HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Exposé
für
eine Bachelorarbeit in Medieninformatik

Konzeption, Evaluation und Entwicklung von Netzwerkkomponenten für eine auf die Lehre spezialisierte Game-Engine

Referent: Prof. Jirka Dell'Oro

Korreferent: Prof. Dr. Ruxandra Lasowski

vorgelegt am: 03.07.2019

vorgelegt von: Falco Böhnke
250100
Im Großacker 28
79252, Stegen
falco.boehnke@hs-furtwangen.de

1. Problemstellung

Die Videospielindustrie ist in den letzten Jahren massiv gewachsen. Allein der Markt für Videospiele in den Niederlanden wuchs von 518 Millionen Euro im Jahr 2012 auf 1.65 Milliarden Euro im Jahr 2019 an, mit vergelcihbaren Tendenzen weltweit mit Wachstumszahlen von 11% über die letzten 10 Jahre (Wijman 2018)(Kazimier 2017). Der starke Anstieg der Umsatzzahlen hat zu einem gesteigerten Konkurrenzkampf innerhalb des Videospielmarktes geführt. Innovationen sind unerlässlich geworden um wettbewerbsfähig zu bleiben, wodurch auch die Ansprüche der Kunden an die Produkte immer weiter ansteigen. Besonders der Markt der Online-Spiele liegt dabei im Fokus der Videospielhersteller, da sich dort das größte Potential für Umsatz und dauerhafte Kundenbindung finden. So wächst der Druck auf Spieleentwickler, besonders auf qualifizierte Fachkräfte im Bereich Netzwerkkommunikation, die sich mit immer komplexeren Projekten konfrontiert sehen.

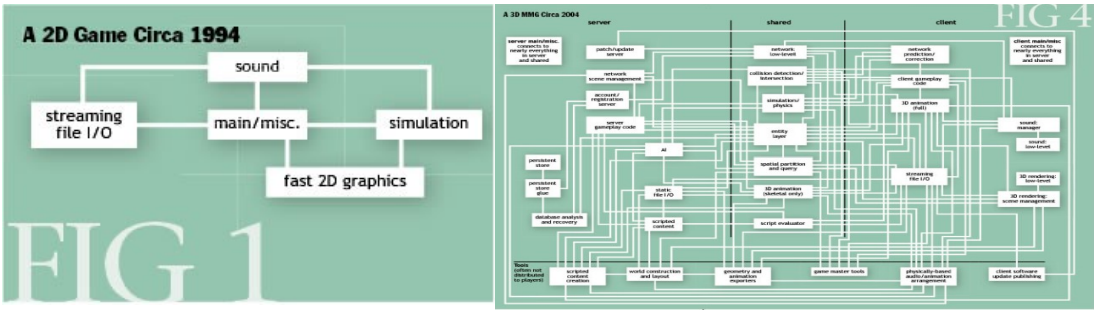


Abb. 1: Vergleich der benötigten Komponenten für ein Videospiel im Jahr 1994 und 2004 in Anlehnung an Blow (2004)

Gameengines, Entwicklungsumgebungen die für die Produktion von Videospielen optimiert sind, finden daher verstärkt Einsatz um die Effizienz der Entwicklung zu steigern. Doch auch hier hat die Komplexität stark zugenommen. Automation und Abstraktion sollen Entwicklern fundamentale Arbeiten abnehmen und so den Arbeitsaufwand verringern. Doch die Undurchsichtigkeit der abstrahierten Systeme und die oftmals komplizierte Struktur der Game-Engines macht es neuen Entwicklern schwer die grundlegende Funktionsweise der einzelnen Elemente eines Videospiels zu erfassen und zu erlernen. So sind Entwickler gezwungen statt grundlegender Konzepte die Funktionsweise ihrer ausgewählten Entwicklungsumgebung zu lernen. Ist dann ein Wechsel auf eine andere Entwicklungsumgebung notwendig, müssen Entwickler sich erneut an die Umgebungsentwicklung anpassen, ohne die grundlegenden Funktionsweisen von Videospielen verstanden zu haben. Besonders der Mangel einer von grundauf für die Lehre konzipierten Game-Engine verschärft diese Problematik weiter.

Um diese Lücke zu füllen wurde das Projekt ‚Fudge' von Prof. Jirka Dell'Oro-Friedl ins Leben gerufen. Fudge ist eine Game-Engine und ein Editor, der die Strukturen und Prozesse eines Videospiels offen legt. Zwar bietet auch Fudge Werkzeuge die das Leben der Entwickler erleichtern sollen, diese sollen jedoch dem Grundsatz der Lesbarkeit folgen und es so vereinfachen zu verstehen wie sie funktionieren. Insbesondere im Bereich der Netzwerkkommunikation, die von vielen Game-Engines komplett automatisiert wird, soll der Prozess der Kommunikation menschenlesbar und verständlich sein.
Das Ziel dieser Arbeit ist es also, unter Berücksichtigung der oben genannten, übergeordneten Leitsätze, Netzwerkkomponenten für ‚Fudge' zu entwickeln. Diese Netzwerkkomponenten sollen es ermöglichen Prototypen für Online- und Mehrspielerspiele rapide zu entwicklen und zu testen und gleichzeitig vollständig anpassbar und erweiterbar zu bleiben. Besonderes Augenmerk liegt darauf die Abstraktion möglichst gering zu halten, damit die Funktionsweise der Komponenten direkt am Quellcode von ‚Fudge' ersichtlich ist.

