Softwareprojektmanagement in der Planung

Aufgabe 2: Projektentwurf

Hochschule Bielefeld
Campus Minden
Bachelor of Computer Science
Softwareprojektmanagement
Prof. Dr. Jörg Brunsmann
Wintersemester 2023/2024

TeamA1
Willi Schäfer, Kevin Zuber, Tim Röckemann, Ole Löffler, Timo Haverich, Tobias Wegner, Lewin Wanzek

GitHub-Link
https://github.com/TeamA1-SPM

Organisatorisch

Um die Anforderungen an das Softwareprodukt zu beschreiben und zu priorisieren, sollte das Endergebnis ein verteiltes Softwaresystem sein, bei dem jedes Teammitglied die Verantwortung für ein geeignet großes Softwaremodul übernimmt. Das konkrete Ziel besteht in der Implementierung und Umsetzung eines verteilten Softwaresystems in Form eines Multiplayer Car Racing Games. Hierbei liegt die Priorisierung auf einer erfolgreichen Zusammenarbeit, einer umfassenden Planung und einer erfolgreichen Implementierung des Softwareprojekts, wobei das Endergebnis als zweitrangig betrachtet wird, um den Fokus auf eine effektive und gute Zusammenarbeit zu legen.

Das organisatorische Vorgehen setzt auf ein agiles Projektmanagement nach dem Scrum-Ansatz, wobei alle Mitglieder in Themengruppen eingeteilt werden. Konkrete Zuständigkeiten sind wie folgt verteilt:

Timo Haverich, Tobias Wegner, Lewin Wanzek sind für die Integration von Socket.io verantwortlich. Der Fokus liegt auf der Entwicklung des Servers. Dabei werden verschiedene Aspekte wie Spiellogik, Spielstand, Lokalisierung und die Wahl des Kommunikationsprotokolls eingehend geprüft.

Der Python-Client wird von Tim und Ole betreut, wobei eine gründliche Analyse des Source Codes zu Beginn des Projekts durchgeführt wird. Technische Fragen, die sich während der Implementierung ergeben, werden sorgfältig beantwortet und dokumentiert, um eine reibungslose Entwicklung sicherzustellen. Ebenso werden klare Standards und Richtlinien festgelegt, die den gesamten Entwicklungsprozess des Python-Clients leiten sollen.

Der Java-Client wird von Willi und Kevin entwickelt, wobei ähnliche Schritte wie beim Python-Client verfolgt werden. Auch hier erfolgt eine gründliche Untersuchung des Source Codes zu Beginn des Projekts, gefolgt von der Beantwortung und Dokumentation technischer Fragen, um eine effiziente Entwicklung zu gewährleisten. Klar definierte Standards und Richtlinien sollen den Entwicklungsprozess des Java-Clients leiten.

Das Scrum-Framework wird im Projekt verwendet, wobei ein Sprint Backlog mit allen notwendigen Teilschritten für ein erfolgreiches Ergebnis erstellt wird. Dieser wird von den jeweiligen Themengruppen erstellt und ist für alle Teammitglieder im Kanban-Board einsehbar. Sprints werden durchgeführt, um das Projekt schrittweise umzusetzen und Teilergebnisse zu erarbeiten. Am Ende jedes Sprints finden eine Retrospektive zur Beurteilung der Zusammenarbeit und ein Review zur Beurteilung der Ergebnisse statt.

Die Rolle des Scrum-Masters wird von Timo Haverich übernommen, während alle Teammitglieder als Product-Owner fungieren. Das gesamte Team arbeitet zusammen, um die Ziele des Projekts zu erreichen. Tobias Wegner übernimmt die Rolle des Repo/CI-Maintainers, während Lewin Wanzek als Projektleiter agiert, um die Koordination und Leitung des gesamten Projekts sicherzustellen. Diese Struktur gewährleistet eine effiziente und effektive Umsetzung des Multiplayer Car Racing Games und fördert eine kooperative Arbeitsweise innerhalb des Teams.

