SEG BERICHT

The Blinking LightHouse 3000

Micha Burkhardt, Nico Johnsen, Alexander Neumann & Erik Nissen

Inhalt

Exposé

Wir wurden von der Evil Corporation beauftragt Unterhaltungssoftware zu entwerfen, die die Gestik des Benutzers erfasst und es ihm ermöglicht auf einer Hauswand, in deren Fenstern LEDs befestigt wurden, zu zeichnen. Umgesetzt wird die Erfassung der Gestik mithilfe eines Leap Motion Controllers (LMC). Die so erfassten Daten werden über das OSC-Protokoll an das Lighthouse übertragen.

Primär sollen die Positionsdaten des Zeigefingers des Nutzers erfasst werden, um eine hohe Präzision zu gewährleisten. Je nach Position des Zeigefingers leuchten die LEDs des Lighthouse für eine gewisse Zeit auf, wodurch das Zeichnen oder Malen auf dem Lighthouse ermöglicht wird.
Um eine reibungslose Kommunikation der verschiedenen Hardware-Elemente zu erreichen, wird für dieses Projekt die Programmiersprache Java zum Einsatz kommen.

Die Umsetzung der Projektidee wird durch einige Teilprobleme erschwert. So dürfen wir aufgrund der beschränkten Netzwerkkapazität nicht mehr als 50 Signale in der Sekunde versendet werden. Lösen lässt sich das Problem, indem wir die Geschwindigkeit, in der der Nutzer zeichnen kann, beschränken.

Des Weiteren müssen sowohl Lighthouse als auch der LMC über den gleichen Koordinatenbereich verfügen. Dies können wir über eine einheitliche Skalierung beider Geräte erreichen.

Zu Fehlern kann es außerdem kommen, wenn sich mehrere Hände im Erfassungsbereich des LMC befinden. Eine mögliche Lösung wäre die Deaktivierung der Erfassung weiterer Hände.

Aufwandsabschätzung

Exposé:	1 Manntag
User Stories:	1 Manntag
Lastenheft:	2 Manntage

Use Cases

Die folgenden Anwendungsfälle decken die wichtigsten Funktionen der Software ab:

Titel: Auf dem Lighthouse zeichnen
• **Kurzbeschreibung:** Als Nutzer möchte ich mit meinem Zeigefinger verschiedene LEDs an einer Hauswand ansteuern können.
• **Akteure:** Benutzer
• **Vorbedingungen:** Kommunikation zwischen LMC und Lighthouse ist sichergestellt
• **Beschreibung des Ablaufs:** Der Benutzer hält seine Hand in den Erfassungsbereich des LMC. Sein Zeigefinger wird erfasst und entsprechend den erfassten Positionsdaten, leuchtet eine LED auf dem Lighthouse. Das Programm differenziert zwischen den Zeigefingern der linken und rechten Hand und stellt dem entsprechend unterschiedliche Zeichenmodi zur Verfügung.
• **Auswirkungen:** Zeichnet der Nutzer mit dem rechten Zeigefinger, so bleiben die angesprochenen LEDs aktiviert, bis ein manueller Reset vom Nutzer ausgeführt wird. Wird der linke Zeigefinger genutzt, so werden die LEDs nach einer gewissen Zeit automatisch wieder ausgeschaltet.

Titel: Zeichenfarbe für das Lighthouse wählen
• **Kurzbeschreibung:** Der Nutzer beeinflusst die Farbe des LightHouse
• **Akteure:** Benutzer
• **Vorbedingungen:** Die Anwendungsanforderung „Auf dem Lighthouse zeichnen“ muss erfüllt sein
• **Beschreibung des Ablaufs:** Während der Nutzer die LEDs über die X- und Y-Achsen ansteuert, kann er mit der Z-Achse die Helligkeit der Farbe beeinflussen
• **Auswirkungen:** Neu angesprochene LEDs leuchten in der gewählten Farbe, bis durch Veränderung der Z-Position eine andere gewählt wird.

Titel: Zeichnung resetten
• **Kurzbeschreibung:** Der Nutzer gibt den Befehl, alle LEDs zurückzusetzen, um seine Zeichnung zu löschen.
• **Akteure:** Benutzer
• **Vorbedingungen:** Eine Zeichnung muss auf dem Lighthouse vorhanden sein
• **Beschreibung des Ablaufs:** Der Nutzer führt mit seiner Hand eine Wischbewegung im Erfassungsbereich des LMC aus.
• **Auswirkungen:** Alle LEDs werden ausgeschaltet

Test-Cases

Im Rahmen des Test Driven Development wurden bereits früh in der Entwicklung Testfälle basierend auf den bereits vorgestellten Use Cases generiert. Dabei wurde die Funktionsweise der einzelnen Use Cases genau analysiert und auf eventuelle Testbarkeit geprüft. Daraus entstanden sind insgesamt vier Testfälle. Folgend ist ein Beispiel dargestellt, welches sich mit der Elementarfunktionalität des Zeichnens auf dem Lighthouse befasst. Weitere Testfälle befinden sich im Anhang.

1. Auf dem Lighthouse zeichnen
Vorbedingungen: Gestenerkennung funktioniert und Daten werden an Lighthouse übermittelt
Testobjekt: Lighthouse
Eingabedaten: Positionsdaten eines Zeigefingers
Durchführung: Testperson hält einen Zeigefinger in den Erfassungsbereich des LMC und bewegt ihn an verschiedene X/Y-Positionen.
Erwartetes Ergebnis: Entsprechend der Positionsdaten leuchten die zugehörigen LEDs auf dem Lighthouse.
Prüfanweisungen: Entsprechen die eingelesenen Positionsdaten des Zeigefingers genau den Koordinaten der aufleuchtenden LEDs?

