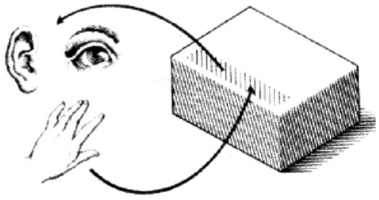
Projekt Hand_Griff
Bauhaus-Universität Weimar
Sommersemester 2003

Fakultät Gestaltung
Prof. Wolfgang Sattler
Dipl.-Ing. Stefan Kraus

Fakultät Medien
Prof. Dr. Bernd Fröhlich
Dipl.-Inf. (FH) Jan Hochstrate



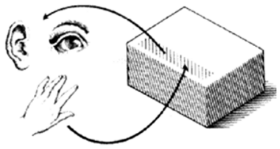
Wind_Craft

Robert Gerling (MS), Jonas Schild (MS), Tina Schenk (PD)

Kontakt:
http://ww.uni-weimar.de/medien/vr
robert.gerling@medien.uni-weimar.de
tina.schenk@gestaltung.uni-weimar.de
jonas.schild@medien.uni-weimar.de

Inhaltsverzeichnis

1. Hand_Griff – das Projekt



Greifen und Begreifen im Informationszeitalter – welche Rolle spielt dabei Ergonomie?

Das Projekt Hand_Griff beschäftigte sich mit der Gestaltung von Eingabegeräten und 3D-Anwendungen in der Virtuellen Realität. Die Teilnehmer aus den Studiengängen Produktdesign und Mediensysteme bildeten Gruppen, in denen interdisziplinäre Lösungen gesucht wurden. Die so entstandenen Prototypen setzen sich mit Ergonomie und neuen Inputs auseinander und zeigen neue Möglichkeiten auf, die sich von den herkömmlichen Geräten wie der Maus unterscheiden.

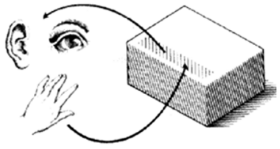
2. Projektverlauf

Der Einstieg in das Projekt begann mit kleinen Einführungsvorträgen der Projektteilnehmer. Während die Produktdesigner Studien zu Gesten, Gebärden und Kommunikation zeigten, gaben die Studenten des Studiengangs Mediensysteme Einblicke in die Techniken, Anwendungen und Möglichkeiten der Virtuellen Realität. Im weiteren Verlauf wurden Gruppen gebildet, die erste Anwendungen und Geräte realisierten. Diese Ideen bildeten die Grundlage für die projektinterne Zwischenpräsentation. Zum Rundgang waren Prototypen und Anwendungen zu sehen und konnten auch ausprobiert werden. Die Endpräsentation bildete den Abschluß des Projektes.

Unsere Gruppe (Robert Gerling, Jonas Schild, beide Mediensysteme und Tina Schenk, Produktdesign) hatte auf Grund ihrer Größe den Vorteil, dass die Zusammenarbeit untereinander sehr gut funktioniert hat. Regelmäßige Treffen ausserhalb des Plenums waren die Basis für die gemeinsamen Entscheidungen. Wir haben die Ideen und Konzepte für unsere Anwendungen gemeinsam erarbeitet. In der Umsetzung hat jeder in seinem Bereich gearbeitet, aber immer in Rück- und Terminabsprache mit den anderen.

Wind in der virtuellen Realität und Pusten als Input – das ist der Schwerpunkt der interdisziplinären Zusammenarbeit.Wir haben unsere Arbeit unter dem Titel Wind_Craft zusammengefasst, da sich gezeigt hat, wie wichtig nicht nur digitale Werte sind, sondern auch die handwerklichen Fähigkeiten, um ein solches Projekt zu verwirklichen. Jeder von uns hat nach seinen Möglichkeiten sein „Handwerk“ ausgeübt, damit am Ende zwei unterschiedliche Anwendungen mit dem gleichen Input entstehen. Die unterschiedlichen Herangehens- und Arbeitsweisen der beiden Fakultäten stellten einen Vorteil dar und trugen zur Ideenvielfalt bei.

Ein entscheidender Faktor unserer Arbeit war die jeweilige Abhängigkeit einzelner Faktoren untereinander. Das Eingabegerät mit seiner Interaktionsmöglichkeit, seiner Sensorik und



dem Design beeinflussen die Anwendung in ihrem Konzept und in der Visualisierung. Diese beiden Faktoren beeinflussen sich gegenseitig, sowie die Zielgruppe, die wiederum ihre Anforderungen an das Gerät und die Anwendung stellt. Es entstehen also Wechselwirkungen, die bei der Realisierung beachtet werden müssen. Diese Abhängigkeit der Faktoren zieht sich wie ein roter Faden durch das ganze Projekt. Als Zielgruppe haben wir uns Spielekonsumenten, Science-Center und Museen, sowie die Industrie herausgenommen. Die Arbeiten, die im Rahmen des Projektes entstanden sind, werden in chronologischer Reihenfolge beschrieben. Die Erfahrungen des Rundgangs und der Testphasen unserer Anwendungen gehen in die Evaluation dieser Dokumentation ein.