Regelmäßige Meetings über Discord dienen der Synchronisation und dem Austausch von Aufgaben und Fortschritten. Außerhalb der Meetings werden auftretende Fragen und Probleme entweder im GitHub, in den entsprechenden Issues erfasst oder direkt im Discord besprochen.

Die Verfolgung des Projektfortschritts erfolgt über ein Kanban-Board in GitHub, das eine transparente Darstellung und effiziente Verteilung der Aufgaben ermöglicht. Meilensteine werden mit den entsprechenden Issues verknüpft, um einen klaren Überblick über den Projektfortschritt zu gewährleisten.

Zur Definition und zum Sammeln von Anforderungen, als auch Ideen werden User-Stories formuliert.

Die Dokumentation des Fortschritts beinhaltet die Erklärung des Codes und der Logik, die zeitliche Dokumentation sowie die Erläuterung sämtlicher Funktionalitäten des Softwareprodukts. Die Dokumentation wird dabei einfach gehalten und für alle Mitglieder einsehbar sein.

Als potenzielle Risiken werden spezifische Punkte wie die Gefahr, den Fokus auf das Lernen zu vernachlässigen, das Henne/Ei-Problem, die Notwendigkeit der gegenseitigen Unterstützung über Zuständigkeitsbereiche hinweg sowie die korrekte Nutzung des GitHub-Projekts deutlich hervorgehoben, um mögliche Hürden während des Projektverlaufs zu minimieren.

Socket.io

Socket.io ist eine JavaScript-Bibliothek, die eine bidirektionale Echtzeitkommunikation zwischen Web Clients und Servern ermöglicht. Sie ermöglicht eine reibungslose und effiziente Datenübertragung in Echtzeit, was sie zu einer geeigneten Wahl für die Implementierung des Multiplayer Racing Games macht. Die Bibliothek bietet eine Vielzahl von Funktionen, die speziell für die Handhabung von Websockets entwickelt wurden, was eine schnelle und zuverlässige Echtzeitkommunikation ermöglicht.

Für den Java-Client wird socket.io mit Hilfe von Bibliotheken wie "socket.io-client" in die Java-Umgebung integriert. Hierbei müssen die geeigneten Abhängigkeiten und Konfigurationen berücksichtigt werden, um eine reibungslose Integration und Kommunikation mit dem Server zu gewährleisten. Es ist wichtig, dass Schnittstellen und Protokolle klar definiert werden, um eine effektive Kommunikation und Synchronisation zwischen dem Java-Client und dem Server zu ermöglichen.

Für den Python-Client wird socket.io ebenfalls in die entsprechende Python-Umgebung integriert, wobei ähnliche Schritte wie beim Java-Client befolgt werden. Die Auswahl der geeigneten Bibliotheken und Module, wie beispielsweise "python-socketio", ist entscheidend, um eine reibungslose Kommunikation zwischen dem Python-Client und dem Server sicherzustellen. Es ist wichtig, dass die verwendeten Versionen und Konfigurationen miteinander kompatibel sind, um potenzielle Konflikte und Probleme während der Implementierung zu vermeiden.

Um eine erfolgreiche Implementierung von socket.io in das Multiplayer Racing Game zu ermöglichen, müssen verschiedene Aspekte berücksichtigt werden, um eine nahtlose und effiziente Kommunikation zwischen den Spielern zu gewährleisten.

1. Echtzeitkommunikation: Durch die Verwendung von socket.io kann eine Echtzeitkommunikation zwischen den Spielern und dem Server etabliert werden. Dies ermöglicht eine schnelle und zuverlässige Übertragung von Spielstatus, Spieleraktionen und anderen relevanten Informationen.

2. Spielerinteraktion: Die Integration von socket.io ermöglicht es den Spielern, in Echtzeit miteinander zu interagieren, sei es durch Chats, Emotes oder andere Formen der Kommunikation. Dies fördert das Gemeinschaftsgefühl und verbessert das Spielerlebnis.