Lastenheft

Zielbestimmung

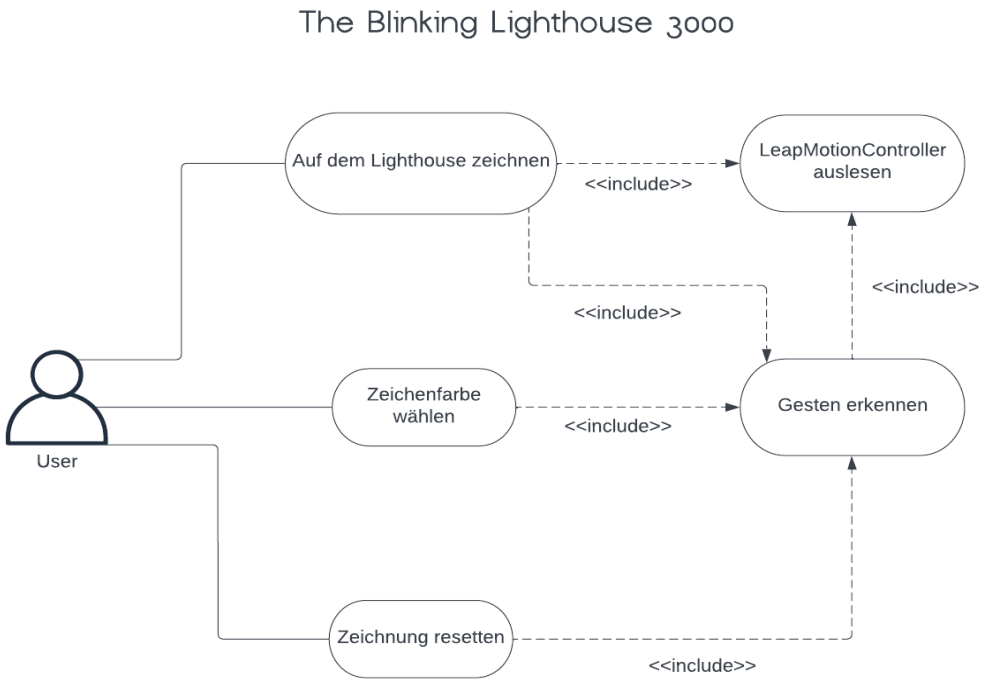
Die Firma EvilCorp erwartet ein vorzeigbaren Prototyp, dem der Leap Motion Controller und das Lighthouse zugrunde liegen.

Produkteinsatz

Das Endprodukt dient der Unterhaltung. Zielgruppe sind alle Altersklassen und Personengruppen. Kunden können die Hardware des Leap Motion Controllers kennenlernen und mit diesem auf dem Lighthouse zeichnen.

Produktübersicht

Zur besseren Visualisierung der erstellten Use Cases haben die Projektteilnehmer diese in ein Diagramm eingearbeitet. Zusätzlich zu den drei beschriebenen, wichtigsten Use Cases erscheinen im Diagramm noch die Punkte „Leap Motion Controller auslesen“ und „Gesten erkennen“, welche elementare Bestandteile der übergeordneten Use Cases sind und das Zusammenspiel verdeutlichen.



Produktfunktionen

/LF10/ **Prozess:** Zeichenmodus eins

Akteur: Benutzer

Beschreibung: Der Bediener möchte mit seinem Zeigefinger auf dem Lighthouse zeichnen, wobei die gezeichnete Linie nach kurzer Zeit wieder verschwindet.

/LF20/ **Prozess:** Zeichenmodus zwei

Akteur: Benutzer

Beschreibung: Der Bediener möchte mit seinem Zeigefinger auf dem Lighthouse zeichnen, wobei die gezeichnete Linie bestehen bleibt.

/LF30/ **Prozess:** Reset der Zeichnung

Akteur: Benutzer

Beschreibung: Mit einer Geste ist es dem Benutzer möglich die erstellte Zeichnung zu löschen

/LF40/ **Prozess:** Farbwahl

Akteur: Benutzer

Beschreibung: Indem der Nutzer die Z-Achse beeinflusst, ist er in der Lage die Farbe zu beeinflussen.

Produktdaten

Durch den theoretischen Charakter der Veranstaltung sind keine Produktdaten vorhanden

Produktleistungen

/LL10/ Die Funktionen /LF10/ und /LF20/ hat eine so geringe Verzögerung, dass der Nutzer damit umgehen kann.

Qualitätsanforderungen

Zur besseren Einordnung der Prioritäten im Entwicklungsverlauf wurden die Qualitätsanforderungen an das Produkt visuell in einer Tabelle dargestellt.

