Simplifizierte Simulation stapelbarer Objekte in einer virtuellen Umgebung

Bachelorarbeit

David Brinkmann

Prüfer der Bachelorarbeit:	1. Prof. Michael Beetz, PhD
	2. Prof. Dr. Gabriel Zachmann
Betreuer:	Andrei Haidu

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Bremen, den 5. September 2017

David Brinkmann

Kurzfassung

In dieser Arbeit wurden mittels einer Implementation Mechanismen entwickelt, in der es möglich ist, innerhalb einer virtuellen Umgebung mit Gegenständen zu interagieren. Interaktionen sind z.B. das Öffnen und Schließen von Schubladen und Türen, Aufheben, Ablegen und Stapeln von Gegenständen. Es sollte herausgefunden werden, welche Strategien zum Decken eines Frühstückstisches optimal sind. Dazu wurden drei unterschiedliche Modi implementiert: Ein Modus, in dem nur mit einer Hand interagiert werden konnte, ein Modus für Interaktionen mit zwei Händen und ein Modus, in der es möglich war, Stapel von Gegenständen zu bilden.

Es wurden Testreihen durchgeführt, in der ProbandInnen drei verschiedene Szenarien durchliefen. Jedes Szenario wurde in allen drei Modi gespielt. Es stellte sich heraus, dass es viel Übung bedarf, bis der Stapelmodus effizienter ist als der Zwei-Hand-Modus. Unerfahrene Spieler benötigten daher mehr Zeit im Stapelmodus als im Zwei-Hand-Modus. Eine zusätzliche Versuchsreihe mit einem erfahrenen Spieler machte deutlich, dass der Stapelmodus dennoch effizienter sein kann, wenn sich erst einmal mit der Steuerung und der Umgebung vertraut gemacht worden ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Greifen und Platzieren von Gegenständen ist für den Menschen eine alltägliche Tätigkeit, welche kein intensives Nachdenken oder Planen erfordert. Der Grund dafür, dass Greifmechanismen fast gänzlich unterbewusst ablaufen, liegt zu großen Teilen daran, dass der Mensch diese während seiner Entwicklung gelernt und verinnerlicht hat. Ihm ist klar, dass er einen Gegenstand nur dann heben kann, wenn dieser nicht zu schwer oder sperrig ist. Des weiteren ist ihm ebenso klar, dass er einen Gegenstand, auf dem ein weiterer Gegenstand liegt, nicht anheben kann, ohne den darauf liegenden Gegenstand ebenso mit anzuheben. Weiterhin bedarf es für einen Menschen keine lange Bedenkzeit, um einschätzen zu können, ob er zum Tragen eines Objekts eine Hand oder beide Hände benötigt und ob ein Stapel von Gegenständen transportierbar sein wird. Mit dieser Wissensbasis ist der Mensch in der Lage, Objekte in seiner näheren Umgebung zu manipulieren und dies, mit zunehmender Erfahrung, so effizient wie möglich zu tun, das heißt, er hat Strategien zum Aufheben und Platzieren von Gegenständen entwickelt und diese im Laufe seines Lebens im Rahmen seiner motorischen Fähigkeiten optimiert.

Roboter werden dem Menschen in Form und Funktion immer ähnlicher. Angefangen von Industrieroboter an Fließbändern, die sehr simple und einseitige Bewegungen vollziehen bis hin zu modernen, vielseitigen Robotern, die in der Lage sind, unterschiedlichste Gegenstände zu handhaben. Ein Beispiel dafür ist der Roboter *Baxter* [1], dem man unterschiedlichste Aufgaben - vom Beschicken von Maschinen bis zum Ver- und Entpacken von Objekten in bzw. aus Kisten und Kartons - zuweisen kann. Oder *Boxy* [2] - ein Projekt der Universität Bremen - der innerhalb einer realen Küchenumgebung z.B. Popcorn zubereiten kann.

Aber wie bringt man einem Roboter effiziente Strategien zum Greifen, Platzieren und Stapeln von Gegenständen bei? Mit dieser Frage beschäftigt sich diese Bachelorarbeit und versucht dadurch Wissen über effiziente Strategien zum Manipulieren von Objekten zu erlangen, indem menschliche SpielerInnen Aufgaben innerhalb einer virtuellen Umgebung erledigen. Dabei geht der wichtigste Teil dieser Arbeit der Frage nach, wie Gegenstände stabil gestapelt werden können, sodass auch ein Roboter diese transportieren könnte.

Damit lässt sich die Implementation dieser Arbeit in die Kategorie “Games with a purpose“ [3], [4] einordnen, deren Zweck darin besteht, die Fähigkeiten menschlicher SpielerInnen zu

nutzen, um unter anderem Daten zu gewinnen, die wiederum für Wissensdatenbanken und Roboterlernen genutzt werden können.

Kapitel 2

Zielsetzung und Motivation

Ziel dieser Arbeit ist es, SpielerInnen in eine virtuelle Küchenumgebung zu platzieren, in der sie vorgefertigte Aufgaben erledigen. Diese Aufgaben konzentrieren sich auf das Heben und Platzieren von Objekten, die unter anderem in Schubladen oder Schränken platziert sind, so dass diese erst geöffnet werden müssen. Während der Interaktion zwischen den SpielerInnen und der virtuellen Umgebung werden Messdaten generiert. Anschließend wird versucht zu folgern, welche Aufgabe wie viele Aktionen und wie viel Zeit erfordert hat und welche Strategien am effizientesten waren.

Um unterschiedliche Strategien herauszufordern, gibt es drei differenzierte Modi:

Ein-Hand-Modus: Den SpielerInnen stehen nur eine Hand zum Manipulieren der Umgebung zur Verfügung. Dies hat zur Folge, dass Gegenstände erst abgelegt werden müssen, um z.B. eine Schublade zu öffnen.

Zwei-Hand-Modus: Den SpielerInnen stehen beide Hände zur Verfügung. Schubladen und Türen können mit einer freien Hand manipuliert werden.

Stapelmodus: Die SpielerInnen haben beide Hände zur Verfügung und können Gegenstände aufeinander stapeln und diesen Stapel als Ganzes transportieren.

Es wird versucht, die eingeschränkte Interaktion durch Maus und Tastatur so zu gestalten, dass sich Interaktionen *natürlich* anfühlen, z.B. das Öffnen und Schließen einer Schublade, in dem die Maus vor bzw. zurück bewegt wird.

Der wichtigste Bestandteil dieser Arbeit ist die Komponente zum Stapeln von Gegenständen und zur Überprüfung, ob der erzeugte Stapel stabil genug ist, um transportiert werden zu können und somit das Decken eines Tisches mit vorgegeben Gegenständen in Hinsicht auf die benötigte Zeit und Anzahl der einzelnen Aktionen zu optimieren.

Mit den so gewonnenen Daten könnte zukünftig eine Wissensdatenbank aufgebaut werden, die effiziente Strategien für unterschiedlichste Aufgaben zum Heben und Platzieren von Gegenständen enthält. Dies liegt jedoch außerhalb des Rahmens dieser Arbeit.

2.1. Abgrenzung und Einschränkungen

Diese Bachelorarbeit beschränkt sich lediglich darauf, eine virtuelle Küchenumgebung zu schaffen, in der SpielerInnen sich bewegen und mit bestimmten Objekte interagieren können. Hinzu kommt das Sammeln einiger wichtiger Schlüsseldaten während der Interaktion der SpielerInnen mit der virtuellen Spielwelt. Inwieweit diese generierten Daten tatsächlich für das Lernen von Interaktionsstrategien von Robotern genutzt werden können, liegt nicht im Rahmen dieser Arbeit.

Eingeschränkt sind die SpielerInnen innerhalb ihrer Möglichkeiten dadurch, dass sie zur Bedienung ihres virtuellen Charakters nur die Tastatur und die Maus als Eingabemöglichkeit zur Verfügung stehen haben. Dies hat den Nachteil, dass eine einfache und intuitive Interaktion mit Objekten immer nur höchstens zweidimensional möglich sein kann, z.B. ließe sich ein Objekt mit der Maus vor, zurück, nach links und rechts bewegen, aber nicht rotieren oder nach oben und unten bewegen, da die Maus nicht ohne Weiteres Objekte in alle Richtungen bewegen kann. Dazu müsste zusätzlich die Tastatur verwendet werden, jedoch soll die Anzahl verwendeter Tasten so gering wie möglich gehalten werden. Wenngleich versucht wird, die Eingabe via Tastatur und Maus natürlich zu gestalten, kann nicht der Realitätsgrad eines *Virtual-Reality*-Controllers erreicht werden, mit dem in allen Freiheitsgraden Gegenstände auf intuitive Weise bewegt werden können.

Auch bei der Überprüfung der Stabilität eines Stapels von Objekten müssen Einschränkungen hingenommen werden. Ziel ist es, eine schnelle Simulation des Objektstapels durchzuführen, um zu entscheiden, ob dieser transportfähig ist. Diverse Eigenschaften der simulierten Physik schränken diesen Prozess jedoch ein. Folgende Ungenauigkeiten sind daher unvermeidlich:

- Je schneller die Stabilitäts-Simulation abläuft, desto ungenauer ist das Ergebnis. Ein Stapel, der als *stabil* befunden wurde, würde bei einem längeren Test evtl. das gegenteilige Ergebnis liefern.

- Auf Grund der verwendeten Physik, kann der Stabilitätstest bei sehr leichten bzw. kleinen Gegenständen ein *falsch-negatives* Ergebnis liefern. Die Ursache dafür ist, dass solche Objekte bei einem dezenten “Rütteln“ des Stapels zu weit von ihrer Ursprungsposition verrückt werden, was als *instabil* gewertet wird.

- Kollisionen und Physik funktionieren gut für wenige Gegenstände. Befinden sich jedoch dünne Objekte in einem Stapel, z.B. Teller, kann es mitunter passieren, dass der Stapel sich von alleine bewegt, da die Teller ständig miteinander kollidieren oder sogar ineinander gleiten. Dieses Verhalten konnte bei Tellerstapeln mit mehr als 5 Tellern beobachtet werden.

- Da die Simulation nicht deterministisch ist, können die Ergebnisse von Versuch zu Versuch variieren, das heißt, dass ein instabiler Stapel direkt im zweiten Versuch ein stabiler Stapel sein kann.

Kapitel 3

Verwendete Software

In diesem Abschnitt werden kurz die verwendete Engine und Plugins vorgestellt. Bei den Plugins handelt es sich um Software, die von der `Arbeitsgruppe Künstliche Intelligenz` entwickelt wird. Diese stellen zusätzliche Funktionen innerhalb der Unreal Engine zur Verfügung und haben unter anderem den Zweck, die Implementierung dieser Bachelorarbeit zur späteren Nutzung innerhalb der Arbeitsgruppe vorzubereiten bzw. nutzbar zu machen.

3.1. Unreal Engine

Die Unreal Engine ist eine der am häufigst verwendeten Spiele-Engines und wurde 1998 von Epic Games [5] veröffentlicht. Sie stellt die Grundlage der Entwicklung der virtuellen Umgebung dar. In ihr wurde das Modell der virtuellen Küche des Projekts *Robot Comonsense Games* [6] erstellt, welches auch innerhalb dieser Bachelorarbeit genutzt wird. Wie für moderne Spiele-Engines bietet sie die grafische Darstellung von Objekten, Lichter-Rendering, Ein- und Ausgabenkontrolle, Physik etc. an. Funktionen können über sogenannte *Blueprints* oder über C++-Code erstellt werden. Innerhalb dieser Arbeit wird fast jede Funktionalität in Code erstellt.

3.2. Plugins

3.2.1. UTags

Tags sind Zeichenfolgen, die jedem Element innerhalb der Spielwelt zugewiesen werden können. Auf diese Tags kann anschließend via Code zugegriffen werden. So kann eine Schublade als *OpenClosable* getaggt sein, eine Tasse als *Pickupable*. Innerhalb des Codes kann nach dem Auslesen des Tags unterschiedliche Funktionen aufgerufen werden, z.B. wird eine Tür geöffnet und nicht wie die Tasse aufgehoben.

