# Interaktionskonzept

Um zu Interagieren wurden zwei Bereiche definiert. Der feingranulare Interaktionsbereich (FGI) und der Grobselektionsbereich (GS).

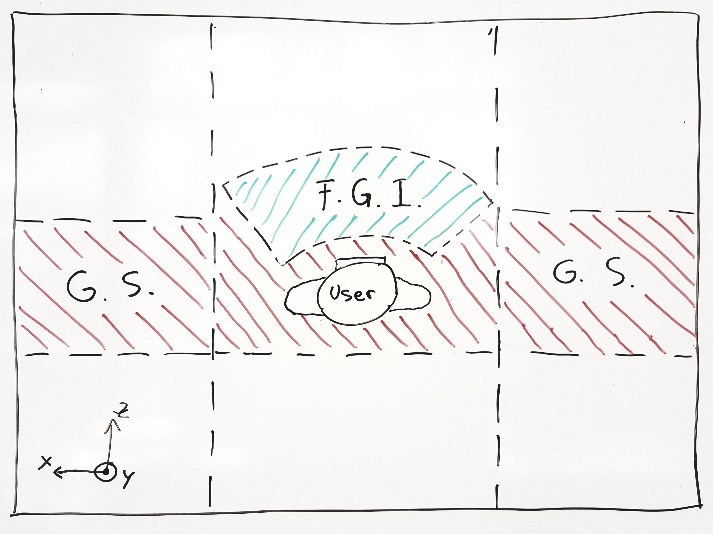


Abbildung 1 Interaktionsbereiche

Im Bereich des FGI liegt der Fokus auf Interaktionen mit Objekten und der Umgebung mit den Händen, deshalb bedarf dieser Bereich hochauflösender Sensorik. Der FGI-Bereich ist immer im Sichtfeld des Users.

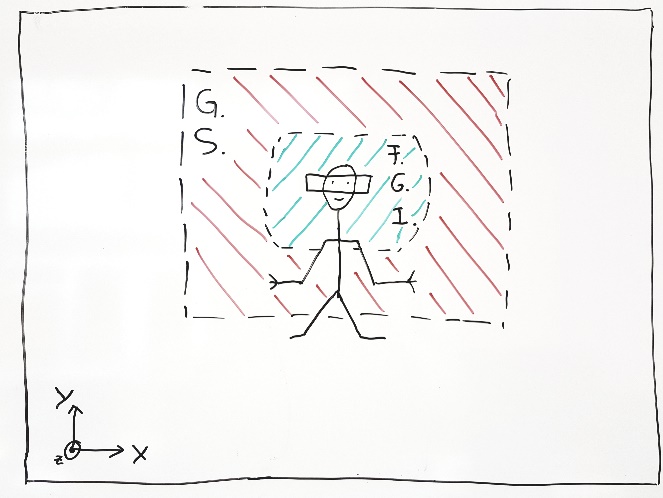


Abbildung 2Abgrenzung GS/FGI

Im Grobselektionsbereich (GS) werden Menüs bedient. Im Bereich des GS soll der ganze Körper verwendet werden, deshalb kann der Bereich mit niedrig auflösenden Sensoren bedient werden.

Es gibt 1 Top-Level-Menü, welches weitere Untermenüs haben kann. Die Menüpunkte können weitere Untermenüs aufrufen oder ein Objekt festlegen welches in der virtuellen Umgebung manifestiert werden soll. Die Menüs werden im Blickfeldmittelpunkt des Users angezeigt und folgen dem Sichtfeld bei Bewegungen des Kopfes.

Durch einfaches schließen der Hand im GS wird das Menü aufgerufen. Um im Menü zu navigieren, werden mit Rotationsbewegungen des Arms Menüpunkte selektiert. Der Menü-Mittelpunkt wird auf die linke Schulter abgebildet. Der Winkel der linken Hand zur linken Schulter bestimmt die Selektion des Menüpunktes.