Produktqualität	Sehr gut	Gut	Normal	Nicht relevant
Funktionalität	X			
Zuverlässigkeit	X			
Effizient		X		
Änderbarkeit			X	
Übertragbarkeit			X	

Ergänzungen

Benutzer müssen mindestens eine Hand haben

Systemmodellierung – Grobentwurf

Aufwandsabschätzung

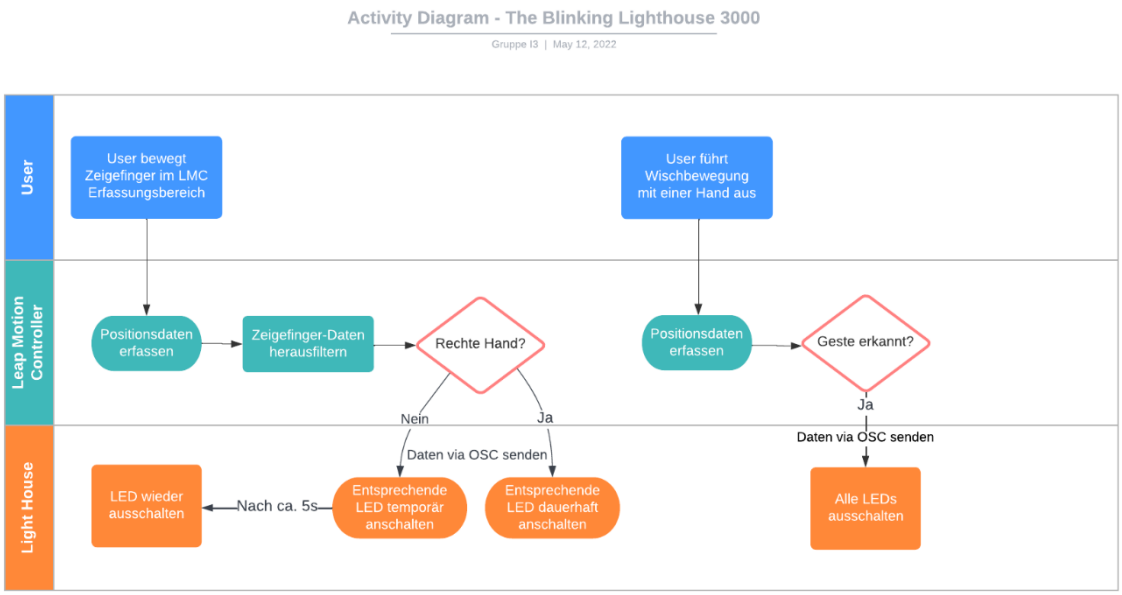
Aufwandsabschätzung für den Grobentwurf in Manntagen:
Verhaltensdiagramm – 0,5 Manntage

OOA: 0,25 Manntage

Strukturdiagramm: 0,75 Manntage

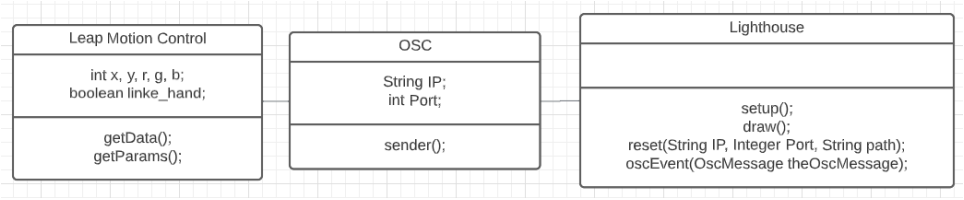
Klassendiagrammpakete: 0,1 Manntage

Verhaltensdiagramm



Das Activity Diagram visualisiert die möglichen Benutzerinteraktionen mit der Software, sowie die darauffolgenden inneren Abläufe. Eine Interaktion erstreckt sich dabei immer über die Ebenen „User“, „Leap Motion Controller“ und „Lighthouse“.

Klassendiagramm

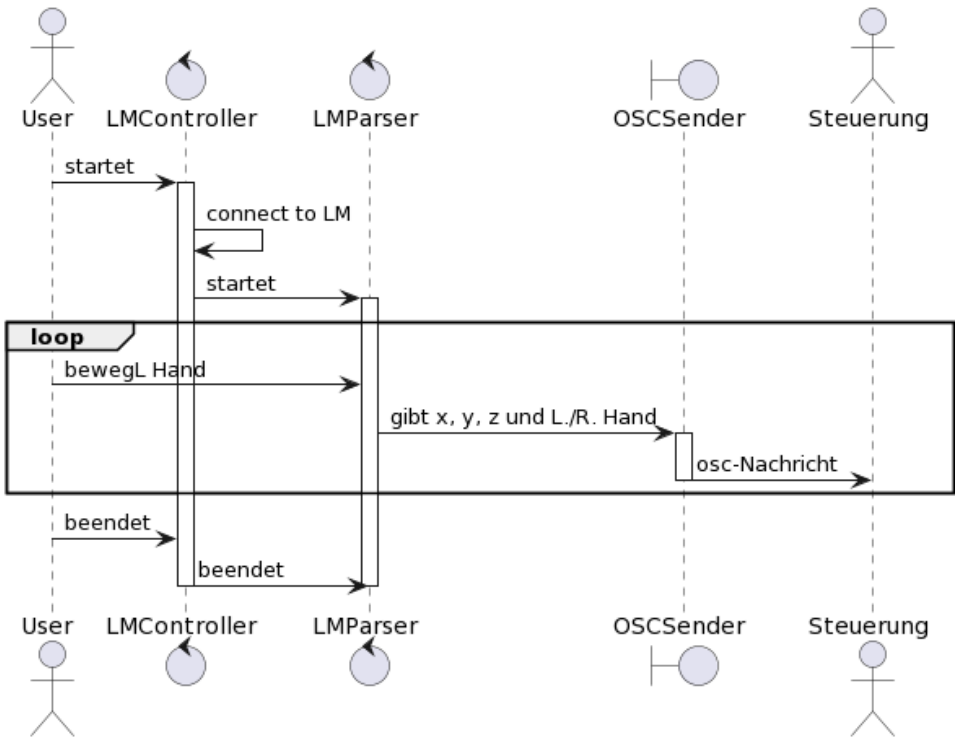


Im Rahmen einer objektorientierten Analyse wurde zunächst eine grobe Klassenstruktur erstellt. Diese plant zunächst jeweils eigenen Klassen für die verwendete Hardware sowie für das Netzwerkprotokoll OSC ein.