Das UTags-Plugin [7] hilft einerseits dabei, die Tags auszulesen und andererseits dient es der Vereinheitlichung von Tags über mehrere Projekte, was eine leichtere Integration der einzelnen Projekte in ein gemeinsames Projekt ermöglicht.

3.2.2. USemLog

Das USemLog-Plugin [8] dient dem Aufzeichnen von Aktionen, welche die SpielerInnen während des Lösens der Aufgaben tätigen. Dabei wird für den Zweck dieser Arbeit die Aktion, die Dauer und wenn sinnvoll das Resultat aufgezeichnet. Das Plugin wird auch in anderen Projekten wie *Robot Comonsense Games* verwendet. Es dient dazu, eine semantische Karte von allen Gegenständen innerhalb der virtuellen Umgebung zu erzeugen und bestimmte Ereignisse aufzuzeichnen. Für den Zweck dieser Bachelorarbeit werden jedoch nur einige dieser Daten verwendet und zur besseren Verwendung weiterverarbeitet. Dennoch können die gewonnen Daten in bereits bestehenden Projekten wie *openEASE* [9] verwendet werden.

Kapitel 4

Grundlagen und Bedienung

In diesem Kapitel werden kurz die grundlegende Bedienung und Möglichkeiten der Interaktion innerhalb der virtuellen Küche vorgestellt. Haupteingabeinstrumente sind dabei die Tastatur und die Maus. Es wurde Wert darauf gelegt, die Steuerung schlicht und intuitiv zu gestalten, das heißt, es sollten keine komplizierten Tastenkombinationen notwendig sein.

Alle Tastenzuweisungen können nach Wunsch neu definiert werden, daher einigen wir uns auf die Aktionsnamen und nicht auf spezielle Tasten.

4.1. Aktionsbezeichnungen

Bewegungs-Tasten Diese Tasten werden genutzt, um den Charakter durch die virtuelle Umgebung zu bewegen

LinkeHand-Taste Die Taste, die für eine Aktion mit der linken Hand zuständig ist

RechteHand-Taste Die Taste, die für eine Aktion mit der rechten Hand zuständig ist

Abbrechen-Taste Mit dieser Taste können einige Aktionen (z.B. Aufheben, Ablegen von Objekten) abgebrochen werden

Ziehen-Taste Mit dieser Taste kann ein Objekt auf der Oberfläche, auf der das Objekt steht, verschoben werden, ohne es aufzuheben

Bewegungs-Tasten Mittels dieser Tasten können sich die SpielerInnen vor, zurück und seitlich bewegen

Ducken-Taste Mit dieser Taste können sich SpielerInnen ducken, um an tiefer gelegene Objekte zu gelangen

PickSingleItem-Taste Mittels dieser Taste kann erzwungen werden, dass nur ein einzelner Gegenstand aufgehoben wird und nicht auf einen Stapel geprüft wird

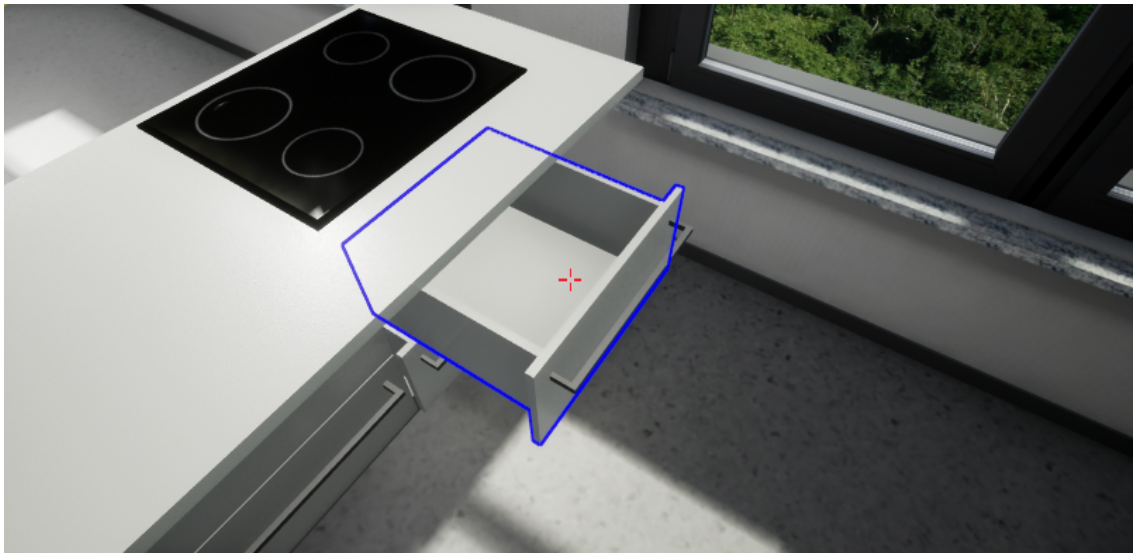
Rotieren-Taste Diese Taste hat den Zweck, einen Gegenstand während des Ablegens um schrittweise 90 Grad zu drehen

4.2. Benutzerinterface, Fokussierung von Objekten, Interaktionssteuerung

Das Benutzerinterface ist sehr einfach gehalten. Zum besseren Anvisieren von Objekten ist mittig ein Fadenkreuz platziert. Info-Texte werden zeitlich begrenzt in der oberen linken Ecke angezeigt. Um den SpielerInnen ein visuelles Feedback zu geben, welchen Gegenstand sie gerade anvisieren und ob mit diesem interagiert werden kann, werden die entsprechenden Gegenstände farblich umrandet dargestellt (siehe Abb. 4.1). Das Post-Processing-Material für die farbliche Hervorhebung wurde dem Projekt *RobCoG* [10] entnommen.

Die grundlegende Steuerung und Interaktion mit Gegenständen findet über die *LinkeHand-Taste* und *RechteHand-Taste* statt.

Abbildung 4.1.: *Fokussieren einer Schublade*



4.3. Bewegung

Mittels der *Bewegungs-Tasten* bewegen sich die SpielerInnen durch die virtuelle Küche. Um mit Dingen interagieren zu können, müssen die Gegenstände in Reichweite sein. Die Reichwei-

te kann individuell eingestellt werden. Bei einer Reichweite von standardmäßig etwa einem Meter zwanzig ist es den SpielerInnen nicht möglich, Dinge im Stand vom Boden aufzuheben. Hierzu müssen sie sich mittels der *Ducken-Taste* ducken, bevor sie das Objekt aufheben können.

Standardmäßig aktiviert ist die Einstellung, dass getragene Gegenstände die Geschwindigkeit der SpielerInnen beeinflusst. Wird z.B. ein großer Stapel Teller getragen, sind die SpielerInnen langsamer, als wenn sie nur einen Löffel in den Händen halten. Diese Einstellung kann deaktiviert werden.

4.4. Öffnen und Schließen

Zum Öffnen und Schließen einer Schublade oder Tür, muss

- diese fokussiert und in Reichweite sein
- die SpielerInnen die entsprechende Hand frei haben

Sind diese Bedingungen erfüllt, kann durch Gedrückthalten der entsprechenden Taste (*LinkeHand-Taste* bzw. *RechteHand-Taste*) die Schublade bzw. Tür geöffnet oder geschlossen werden. Dazu bewegen die SpielerInnen bei gedrückter Taste die Maus zurück (öffnen) bzw. nach vorn (schließen). Abhängig davon, wie schnell die Maus nach vorn bzw. hinten bewegt wird, wird mehr Kraft auf die Schublade oder Tür ausgeübt, was diese dann schneller öffnet bzw. schließt.

4.5. Anheben von Objekten

Zum Anheben von Objekten müssen folgende Bedingungen erfüllt sein:

- Das Objekt muss fokussiert sein und sich in Reichweite befinden
- Die von den SpielerInnen genutzten Hand muss frei sein
- Der aufzuhebende Gegenstand darf ein eingestelltes Gewicht oder Volumen nicht überschreiten (Diese Einschränkungen sind optional und können deaktiviert werden).
- (Optional) Es darf sich nichts zwischen dem Objekt und den SpielerInnen befinden. Diese Option ist standardmäßig deaktiviert.

Sind alle Bedingungen erfüllt, können die SpielerInnen mittels der *LinkeHand-Taste* bzw. *RechteHand-Taste* auf den Gegenstand klicken. Wird die Taste gehalten, ist eine halbtransparente, einfar-

bige Kopie des Gegenstandes (nachfolgend **Schattenitem** genannt) zu sehen und signalisiert den SpielerInnen die Endposition des Gegenstands. Sind die Einstellungen so gewählt, dass der aufzuhebende Gegenstand mit nichts zwischen diesem und den SpielerInnen kollidieren darf, und befindet sich ein anderer Gegenstand zwischen ihnen, wird das Schattenitem an der Position der Kollision dargestellt. Abbildung 4.2 zeigt einen erfolgreichen Test, ob der Gegenstand aufgehoben werden kann. Die Endposition ist die Hand der SpielerInnen. In Abbildung 4.3 hingegen befindet sich etwas im Weg zwischen dem bzw. der SpielerIn. Es wird die Position der Kollision angezeigt.

Befindet sich kein Gegenstand im Weg (bzw. ist die entsprechende Einstellung deaktiviert), d.h. kann der Gegenstand problemlos aufgehoben werden, wird dieser in die entsprechende Hand der SpielerInnen teleportiert, wenn diese die *LinkeHand-Taste* bzw. *RechteHand-Taste* loslassen. Das Aufheben kann mittels der *Abbrechen-Taste* abgebrochen werden.

Da es vor allem bei kleineren Objekten passieren kann, dass sie ungünstig aufeinander liegen (z.B. Besteck) kann die *PickSingleItem-Taste* genutzt werden, um nur exakt den fokussierten Gegenstand aufzuheben, ohne auf einen Stapel (siehe Abschnitt 4.9) zu prüfen.

Abbildung 4.2.: *Aufheben eines Gegenstandes*

Abbildung 4.3.: *Blockierter Gegenstand beim Aufheben*

4.6. Ablegen von Objekten

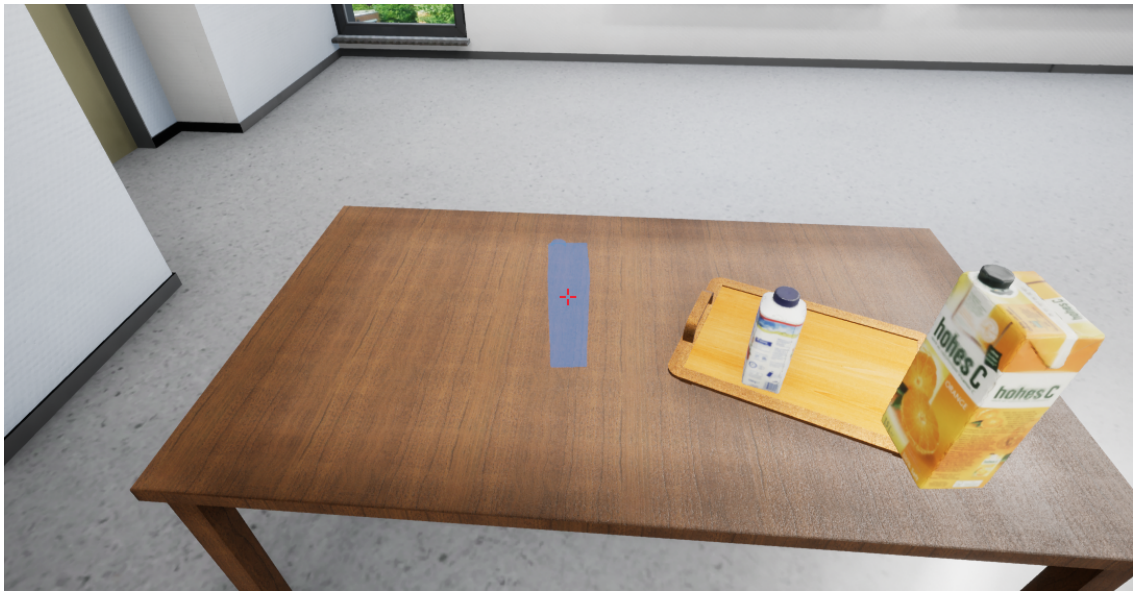
Das Ablegen eines Gegenstandes aus einer Hand ist ähnlich wie das Aufheben. Es müssen folgende Bedingungen vorhanden sein:

- Die Oberfläche, auf der der Gegenstand abgestellt wird, muss in Reichweite sein
- Es darf sich nichts zwischen den SpielerInnen und dem Punkt, auf dem das Objekt abgestellt werden soll, befinden. Diese Einstellung kann optional deaktiviert werden.