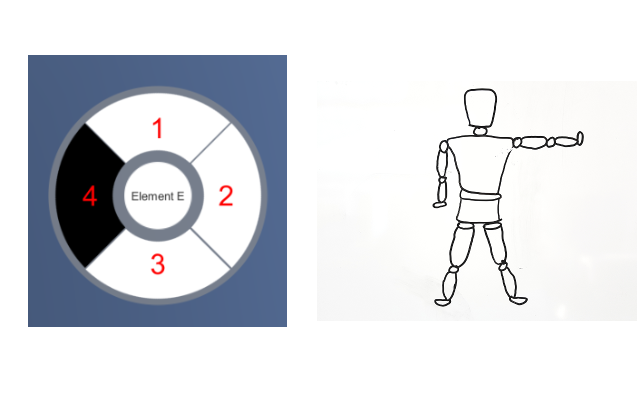


Abbildung 3 Selektion des linken Menüpunktes

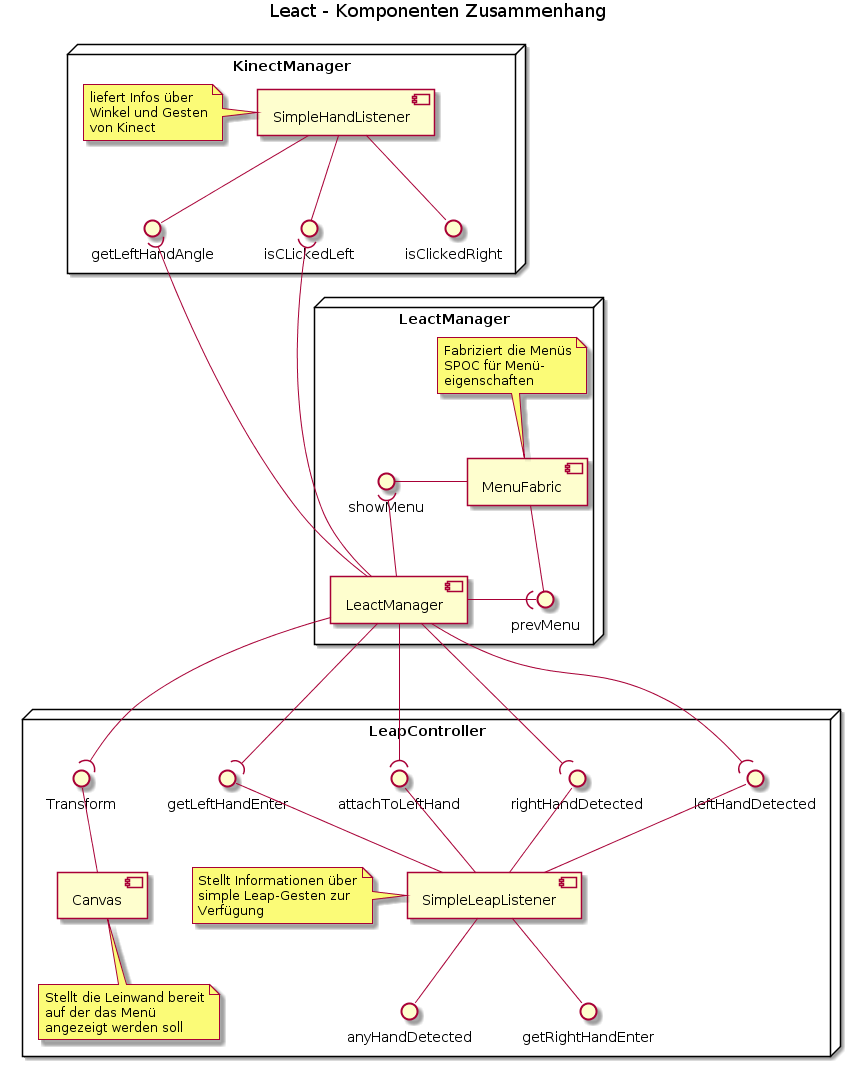
Durch erneutes Schließen der Hand wird ein selektierter Menüpunkt aktiviert. Wenn der selektierte Menüpunkt ein Untermenü enthält, wird das Menü ausgeblendet und das neue Untermenü wird angezeigt. Um zum vorherigen Menü zurück zu kehren, ist das oberste Menüelement mit der „Zurück“-Funktion belegt. Das momentane Menü wird ausgeblendet und das vorherige Menü wieder sichtbar gemacht.