2. Entwicklungsmethoden

Fudge als Gameengine basiert auf *Electron* und verwendet *Typescript* für die Entwicklung. Electron ist ein auf *Chromium* basierendes Framework, welches es ermöglicht mithilfe von Webtechnologien eigenständige Anwendungen zu entwickeln. Zu diesen Webtechnologien gehören *HTML*, *CSS* und *JavaScript*. TypeScript, ein Superset für JavaScript, erlaubt die objektorientierte Programmierung mit JavaScript und vermeidet dadurch auch einige Fehlerquellen, wie zum Beispiel fehlerhafte Typisierung von Variablen.
Um die Netzwerkkommunikation möglich zu machen, wird zum einen *WebRTC* verwendet, zum anderen *WebSocket*. Beide Technologien sind gängige und zukunftssichere Standards für die Netzwerkkommunikation in browserähnlichen Umgebungen, worum es sich bei Chromium handelt.
WebRTC bewerkstelligt die Kommunikation über *UDP* (User Datagram Protocol), die bei der möglichst latenzfreien Übertragung von Nachrichten hilft und so Kernstück der meisten Onlinespiele darstellt. Websocket wird für die zuverlässige, aber langsamere, Kommunikation verwendet, beispielsweise für die Übertragung von Dateien oder wichtigen Netzwerknachrichten wie Logindaten.

Die Planung wird mit Klassendiagrammen, sogenannten Unified Modeling Language Diagrammen, und Aktivitätsdiagrammen umgesetzt. So lässt sich einerseits das Vorgehen festlegen und andererseits eine allgemein gültige und standardisierte Dokumentation der Komponenten sowie ihrer Funktionsweise erstellen.
Anschließend werden die Diagramme evaluiert und, auf die Zielsetzung eine schlichte und durchsichtige Gameengine zu entwickeln, angepasst.
Ist die Evaluation abgeschlossen, beginnt die Umsetzung der Pläne. Als Entwicklungsumgebung dient dabei *VisualStudio Code*, das ebenfalls auf Electron basiert und sich durch Modularität und einen großen Marktplatz an Erweiterungen auszeichnet.
Zur Versionskontrolle wird *Github* verwendet. Github ermöglicht es frühere Entwicklungsstände wiederherzustellen, sichert die Ergebnisse auf einem Server, erlaubt einfache Organisation mithilfe von digitalen Karteikarten und erlaubt das fehlerfreie Zusammenfügen großer Codeabschnitte.
Ist die Entwicklung der Netzwerkkomponenten abgeschlossen, werden sie in Fudge integriert.