Die SpielerInnen legen einen Gegenstand ab, indem sie mit der entsprechend *LinkeHand-Taste* bzw. *RechteHand-Taste* auf eine Oberfläche oder einen anderen Gegenstand klicken und die Taste halten. Es wird ein Schattenitem angezeigt, um den SpielerInnen eine Vorschau auf die endgültige Position des Objekts nach Ablage anzuzeigen (siehe Abb. 4.4). Wird das Objekt auf dem Weg zur gewünschten Position von einem anderen Gegenstand blockiert, wird das Schattenitem an der Kollisionsstelle angezeigt. Auch hier haben die SpielerInnen die Möglichkeit, die Aktion mittels der *Abbrechen-Taste* abzubrechen.

Lassen die SpielerInnen die *LinkeHand-Taste* bzw. *RechteHand-Taste* los, wird der Gegenstand aus der Hand zur angezeigten Position teleportiert.

Abbildung 4.4.: *Vorschau beim Abstellen eines Gegenstandes*



4.7. Ziehen von Objekten

Mittels der *Ziehen-Taste* kann ein Gegenstand auf der Oberfläche, auf der er sich gerade befindet, bewegt werden, ohne aufgehoben werden zu müssen. Dazu muss

- der Gegenstand in Reichweite und fokussiert und
- die benutzte Hand nicht bereits durch einen Gegenstand belegt sein

Halten die SpielerInnen die *Ziehen-Taste* gedrückt, wird der Ziehen-Modus aktiviert. Wird nun auf ein Gegenstand mit einer der *LinkeHand-Taste* bzw. *RechteHand-Taste* geklickt und gehalten, bewegt sich der gezogene Gegenstand entsprechend der Mausbewegung der SpielerInnen.

4.8. Zwei-Hand-Simulation

Sind die Optionen aktiviert, dass für schwere oder sperrige Gegenstände zwei Hände benötigt werden, wird beim Aufheben eines solchen Gegenstandes dieser in eine Position direkt vor den SpielerInnen teleportiert, sofern beide Hände frei sind. Beim Ablegen dieses Gegenstandes spielt es keine Rolle, ob man die *LinkeHand-Taste* oder *RechteHand-Taste* zum Ablegen benutzt.

4.9. Stapel-Simulation

Ist die Option zur Stapel-Simulation aktiviert, und wird ein Gegenstand mittels der *LinkeHand-Taste* bzw. *RechteHand-Taste* aufgehoben, wird geprüft, ob sich weitere Gegenstände direkt über dem angewählten Gegenstand befinden. Ist dies der Fall, wird ein Stapel gebildet. Dieser wird im Anschluss auf Stabilität getestet. Um den SpielerInnen ein Feedback über den Verlauf des Tests zu geben, wird ein kleines Fenster mittig eingeblendet, das den Stapel (bzw. eine exakte Kopie davon) darstellt und die gemachten Bewegungen (Rütteln und Rotieren) verdeutlicht. Zusätzlich wird in der oberen linken Bildschirmecke die noch benötigte Zeit für den Test angezeigt. Anschließend wird der Stapel in die Hände der SpielerInnen teleportiert (wenn Stabilitäts-Check erfolgreich) oder es wird eine Information angezeigt, dass der Stapel instabil ist und somit nicht aufgehoben werden kann. Abbildung 4.5 zeigt den aktiven Stabilitäts-Check in einem zusätzlichem Fenster.

Abbildung 4.5.: *Stability-Check*

Kapitel 5

Umsetzung

In diesem Kapitel wird die Umsetzung und Implementierung der wichtigsten Elemente der Arbeit aufgezeigt und gliedert sich in die Unterkapitel für die Umsetzung des Öffnens und Schließens, Aufheben und Ablegen von Objekten und dem Stapeln von Gegenständen inklusive der Überprüfung der Stabilität eines Stapels. Die Implementation für Bewegung ist trivial und werden nicht näher erläutert.

5.1. Spielercharakter

Der grundlegende Aufbau besteht in je einer separaten Klasse für zusammenhängenden Funktionalitäten. Das heißt, es gibt eine Klasse, die ausschließlich für die Bewegung zuständig ist, eine Klasse für das Öffnen und Schließen etc. Da diese Funktionalität nur dann sinnvoll nutzbar ist, wenn es auch einen Spieler-Charakter in der Welt gibt, sind diese keine eigenständigen `Actors` (Elemente innerhalb der Spielwelt) sondern `ActorComponents`. Sie sind also Bestandteil des Spieler-Charakters.

Die Hauptklasse, der all diese Komponenten zugewiesen wird, ist in dem Skript `CharacterController` hinterlegt. Dort wurde die Funktionalität des Fokussierens implementiert, ebenso das Highlighting interagierbarer Gegenstände.

Diesem Controller werden unter anderem folgende wichtige Komponenten untergeordnet:

CMovement bewegt die SpielerInnen auf Tastendruck

COpenClose ist die Komponente, zum Öffnen und Schließen von Türen und Schubladen

CPickup ist die Komponente zu Aufnehmen, Ziehen und Ablegen von Objekten

CLogger dient dem Mitschneiden einzelner Aktionen

Die Umsetzung der Kernkompontenten werden in den folgenden Abschnitten genauer erläutert.

5.2. Öffnen und Schließen

Damit sich ein Objekt öffnen und schließen lässt, benötigt es den *Tag* `OpenCloseable,True;` (Kompletter *Tag*:`ClickInteraction;Interactable,True;OpenCloseable,True;`). Dieser kann durch das Plugin UTags (siehe Kapitel 3.2.1) ausgelesen werden. Für einen schnellen Zugriff werden alle Objekte mit diesem *Tag* bei Spielstart gesammelt und in einem `Set` gespeichert. Nutzten die SpielerInnen nun eine der beiden Interaktionstasten (*LinkeHand-Taste* bzw. *RechteHand-Taste*) und ist diese Hand nicht durch einen Gegenstand belegt, teilt die `CPickup`-Komponente dem `CharacterController` mit, dass sie nun exklusive Kontrolle hat, das heißt, dass sich die SpielerInnen während sie die Interaktionstaste gedrückt halten, weder bewegen noch eine andere Aktion ausführen können. Solange nun die Interaktionstaste gedrückt bleibt, wird die relative Position der Maus ausgewertet. Somit wird erkannt, in welche Richtung die Maus bewegt wurde. Entsprechend der Stärke der Bewegung wird nun eine Kraft auf das fokussierte Objekt angewendet. Somit wird eine Tür oder eine Schublade geöffnet bzw. geschlossen. Voraussetzung dafür ist jedoch, dass die Richtung, in der das Objekt geöffnet bzw. geschlossen wird, dem Vorwärtsvektor (X-Vektor des Objekts) entspricht. Dies muss bei der Modellierung der 3D-Objekte berücksichtigt werden.

5.3. Aufheben und Ablegen von Gegenständen

Das Aufheben und Ablegen ist in der Komponente `CPickup` implementiert. Damit ein Objekt aufgehoben werden kann, benötigt es den *Tag* `Pickup,True;` welcher - wie auch beim Öffnen und Schließen - durch das Plugin UTags (siehe Kapitel 3.2.1) ausgelesen wird. Alle Objekte dieser Art werden in einem `Set` gespeichert, um die Zugriffszeiten zu verringern.

Sowohl das Aufheben als auch das Ablegen läuft in drei Stufen ab:

1. Start des Aufhebens bzw. Ablegens. Dies wird durch den ersten Druck eine der Interaktionstasten ausgelöst

2. Schattenitems anzeigen. Dies läuft in einer Schleife, solange die Interaktionstaste gedrückt ist

3. Aufheben bzw. Ablegen. Die finale Aktion. Sie wird ausgelöst, sobald die Interaktionstaste losgelassen wird

5.3.1. Aufheben

5.3.1.1. StartPickup

Der Start des Aufhebens dient dazu, zu bestimmen, wie schwer das Objekt ist, ob es sich vielleicht um einen Stapel handelt und ob eine oder zwei Hände dafür benötigt werden bzw. ob die Hände, die genutzt werden sollen auch frei sind. Ist es den SpielerInnen möglich, den Gegenstand grundsätzlich aufzuheben, wird ein Log-Event erzeugt und bereits die neue Bewegungsgeschwindigkeit der SpielerInnen in Abhängigkeit des Gewichts des Objekt berechnet. Für den Fall, dass ein Aufheben unmöglich ist (z.B. weil keine Hände frei sind oder das Objekt zu schwer ist) wird eine Fehlermeldung angezeigt.

5.3.1.2. ShadowPickupItem

Diese Funktion hat den Zweck, den SpielerInnen visuell zu signalisieren, an welcher Stelle sich das Objekt befinden würde, wenn es aufgehoben wird bzw. wenn es unterwegs durch einen anderen Gegenstand blockiert wird. Solange die Interaktionstaste gedrückt gehalten wird, werden diese Schattenitems angezeigt. Die originalen Objekte werden dabei lediglich kopier und die Kopien erhalten ein halbtransparentes Material. Da in dieser Funktion auf Kollisionen zwischen dem Objekt und dem bzw. der SpielerIn geprüft wird, wird auch gleichzeitig entschieden, ob der Gegenstand überhaupt beim Loslassen der Interaktionstaste aufgehoben werden kann oder ob eine Kollision dies verhindert. Bei Letzterem passiert nach dem Loslassen der Interaktionstaste einfach nichts.

5.3.1.3. PickupItem

In dieser Funktion entscheidet sich schließlich, in welche Hand das Objekt teleportiert wird, sofern dieser aufhebbar ist. Zusätzlich werden (wenn eingestellt) alle Kollisionen und die Physik dieser Gegenstände deaktiviert. Zusätzlich wird ein weiteres Log-Event erzeugt mit der Meldung, dass der Gegenstand aufgenommen wurde.

5.3.1.4. Probleme und Einschränkungen beim Aufheben von Objekten

Bei kleineren Gegenständen, die ungünstig nahe beieinander liegen, wie z.B. Besteck oder kleinere Schüsseln, kann nicht präzise entschieden werden, dass nur ein einzelner Gegenstand aufgehoben werden soll oder ob es sich um einen Stapel handelt. Um zu vermeiden, dass beim Aufheben z.B. eines Löffels die umliegenden Löffel als Gegenstände erkannt werden, die man zu einem Stapel zusammenführen kann, kann die *PickSingleItem-Taste* (siehe Kapitel

4.1) genutzt werden. Dies hat aber auch den Nebeneffekt, dass damit Gegenstände mitten aus einem Stapel (z.B. ein Tellerstapel) heraus gegriffen werden können.

5.3.2. Ablegen

Die Grundprinzipien des Ablegens ähneln denen des Aufhebens.

5.3.2.1. StartDropItem

Im Vergleich zu `StartPickup` wird hier lediglich entschieden, aus welcher Hand der Gegenstand abgelegt werden soll. Zusätzlich wird auch hier ein entsprechender Eintrag im Log erzeugt.

5.3.2.2. ShadowDropItem

Ist die Option aktiviert, dass beim Ablegen auf Kollision geprüft wird, fungiert diese Funktion analog zu `ShadowPickupItem`. Ist dies jedoch nicht der Fall, so muss zusätzlich die finale Position des Gegenstands berechnet werden. Der Grund dafür ist, dass der Gegenstand sonst **in** die Oberfläche, auf die der Spieler bzw. die Spielerin schaut gelegt wird. Abhängig von der Höhe des Objekts muss daher ein Offset zur Position des Gegenstandes beim Platzieren aufgerechnet werden.