Um Menüs auszublenden, müssen die Hände den FGI-Bereich betreten. Durch Verlassen des FGI-Bereichs und erneutes Schließen der linken Hand wird das zuletzt aufgerufene Menü erneut angezeigt.

Wenn ein Menüelement selektiert und aktiviert wurde, welches kein Untermenü enthält sondern ein Objekt, welches in der Welt manifestiert und manipuliert werden soll, so ist dieses Objekt bei betreten der Hände des FGI, an der linken Hand verankert. Es kann von dort mit der rechten Hand entnommen und weiter interagiert werden.

# Realisierung

Die Realisierung ist in Komponenten aufgeteilt.



## KinectManager

Der KinectManager ist eine Komponente die es erlaubt Kinect V2 und VR gemeinsam zu verwenden. Die Komponente sorgt dafür dass der Körper des Users getrackt wird. Die Kinect V2 wird verwendet um im GS mit Menüs zu interagieren.

Das Skript SimpleHandListener (SHL) ist ein Bestandteil des KinectManager und musste erweitert werden. Es wird die Information über den Winkel von linker Hand zur linken Schulter benötigt. Diese Information ist eine Erweiterung die im Skript umgesetzt werden muss. Die linke Schulter sowie die linke Hand sind Gelenke, welche die Kinect V2 erkennt und somit die Bewegung trackt.

Das Schließen der linken Hand ist als Linksklick definiert. Das Skript muss um die Methode *isClickedLeft* erweitert werden, welche die Information über den Handzustand (geöffnet / geschlossen) bereitstellt.

Generell arbeitet das SHL-Skript mit den Rohdaten der Kamera und stellt vorgefertigte Funktionalitäten als Fassade bereit.

## LeapController

Der LeapController ist ein Bestandteil des Orion-SDKs für LM. Er sorgt für das Tracking und Darstellen der Hände und Unterarme. Die Auflösung der LM ist höher gegenüber der Kinect, weshalb die LM für Objekt Manipulation im FGI-Bereich vorgesehen ist. Aus diesem Grund beschränkt sich auch der FGI Bereich auf das Field of View der LM.

Da der LeapController die Hauptkamera für VR enthält, muss die Kamera eine weitere Fläche zum Darstellen der Menüs (Canvas) enthalten. Die Fläche ist das Eltern-Element des Menüs damit diese sich immer im Sichtbereich des Users befinden.

Der LeapController wird um ein selbst implementiertes Skript *SimpleLeapListener* (SLL) ergänzt. Der SLL ist eine Fassade die Informationen bezüglich der LM Sensoren bereitstellt. Um den Zugriff auf diese Informationen zu gewähren, wird durch den Controller auf die aufbereiteten Sensordaten zugegriffen. Der Controller hat Daten der LM über vergangene Frames, aus denen festgestellt wird ob eine Hand den Sichtbereich der LM betritt. Dies ist durch einen Vergleich von dem vorherigen auf den aktuellen Frame realisiert, welcher ein plötzliches Erkennen einer oder beider Hände feststellt.

Die LM Beispiele liefern die Funktionalität einen Anker an der Hand zu definieren. Der SLL kann GameObjects, welche ein AnchorableBahaviour haben, an der linken Hand verankern.

## LeactManager

Der LeactManager ist eine selbst entwickelte Komponente. Er vereint die Daten der Kinect -und LM-Fassade, wodurch das Verhalten bei dem Übergang von FGI zu GS definiert wird. Des weiteren stellt er noch eine MenuFactory, die Menüs generiert und die Menü Historie verwaltet. Die Informationen der erkannten LM Gesten oder Kinect Gesten, werden durch Polling bei jedem Frame gesammelt.

