**Ziel**

Das Ziel des Projekts war die Erstellung eines Frameworks zur Nutzung von Sensordaten von Androidgeräten auf javafähigen Endgeräten.

**Requirements**

Um die Funktionalität des Frameworks zu demonstrieren, haben wir uns zu Anfang die Umsetzung der folgenden drei Anwendung als Ziel gesetzt: Eine Maussteuerung, eine Spielsteuerung und eine Robotersteuerung.

All diese Anwendungen stellen schwer zu definierende Anforderungen an die Latenz und Frequenz der Sensordaten. Wir haben versucht, uns beim festlegen der Grenzwerte auf bekannte Geräte zu beziehen, die keine Probleme bei der Bedienbarkeit haben.

Für die Frequenz der Sensordaten haben wir uns an der Playstation 3 von Sony orientiert: Diese stellt die laufenden Spiele mit 24 Bildern pro Sekunde dar, ein Wert mit dem die meisten Menschen problemlos auch schnelle Actionspiele bedienen können. Daraus entstand für uns die Anforderung, mindestens 30 Updates der Sensordaten pro Sekunde unterstützen zu können.

TODO frequenzgraph

Eine akzeptable Grenze für die Latenz festzulegen war ebenfalls schwierig, da Latenzen im Millisekundenbereich nur mit großem Aufwand zu messen sind. Wir haben uns daher über die Latenzen professionell hergestellter kabelloser Eingabegeräte informiert. Diese haben eine Latenz von ungefähr TODO ms, TODO

TODO latenzgraph

TODO jittergraph

Für die Maussteuerung hatten wir uns folgende Requirements gesetzt: Die Steuerung sollte sehr direkt sein, damit sie die Hauptaufgaben einer Maus übernehmen kann. Das sind Rechts- und Linksklick, damit verbunden der Doppelklick und das Markieren und Bewegen von Dateien, sowie das Bedienen des Kontextmenüs.  
Für die Spielsteuerung haben wir einen SNES-Controller imitiert, mit dem wir das Spiel „Super Mario Kart“ spielen können. Unser Ziel war es auf der 1. Strecke mit der Gesamtzeit eines nativen Controller konkurrieren zu können. Nach einigen Tests ergab sich daraus eine Zeit von 1:20. Für den Mehrspielermodus brauchten wir dann eine Multiclientunterstützung. Um mehr Sensoren zu verwenden haben wir beschlossen die Item Mechanik durch einen weiteren Sensor zu steuern.   
Die Robotersteuerung ist eine Umsetzung des Kuka Roboters „KUKA LBR Iiwa 7 R800“, der ein Murmellabyrinth bedienen soll. Dazu wird er in eine bestimme Position gebracht, und es lässt sich nur das letzte Neigungs-Rotationspaar des Roboterarms bewegen. Dadurch ist es möglich das Labyrinth durch geschickte Neigungen zu lösen.  
Unser oberstes Ziel war es, die App und den Server sehr benutzerfreundlich zu gestalten. Die App sollte sehr leicht bedienbar sein, sodass sehr wenige Konfigurationen notwendig sind. Den App-Code selber muss man dafür nicht verändern, neue Funktionen sind am Server schnell hinzufügbar. So kann man z.B. das Layout für die Buttons, die vom Server gesetzt werden, durch eine XML-Datei anpassen. Für eine angenehme und genaue Steuerung sollte man durch einfache Datenfilter im Server die Sensibilität selber einstellen können. Die Server sollten natürlich auch leicht zu finden, und auch zu wechseln sein. Zur Benutzerfreundlichkeit gehört auch, dass die App Cross-Plattform einsetzbar ist. Unser Server ist auf allen Geräten mit JDK 8 und Zugang zu WLAN-Netzwerken zu nutzen, die App auf allen Android Geräten (Android 4.4 oder höher). Zum Verbinden muss man lediglich den Discovery Port einstellen, für die restlichen Verbindungen werden freie Ports automatisch gesucht.

**Lessons learned**

Es ist deutlich effektiver sich gemeinsam zu festen Zeiten zu treffen um zusammen zu arbeiten. Dadurch haben wir direktes Feedback zu neuen Ideen und deren Umsetzungen von den anderen Teammitgliedern bekommen. Pairprogramming erlaubt Diskussionen und konstruktive Lösungen von neuen Problemen durch andere zu erhalten.  
Uns fiel es schwer messbare Requirements zu erstellen, die das Projekt beschreiben.