5.3.2.3. DropItem

Die `DropItem`-Funktion macht alles rückgängig, was die `PickupItem`-Funktion getan hat. Das bezieht sich hauptsächlich auf die Einstellungen der Physik und Kollisionen, da sich die Gegenstände nach dem Ablegen wieder so verhalten sollen, wie vorher. Außerdem wird auch hier ein Event im Log erzeugt und die Bewegungsgeschwindigkeit der SpielerInnen neu berechnet.

5.3.3. Neuberechnung der Bewegungsgeschwindigkeit

Um de Realitätsgrad zu erhöhen, verringert sich die Bewegungsgeschwindigkeit abhängig von der Masse, welche die SpielerInnen tragen. Einbezogen in die Berechnung der Geschwindigkeit wird die Masse, die bereits von den SpielerInnen getragen wird (*MassToCarry*), die maximale Masse (*MaximumMassToCarry*), die getragen werden kann und die maximale und

minimale Spielergeschwindigkeit (*MinSpeed* bzw. *MaxSpeed*). Es wurden zwei Ansätze implementiert, ein linearer und ein quadratischer.

Für die lineare Berechnung gilt:

$$v_l = (MinSpeed - MaxSpeed) \cdot \frac{MassToCarry}{MaximumMassToCarry} + MaxSpeed$$

Für die quadratische Berechnung gilt:

$$v_q = (MinSpeed - MaxSpeed) \cdot \left(\frac{MassToCarry}{MaximumMassToCarry}\right)^2 + MaxSpeed$$

Abbildung 5.1 verdeutlicht die beiden Ansätze grafisch. Als maximal tragbare Masse sind hier 10 kg gewählt. Der Funktiongraph f entspricht der linearen, der Funktionsgraph g der quadratischen Berechnung.

Die quadratische Berechnung hat den deutlichen Vorteil, die Bewegungsgeschwindigkeit für eher leichte Objekte weniger stark zu beeinflussen, was der Realität näher kommt, als die lineare Berechnung.

Abbildung 5.1.: *Verhalten der unterschiedlichen Ansätze zur Berechnung der Spielergeschwindigkeit*

5.4. Stapeln

Das Erzeugen eines Stapels wird durch die beiden Funktionen `GetItemStack` und `FindAllStackableItems` verwirklicht. Dabei ist Letztere das Herzstück dieser Funktionalität. Um einen Stapel zu erzeugen, wird das `Basis-Item`, also das, auf den der Spieler bzw. die Spielerin klickt, simuliert nach oben bewegt. Während dieser simulierten Bewegungen werden auch alle potentielle Kollisionen erkannt (`sweeping`). Ist der Gegenstand, mit dem kollidiert wird, ein Gegenstand, der auch aufgehoben werden kann, d.h. hat dieser auch den *Tag* `Pickup,True;`, wird dieser zu einer Liste hinzugefügt. Wird das erste mal mit einem Gegenstand ohne diesen *Tag* kollidiert, endet die Suche nach weiteren Gegenständen, denn die simulierte Bewegung ist z.B. auf ein Regalboden gestoßen.

Anschließend werden alle gefundenen Gegenstände dem Basis-Item angehängt (`attach`) und zusammengeschweißt (`welding`). Dadurch agiert der komplette Stapel so, als wäre er ein einziger Gegenstand.

5.4.1. Probleme beim Sweeping

Die erste Implementierung der simulierten Bewegung mit Kollisionserkennung hat gezeigt, dass diese nicht sonderlich genau von der Unreal Engine umgesetzt wurde. So passierte es oft, dass bei mehreren Gegenständen einige nicht erkannt wurden, wenn diese zu nah beieinander standen und sich somit auf eine gewisse Weise “verdeckten“. Um dem entgegenzuwirken, wird eine `do while`-Schleife verwendet. Innerhalb dieser läuft das oben genannte Prozedere ab. Dabei werden alle neu gefundenen Gegenstände gezählt. Die Schleife wird genau dann verlassen, wenn keine neuen Gegenstände mehr durch das Sweeping gefunden wurden.

5.4.2. Stabilitätsprüfung

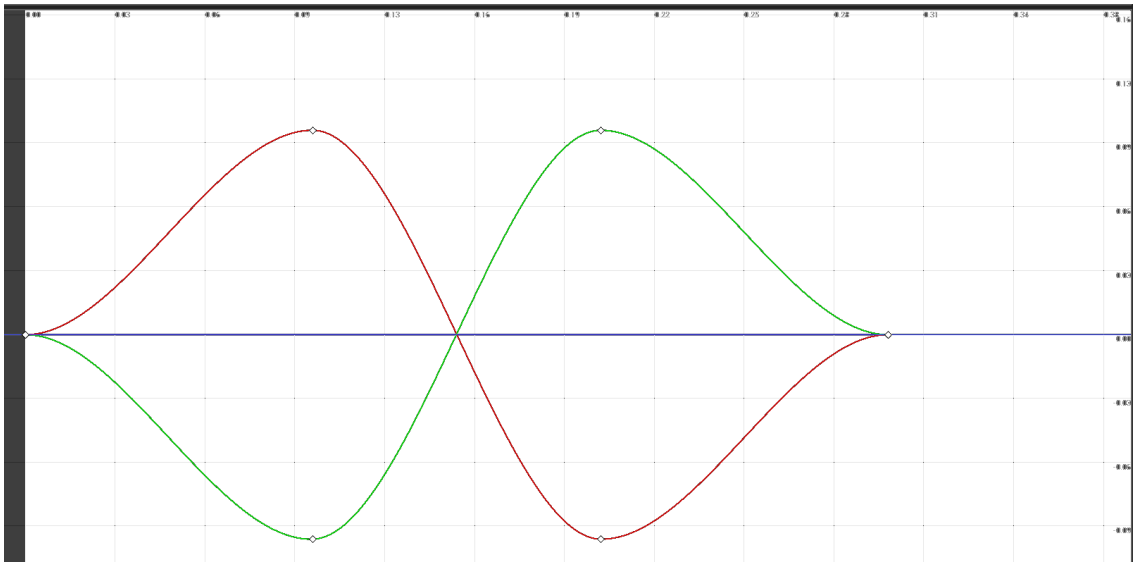
Die Stabilitätsprüfung ist ein eigenes Objekt (`Actor`) innerhalb der Spielwelt. Die grundlegende Idee ist es, die durch die Unreal Engine bereitgestellte Physik zu nutzen, um zu überprüfen, wie sich ein Stapel von Objekten verhält, wenn er bewegt wird. Es wurde darauf Wert gelegt, diese Funktionalität erweiter- und anpassbar zu halten. Es gibt daher zwei Funktionen, die alle Bewegungsgrade abdecken. Eine Funktion für das Verändern der Position (`Shaking`) und eine für das Verändern der Rotation. Beide Funktionen erhalten ihre Informationen durch Vektoren, die durch eine `Animation Curve` bereitgestellt werden. In dieser lassen sich Bewegungen und Rotationen in allen drei Dimensionen anfertigen, die während des Tests ausgeführt werden. Abbildung 5.2 zeigt eine solche Animation Curve, die für ein Shaking zuständig ist. Jeder Wert auf der Kurve stellt die Änderungsrate der Position bzw. Rotation dar, demzufolge entspricht das Integral zwischen der Kurve und der Abszissenachse der absoluten Verschiebung bzw. Rotation. Bei einer vollständigen, symmetrischen Schwingung (punktsymmetrisch zum Wendepunkt)) wie in Abbildung 5.2 zu sehen, befindet sich der Stapel am Ende wieder am

Ausgangspunkt. Die in dieser Abbildung gezeigten Kurve hat eine Dauer von 0,3 Sekunden, die maximale Änderungsrate entspricht 0.1 $\frac{m}{s}$. Durch Setzen der Änderungsrate anstelle der tatsächlichen Position bzw. Rotation lassen sich die Bewegungen feiner kontrollieren, was zu weniger Fehlern bei der Kollisionserkennung führt.

Auf Wunsch kann nach den Bewegungen eine Pause aktiviert werden, in der sich der Stapel "beruhigen" bzw. endgültig zu Fall kommen kann. Die Länge der Pause kann eingestellt werden.

Bevor die Bewegungen stattfinden, werden alle relativen Positionen aller Gegenstände zum Basis-Item aufgezeichnet. Nachdem das Shaking und Rotieren beendet ist, werden die neuen relativen Positionen mit den alten vergleichen. Überschreitet die Abweichung einen vorher eingestellten Wert, gilt der Stapel als instabil. Die SpielerInnen erhalten eine entsprechende Meldung und der gewünschte Stapel kann nicht angehoben werden.

Abbildung 5.2.: *Stability-Check Animation Curve (Shaking) 0,3 Sekunden*



5.4.2.1. Einschränkungen von Stapeln und der Stabilitätsprüfung

Wie bereits im Kapitel 2.1 (Abgrenzungen und Einschränkungen) erwähnt, kann es bei der Stabilitätsprüfung passieren, dass vor allem leichte und kleine Gegenstände durch das Shaking vom Stapel geworfen werden. Dies wirkt unrealistisch kann aber nur dadurch gelöst werden, in dem das Gewicht dieser Objekte entsprechend erhöht wird, was jedoch nachteilige Effekte auf andere Physik basierenden Funktionen haben könnte. Diese Einschränkung tritt jedoch selten genug auf und kann bei wiederholtem Versuch dennoch zu einem stabilen Stapel führen.

Kapitel 6

Experimente

6.1. Stabilitätstest

Um ein geeignetes Muster zum Testen der Stapel zu finden, wurden unterschiedliche `Animation Curves` erzeugt und diese an zwei Stapeln getestet. Diese Stapel sind so konstruiert, dass einer davon das Resultat *instabil* der andere das Resultat *stabil* liefern sollte. Anschließend werden verschiedene `Animation Curves` auf diese Stapel angewendet.

Abbildung 6.1 zeigt die zwei verwendeten Stapel. Stapel A ist als instabil anzunehmen, da dieser in der Realität durch eine leichte Bewegung zum Einsturz gebracht werden würde. Stapel B hingegen gilt als transportierbarer Stapel.

Bei jedem Versuch wird nach dem Shaking bzw. der Rotation eine Pause von einer Sekunde eingestellt, bevor der Stapel geprüft wird. Jeder der beiden Stapel ist zu Beginn der Simulation stabil, droht im Stillstand also noch nicht, umzukippen.

Die in den Abbildungen 6.2 bis 6.7 gezeigten `Animation Curves` wurden verwendet. Die Farben sind folgenden Koordinatenachsen innerhalb der virtuellen Umgebung zugeordnet:

- Rot: X-Achse
- Grün: Y-Achse
- Blau: Z-Achse

Abbildung 6.1.: *Versuchsstapel: Stapel A instabil, Stapel B stabil*

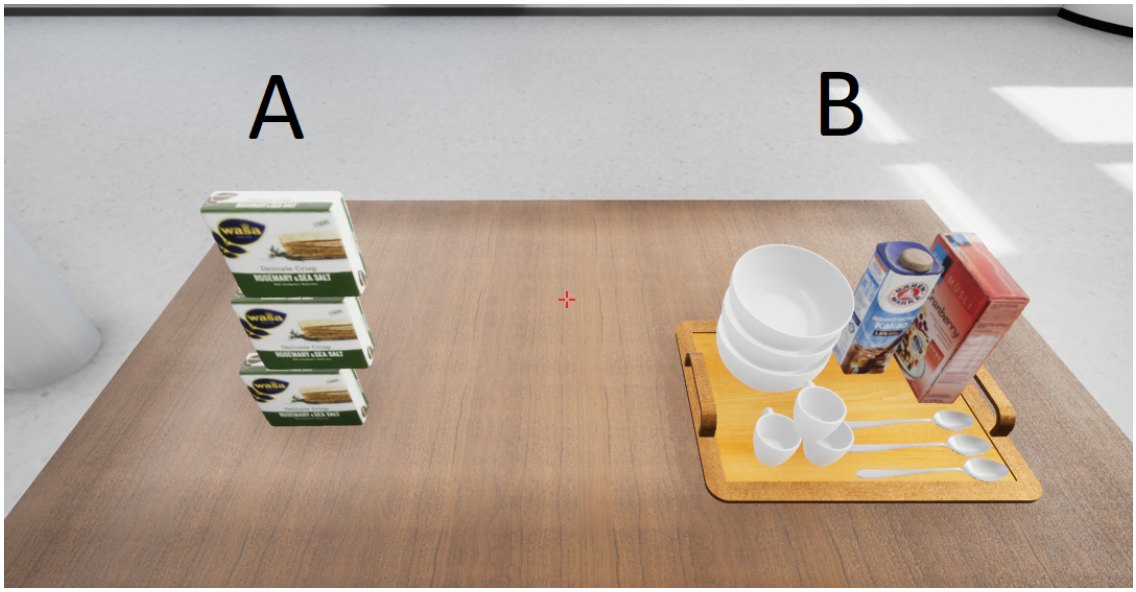


Abbildung 6.2.: *Animationskurve: TestRotation_Z_3sec*

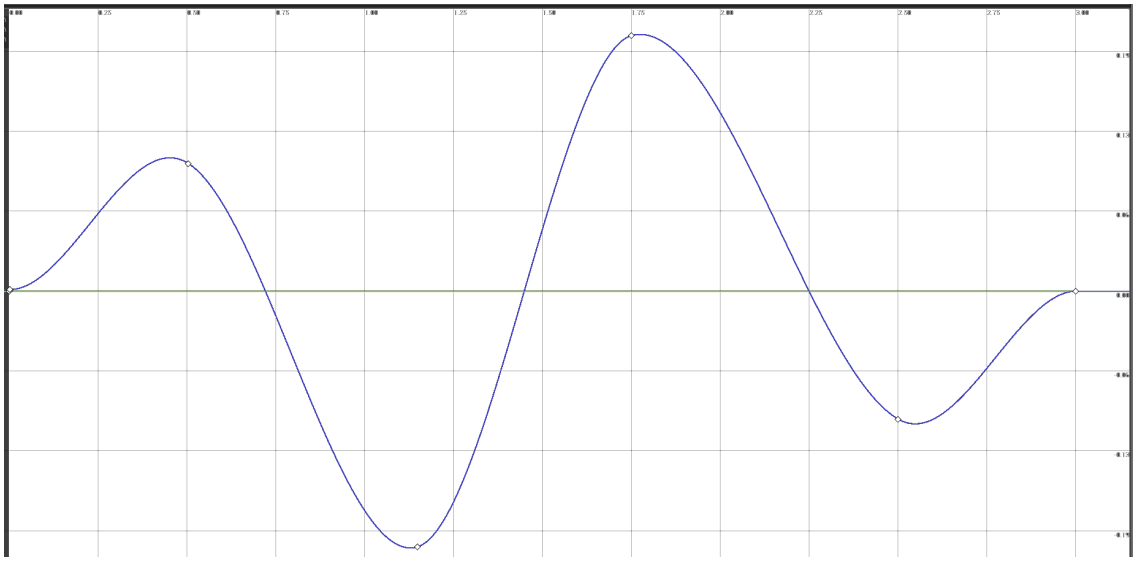


Abbildung 6.3.: *Animationskurve: TestRotation_Y_Z_3sec*

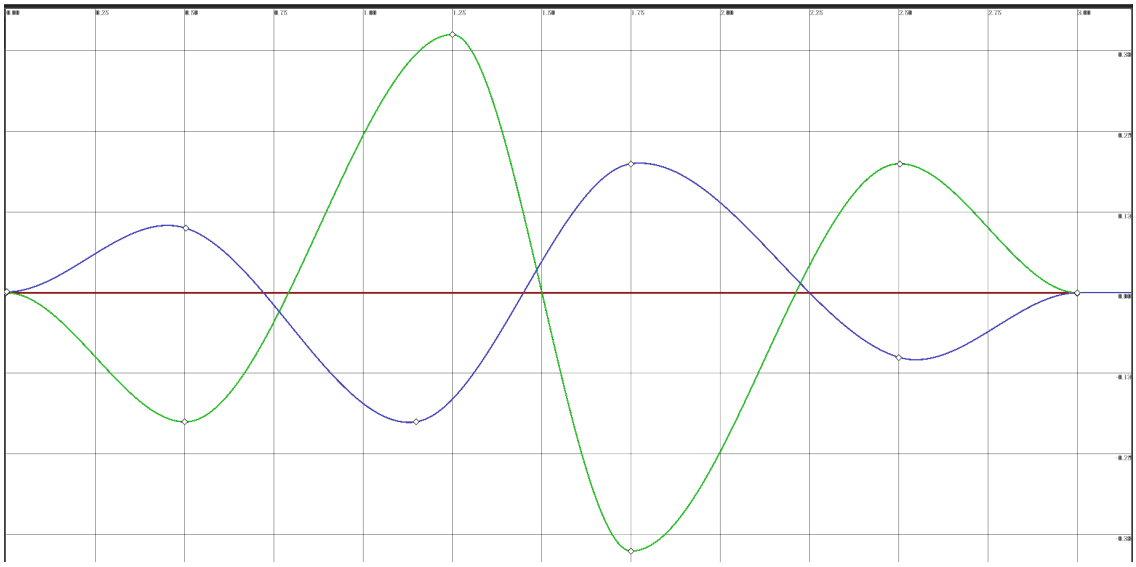


Abbildung 6.4.: *Animationskurve: TestRotation_Y_Z_1sec*

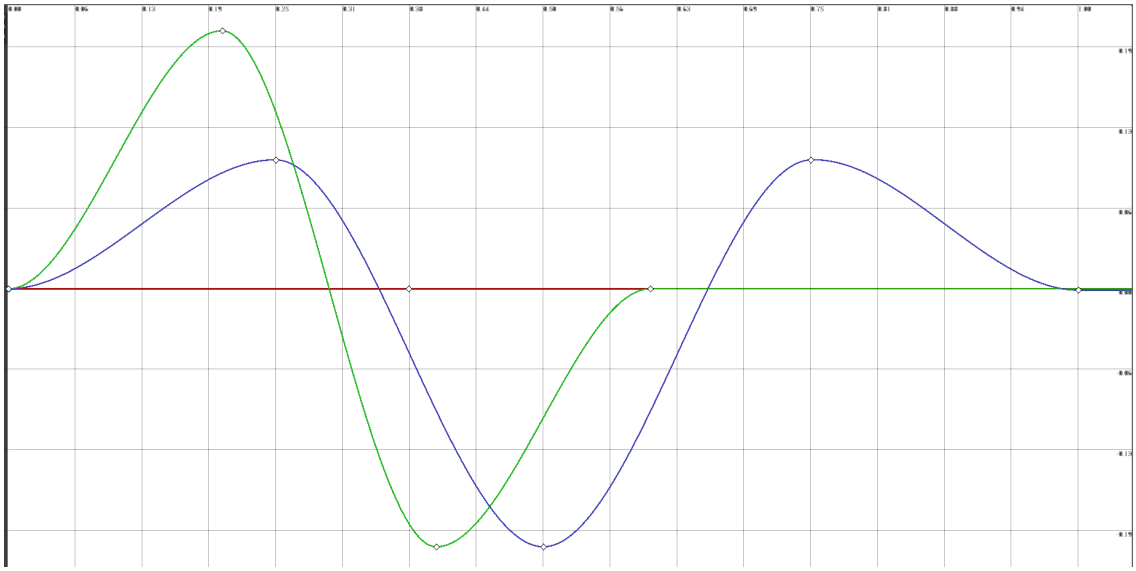


Abbildung 6.5.: *Animationskurve: TestShaking_X_Y_0_3sec*

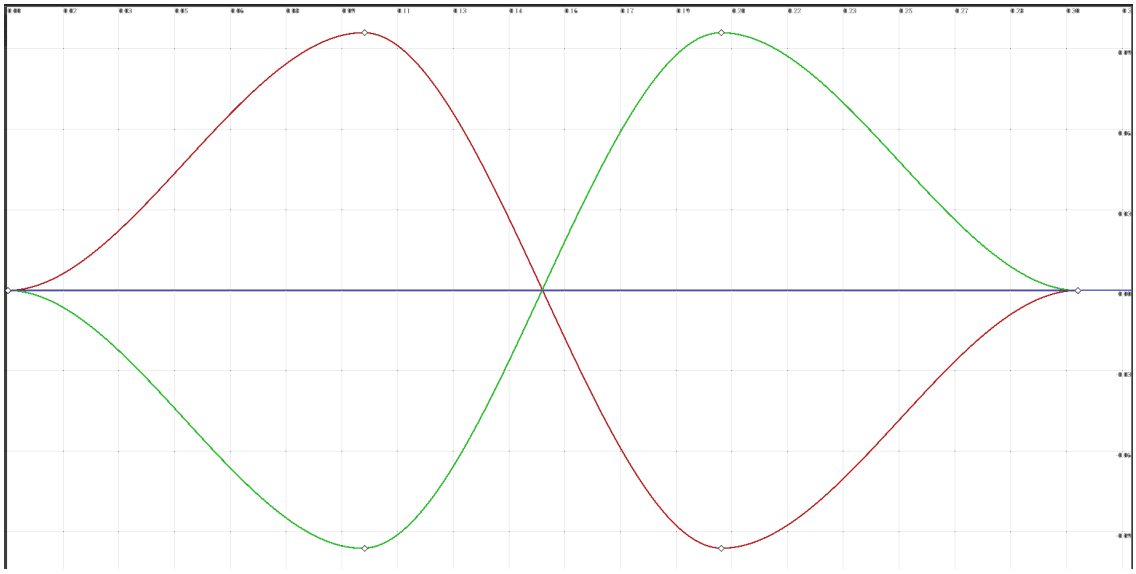


Abbildung 6.6.: *Animationskurve: TestShaking_X_Y_0_5sec*

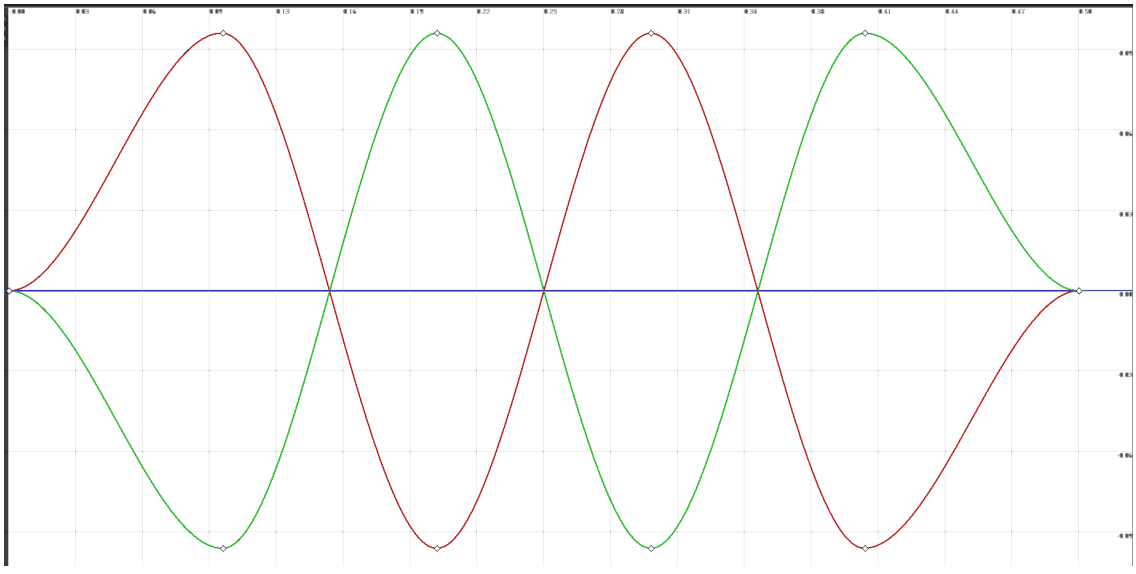
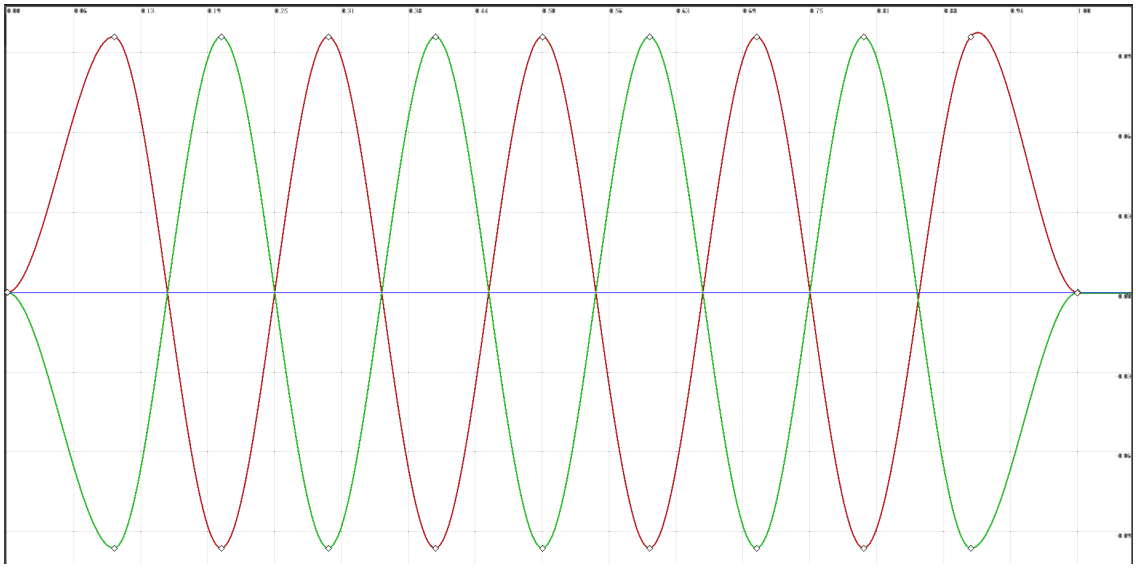