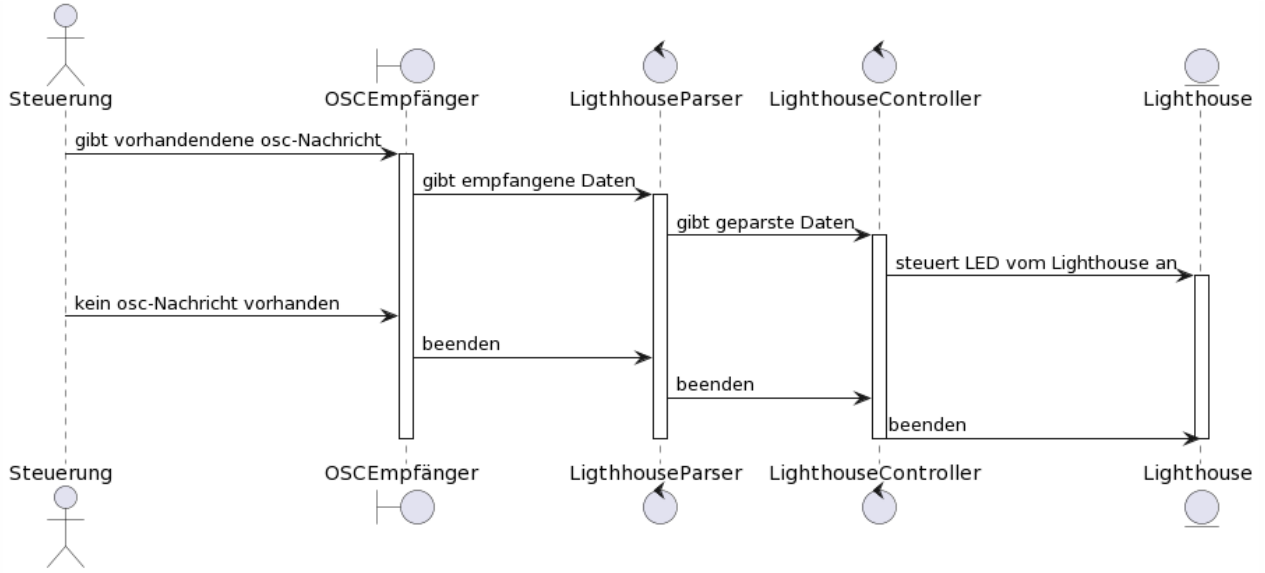
Die Leap Motion Klasse dient hierbei dazu, die eingelesenen Positionsdaten einer Hand zwischenzuspeichern und auszuwerten, um sie anschließend via OSC-Sender an das Lighthouse weiterzureichen. Letzteres verfügt entsprechend über Methoden, um anhand der empfangenen Daten bestimmte LEDs anzusteuern oder alle LEDs zu resetten.

Sequenzdiagramme

Im Folgenden wird ein typischer Programmablauf anhand zweier Sequenzdiagramme abgebildet. Zunächst wird eine Nutzereingabe, das Entgegennehmen und Auswerten dieser durch den Leap Motion Controller, sowie das anschließende Weitersenden über OSC visualisiert:

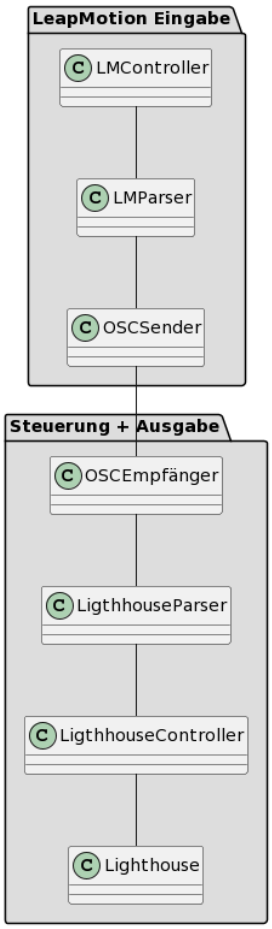


Im folgenden Diagramm ist das Empfangen der OSC-Nachricht durch das Lighthouse, sowie das anschließende Parsen und Ausgeben in Form leuchtender LEDs dargestellt:



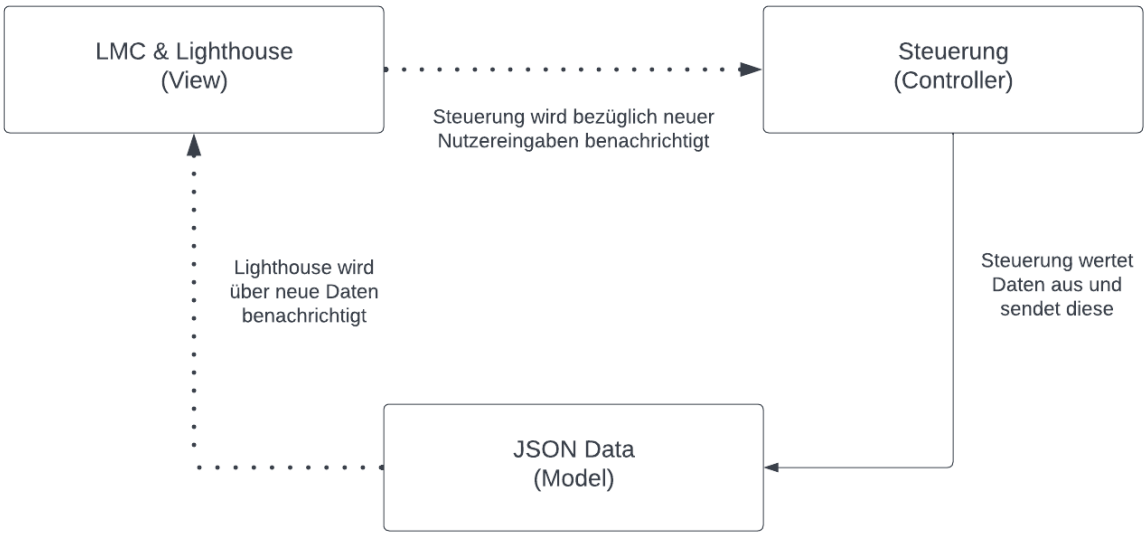
Klassenpakete

Im Zuge der Ausarbeitung der Sequenzdiagramme wurde die Klassenstruktur um weitere Klassen erweitert und anschließend in zwei Pakete unterteilt:



Systemmodellierung – Feinentwurf

Architekturmuster

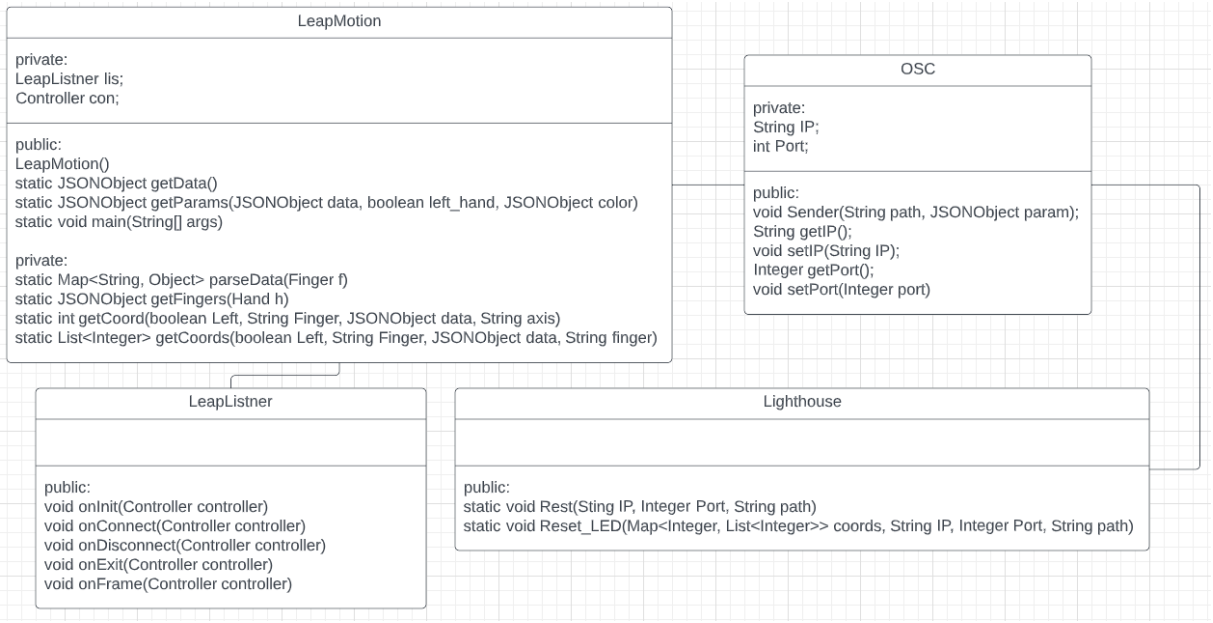


Die Struktur der Software kann mithilfe des Architekturmusters „Model-View-Controller“ abgebildet werden. Leap Motion Controller und Lighthouse bilden als Kombination aus Nutzer Ein- und Ausgabe die View, die zwischengeschaltete Steuerung verwaltet als Controller entsprechend den eingelesenen Eingaben die zu sendenden JSON-Daten, welche das Model darstellen.

Ausgearbeitetes Klassendiagramm

Im Zuge der vorangeschrittenen Planung, sowie den ersten Schritten der Implementierung, wurde das Klassendiagramm deutlich verfeinert.

Die einzelnen Klassen wurden um neue Attribute und Methoden erweitert, während vereinzelte Methoden verschoben oder entfernt wurden



Entwicklung

Aufwandsabschätzung

Zu Beginn der Entwicklung wurden Aufwandsabschätzungen basierend auf dem verfeinerten Klassendiagramm vorgenommen. Im Zuge der laufenden Entwicklung wurden die tatsächlichen Aufwände in das Diagramm übertragen. So ist es möglich einen direkten Vergleich vorzunehmen. Wie man dem Gantt Diagramm entnehmen kann, lag das Team mit seinen Schätzungen überwiegend sehr dicht am real erfassten Aufwand. Die größte Differenz entstand bei der Implementierung der Tests, da der Aufbau eines Unit Tests in der Literatur als sehr simpel dargestellt wurde, den Programmierern bei der Übertragung auf das aktuelle Projekt jedoch sehr schwergefallen ist.

