 |
| Score |  | 0.3 | 0.6 |

~ Gioele: Grundkenntnisse C#

~Gian-Luca: Mittlere C# Kenntnisse

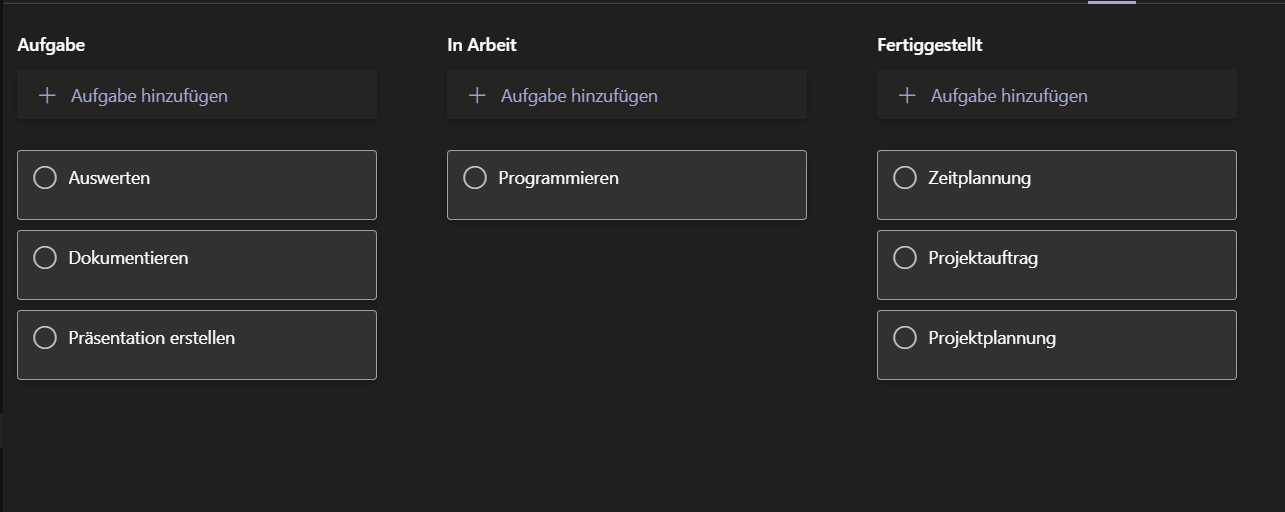
***Wie organisieren wir uns? [B] «P3»***

|  |  |  |  |
| --- | --- | --- | --- |
| Kriterien: | Gewichtung: | Teilung | Zusammen |
| Lerneffekt | 10 | 3 | 10 |
| Geschwindigkeit | 8 | 10 | 4 |
| Gesamtwert | 18 | 13 | 14 |
| Score |  | 0.7 | 0.8 |