Abbildung 6.7.: *Animationskurve: TestShaking_X_Y_1sec*



6.1.1. Auswertung

Jeder Stapel wurde jeweils 5-mal getestet und die Ergebnisse in der nachfolgenden Tabelle 6.1 eingetragen.

Wie gut zu erkennen ist, erzeugt eine alleinige Rotation um Z- und Y- Achse (Zeile 1, 5 und 9) ein Falsch-Positiv - der instabile Stapel wird als stabil erkannt. Die Kombination *TestRotation_Z_3sec* und *TestShaking_X_Y_0_5sec* erkennt den stabilen Stapel hingegen dreimal als instabil (Zeile 3). Besonders interessant sind die Daten der Zeilen 9, 10 und 11. Mit zunehmender Shaking-Dauer verbessert sich die Erkennung des instabilen Stapels.

6.1.2. Fazit

Sowohl die Kombination *TestRotation_Y_Z_1sec* und *TestShaking_X_Y_0_5sec* als auch *TestRotation_Y_Z_1sec* und *TestShaking_X_Y_1sec* erzeugen gute Resultate in einer annehmbaren Zeit. Aus diesem Grund wird die Kombination *TestRotation_Y_Z_1sec* und *TestShaking_X_Y_1sec* für die weiteren Experimente verwendet.

Tabelle 6.1.: *Ergebnisse verschiedener Animationskurven des Stabilitätstests*

	Rotations-Kurve	Shaking-Kurve	Verhältnis Stabil : Instabil Stapel A	Verhältnis Stabil : Instabil Stapel B
1	TestRotation_Z_3sec	Keine	0:5	5:0
2	TestRotation_Z_3sec	TestShaking_X_Y_0_3sec	0:5	5:0
3	TestRotation_Z_3sec	TestShaking_X_Y_0_5sec	0:5	2:3
4	TestRotation_Z_3sec	TestShaking_X_Y_1sec	0:5	5:0
5	TestRotation_Y_Z_3sec	Keine	5:0	5:0
6	TestRotation_Y_Z_3sec	TestShaking_X_Y_0_3sec	0:5	5:0
7	TestRotation_Y_Z_3sec	TestShaking_X_Y_0_5sec	0:5	5:0
8	TestRotation_Y_Z_3sec	TestShaking_X_Y_1sec	0:5	5:0
9	TestRotation_Y_Z_1sec	Keine	5:0	5:0
10	TestRotation_Y_Z_1sec	TestShaking_X_Y_0_3sec	3:2	5:0
11	TestRotation_Y_Z_1sec	TestShaking_X_Y_0_5sec	0:5	5:0
12	TestRotation_Y_Z_1sec	TestShaking_X_Y_1sec	0:5	5:0
13	Keine	TestShaking_X_Y_0_3sec	1:4	5:0
14	Keine	TestShaking_X_Y_0_5sec	0:5	5:0
15	Keine	TestShaking_X_Y_1sec	0:5	5:0

6.2. Tischdeck-Szenarien

In diesem Kapitel werden einige Szenarien und Aufgaben erzeugt, welche die SpielerInnen zu erfüllen haben. Den einzelnen SpielerInnen wird dabei die Steuerung und Aufgabe erklärt. Anschließend können sie sich mit der Umgebung und der Steuerung vertraut machen. Wenn die SpielerInnen sich vorbereitet fühlen, können sie den Versuch starten, bei dem die Aktionen mit aufgezeichnet werden. Sie erhalten zusätzlich zu der Liste an Gegenständen eine Beispielgrafik, an der sie sich orientieren können.

Ein Szenario gilt als beendet, wenn sich alle Gegenstände auf dem Tisch befinden, alle Schubladen und Türen wieder geschlossen sind und (wenn genutzt) das Tablett sich wieder am Ursprungsort befindet. Auf Wunsch können die ProbandInnen einen Versuch wiederholen, sollten sie mit ihrer Leistung nicht zufrieden sein. Es wird jedoch pro ProbandIn nur ein Datensatz erhoben.

Jedes Szenario wird in allen drei Modi gespielt:

- Den SpielerInnen stehen nur eine Hand zur Verfügung (Ein-Hand-Modus)
- Den SpielerInnen stehen beide Hände zur Verfügung (Zwei-Hand-Modus)

- Den SpielerInnen stehen beide Hände zur Verfügung und sie können Stapel erzeugen (Stapelmodus)

6.2.1. Einstellungen

Folgende Einstellungen der Simulation werden für alle Experimente gesetzt:

6.2.1.1. Stability Check

Während der Prüfung der Stapelstabilität wird dieser für eine Sekunde um die Z-, und Y, Achse gedreht. Währenddessen wird der Stapel zusätzlich auch noch für eine Sekunde geschüttelt. Dies entspricht den `Animation Curves` *TestRotation_Y_Z_1sec* und *TestShaking_X_Y_1sec*.

Die maximale Abweichung der Position der einzelnen Gegenstände innerhalb des Stapels darf nicht mehr als **20** cm betragen. Nach dem Rotieren und Rütteln tritt eine Verzögerung von **1** Sekunde ein, bevor geprüft wird, ob der Stapel stabil ist.

6.2.1.2. Spielereinstellungen

Die Reichweite, in der die SpielerInnen Gegenstände manipulieren können, beträgt **120** cm und entspricht in etwa einer Reichweite die ein erwachsener Mensch im Stand mit seinen Händen erreichen kann, wenn er sich zusätzlich noch etwas nach vorn beugt. Für eine Strecke von 12 Metern (dies ist die Breite des Labors, in der sich die reale Küche befindet) werden bei normalem Gang 8 Sekunden benötigt. Daraus ergibt sich eine Schrittgeschwindigkeit von **1,5** $\frac{\text{m}}{\text{s}}$. Dies wird als maximale Geschwindigkeit gesetzt und entspricht dem Geschwindigkeitsmultiplikator von **0,25** innerhalb der Unreal Engine. Für die minimale Geschwindigkeit wird ein Geschwindigkeitsmultiplikator von **0,05** gesetzt, was einer Realgeschwindigkeit von **0,3** $\frac{\text{m}}{\text{s}}$ entspricht.

Die Werte für das maximale Gewicht eines Gegenstandes, den die SpielerInnen mit einer Hand heben bzw. für das maximale Gewicht, welches die SpielerInnen insgesamt tragen können, wurden abgeschätzt, basierend auf einem durchschnittlichen Menschen.

Einstellungen zum aufheben von Gegenständen:

- Für Gegenstände ab einem Gewicht von **1.5** kg werden beide Hände benötigt
- Das Volumen der Gegenstände hat **keine** Auswirkung darauf, ob beide Hände verwendet werden müssen

- Das Gewicht der getragenen Objekte beeinflusst die Geschwindigkeit. Es wird die **quadratische** Berechnung verwendet (siehe Kapitel 5.3.3)
- Das maximale Gewicht, das getragen werden kann, beträgt **10** kg
- Beim Aufheben von Objekten wird **nicht** auf Kollisionen geprüft, da dies die Steuerung für unerfahrene SpielerInnen zu kompliziert machen würde
- Beim Ablegen von Objekten **wird** auf Kollisionen geprüft

6.2.2. Szenarien

Im Folgenden werden die unterschiedlichen Szenarien erläutert.

6.2.2.1. E1: Kleines Frühstück für zwei Personen (8 Objekte)

Es soll ein Frühstückstisch für zwei Personen gedeckt werden. Den SpielerInnen wird vorab erklärt, wo welche Gegenstände zu finden sind. Die Anzahl der Objekte ist ähnlich der Anzahl der Gegenstände im Szenario E2, doch ist das Arrangieren der Objekte etwas herausfordernder, da mitunter um den Tisch herumgegangen werden muss.

Der Tisch sei mit folgenden Gegenständen zu decken:

- 2 mittelgroße Schüsseln
- 2 Kaffeetassen
- 2 Löffel
- Eine selbstgewählte Packung Frühstückszerealien
- Eine Packung Milch

Abbildung 6.8 zeigt beispielhaft, wie ein Endergebnis aussehen könnte.

Abbildung 6.8.: *Beispiel: Kleines Frühstück für 2 Personen*

6.2.2.2. E2: Großes Frühstück für eine Person (10 Objekte)

In diesem Szenario soll ein größeres Frühstück für eine Person aufgedeckt werden. Folgende Gegenstände sollen auf dem Tisch arrangiert werden:

- Eine kleine Schüssel
- Eine Kaffeetasse
- Ein Glas
- Ein mittelgroßer Teller
- Ein Messer und ein Löffel
- Eine selbstgewählte Packung Frühstückszerealien
- Eine Packung Milch
- Eine Packung Zwieback oder Knäckebrot
- Eine Packung Saft

Abbildung 6.9 illustriert, wie ein Tisch innerhalb dieses Szenarios gedeckt sein könnte.

Abbildung 6.9.: *Beispiel: Großes Frühstück für eine Person*

6.2.2.3. E3: Großes Frühstück für vier Personen (32 Objekte)

In diesem Szenario soll ein größeres Frühstück für vier Person aufgedeckt werden. Folgende Gegenstände sollten auf dem Tisch platziert werden:

- 4 kleine Schüsseln
- 4 Kaffeetassen
- 4 Gläser
- 4 mittelgroßer Teller
- 4 Messer und 4 Löffel
- 2 selbstgewählte Packungen Frühstückszerealien
- 2 Packungen Milch
- 2 Packungen Zwieback oder Knäckebrot
- 2 Packungen Saft

Abbildung 6.10 illustriert, wie ein Tisch innerhalb dieses Szenarios gedeckt sein könnte. In diesem Szenario stellt die große Anzahl der Gegenstände die SpielerInnen vor eine größere Herausforderung.

Abbildung 6.10.: *Beispiel: Großes Frühstück für 4 Personen*

6.2.3. Auswertung

Für die Auswertung wurden folgende Daten erhoben:

- Anzahl der durchgeführten Versuche für jedes Szenario
- Gesamtspielzeit jedes Szenarios
- Die Summe der Zeiten aller Aktionen (Aufheben, Ablegen, Ziehen eines Objekts und Öffnen und Schließen von Schubladen und Türen) pro Szenario
- Die Anzahl der durchgeführten Aktionen je Szenario
- Auch fehlgeschlagene Stapel gelten als gültige Aktion und werden ausgewertet
- Die ProbandInnen werden nicht untereinander verglichen. Lediglich die Modi und Szenarien werden verglichen

Über alle Testreihen wurde ein Durchschnitt ermittelt. Zusätzlich enthält jede Tabelle einen weiteren Durchschnittswert der einzelnen Modi über alle Szenarien hinweg, um zu verdeutlichen, welcher Modus z.B. wie viel Zeit im Durchschnitt benötigt.