3. Synchronisation des Spielzustands: Mithilfe von socket.io kann der Spielstand zwischen den verschiedenen Clients und dem Server synchronisiert werden, um sicherzustellen, dass alle Spieler eine konsistente und aktuelle Ansicht des Spiels haben. Dies ist besonders wichtig, um Unstimmigkeiten und Verzögerungen zu vermeiden.

4. Mehrspieler-Modus: Mit socket.io kann ein reibungsloser Mehrspieler-Modus implementiert werden, der es den Spielern ermöglicht, in Echtzeit gegeneinander anzutreten. Durch die schnelle Übertragung von Spielaktionen und -ereignissen können die Spieler ein intensives und interaktives Spielerlebnis genießen.

5. Lobby-Management: Die Verwendung von socket.io erleichtert die Verwaltung von Spiellobbys, indem es eine nahtlose Kommunikation zwischen dem Server und den Clients ermöglicht. Dies umfasst Funktionen wie das Erstellen, Beitreten und Verlassen von Spiellobbys sowie die Verwaltung von Spielerlisten und Spielparametern.

6. Fehlerbehandlung und Wiederherstellung: Durch die Integration von socket.io kann auch eine effektive Fehlerbehandlung und Wiederherstellung implementiert werden, um sicherzustellen, dass die Spieler bei eventuellen Verbindungsproblemen oder Unterbrechungen des Spielverlaufs nahtlos wieder in das Spiel einsteigen können, ohne dabei den Spielfortschritt zu verlieren.

Durch eine sorgfältige Planung und Implementierung von socket.io können diese Aspekte in das Multiplayer Racing Game integriert werden, um ein interaktives und reibungsloses Spielerlebnis zu gewährleisten, das den Anforderungen eines ansprechenden Multiplayer-Spiels gerecht wird.

Zusatz:

Mithilfe von Node.js kann ein Webserver erstellt werden, auf welchen durch die Socket-Verbindung zugegriffen werden kann.
Zur Synchronisierung muss entschieden werden, welche Daten zwischen den Clients ausgetauscht werden (Request and Response) und in welchem Format diese versendet werden sollen. Ebenso ist es wichtig, dass der Empfänger die Daten verarbeiten und einbinden kann. Die Definition und Auslegung der Kommunikationsprotokolle stellt den Schwerpunkt der Aufgabe dar. 7

Technik Python/Java

Wir verwenden verschiedene Programmiersprachen für unser Projekt. Für einen Client nutzen wir Python, während wir für den anderen Client Java einsetzen. Unser Source-Code wird über GitHub geteilt.Die Projekttagebücher, Meetingprotokolle und die Projektdokumentation sind in einem gesonderten Repository verfügbar.

Jeder Entwickler hat unterschiedliche Repositories für Python, Java und den Server. Diese Repositories sind auf dem Server verfügbar, und jeder Entwickler hat zudem ein lokales Repository auf seinem Rechner.

Die Testfälle werden in einzelnen Testklassen erstellt und durchlaufen. Wir planen, das Softwareprodukt entweder über einen lokalen Server oder über einen Cloud-Dienst bereitzustellen.

Es gibt Schnittstellen zwischen den Softwaremodulen, hierfür nutzen wir socket.io als Schnittstelle für die Kommunikation zwischen Client und Server. Für die Kommunikation verwenden wir Websockets, HTTP und JSON.

Wir verwenden verschiedene Software-Werkzeuge, darunter PyCharm als IDE, GitHub als Versionskontrollsystem und PyGradle für Tests und Qualitätssicherung. In Java wird äquivalent dazu VsCode und IntelliJ als Entwicklungsumgebung genutzt, sowie JUnit für Tests eingesetzt. Die Versionskontrolle erfolgt ebenfalls über Github. Wir folgen Coding-Richtlinien, um Standards für den Source-Code einzuhalten.

Bei Code-Reviews und Pull Requests werden Funktionalität, Codequalität, Codestyle, Lesbarkeit, Dokumentation, Effizienz und Performance sowie Testabdeckung überprüft. Die Code-Reviews werden von dem jeweils anderen Team durchgeführt und bei Team Meetings kann der Code zusätzlich vorgestellt werden.