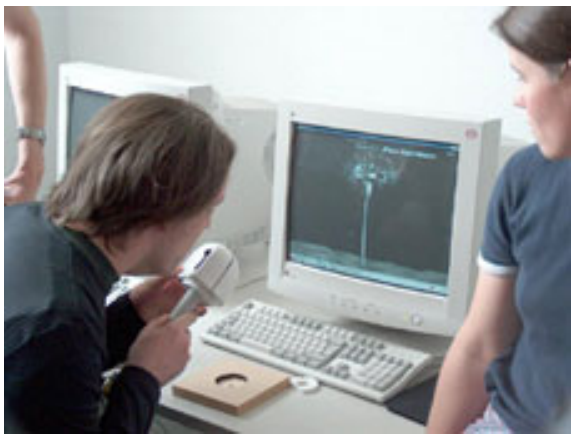
2.1 Pusten als Input

Skizze der Anwendung

Auf der Suche nach einem ersten Interaktionsszenario mit einem Eingabegerät und einer kleinen Anwendung haben wir uns auf ein Gerät mit Steckprinzip und eine kleine Spielszene geeinigt. Für den Bau standen uns diverse feinelektronische Bauteile zur Verfügung, unter anderem Buttons, Pontentiometer, Minijoysticks und A/D-Wandler. Dieser Umwandler sorgt dafür, dass die Eingabegeräte über die USB-Schnittstelle an den Computer angeschlossen werden. Die Idee, mit Hilfe von Schiebe- oder Drehpotentiometern kleine virtuelle Ventilatoren anzutreiben, um so Federn oder kleine Kuglen in der Luft zu halten, führte dazu, Pusten direkt als Input zu verwenden. Mit Hilfe eines handelsüblichen CPU-Lüfters, der normalerweise zum Kühlen von Prozessoren verwendet wird, haben wir es geschafft, eine Spannung zu induzieren (Generatorprinzip).

Steckmodell

Diese Spannung kann von einem Analog/Digitalwandler erkannt werden und liefert Daten an den Computer. Der nächste Schritt bestand darin, ein erstes Funktionsmodell zu bauen, während parallel dazu die Anwendung programmiert wurde.

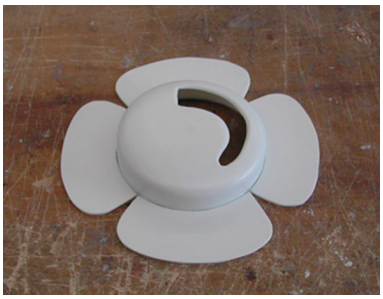


Ausprobieren der Spielszene bei der ersten Präsentation

3. Pusteblume

3.1 Konzept des Eingabegerätes

Das Design des Eingabegerätes ist als Analogie zu verstehen. Man nimmt eine Spielzeugblume in die Hand und kann damit eine virtuelle ansteuern. Unsere Blume hat Blütenblätter, da sich daraus ein größerer Wiedererkennungswert ergibt. Ein Löwenzahnkopf ohne Samen ist nicht wirklich eindeutig erkennbar, wenn man versucht, ihn in seiner Formensprache zu reduzieren. Ausserdem war es uns wichtig, gleich sehen zu können, dass es sich bei der Anwendung um ein Spiel handelt. Daher rührt die schon fast comicartige Formensprache.


Halbschale Deckel


Rückseite Deckel, am Griff befestigt.

3. 1. 1 Umsetzung

Die Technik ist die gleiche, die wir auch schon bei unserem ersten Funktionsmodell verwendet haben. Der CPU-Lüfter ist auch wieder in den Kopf der Blume integriert, der A/D-Wandler befindet sich im Stiel und die Resettaste im Fuß. Die Proportionen ergeben sich aus der Größe der integrierten Teile. Der Lüfter hat einen Durchmesser von acht Zentimetern. Daher war es nötig, zwei recht große Halbschalen für den Kopf herzustellen. Die Schalen bestehen aus Polystyrol und sind tiefgezogen. Der geneigte Griff aus PVC-Schlauch ermöglicht es, den Standfuß recht klein zu halten, da durch die Neigung der Schwerpunkt so verlagert wird, dass trotz relativ kleiner Standfläche ein sicheres Abstellen und Drücken der Resettaste möglich ist. Der Fuß besteht aus Polyurethanschaum und wurde auf der Drehbank hergestellt. Es wäre auch möglich gewesen, diese Arbeit mit Hilfe der CNC-Fräse durchzuführen, allerdings dauert der Fräsvorgang wesentlich länger, als das Drehen. Die einzelnen Teile des Gerätes wurden mit Technik bestückt, zusammengesteckt und zum besseren Halt miteinander verklebt. Das Gerät ist trotz seiner Größe leicht und liegt gut in der Hand.