Der LeactManager kann den Übergang von GS zu FGI detektieren, indem der SLL die Information über das Eintreten einer Hand in den FGI bereitstellt. Durch Betreten des FGI werden Modelle der Hand sichtbar, was das Manipulieren von Objekten mit der Hand möglich macht. Sichtbare Menüs werden ausgeblendet indem ihre zugehörigen GameObjects gelöscht werden. Sobald eine der beiden Hände im FGI ist, kann der GS nicht aktiviert werden.

Wenn vor Betreten des FGI ein Objekt in dem jeweiligen Menü gewählt wurde, kann es in der VR manifestiert werden. Dafür wird es bei Betreten des FGI an der linken Hand verankert. Die Fähigkeit Objekte an der Hand zu verankern wird von dem LM-Framework gestellt. Im Anwendungsfall-Kontext eines UML-Editors könnten diese Objekte Modellelemente sein, die in der Umgebung platziert und manipuliert werden sollen.

Wenn sich keine Hände im FGI befinden, kann mit einem einfachen Klick der linken Hand das zuletzt besuchte Menü aufgerufen werden, da sich die MenuFactory die Historie durch eine ID der zuletzt aufgerufenen Menüs merkt.

Diese eindeutigen IDs werden den jeweiligen Menüs und Objekten zugeordnet. Das Menüframework muss um diese Verwaltungsfunktionalität erweitert werden. Anhand der ID wird definiert welche Menüelemente ein Nachfolgemenü haben und wie viele Menüelemente der Nachfolger hat. Die Factory generiert anhand von Prefabs zur Laufzeit neue Instanzen und passt diese entsprechend an, was ein einfaches Anpassen der Menüstruktur ermöglicht.

Das Menüverhalten, also unterschiedliche Handling von Objekten und/oder Menüs, kann komplett in der MenuFactory konzentriert werden. Dies wird durch die ID realisiert, welche bei Objekten grundsätzlich anders ist als bei Menüs. So bleibt die Kontrolle der aufrufbaren Elemente bei der Factory da sie entweder ein neues Menü erzeugt oder ein Objekt an der Hand verankert.

Es ist definiert wie

* Kontinuierliches polling der daten
* Menüfunktion wird in factory definiert
* Mach meü unsichtbar indem es zerstört wird um den übergang in interaktionsmodus zu veranschaulichen
* Factory stellt menü wieder her, hält sich also die information welches menü zuletzt benutzt wurde

## Verwendete Bibliotheken / Frameworks

Radial Menu Framework[[1]](#footnote-1): Dieses kostenlose Framework ist Teil des Unity-Assetstore. Es stellt leicht anpassbare Kreismenüs zur Verfügung, die sich leicht in VR integrieren lassen. Es ist relativ gut dokumentiert, was dabei hilft es für die Verwendung mit der MenuFactory anzupassen.

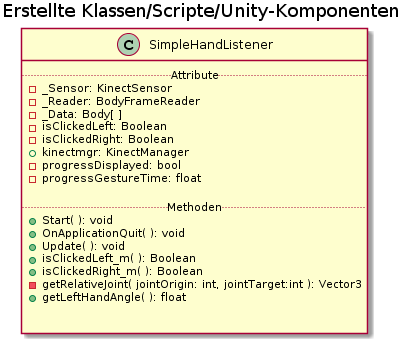
Kinect-v2 VR Examples von Rumen Fielkov[[2]](#footnote-2): Diese Beispiele wurden von Rumen Fielkov auf Anfrage bereitgestellt. Sie vereinen die Kinect-V2 mit VR und machen die Funktionalität von Kinect in VR nutzbar. Durch die enthaltenen Beispiele und die online Dokumentation ist die Benutzung und Handhabung gut zu verstehen und leicht zu lernen.

Leap Motion Orion[[3]](#footnote-3): Die Unity-Core Assets für Leap-Motion bieten Controller und Beispiele für die Leap Motion. Es wurde sich für die Interaktion im FGI und dem verankern an der Hand an einem LM Beispiel orientiert. Die Beispiele und die Verwendung von LM sind gut Dokumentiert.