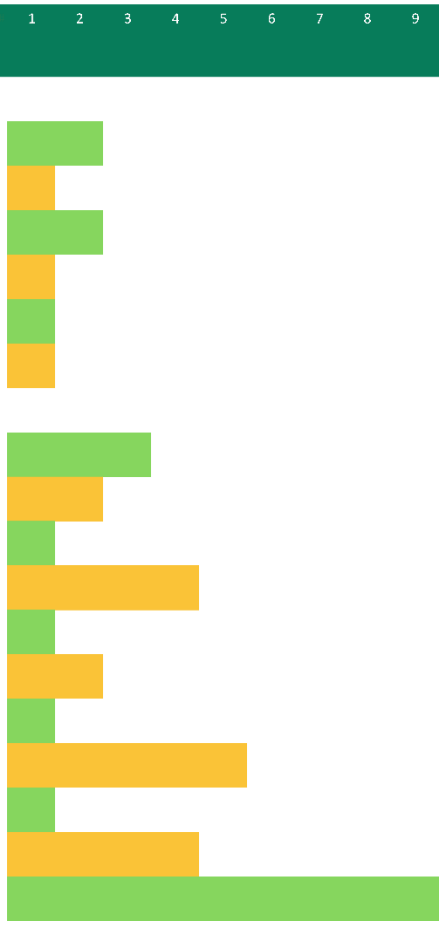
LightHouse3000

Evil Corp

Projektanfangsdatum: 28.03.2022

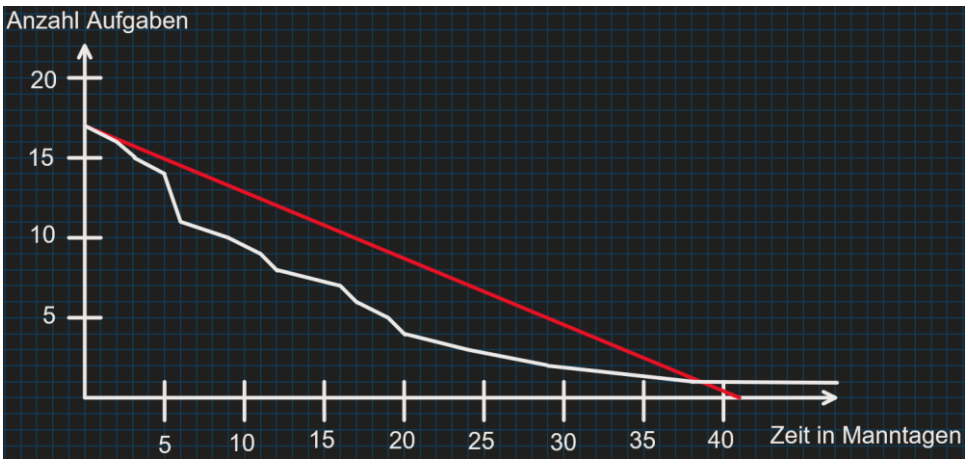
Meilensteinmarkierung: 1

Meilensteinbeschreibung	Zugewiesen zu	Fortschritt	Tage
Vorbereitung			
Recherche Leap Motion		100%	1
Recherche OSC		100%	1
Einrichtung IDE		100%	1
Treiber installieren		100%	1
LH Emulator einrichten		100%	1
Einrichtung Repo		100%	1
Programmierung			
LMC Handerfassung		100%	2
LMC Koordinatenverarbeitung		100%	3
OSC Sender implementieren		100%	2
Daten formatieren		100%	3
Daten senden		100%	1
Zeichenmodus 1		100%	4
Zeichenmodus 2		100%	2
Farbe auswählen		100%	5
Gesten erkennen		50%	2
Farbraum konvertieren		100%	3
Tests implementieren		27%	6



Burndownchart

Über ein jederzeit für alle Teilnehmer zugängliches Burndownchart wurde der Projektfortschritt jederzeit überwacht. Das Zeitmanagement war über die gesamte Projektdauer überdurchschnittlich erfolgreich. Das Team sah sich bis zuletzt gut im Zeitplan, jedoch haben externe Einflüsse und die Komplexität der Tests dazu geführt, dass ein einzelnes Feature nicht implementiert werden konnte.



Probleme

Bei der Entwicklung traten diverse Komplikationen auf, die an dieser Stelle erläutert werden. Zu Beginn bestand das Team aus insgesamt fünf Personen. Die Einteilung der Aufgabenbereiche geschah zügig, da sich schnell zwei Personen gefunden haben, die sich eher auf die Programmierung als auf das Schreiben des Berichtes konzentrieren wollten. Leider wurden drei vereinbarte Termine durch einen Entwickler nicht wahrgenommen. Somit wurde nach einer langen Besprechung entschlossen, jenen Entwickler aus dem Team zu entfernen. Daraus resultierend war nur noch ein Entwickler übrig, welcher die Entwicklung des Programmes fortgeführt hat, jedoch von den anderen unterstützt worden ist, falls diese Zeit dafür aufbringen konnten. Leider wurde er aufgrund der hohen Belastung des Berichtsteams nicht in dem ihm zustehenden Maße unterstützt.

Da es an Zeit mangelte, war es dem Team nicht möglich den Reset der Zeichnung mit einer Wischgeste zu implementieren. Stattdessen muss sich das Team zum jetzigen Zeitpunkt damit zufriedengeben, dass die Zeichnung in Zeichenmodus zwei dadurch zurückgesetzt wird, wenn die linke Hand anstatt der rechten in den Erfassungsbereich des Sensors bewegt wird.