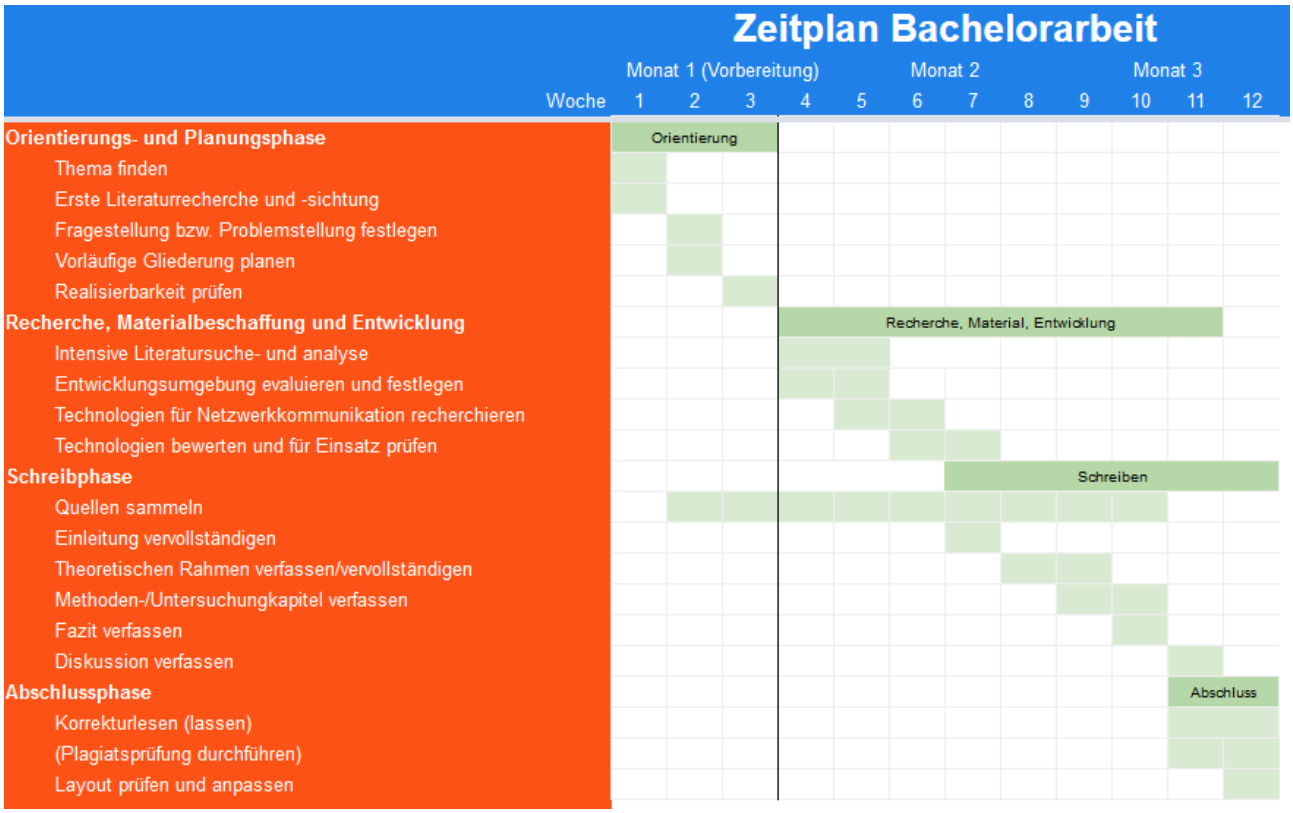
3. Zielsetzung

Das Ziel dieser Bachelorarbeit ist es, Komponenten für die Netzwerkkommunikation in der Gameengine ‚Fudge' zu entwickeln. Dies beeinhaltet die Evaluation der zu verwendenden Technologien, sowie Planung der Netzwerkkomponenten, Dokumentation und Entwicklung. Bei der Planung ist zu beachten, dass ‚Fudge' eine auf die Lehre spezialisierte Gameengine ist und dementsprechend Komplexität und Abstraktion auf einem geringen Niveau gehalten werden sollen. Dies muss in der Planung ersichtlich sein und so dokumentiert werden.

Am Ende des Projekts steht angehenden Entwicklern ein Werkzeug zur Verfügung, das es ihnen ermöglicht vollständige Online-Multiplayerspiele mit ‚Fudge' zu entwickeln, die gegebenen Komponenten zu erweitern und beliebig anzupassen.

4. Provisorisches Inhaltsverzeichnis

5. Zeitplan


Zeitplan Bachelorarbeit
Monat 1 (Vorbereitung)
Monat 2
Monat 3
Woche
1
2
3
4
5
6
7
8
9
10
11
12
Orientierungs- und Planungsphase
Orientierung
Thema finden
Erste Literaturrecherche und -sichtung
Fragestellung bzw. Problemstellung festlegen
Vorläufige Gliederung planen
Realisierbarkeit prüfen
Recherche, Materialbeschaffung und Entwicklung
Recherche, Material, Entwicklung
Intensive Literatursuche- und analyse
Entwicklungsumgebung evaluieren und festlegen
Technologien für Netzwerkkommunikation recherchieren
Technologien bewerten und für Einsatz prüfen
Schreibphase
Schreiben
Quellen sammeln
Einleitung vervollständigen
Theoretischen Rahmen verfassen/vervollständigen
Methoden-/Untersuchungkapitel verfassen
Fazit verfassen
Diskussion verfassen
Abschlussphase
Abschluss
Korrekturlesen (lassen)
(Plagiatsprüfung durchführen)
Layout prüfen und anpassen

Literaturverzeichnis

1: Blow, Jonathan (2004): Game Development: Harder than you think. In: Queue – Game Development. (02/2004), S. 29-37.

2: Kazimier, Martin, de Visser, Jeroen (2017): Video games. Playtime is over; with revenue surpassing one billion euro in 2018, video games are serious business and here to stay. https://www.pwc.nl/en/publicaties/dutch-entertainment-and-media-outlook-2017-2021/video-games.html (2019-06-27)

3: Wijman, Tom (2018): Mobile Revenues Account for More Than 50% of the Global Games Market as It Reaches $137.9 Billion in 2018. https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/ (2019-07-03)

4. Williams, Andrew (2017): History of Digital Games. Waltham, Massachusetts, United States: Focal Press

5. Ristic, Dan (2015): Learning WebRTC. Develop interactive real-time communication applications with WebRTC. Birmingham, UK: Packt Publishing. E-Book

6. Sergiienko, Andrii (2014): WebRTC blueprints. Develop your very own media applications and services using WebRTC. Birmingham, UK: Packt Publishing. E-Book

7. Chopra, Varun (2015): WebSocket essentials, building apps with HTML5 WebSockets. build your own real-time web applications using HTML5 WebSockets. Birmingham, UK: Pack Publishing. E-Book

8. Rieseberg, Felix (2018): Introducing Electron. Desktop apps with JavaScript. Sebastapol, CA: O‘Reilly Media. E-Book

9. Jasim, Muhammed (2017): Building cross-platform desktop applications with Electron. create impressive cross-platform desktop applications with Electron and Node. Birmingham, UK: Packt Publishing. E-Book