Die Idee der Leact-Erweiterung für Unity ist es, Daten von Kinect und LM zu vereinen um möglichst wenig Interaktion mit echten Kontrollern zu haben. Es geht darum allein mit dem Körper in VR mit der Umgebung zu interagieren. Der Schwerpunkt lag hier auf Menüführung mit Kinect-Gesten und Interaktion mit Objekten durch LM. Dazu mussten Die Kontroller für Kinect und LM erweitert werden und ein komplett neuer Kontroller – der beide Daten fusioniert – erstellt werden.

Die Leact-Erweiterung ist eine Erweiterung für Unity. Um die Erweiterung einzubinden, muss das Skript SimpleHandListener.cs dem GameObject Kinect Manager hinzugefügt werden. Das Skript SimpleLeapListener.cs muss dem LeapController hinzugefügt werden. Der Kinect Manager sorgt für Schnittstellen mit der Kinect und sorgt dafür das Körper getrackt wird und in VR verfügbar ist. Der LeapController ist der Controller der LM. Dem LeapController muss zudem noch ein Canvas (Leinwand) hinzugefügt werden. Diese Sorgt dafür das das Menü immer direkt im Zentrum des Sichtfeldes des Spielers ist.

~~Die Skripte sind im Unity-Kontext als Komponenten zu verstehen~~. Der SimpleHandListener im Kinect Manager erweitert die von Kinect bereit gestellten Standardgesten. Der SimpleLeapListener erweitert das Standard Gestenset von LM. Der LeactManager ist der neue Kontroller der die neuen Gesten definiert und eine Factory für Menüs stellt. In der Menüfabrik wird die Menüstruktur definiert und das Layout sowie Funktionalität festgelegt



Der SimpleHandListner stellt einfache Gesten zur Verfügung. Er stellt Informationen wie ein Click mit der Hand bereit. Er stellt auch den Winkel bereit, den die linke Hand zur linken Schulter hat. Die linke Hand sowie die linke Schulter sind Gelenke die die Kinect erkennen kann. Es hat sich herausgestellt, dass es die Bedienung deutlich angenehmer macht, wenn der Winkel zwischen Schulter und Hand bestimmt wird. Der bestimmte Winkel zwischen Schultermitte und Hand sorgte für Überstreckung des Körpers, was unangenehm war und zu Fehlern in der Navigation durch Menüs führte.

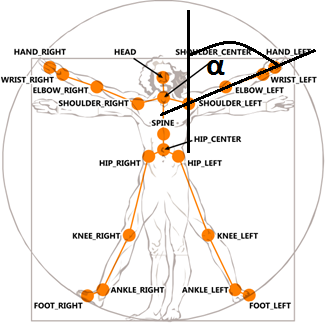
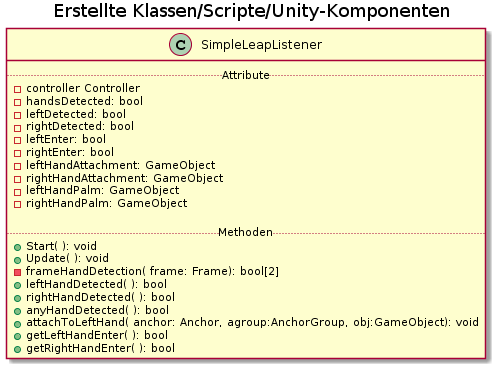
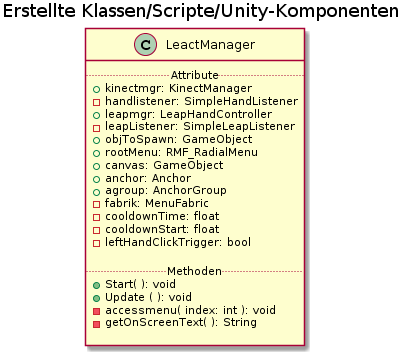


Abbildung 3 https://possiblywrong.wordpress.com/2012/11/04/kinect-skeleton-tracking-with-visual-python/



Der SimpleLeapListener stellt Gesten für LM bereit. Er stellt fest ob Hände zeitlich betrachtet vom letzten Frame bis zu diesem den Sichtbereich (FOV, field of view) betreten. Dadurch wurde eine Selektions-Geste realisiert, mit der es möglich war eine Selektion aus dem Menü ins Bild zu ziehen. Per Menü wird z.B. gewählt was für ein Objekt erstellt werden soll und per „reinziehen“-Geste ist das Objekt an der Hand verankert. Der Anker der verwendet wurde ist ein Anker aus einem LM Beispiel. Objekte die durch die LM Hände manipuliert werden können sollen, sowie im Anker verankert werden sollen, müssen bestimmte Komponenten besitzen. Bewährt hat sich ein bestehendes Prefab zu kopieren und an die eigenen Bedürfnisse anzupassen.