Die Einzelauswertungen sind im Anhang der digitalen Version dieser Arbeit zu finden. Insgesamt wurden 12 Versuchsreihen unternommen, was 108 Spieldurchläufen entspricht. Bereits nach den ersten Versuchen wurde jedoch deutlich, dass sich die Spielweise der ProbandInnen untereinander sehr ähnelten. Z.B. gaben sich alle ProbandInnen sehr große Mühe, im Stapelmodus das Tablett so gut es geht zu nutzen, d.h. es mit so vielen Gegenständen wie möglich zu beladen. Daher wurde schnell deutlich, dass sich auch die gemessen Daten in Bezug auf die Verhältnisse zwischen den unterschiedlichen Modi zwischen unterschiedlichen ProbandInnen kaum voneinander unterscheiden werden. Aus diesem Grund war es nicht zweckmäßig, übermäßig viele Versuche durchzuführen.

Tabelle 6.2 fasst die Gesamtspielzeit zusammen. Für den Stapelmodus wurde dabei die Zeit, die für den Stabilitätstest verbraucht wurde, herausgerechnet.

Tabelle 6.2.: *Gesamtspielzeit in Sekunden pro Szenario*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	140.38	94.74	128.86
E2: (10)	181.87	121.86	166.10
E3 (32)	358.46	242.23	348.61
Ø	226.90	152.94	214.52

Tabelle 6.3 zeigt die Gesamtdauer der einzelnen Aktionen, Tabelle 6.4 die Anzahl aller Aktionen. Wie auch in Tabelle 6.2 wurden die Daten abzüglich der Zeit bzw. Anzahl der Aktionen des Stabilitätstests aufgelistet.

Tabelle 6.3.: *Gesamtzeit in Sekunden der Aktionen*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	24.32	19.73	30.20
E2: (10)	30.77	26.11	43.00
E3 (32)	47.90	43.60	86.37
Ø	34.33	29.81	53.19

Tabelle 6.4.: *Anzahl an Aktionen*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	31.83	30.75	53.42
E2: (10)	37.08	37.17	63.58
E3 (32)	89.33	88.67	173.50
Ø	52.75	52.19	96.83

In Tabelle 6.5 wurde die Differenz zwischen Spielzeit und der Zeit aller Aktionen pro Szenario zusammengefasst. Dies entspricht der Zeit, in der sich die SpielerInnen bewegen oder nachgedacht haben.

Tabelle 6.5.: *Differenz von Spielzeit und Aktionszeit*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	116.06	75.01	98.65
E2: (10)	151.10	95.75	123.10
E3 (32)	310.56	198.63	262.23
Ø	192.57	123.13	161.33

In Tabelle 6.6 ist das Verhältnis zwischen Aktionszeit und Aktionsanzahl aufgezeigt und verdeutlicht, wie lange im Durchschnitt eine Einzelaktion dauerte.

Tabelle 6.6.: *Verhältnis zwischen Aktionszeit und Aktionsanzahl*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	0.76	0.64	0.57
E2: (10)	0.83	0.70	0.68
E3 (32)	0.54	0.49	0.50
Ø	0.71	0.61	0.58

6.2.3.1. Verhältnisse zwischen einzelnen Modi

Da in dieser Arbeit untersucht wird, wie sich die einzelnen Modi zueinander verhalten, werden diese in Verhältnis gesetzt. Tabellen 6.7 bis 6.11 verdeutlichen diese Verhältnisse. Es wurden jeweils der Ein-Hand- mit dem Zwei-Hand- und der Stapelmodus mit dem Zwei-Hand-Modus verglichen.

Tabelle 6.7.: *Verhältnis der Modi: Spielzeit*

Szenario (#Objekte)	Ein-Hand : Zwei-Hand	Stapel : Zwei-Hand
E1: (8)	1.48	1.36
E2: (10)	1.49	1.36
E3 (32)	1.48	1.44
Ø	1.48	1.40

Betrachtet man das Verhältnis der Spielzeit zwischen den einzelnen Modi, ist zu erkennen, dass der Ein-Hand-Modus im Durchschnitt 48% und der Stapelmodus 40% mehr Zeit gegenüber dem Zwei-Hand-Modus benötigt (Tabelle 6.7). Diesen Werten zufolge ist der Zwei-Hand-Modus der schnellste gewesen.

Tabelle 6.8.: *Verhältnis der Modi: Aktionszeit*

Szenario (#Objekte)	Ein-Hand : Zwei-Hand	Stapel : Zwei-Hand
E1: (8)	1.23	1.53
E2: (10)	1.18	1.65
E3 (32)	1.10	1.98
Ø	1.15	1.78

Besonders groß ist der Unterschied in puncto Aktionszeit zwischen Stapel- und Zwei-Hand-Modus (Tabelle 6.8). Der Stapelmodus benötigt vor allem im Szenario E3 fast doppelt so

viel wie der Zwei-Hand-Modus; im Durchschnitt 78% mehr Zeit gegenüber des Zwei-Hand-Modus.

Tabelle 6.9.: *Verhältnis der Modi: Aktionsanzahl*

Szenario (#Objekte)	Ein-Hand : Zwei-Hand	Stapel : Zwei-Hand
E1: (8)	1.04	1.74
E2: (10)	1.00	1.71
E3 (32)	1.01	1.96
Ø	1.01	1.86

Zu Tabelle 6.9: Wie zu erwarten gewesen, ist die Anzahl der Einzelaktionen im Zwei-Hand- und Ein-Hand-Modus nahezu identisch. Der Stapelmodus benötigt hingegen fast doppelt so viele Aktionen wie der Zwei-Hand-Modus (86 % im Durchschnitt).

Tabelle 6.10.: *Verhältnis der Modi: Differenz zwischen Spielzeit und Aktionsanzahl*

Szenario (#Objekte)	Ein-Hand : Zwei-Hand	Stapel : Zwei-Hand
E1: (8)	1.55	1.32
E2: (10)	1.58	1.29
E3 (32)	1.56	1.32
Ø	1.56	1.31

Die Zeit, die nicht mit Aktionen wie Ablegen, Aufheben, Öffnen und Schließen verbracht wurde, ist beim Ein-Hand-Modus um 56 %, beim Stapelmodus um nur 31 % größer gegenüber dem Zwei-Hand-Modus (Tabelle 6.10).

Tabelle 6.11.: *Verhältnis der Modi: Quotient aus Aktionszeit und Aktionsanzahl*

Szenario (#Objekte)	Ein-Hand : Zwei-Hand	Stapel : Zwei-Hand
E1: (8)	1.19	0.88
E2: (10)	1.18	0.96
E3 (32)	1.09	1.01
Ø	1.16	0.95

Die Zeit, die im Schnitt pro Aktion benötigt wurde, ist im Ein-Hand-Modus um 16 % höher als im Zwei-Hand-Modus. Der Stapelmodus hingegen liegt nahezu gleichauf mit dem Zwei-Hand-Modus (Tabelle 6.11).

6.2.4. Deutung der Ergebnisse

6.2.4.1. Vergleich: Ein-Hand- und Zwei-Hand-Modus

Vergleicht man die ausgewerteten Daten zwischen Ein- und Zwei-Hand-Modus, ist erkennbar, dass ersterer weniger effizient ist und etwa 48 % mehr Gesamtzeit beansprucht (Tabelle 6.7). Da der Ein-Hand-Modus kein großes Planen erfordert, kann der Unterschied in der Differenz zwischen Spiel- und Aktionszeit (Tabelle 6.5) nur daraus resultieren, dass die SpielerInnen häufiger hin und her laufen mussten. In beiden Modi ist die Anzahl an Aktionen jedoch nahezu gleich, da unabhängig vom Modus je eine Aktion pro Hand aufgezeichnet wird.

Eine mögliche Erklärung dafür, dass die Aktionszeit im Ein-Hand-Modus etwas größer ist als im Zwei-Hand-Modus, könnte die Tatsache sein, dass jedes Szenario im Ein-Hand-Modus gestartet wird und sich die SpielerInnen somit erst an das Arrangieren der Gegenstände auf dem Tisch gewöhnen müssen. Das heißt, dass z.B. zwischen dem Start und dem Ende der Ablegen-Aktion die Interaktionstaste länger gedrückt gehalten wurde und somit die Aktionszeit verlängert wird.

6.2.4.2. Vergleich: Zwei-Hand-Modus und Stapelmodus

Anders als erwartet, ergab die Fähigkeit, Stapel zu erzeugen, keine kürzeren Zeiten. Die Gründe dafür können unterschiedlichen Ursprung haben. Zum einen hatten einige SpielerInnen Schwierigkeit mit der Steuerung. Z.B. war Ablegen oft nicht präzise, so dass ein Gegenstand nochmals aufgehoben werden musste. Dies war vor allem beim Bestücken des Tabletts häufig der Fall und kostete Zeit. Die Ursache für dieses unpräzise Verhalten liegt an der mangelnden räumlichen Wahrnehmung innerhalb der virtuellen Umgebung. Dies hat zur Folge, dass beim Platzieren von Objekten nicht immer sofort deutlich ist, ob mit einem anderen Gegenstand kollidieren wird. Beim Ablegen fielen daher oftmals Gegenstände auf den Boden und mussten erneut platziert werden.

Während der Beobachtung der ProbandInnen konnte auch festgestellt werden, dass der einfache Zwei-Hand-Modus (ohne Stapel) kein großes Nachdenken erforderte und dennoch effizient genug für die SpielerInnen gewesen ist, während im Stapelmodus öfter nachgedacht wurde, auf welche Weise ein günstiger Stapel gebaut werden könnte. Im Stapelmodus wurden deswegen weitaus mehr Aktionen durchgeführt (Tabelle 6.4). Betrachtet man jedoch das Verhältnis zwischen der Zeit der Aktionen und die Anzahl der Aktionen (Tabelle 6.6), ist zu erkennen, dass die Durchschnittswerte für beide Modi annähernd gleich sind. Dass die SpielerInnen mehr mit Denken beschäftigt waren, kann auch durch die Differenz zwischen Spielzeit und Aktionszeit interpretiert werden. Wie in Tabelle 6.5 zu sehen ist, wird mehr Zeit “aktionslos“ verbraucht. Da sich die zurückgelegte Entfernung beim Transport der Gegenstände zwischen den Modi jedoch nicht änderte, und demnach diese Zeit nicht durch Herumlaufen verbraucht wurde, lässt sich vermuten, dass in dieser Zeit nachgedacht bzw. geplant wurde.

Je komplexer das Szenario und je mehr Gegenstände es zu transportieren galt, desto mehr Zeit wurde gebraucht, um sich Gedanken über das Stapeln zu machen.

Des Weiteren hat der Stapelmodus 86 % mehr Aktionen benötigt (Tabelle 6.9). Dies ist dadurch begründet, dass ein Gegenstand in der Regel aus dem Schrank genommen und auf das Tablett gelegt wurde (2 Aktionen), das Tablett in die Hand genommen und wieder abgelegt (2 Aktionen) und anschließend alle Gegenstände wieder vom Tablett genommen und auf dem Tisch abgelegt wurden (2 Aktionen). Im einfachen Zwei-Hand-Modus hingegen wurde ein Gegenstand lediglich aufgenommen (1 Aktion) und wieder auf den Tisch abgelegt (1 Aktion). Für den Transport vieler Gegenstände können die Aktionen zum Anheben und Ablegen des Tabletts im Stapelmodus vernachlässigt werden, woraus sich ergibt, dass fast doppelt so viele Aktionen im diesem Modus gemacht wurden, als im einfachen Zwei-Hand-Modus.