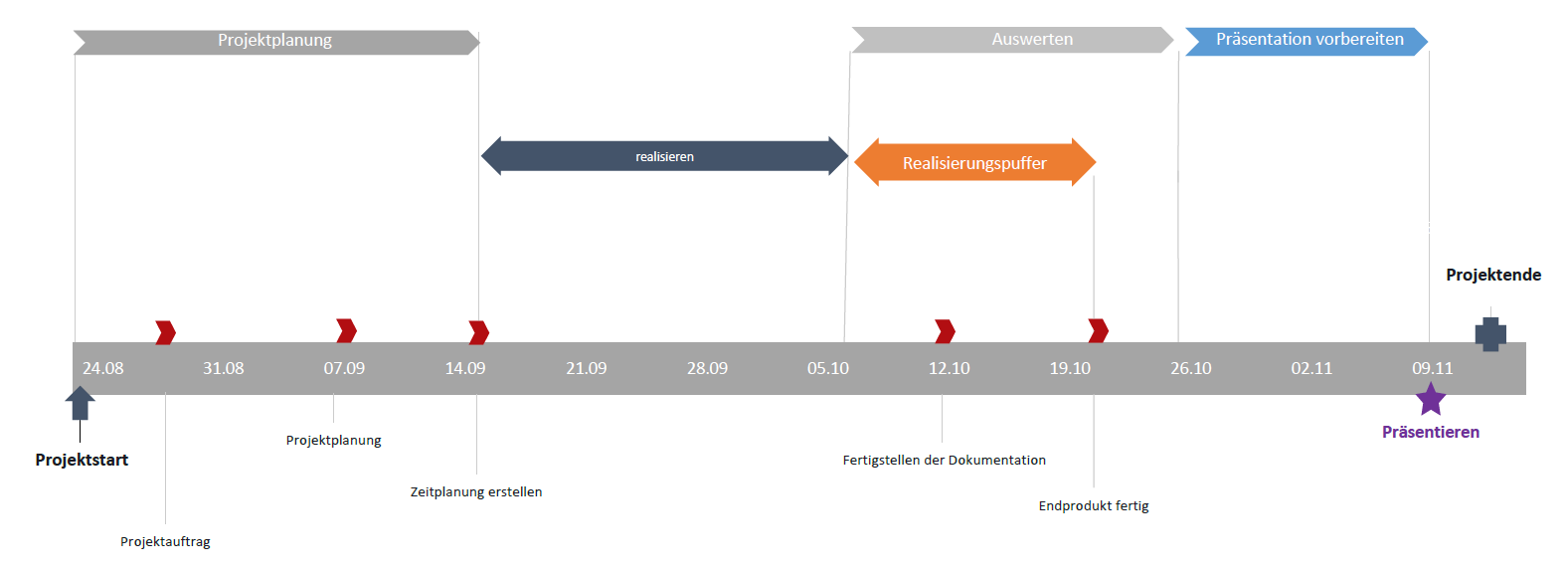
~Die Arbeiten an der Dokumentation werden wir uns aufteilen. Am Code aber werden wir, um den grössten Lerneffekt zu erhalten, wenn möglich zusammenarbeiten.

# Planen

Nachdem wir nun unseren Auftrag haben und unsere Fragen geklärt sind, müssen wir noch den ganzen Ablauf unseres Projekts planen. Um uns einen groben Überblick über unsere Aufgaben zu verschaffen, haben wir uns mit Teams Aufgaben erstellt, welche uns helfen sollten, unsere Zeit einzuplanen.



Mit diesen Aufgaben, haben wir uns dann einen Zeitplan erstellt. Dieser sollte uns helfen, unsere Aufgaben und Deadlines sowie auch die Meilensteile unseres Projekts besser und einfacher ersichtlich zu machen so, dass wir immer alles im griff haben. Ausgesehen hat der Ganze Plan dann so



# Entscheiden

Um unser Projekt nun überhaupt realisieren zu können, mussten wir uns etwas einfallen lassen. Schliesslich, wissen wir im ersten Lehrjahr noch nicht wirklich viel vom Programmieren. Das Snake aber braucht schon die einte oder andere Fähigkeit.

Also haben wir uns wieder unsere altbekannte Tabelle gezeichnet und Zwischen zwei Optionen verglichen:

|  |  |  |  |
| --- | --- | --- | --- |
| Kriterien: | Gewichtung: | Alleine losprogrammieren | Mit einem Tutorial programmieren |
| Zeit | 8 | 1 | 5 |
| Lerneffekt | 10 | 5 | 3 |
| Zusammenarbeit | 4 | 2 | 5 |
| Komplexität | 5 | 2 | 4 |
| Gesamtwert: |  | 10 | 17 |
| Score: |  | 3 | 6 |

Gewonnen hat dabei die Idee mit einem Tutorial zu arbeiten, da diese vor allem im Zeitaspekt und bei der «Zusammenarbeit» also ob man gut gleichzeitig daran arbeiten kann gepunktet hat.

Wir suchten also im Internet nach einem Tutorial. Also einer Anleitung, mit welcher wir Schritt für Schritt unser Snake spiel Programmieren können und auch verstehen, wie das ganze funktioniert. So konnten wir am Ende einen sehr hohen Lerneffekt erzielen.

Falls euch das Tutorial auch interessiert:

<https://www.mooict.com/c-tutorial-create-a-classic-snake-game-in-visual-studio/>

# Realisieren

Nun ging es endlich ans Realisieren und wie wir uns vorher geeinigt hatten, begannen wir damit, uns ein Git-Repository zu erstellen. Darin sollten wir dann später unsern Code so abspeichern können, dass wir beide von unseren Computern darauf zugreifen können.