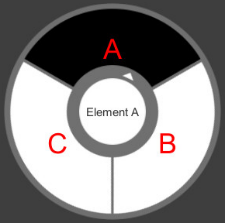
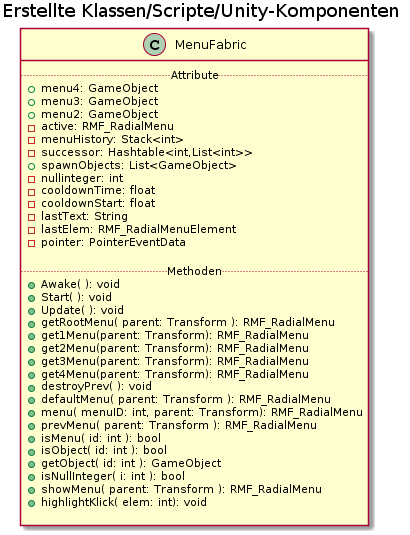
Der LeactManager bezieht die vom SimpleHandListener und SimpleLeapListener bereitgestellten Gesten und Daten und kombiniert sie zu neuen Gesten. Hier ist definiert wie das Menü aufgerufen wird, wie im Menü navigiert wird und wie man Objekte in der Welt erstellt. Mit einem einfachen Klick am linken Bildschirmrand wird das Menü aufgerufen. Mit Rotationsbewegungen des linken Arms können im Radialmenü (Abb. 6) Elemente Gewählt werden. Die Hände dürfen nicht in den Sichtbereich der LM erscheinen denn dann wird das Menü ausgeblendet um Interaktionen mit der Umgebung nicht zu stören. Mit erneutem Klick auf ein Menüelement wird das dazugehörige Untermenü aufgerufen. Es wird ein neues Menü mit entsprechenden Elementen generiert und an Stelle des alten Menüs angezeigt. Um ein Objekt aus dem Menü in der Welt zu erzeugen wird das Menüelement in den Sichtbereich der LeapMotion gezogen. Das Objekt ist dann an der linken Hand verankert und kann dort mit der rechten Hand entnommen werden. Um in Menüs zurück zu vorherigen Menüs zu navigieren, ist das oberste Element standardmäßig mit der Zurück-Funktion belegt.

Abbildung 6

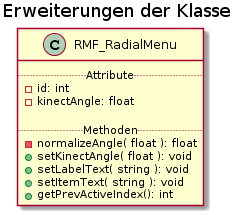
In der MenüFactory ist die Komplette Menü Struktur und Verhalten definiert.



Es ist festgelegt wie viele Elemente jedes Menü hat und ob sie auf ein weiteres Menü oder auf ein Element zeigen. Änderungen oder Erweiterungen des Menüs müssen nur hier vorgenommen werden. Die Factory merkt sich den bisherigen Menü Pfad, sodass zurückgehen bis zum ersten Aufruf des Menüs möglich ist.

Das Menü wird in der Factory generiert, was hohe Austauschbarkeit der Menüs gewährleistet.

Das Verwendete Menü ist aus dem Asset Store von Unity Bezogen. Das Menü Skript wurde geringfügig erweitert um es mit der MenuFactory benutzbar zu machen.



1. https://assetstore.unity.com/packages/tools/gui/radial-menu-framework-50601 [↑](#footnote-ref-1)
2. https://rfilkov.com/2016/05/07/kinect-v2-mobile-vr-examples/ [↑](#footnote-ref-2)
3. https://developer.leapmotion.com/unity/ [↑](#footnote-ref-3)