6.2.5. Expertentest

Da alle ProbandInnen keinerlei Erfahrung mit der Umgebung und den Szenarien hatten, verhielten diese sich während der Versuche zueinander sehr ähnlich. Es ließ sich schlussfolgern, dass mehr ProbandInnen-Versuche nicht zu anderen Ergebnissen führen würden. Es stellte sich daher die Frage, ob längeres Training eine Änderung im Verhältnis zwischen Stapel- und Zwei-Hand-Modus bewirken könnte. Es wurde daher von mir persönlich ein weiterer Versuch unternommen, da ich im Laufe der Entwicklung, viele Stunden damit verbracht habe, mich mit der Umgebung, der Position der Gegenstände und der Steuerung vertraut zu machen. Demnach unterlag ich nicht den oben beschriebenen Problemen, welche die ProbandInnen hatten. Zusätzlich machte ich mir vorab Gedanken über geeignete Strategien, wie der Stapelmodus effizient genutzt werden könnte. Die damit gewonnenen Daten (welche nicht in die obige Wertung einflossen) werden in diesem Abschnitt kurz vorgestellt.

Tabellen 6.12 und 6.13 verdeutlichen, dass mit mehr Übung die Gesamtspielzeit im Stapelmodus durchaus von allen drei Modi die geringste sein kann. In diesem Fall benötigte der Stapelmodus 9 % weniger Zeit als der Zwei-Hand-Modus. Vergleicht man die Gesamtspieltzeit des Versuches mit der Spielzeit der unerfahrenen SpielerInnen (Tabelle 6.2), sind große Unterschiede erkennbar.

Tabelle 6.12.: *Spieltzeit einer geübten Person*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	77.97	78.7	53.9
E2: (10)	120.41	71.76	83.12
E3 (32)	282.13	203.8	184.63
Ø	160.17	118.09	107.21

Tabelle 6.13.: *Verhältnis der Spieltzeit einer geübten Person*

Szenario (#Objekte)	Ein-Hand : Zwei-Hand	Stapel : Zwei-Hand
E1: (8)	0.99	0.68
E2: (10)	1.68	1.16
E3 (32)	1.38	0.91
Ø	1.36	0.91

Tabelle 6.14 zeigt deutlich, dass im Unterschied zu ungeübten Personen (Tabelle 6.3) die Aktionszeiten im Stapelmodus weitaus geringer sind. Zwei-Hand-Modus und Stapelmodus unterscheiden sich bei einer geübten Person nur noch um wenige Sekunden. Im Vergleich dazu, braucht eine ungeübte Person 78 % mehr Zeit für Einzelaktionen (siehe Tabelle 6.8). Demnach denkt eine geübte Person weniger lange nach, wie sie ein Objekt platziert bzw. benötigt für diese Aktionen weniger Zeit und braucht im Schnitt nur knapp 30 % mehr Zeit im Stapelmodus für Aktionen $\left(\frac{34.40}{26.55} \approx 1.2957\right)$

Tabelle 6.14.: *Aktionszeit einer geübten Person*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	13.2	18.38	17.6
E2: (10)	21.41	14.63	32.26
E3 (32)	40.32	46.63	53.35
Ø	24.98	26.55	34.40

Die Anzahl der Einzelaktionen (Tabelle 6.15) hingegen ist bei einer geübten Person größer, da sich die Spielweise grundlegend unterscheidet. So wurden Gegenstände oftmals den Schubladen entnommen, auf die Anrichte gestellt, die Schublade geschlossen und dann erst zum Tisch transportiert. So musste zum Ende des Szenarios nicht noch einmal zurückgegangen werden, um alle Schubladen wieder zu schließen.

Tabelle 6.15.: *Anzahl an Aktionen einer geübten Person*

Szenario (#Objekte)	Ein-Hand	Zwei-Hand	Stapelmodus
E1: (8)	42	44	44
E2: (10)	52	43	59
E3 (32)	142	138	146
Ø	78.67	75.00	83.00

Ein Beispiel: In Szenario E3 sollen 4 Schüsseln und 4 Teller auf den Tisch gestellt werden. Beide Arten von Gegenständen befinden sich in zwei Schubladen, die direkt untereinander liegen.

Alle 8 Gegenstände wurden zuerst auf die Anrichte gestellt. Dies ist insofern schneller, da der virtuelle Charakter sich nicht von seiner Position bewegen muss. So musste ausschließlich die Maus auf und ab bewegt und die Mausknöpfe gedrückt werden, um alle 8 Gegenstände schnell den Schubladen zu entnehmen und wieder abzustellen. Nachdem die Schubladen geschlossen wurden, brauchte man sich nur noch zwischen Anrichte und Tisch hin und her bewegen, um Schüsseln und Teller auf den Tisch zu legen. Dies kostete zwar mehr Aktionen, verringerte jedoch die Gesamtzeit.

Das Aufteilen in einzelne “Phasen“ (Phase 1: Objekte aus Schubladen nehmen und abstellen, Phase 2: Objekte von Anrichte auf Tisch stellen) scheint auch in anderen Fällen eine gute Strategie zu sein und kann leicht angepasst werden. Beispiel: Ebenfalls in Szenario E3 müssen insgesamt 4 Gegenstände aus dem Kühlschrank genommen und auf das Tablett gestellt werden. Phase 1 bestand also daraus, alle Gegenstände aus dem Kühlschrank zu nehmen und auf die Anrichte neben der Spüle zu stellen. In Phase 2 stellt man sich auf eine Position zwischen dem Tablett und den Objekten auf der Spüle, nimmt die Gegenstände von dort, dreht sich ein bisschen und legt sie auf das Tablett ab. Die Bewegungstasten sind für diese Phase nicht notwendig gewesen. Anschließend konnte das Tablett zum Tisch gebracht werden. Beobachtet werden konnte, dass vor allem das Bewegen bzw. Neupositionieren vor einer Schublade unerfahrenen Spielern viel Zeit gekostet hat. Wird dies im Stapelmodus vermieden, verringert sich auch die Gesamtspielzeit in diesem Modus.

Weitere Expertenversuche wurden jedoch nicht durchgeführt, da es den ProbandInnen nicht zumutbar war, sich mehrere Dutzend Stunden mit dem Programm zu beschäftigen und somit genügend Fähigkeiten anzutrainieren, um an einem Expertenversuch teilzunehmen.

Kapitel 7

Fazit

In dieser Bachelorarbeit wurden Funktionalitäten implementiert, die es SpielerInnen in einer virtuellen Küche ermöglichten, Objekte und Einrichtungsgegenstände zu manipulieren. Die SpielerInnen konnten somit mit ihrer Umgebung interagieren, um vorgegebene Aufgaben zu erfüllen.

Es wurden drei Szenarien für ProbandInnen erstellt, in denen sie einen Frühstückstisch zu decken hatten. Jedes Szenario wurde in den Modi “Ein-Hand“, “Zwei-Hand“ und “Stapelmodus“ durchgeführt. Während die ProbandInnen ihre Aufgaben erfüllten, wurden wichtige Daten wie Spielzeit, Anzahl an durchgeführten Aktionen etc. erhoben und später ausgewertet. Es wurde geprüft, welche der drei Modi die geringste Zeit benötigt.

Die Ergebnisse der Testversuche waren überraschend. Es zeigte sich, dass für unerfahrene SpielerInnen der Zwei-Hand-Modus ohne die Fähigkeit zu Stapeln der schnellste ist, also weniger Spielzeit erfordert, um die gegebene Aufgabe zu erfüllen. Auf Grund der mangelnden Erfahrung mit der Umgebung und dem Stapelmechanismus benötigte der Stapelmodus fast doppelt so viel Zeit wie der Zwei-Hand-Modus. Gründe dafür sind unter anderem mangelnde Erfahrung mit der Steuerung und der Spielwelt.

Zusätzlich wurde ein Einzelversuch mit einem erfahrenem Spieler gemacht, um zu zeigen, dass es dennoch möglich ist, mit genügend Training die Aufgaben unter Verwendung des Stapelmodus effizient zu bewältigen. Es liegt die Vermutung nahe, dass mit weiterer Übung noch bessere Ergebnisse erzielt werden können.

Kapitel 8

Ausblick

Wie bereits im Fazit erwähnt, konnte nachgewiesen werden, dass erfahrene SpielerInnen effiziente Strategien für die einzelnen Szenarien entwickeln können, in dem sie den Stapelmodus verwenden. Daraus lässt sich also folgern, dass mit Langzeitversuchen ausreichend gute Strategien aus den Daten gewonnen werden können. Eine Möglichkeit wäre, das Programm über den Webbrowser zugänglich zu machen, um somit eine größere Spieleranzahl zu erreichen, welche eigenständig nach optimalen Lösungen suchen können.

Um den Realitätsgrad der Simulation zu erhöhen, könnte die Stabilitätsprüfung ersetzt werden, in dem die SpielerInnen den Stapel direkt zur Hand nehmen können. Die Physik müsste dann während des Tragens des Stapels aktiviert bleiben, was die SpielerInnen dazu motiviert, sich vorsichtig zu bewegen, damit ihnen der Stapel nicht entgleitet. Dies würde jedoch neue Bewegungsmechanismen voraussetzen (z.B. justierbare Bewegungsgeschwindigkeit bzw. graduell steigende und fallende Geschwindigkeit beim Loslaufen und Stehenbleiben).

Ebenso könnte das visuelle Feedback für die SpielerInnen z.B. beim Ablegen von Gegenständen deutlicher gemacht werden, so dass diese schneller erkennen, auf welcher Position das abzulegende Objekt nach der Ablege-Aktion stehen wird. Ebenso vorstellbar wäre eine Implementierung in Virtual Reality, um eine dreidimensionale Wahrnehmung zu ermöglichen.

Mit den gesammelten Daten, die durch das Plugin USemLog erzeugt werden, könnte daraufhin eine Wissensdatenbank für Roboter erzeugt werden, mittels der sie lernen, effiziente Strategien zum Aufheben und Ablegen von Gegenständen zu entwickeln, um z.B. bestimmte Aufgaben zu erledigen wie es auch in der Arbeit von HAIDU und BEETZ [11] umgesetzt wurde. Ebenso kann dadurch gelernt werden, welche Gegenstände übereinander stapelbar sind und dadurch als Stapel transportiert werden können.

Literaturverzeichnis

[1] "Baxter", http://www.rethinkrobotics.com/de/baxter/, Stand: 30.08.2017.

[2] "Boxy", http://ai.uni-bremen.de/research/robots/boxy, Stand: 30.08.2017.

[3] Luis Von Ahn and Laura Dabbish, "Designing games with a purpose", *Communications of the ACM*, vol. 51, no. 8, pp. 58–67, August 2008, https://cacm.acm.org/magazines/2008/8/5341-designing-games-with-a-purpose/fulltext. Stand: 30.08.2017.

[4] Luis Von Ahn, "Games with a purpose", *IEEE Computer Magazine*, pp. 96–98, June 2006, https://www.cs.duke.edu/courses/cps296.3/spring07/ieee-gwap.pdf. Stand: 30.08.2017.

[5] "Epic Games homepage", https://www.unrealengine.com/, Stand: 30.08.2017.

[6] "Robot Commonsense Games", http://www.robcog.org/games.html, Stand: 30.08.2017.

[7] "UTags", https://github.com/robcog-iai/UTags, Stand: 30.08.2017.

[8] "USemLog", https://github.com/robcog-iai/USemLog, Stand: 30.08.2017.

[9] "openEASE", http://www.open-ease.org/, Stand: 04.09.2017.

[10] "RobCoG", https://github.com/robcog-iai/RobCoG/, Stand: 30.08.2017.

[11] A. Haidu and M. Beetz, "Action recognition and interpretation from virtual demonstrations", in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2833–2838.

Anhang A

Inhalte des Datenträgers

- Eine digitale Kopie dieser Bachelorarbeit im PDF-Format
- Ein Archiv mit dem vollständigen Unreal-Projekt
  - Eigener Quellcode in Ordner: \Plugins\ClickInteraction
- Die Plugins UTags und USemLog in der Version, die während der Implementation genutzt wurde (im Ordner \Plugins\ des Unreal-Projekts)
- Erhobene Datzensätze der ProbandInnenversuche
- Erhobener Datensatz des Expertenversuchs
- Einige Beispiele von `.owl`-Dateien, die durch das Plugin `USemLog` erzeugt wurden