Als nächstes machten wir uns daran, zu programmieren. Trotz des Tutorials, stolperten wir dabei auch über das eine oder das andere Problem. Schliesslich wollten wir nicht nur immer dem Tutorial folgen, sondern auch unsere eigenen kleinen Features einbauen. Zum Beispiel wollten wir, dass unser Snake Programm trotzdem keine verschiedenen Levels hat. Es sollte nämlich selber mit der zeit schwieriger werden. Das soll heissen, dass das Programm schwieriger wird, je besser der jeweilige Spieler ist. In etwa so, dass es sich selber dem Spieler anpasst. Auch die Farben wollten wir in einem etwas anderen Stil gestalten, wobei wir uns zum Ende aber trotzdem für einen Retro-Grauton entschieden haben, da wir es am passendsten gefunden haben.

Bein Coden selber sind wir so vorgegangen, dass wir meist zusammen an einem Pc sassen und dann zusammen die Themen anschauten.

# Kontrollieren

Um unser abgeschlossenes Projekt zu kontrollieren, haben wir mit einer einfachen Checkliste gearbeitet, in welcher wir die grundlegenden Punkte eingetragen haben. Unter den jeweiligen Punkten haben wir dann auch immer einen Vorschlag, eingefügt. Dieser ist immer eine Methode, mit welcher sich die Punkte gut testen lassen.



-Funktioniert das Spiel gemäss den Angaben?

(ausführliches Testing)



-Ist die Dokumentation verständlich und grammatikalisch korrekt?

(von Aussenstehenden kontrollieren lassen)



-Präsentation beinhaltet alles Wichtige?

(mehrfaches kontrollieren)

So konnten wir anhand dieser drei einfachen Punkte ganz einfach sehen, was wir noch kontrollieren müssen und wie wir das am besten machen können.

# Auswerten

Um unser Projekt auszuwerten, haben wir uns nochmals eine Liste erstellt. Diesmal aber mit den einzelnen Punkten von IPERKA. Denn in unserem auswerten Schritt, wollen wir alles, was wir gemacht haben nochmals Review passieren lassen und uns nochmals selbst kontrollieren. Also schauen, ob wir alle Ziele und Punkte eingehalten haben.

Die kurze Antwort auf diese Frage ist kurz und einfach «Ja». Die lange Antwort jedoch ist auch spannend. Fangen wir also ganz am Anfang bei I an:

I: An der ersten Haltestelle von IPERKA haben wir uns für unser Snake Lernprojekt entschieden und uns die Spezifikationen für unser Spiel ausgedacht.

P: Beim Planen haben wir noch alle Fragen zu unserem Projekt geklärt. So, dass wir uns schon vor Beginn im Klaren sind, was wir da eigentlich angehen. Danach haben wir die Aufgabenplanung und auch einen Zeitplan erstellt, welcher uns durch das ganze Projekt hindurch begleitet hat.

E: Die dritte Haltestelle war für uns eher kurz. Wir hatten nämlich schon von Anfang an entschieden, wie wir vorgehen werden.

R: Das Realisieren ist wohl immer die spannendste Stelle. Wir haben verschiedene Tutorials durchgeschaut, bis wir eines gefunden haben welches uns gefällt. Obwohl wir sehr viel nach dem Tutorial gemacht haben konnten wir auch unsere Ideen in unserem Spiel verewigen.

K: Während dem kontrollieren sind wir durch die Punkte gegangen welche wir für unser Projekt gesetzt haben, wir haben auch die Punkte dritt Personen gegeben das wir sicher waren das wir alles erreicht haben.

A: Nun kommen wir zum letzten Punkt von IPERKA und zwar dem Auswerten. Beim Auswerten haben wir uns wohl die meisten Mühen gegeben. Wir wollten das wir unser Projekt ohne Makel abgeben können, darum haben wir alles sehr kritisch angeschaut und auch beim Programm mehrere Debuggs gemacht. Wie auch beim Kontrollieren haben wir auch andere Personen unser Projekt auswerten lassen, um unser Projekt zu verbessern.

Rückblick