Zusätzlich hat sich das Projektteam mit der Implementierung der Tests schwergetan. Nachdem das Testframework AssertJ nicht zu den speziellen Wünschen des Teams gepasst hat, ist die Wahl auf JUnit gefallen. Nach einiger Einarbeitung wurde der erste, einfache Unit Test erstellt. Die Komplexität der erstellten Funktionen, erlaubte es den Entwicklern jedoch nicht wie geplant voranzuschreiten. So wurde eine Testabdeckung von 27% erreicht. Obwohl das Team somit den Anforderungen Genüge getan hatte, lag das Ziel, dass sich das Team gesetzt hat, deutlich höher.

Testen

Code Review

In Zusammenarbeit des Hauptentwicklers und eines weiteren Teammitglieds wurden Code Reviews für die wesentlichen Programmabschnitte durchgeführt. Dabei versuchte der Reviewer, den Code nachzuvollziehen und Verbesserungsvorschläge anzubringen. Diese konnten zumeist gleich im Anschluss von dem Entwickler umgesetzt werden. Dazu gehörten im Wesentlichen:

- Das Wählen sprechender Variablennamen
- Das Aufteilen komplexer Funktionalität auf mehrere, kleinere Methoden
- Das Vereinfachen einiger If-Abfragen
- Das Hinzufügen von methodenbeschreibenden Kommentaren

Im Zuge der Code Reviews war er es den Reviewern zudem möglich, ein besseres Verständnis des Programmcodes zu erhalten und einige offene Fragen in Bezug auf die Implementierung konnten in Absprache mit dem Entwickler geklärt werden.

Code Review Planung

Das Codereview fand jeden Mittwoch mit jeweils unterschiedlichen Teilnehmern statt. Dieser Zeitraum ist gewählt worden, da sich das Projektteam immer mittwochs getroffen hat und somit jede Codezeile direkt geprüft werden konnte. Die folgende Grafik zeigt den Review Zeitpunkt und die Teilnehmer

	Mittwoch	27.04	01.05	11.05	18.05	25.05	01.06	08.06	15.06
Teilnehmer									
Micha		X	X		X	X		X	X
Alex			X	X		X	X		
Nico		X		X	X		X	X	X
Erik (Prog.)		X	X	X	X	X	X	X	X

Test Checklisten

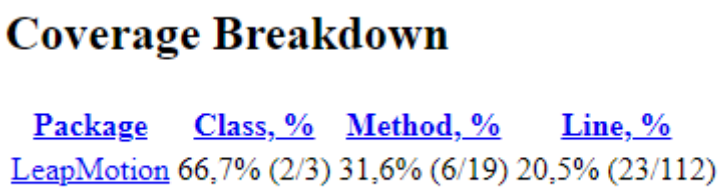
Um ein einheitliches, durchgängiges und nachvollziehbares Ergebnis zu erhalten, wurde eine Checkliste für die Code Reviews erstellt. Diese wurde zusammen mit dem Entwickler während eines Reviews durchgearbeitet. Das Review wurde erst beendet, nachdem alle vom Projektteam definierten Punkte als erfüllt betrachtet werden konnten. Orientiert wurde sich an allgemeinen Clean Code vorgaben, wie der maximalen Zeichenlänge einer Codezeile oder der Nachvollziehbarkeit von einzelnen Funktionen. Die vollständige Checkliste befindet sich im Anhang.

Es hat sich herausgestellt, dass die Erstellung und Bearbeitung der Liste von großem Wert für die Lesbarkeit und vor allem Nachvollziehbarkeit des Codes war. So wurde keines der in der Liste vorhandenen Ziele nach dem ersten Review erreicht und es musste direkt nachgebessert werden, um das Review erfolgreich abzuschließen. In den nachfolgenden Iterationen wurde das Ziel wesentlich schneller erreicht.

Unit Tests

Die Implementierung der Tests erfolgte mit dem Testframework JUnit. Konzentriert wurde sich auf die Klasse LeapMotion.java. Nach einiger Einarbeitungszeit konnte

eine Testabdeckung von 27% erreicht werden. Der komplette Testbericht befindet sich im Anhang.



Coverage Breakdown

Package	Class, %	Method, %	Line, %
LeapMotion	66,7% (2/3)	31,6% (6/19)	20,5% (23/112)

Fazit

Zu Beginn des Projektes wurde die Problemstellung genau analysiert und im Exposé niedergeschrieben. Außerdem sind hier erste Probleme identifiziert worden, die das Team im Laufe des Projektes lösen musste. Die größte Herausforderung dabei war es, sich erst Gedanken über das Projekt zu machen und nicht sofort mit der Implementierung zu starten, wie es das Team bisher gewohnt war. Um mit dieser Problematik besser umgehen zu können, wurden Use Cases entwickelt, die die Funktionalität des Produktes zusammenfassen und verdeutlichen sollten. Somit wuchs auch die konkrete Idee in den Köpfen und das Team war in der Lage das Ziel gemeinsam anzugehen.

