**Konzeptioneller Ansatz: Carrier Tracking**

Verwendete Unity Version: 2019.4.11

Im projekt gewechselt auf 2019.4.13F1 weil Zusammenspiel weil verdacht bestand dass JSON Probleme vorliegen, deshalb auch wechsel auf binary Files

Dot von 2.X auf 4.X version update sonst QR Code Bib nicht nutzbar

**Interessante Punkte:**

anlegen einer virtuellen Umgebung (Raum in Unity) um die Funktionalitäten der QR-Code Erkennung zu simulieren

~~Verbindung von virtuellen Objekten mit der Realität anhand von aufgenommen Fotos (Echte welt in digital überwachen und projezieren)~~

Testen von realen Bildern mit der QR-Code Erkennung mit verschiedenen QR-Code Größen, Distanzen und Kameras

Bilderkennung mit Fokus auf QR-Codes/arbeiten mit vorgefertigten Frameworks zu Bilderkennung

~~Bestimmung der Maße der QR Codes und Positionierung der Kamera (Bilderauswertung, QR Bib auswählen,warum hat er diese gewählt, kostenlose Bib)~~

Serialisierung von Objekten welche von Usern erstellt werden

~~Abspeichern alle angelegte informationen von Stationen und Ladungsträgern in Binary Files~~

Einstieg in Unity, 3D Software Engine (Game Dev – auch 2D )

Anwendung klassischer Softwarepatterns in Unity wie MVC Singleton

Kooperative Softwareentwicklung mit Hilfe von SCRUM und GitHub

~~Irgendwas mit projektmanagement halt // SCRUM~~

**Weitere Punkte welche im Backlog vorgesehen waren aber aus zeitlichen oder technischen Gründen nicht umgesetzt wurden:**

eigene 3D Modele der Ladungsträger zur besseren Visualisierung

~~Platzieren von Abbildern der Ladungsträger anstatt Platzhalter Model~~

Speicherort der QR-Code Bilder auf dem PC frei wählbar (Bilder werden jetzt im Programm gepseichert.

* ~~Freie Ordnerwahl // Shared Ordner~~

virtuelle Räume mit Wänden und Mobiliar gestalten

* ~~Wände zeichnen (temp drin, nicht mehr genutzt funktioniert noch aber wird nicht gespeichert)~~

**Herausforderungen und Probleme:**

Als zentrale Speicherung der einzelnen Codes von verschiedenen Programmieren wurde Github gewählt. Dabei wurden zu anfangs oft Änderungen am Code nicht übernommen oder nicht alle Programmierer konnten diese abrufen. Deshalb hat jeder Programmierer als Backup zusätzlich noch bei sich lokal ein Inkrement seiner jeweils letzten finalen Version vorgehalten.

~~Zusammenspiel von Unity und Github  Schnittstellen dev team github~~

Die Kenntnisse und Erfahrungen im Programmieren waren bei den Teilnehmern unterschiedlich ausgeprägt. Dies führte dazu das erfahrene Programmierer zwar kurzen und performanten Code schreiben konnten, dieser jedoch von anderen nicht verstanden wurde. Hier wurde dann entweder erläutert oder der Code umgeschrieben um dies verständlicher für alle zu machen.

~~Verchiedener wissenstand der teilnehmer  alle auf ein level bringen~~

Durch die Fernlehre, wegen der aktuellen Pandemie, konnten keine Präsenzveranstaltungen stattfinden. Dies führte zu einer erschwerten Zusammenarbeit da nur über Videoanrufe oder Mails der Kontakt zu Dozenten und Mitstudierenden stattfinden konnte. Die ebenfalls wichtige soziale Komponente bei der Zusammenarbeit ging so verloren. Auch eine Besichtigung des Evobus Werkes, um sich ein besseres Bild der Anforderungen zu machen, war deswegen nicht möglich.

~~Corona Pandemie // Präsenztermine // Auswertungen Bilder // Kein werksbesuch alles nur digital tortz praxis alle noch theorie // Kommunikation meist per Mail~~

Programmieren in C# war eine Herausforderung, da die meisten Projekte bisher in Java geschrieben wurden. Zwar sind beides objektorientierte Programmiersprachen, doch teils unterschiedlicher Syntax und Konventionen bereiteten oft Verzögerungen. Nach einigen Wochen Eingewöhnungszeit entwickelte sich hier eine gewisse Routine.

~~Neue programmiersprache~~

Eine weiter Hürde waren die verschiedenen Betriebssysteme (Windows und MacOS) auf denen die einzelnen Entwickler programmiert haben. Es kam vor dass Code von der Windows IDE auf auf der Mac IDE nicht ausgeführt werden konnte.

Crossplattform (Mac und Windows spielen nicht zusammen)