Code Review

DOM:

- DOM (Document Object Model) verbindet Webseiten mit Skripten oder Programmiersprachen, indem es die Struktur eines Dokuments (z.B. HTML) im Speicher darstellt.
- Bezieht sich normalerweise auf JavaScript.
- Stellt ein Dokument in Form eines logischen Baums dar.
- Jeder Zweig des Baums endet in einem Knoten und jeder Knoten enthält Objekte.
- An Knoten können „Event Handlers“ angehängt werden, die bei Auslösen von Ereignissen ausgeführt werden.

var Name:

- Deklaration einer Variablen mit dem Namen „Name“, die ein Objekt repräsentiert.
- In JavaScript gibt es kein Konzept von Klassen.
- Es werden Objekte und Funktionen verwendet.
- Die Funktionen und Eigenschaften in „Name“ gehören zum Objekt.
- Zugriff durch Punktoperator.
- In Java kann eine ähnliche Struktur durch Klassen mit zugehörigen Methoden erstellt werden.

Dateiname: common.js

Abschnitt „minimalist DOM helpers“:

- DOM-Interaktionen und Event-Listener können durch die Verwendung von Swing oder JavaFX in Java emuliert werden.

Abschnitt „general purpose helpers“:

- Enthält verschiedene Hilfsfunktionen im Zusammenhang mit Mathematik und allgemeinen Operationen.
- Die Funktionen können in Java durch Methoden simuliert werden.

Abschnitt „POLYFILL for requestAnimationFrame“:

- Ein Polyfill ist eine Technik, die in älteren Webbrowsern fehlende Funktionen nachbildet oder ergänzt.
- „requestAnimationFrame“ wird verwendet, um Animationen auf eine möglichst effiziente Weise durchzuführen.
- Die Funktionalität kann in Java durch die „java.util.Timer“-Klasse emuliert werden, welche die Animation in regelmäßigen Intervallen ausführt.

Abschnitt „GAME LOOP helpers“:

- Enthält Hilfsfunktionen und ein Objekt namens „Game“, das bei der Erstellung von Schleifen im Spiel und bei der Handhabung von Ressourcen und Eingaben hilft.
- Die Funktionalität kann in Java durch ein oder mehrere Klassen emuliert werden, welche die Spiellogik und Spielsteuerung enthalten.
- Verwendung einer Schleife, um das Spiel zu aktualisieren und zu rendern.
- Methoden für das Laden von Bildern und Ressourcen.
- Methoden für die Behandlung von Tastaturereignissen.
- Methoden zur Überwachung der Spielleistung.

Abschnitt „canvas rendering helpers“:

- Enthält Hilfsfunktionen für das Rendern von 2D-Grafiken in einem Canvas-Element.
- Die Funktionen sind speziell auf die Anforderungen eines 2D-Rennspiels zugeschnitten und ermöglichen das Zeichnen von Straßen, Autos, Hintergründen usw.
- In Java eine Klasse erstellen, die ein „Graphics2D“-Objekt verwendet und die Operationen zum Zeichnen durchführt.
- Verwendung von Swing oder JavaFX.

Abschnitt „RACING GAME CONSTANTS“:

- Definition von Konstanten und Bild-Sprites
- In Java können die Ressourcen wie Bilder als separate Daten gespeichert und geladen werden.
- Verwendung von Swing / AWT zum Zeichnen der Bilder und Grafiken**.**

Dateiname: stats.js

- Enthält eine Leistungsüberwachungsbibliothek namens „Stats“.
- Wird in Webanwendungen verwendet, um die Bildwiederholungsrate (FPS) und die Zeit in Millisekunden (MS) zwischen Frames anzuzeigen.
- In Java kann Swing / AWT verwendet werden, um Elemente für die Benutzeroberfläche bzgl. der Anzeige von FPS und MS zu erstellen.
- Weitere Möglichkeit ist die Verwendung eines Timers, um die Aktualisierung der Anzeige in regelmäßigen Intervallen zu steuern.