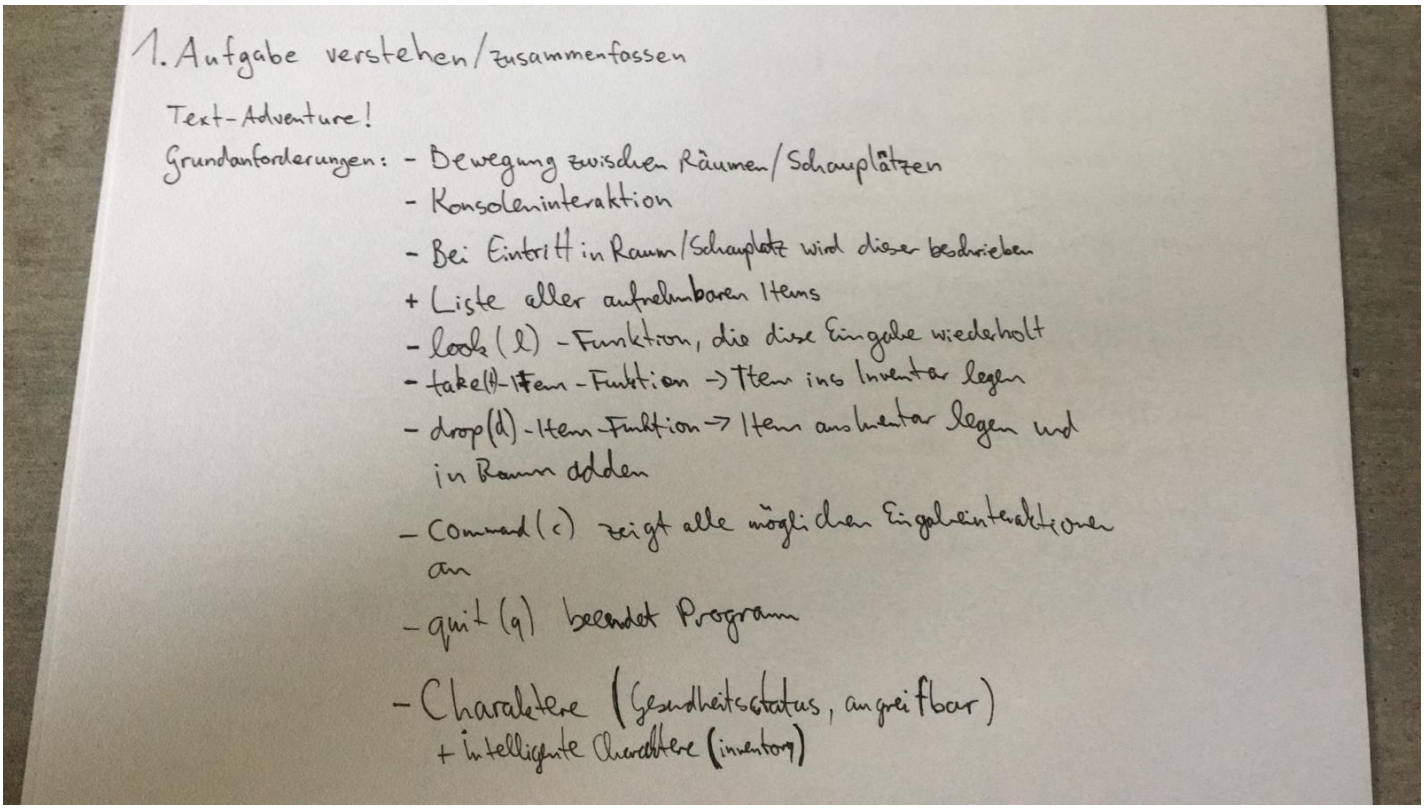
Text-Adventure-Game

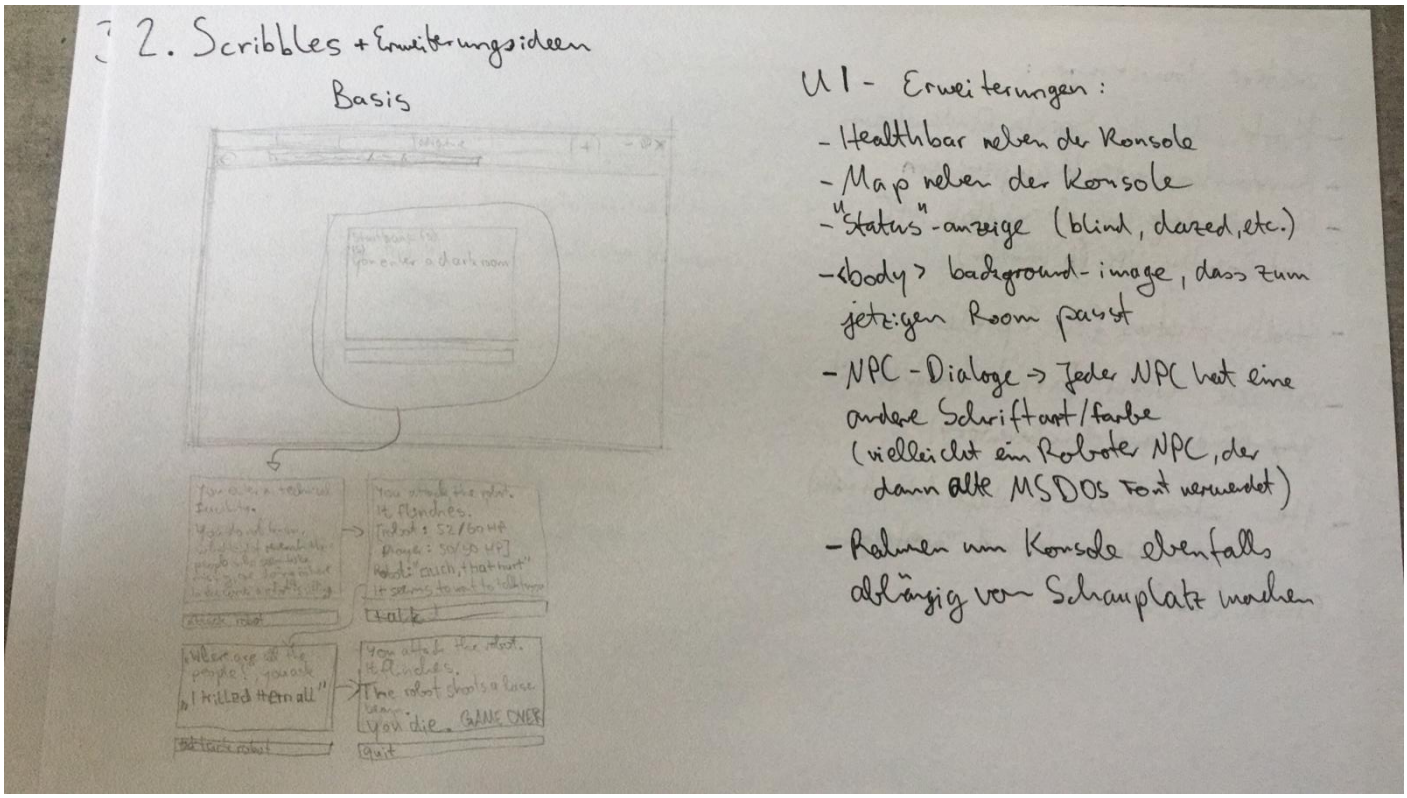
Konzeptionelle Analyse



1. Aufgabe verstehen/zusammenfassen

Text-Adventure!

Grundanforderungen:
- Bewegung zwischen Räumen/Schauplätzen
- Konsoleninteraktion
- Bei Eintritt in Raum/Schauplatz wird dieser beschrieben
+ Liste aller aufnehmbaren Items
- look(l) - Funktion, die diese Eingabe wiederholt
- take(t)-Item-Funktion -> Item ins Inventar legen
- drop(d)-Item-Funktion -> Item aus Inventar legen und in Raum adden
- command(c) zeigt alle möglichen Eingabeinteraktionen an
- quit(q) beendet Programm
- Charaktere (Gesundheitsstatus, angreifbar)
+ intelligente Charaktere (inventory)



2. Scribbles + Erweiterungsideen

Basis

UI - Erweiterungen:
- Healthbar neben der Konsole
- Map neben der Konsole
- "Status"-anzeige (blind, dazed, etc.)
- <body> background-image, dass zum jetzigen Room passt
- NPC-Dialoge -> Jeder NPC hat eine andere Schriftart/farbe (vielleicht ein Roboter NPC, der dann alte MSDOS Font verwendet)
- Rahmen um Konsole ebenfalls abhängig von Schauplatz machen

weitere Erweiterungen:

- Karte, die der Spieler finden kann
- Rundenbasiertes Kampfsystem
- Unterscheidung zwischen Mob, NPC und friendly NPC (z.b. Händler)
- Health-Status für Spieler
- attack with [axe] (Angriff mit Gegenständen aus dem Inventory)
- Item unterscheiden in Waffe (attack with) Consumable (Heiltrank) und Quest-gegenstände (Schlüssel)

3. Erste Gedanken zu Abläufen / Datenstrukturen

- Klasse Game sollte existieren. Sobald instantiiert → initialisierung aller erforderlichen Daten (vor allem map.json muss geladen werden)
- Game besitzt eine Instanz der Klasse Konsole, der Klasse User, eine Liste mit allen Charakteren und eine Liste mit allen Räumen sowie eine Instanz der Klasse Scene

→ Game dient als übergeordnete Klasse, die alle wichtigen Instanzen hat und diese verknüpft.

- Mit dem Observer-Pattern kann z.B. ein Observer auf Konsole dem Game mitteilen, wenn User z.B. "up" eingibt, um so die Scene