Im Rahmen des Test Driven Development wurden direkt Testfälle entwickelt. Hier hat sich gezeigt, dass das Team zwar schnell in der Lage war, mögliche Testfälle zu identifizieren, jedoch bis zuletzt Schwierigkeiten dabeihatten, diese im Projekt umzusetzen bzw. zu implementieren. Direkt im Anschluss wurden die ersten Diagramme zum Bericht hinzugefügt. Aufgrund fehlender Erfahrung im Softwareengineering hatte das Team große Probleme bei der Erstellung dieser. Die Diagramme haben sehr viel Zeit in Anspruch genommen, da das Team lange gezögert hat, externe Hilfe dazu in Anspruch zu nehmen. Nachdem diese Hürde jedoch einmal gemeistert wurde, haben die Teammitglieder direkt nach Hilfe gesucht, wenn sie nicht weiterkamen. So ist dem Team auch die Durchgestaltung von Fein- und Grobentwurf wesentlich leichter gefallen und die geschätzte Aufwandsdauer wurde eingehalten.

Nachdem die Programmierung ohne allzu große Hindernisse abgeschlossen wurde, traten bei der Implementierung der Unit Tests sehr viel größere Probleme auf. Niemand in der Gruppe hatte bisher einen Test schreiben müssen oder auch nur einen implementierten Test gesehen. Der überproportional große Zeitaufwand, der in die Durchdringung und Erstellung von Unit Tests investiert wurde, resultierte in einer geringen Testabdeckung. Das Team hat sich dazu entschieden auf eine hohe Quantität zu verzichten und den Lernerfolg durch qualitativ hochwertigere Tests zu verstärken.

Zusammenfassend konnten viele Probleme im Laufe des Projekts durch Teamarbeit und Offenheit gegenüber externen Quellen gelöst werden. So war es dem Team möglich ein funktionierendes Endprodukt zu kreieren, auch wenn das Feature des Gestenresets bisher noch fehlt.

In zukünftigen Iterationen würde das Team gerne weitere Zeichenmodi hinzufügen sowie weiter an der Qualität des bisher existierenden Produktes arbeiten. Dabei jedoch das Test Driven Development mehr in den Fokus nehmen.

Anhang

Weitere Test Cases

2. Verschiedene Zeichenmodi (linke oder rechte Hand)
Vorbedingungen: Gestenerkennung funktioniert und Daten werden an Lighthouse übermittelt
Testobjekt: Lighthouse
Eingabedaten: Positionsdaten des richtigen Zeigefingers werden erkannt
Durchführung: Testperson hält entweder den linken oder rechten Zeigefinger in den Erfassungsbereich des LMC und bewegt ihn an verschiedene X/Y-Positionen.
Erwartetes Ergebnis: Entsprechend der Positionsdaten und der Erkennung des richtigen Fingers leuchten die zugehörigen LEDs auf dem Lighthouse.
Prüfanweisungen: Wird der richtige Zeichenmodus ausgewählt, wenn der rechte bzw. linke Finger in den Erfassungsbereich gehalten wird

3. Zeichenfarbe auswählen
Vorbedingungen: Test Case 2 erfolgreich abgeschlossen
Testobjekt: Lighthouse
Eingabedaten: Positionsdaten eines Zeigefingers
Durchführung: Testperson hält einen Zeigefinger in den Erfassungsbereich des LMC und bewegt ihn an verschiedene Z-Positionen.
Erwartetes Ergebnis: Die aktuelle Zeichenfarbe des Lighthouse ändert sich
Nachbedingungen: Ändert der Nutzer nun die X/Y-Position seines Zeigefingers, so werden nachfolgende LEDs in der neuen Farbe erleuchtet
Prüfanweisungen: Hat sich die Zeichenfarbe nach Veränderung der Z-Position verändert?

4. Zeichnung auf dem Lighthouse resetten
Vorbedingungen: Test Case 2 erfolgreich abgeschlossen
Testobjekt: Lighthouse
Eingabedaten: Geste mit einer Hand (Wischbewegung)
Durchführung: Testperson führt im Erfassungsbereich des LMC eine Wischbewegung mit einer Hand aus, nachdem eine beliebe Zeichnung auf dem Lighthouse erstellt wurde.
Erwartetes Ergebnis: Alle LEDs werden zurückgesetzt, die Zeichnung verschwindet
Nachbedingungen: Durch Eingabe neuer Positionsdaten eines Zeigefingers kann sofort eine neue Zeichnung erstellt werden.
Prüfanweisungen: Wurden alle LEDs vollständig auf den Startzustand zurückgesetzt?

5. Beide Hände gleichzeitig nutzen
Vorbedingungen: Test Case 2 erfolgreich abgeschlossen
Testobjekt: Leap Motion Controller (LMC)
Eingabedaten: Positionsdaten der Zeigefinger beider Hände
Durchführung: Testperson bewegt beide Hände im Erfassungsbereich des LMC
Erwartetes Ergebnis: Die Positionsdaten des Zeigefingers der linken Hand werden priorisiert, solange sich beide Hände im Erfassungsbereich befinden
Nachbedingungen:
Prüfanweisungen: Werden die Daten der linken Hand vom Lighthouse entgegengenommen und korrekt verwertet, während die Daten der rechten Hand ignoriert werden?

Code Review Checkliste

Code Review Checkliste

Datum:
Teilnehmer:

- ☐ Verständliche, nachvollziehbare Formulierung
- ☐ Passende Benennung der Variablennamen
- ☐ Ausreichende Kommentierung
- ☐ Korrekte Einrückung des Codes
- ☐ Keine ausufernden Funktionen
- ☐ Maximale Zeilenlänge 80

Ergänzungen:
Bitte verwenden Sie sprechende Variablennamen wie z.B. addiereAUndB()

Testbericht

Siehe Bitbucket