Erfahrungsbericht
Jonas Schoeler

1. Sprint

Projektbasis: Das erstellen des Git-Repository war grundsätzlich nichts neues. Konfigurationsarbeit war es dann eine passende git-ignore sowie git-attribute Datei zu erstellen. Mit unterschiedlichen Anleitungen hat sich das Erstellen aber als leichter als gedacht herausgestellt.

Objekte hinzufügen: Unser erster Schritt war es .obj-Dateien in Unity zu laden um damit einen Einstieg in die Thematik zu finden. Nachdem wir hier aber gemerkt habe, dass Unity mit .obj-Dateien zur Laufzeit nicht gut zurechtkommt, sind wir auf .prefabs umgestiegen. Als schwieriger stellte sich dann noch das lokalisieren eines Mausklicks heraus. Hierfür gab es zwar viele Codebeispiele online die wir zu Beginn aber nicht 100% verstanden haben. Erst nach weiteren probieren von anderen Möglichkeiten haben wir nach und nach genau verstanden wie Unity die unterschiedlichen Parameter bezieht.

Am Ende von Sprint 1 waren wir uns immer noch nicht sicher ob wir das Projekt wirklich stemmen können. Die vielen Fragezeichen, die sich während Sprint 1 aufgebaut haben, mussten wir somit in Sprint 2 mitnehmen.

2. Sprint

Objekte entfernen: Das entfernen von Objekten stellte sich als deutlich einfacher heraus, als wir angenommen hatten. Somit war das Sprintziel schneller als geplant fertig.

UI-Gestaltung: Da inzwischen auch von den anderen Teammitgliedern verschiedene Funktionen implementiert worden sind, war es an der Zeit eine UI zu entwerfen und umzusetzen. Niemand aus unserer Gruppe hatte bisher im Bereich UI-Design gearbeitet, was die Aufgabe erschwerte. Auch hier waren für mich Onlinequellen gute Bezugspunkte. Die Programmierung war aber deutlich Anspruchsvoller als ursprünglich angenommen, was sich auch in den Stunden widerspiegelt.

3. Sprint

Einzelnen Bewegungen festlegen: Erst jetzt habe ich mich so richtig mit dem Framework beschäftigt, da meine bisherigen Aufgabe nicht darauf angewiesen waren. Die Grundsätzliche Funktionsweise wurde uns zu Beginn des Projektes kurz vorgestellt. Die Bewegungen an sich sind im Repository von Daimler erklärt und mit Beispielen versehen. Somit war das erstellen einzelner unabhängiger Bewegungen schneller als geplant umgesetzt.

Queue: Was sich im Gegensatz zu den einzelnen Bewegung als großes Hindernis herausstellte, war das Zusammenfügen dieser. Die einzelnen Instructions können mittels Start- und Endconditions aufeinander abgestimmt werden. Nach und nach zeichnete sich aber ab, dass jede Bewegung ein etwas anderes Verhalten zeigte. Somit war es letztendlich viel Trial-and-Error da wir uns nicht immer sicher waren, was genau wann passiert. Über die Stunden wurden wir uns aber immer sicherer und letztendlich hat es mit den, zu dem Zeitpunkt implementierten Bewegungen, auch für eine Vorführung gereicht. Wir wussten aber, dass verschiedene Bewegung immer noch nicht so miteinander interagierten wie sie sollten.

4. Sprint

Laufweg: Hier hatten wir einen Parameter übersehen, denn wir einfach einfügen konnten, um ein abbrechen der Laufwege zu verhindern.

Mosim Update: Um alle Funktionen des Mosim-Framworks zu nutzen, mussten wir das Framework updaten, was dazu führte, dass unser Code teilweise nicht mehr funktionierte. Ausschlaggebend dafür war, dass sich Bezeichnungen änderten und somit der Framework Bewegungen nicht mehr ausführen konnte. Die Änderungen ausfindig zu machen dauerte recht lange und war von stetigen Testen begleitet.

Queue dynamisch an Auswahl anpassen: Das Ziel war es die bekannten Lücken beim Zusammensetzen der Bewegungen zu schließen. Hier zeigte sich immer wieder, dass wir mit dem Framework vorher noch nie gearbeitet hatte, da manches Verhalten für uns nicht erklärbar war. Offene Fragen haben wir entweder durch verschiedenen Einträge in der Doku oder durch weiterleiten an Herrn Gaisbauer gelöst.

5. Sprint

GUI: Unsere GUI brauchte dringend ein Upgrade, da die bisherige Version nicht unsere Anspruch widerspiegelte. Hier haben wir uns einen Crashkurs von Tim Niederer geben lassen um unsere Vorstellungen umzusetzen. Die GUI umzubauen und umzuschreiben hat natürlich nochmal viele Stunden benötigt.

Code Refactoring und Kommentare: Methoden die wir zu Beginn des Projektes implementiert hatten, waren teilweise unnötig teuer oder kompliziert implementiert. Hier ließen sich viele Aufrufe günstiger abwandeln. Den Kommentarstil hatten wir zu Anfang festgelegt und uns auch daran gehalten. Um aber die Dokumentationsarbeit etwas zu erleichtern sind wir zuletzt noch auf die XML-Kommentare umgestiegen.

Sprint 5 war zu Beginn von GUI beherrscht, wurde aber gegen Ende immer offener. Hier war dann das fixen von kleinen Bugs häufig die täglich Arbeit. Aber auch noch spontane Zusätze wie ein Timer ließen das Projekt noch immer weiter wachsen.

Lessons learned:

- Genauere Klassenplannung
- UI ist schwieriger als gedacht
- Stunden genauer dokumentieren