Vorderansicht Eingabegerät


Vorderansicht Eingabegerät

3.2 Anwendung

Entertainment

Rendering Pusteblume

Nach den entsprechenden Vorüberlegungen (siehe *Projektverlauf*) sollte unsere erste Anwendung eine Pusteblume werden, bei der es möglich sein sollte, die Samen wegfliegen zu lassen. Dies geschieht dadurch, dass der Benutzer in das Eingabegerät pustet und dieser Input dann auf den Wind in der Virtuellen Realität (VR) gemappt wird. Durch das direkte Mapping entsteht eine neue Interaktionssituation, die den meisten Benutzern noch fremd ist und überraschende Reaktion hervorruft, wie auch die Erfahrungen vom Rundgang bestätigen. Um den Spielspaß zu verstärken haben wir einen Score und einen Highscore in die Anwendung mit eingebunden, um ihr einen Wettkampfcharakter zu verleihen. Somit kann der Benutzer seine Pustekraft austesten und mit anderen um den Highscore konkurrieren.

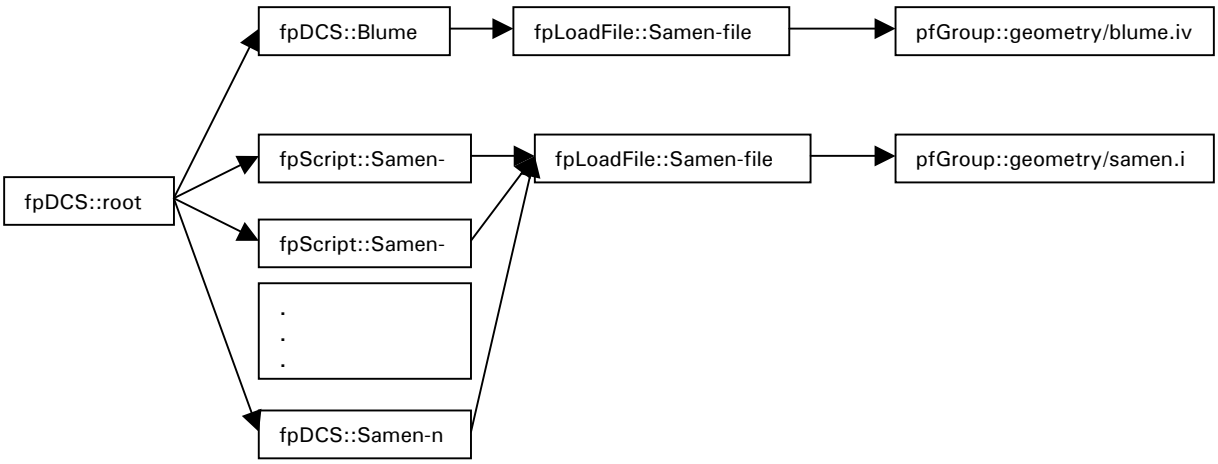
3.2.1 Szene

Rendering Spielszene

Die Anwendung besteht aus zwei Szenen. In der ersten ist eine Pusteblume mit vielen Samen auf ihrem Blütenstempel zu sehen. Sie steht auf einer Wiese und im Hintergrund fordert eine Schrift dazu auf, sie zu pflücken. Allerdings ist das Pflücken auf das Eingabegerät bezogen, da man es erst in die Hand nehmen muss, um mit der Anwendung zu interagieren. Wir entschieden uns in der Planungsphase, dieses reale Pflücken nicht in der VR zu übernehmen, die Pusteblume wird weiterhin als auf der Wiese stehend dargestellt. Hat der Anwender das Eingabegerät nun aufgenommen, verschwindet die Schrift aus dem Hintergrund und er kann hineinpusten. Dies ist die zweite interaktive Szene der Anwendung. Die eingehenden Daten über Dauer und Stärke des Pustens werden nun in Wind in der VR umgesetzt (siehe *Technik*). Die Samen beginnen sich im Wind zu bewegen, aber sitzen noch auf der Blüte. Erst wenn der erzeugte Wind stark genug ist, lösen sie sich vom Blütenstempel und fliegen nach hinten aus der Szenerie, wobei sie natürlich auch aufgrund ihrer „Schwerkraft“ zu Boden sinken. Der Anwender hat die Möglichkeit sooft wie er möchte hineinzupusten, falls er es nicht geschafft hat, alle Samen beim ersten mal wegfliegen zu lassen. Allerdings verringert sich mit jedem Versuch die Punktezahl die man erreichen kann. Beendet wird das Ganze, in dem der Benutzer das Eingabegerät wieder hinstellt. Ein in das Eingabegerät eingelassener Button sorgt dann dafür, dass die Anfangsszene wieder zu sehen ist, also alle Samen wieder auf der Blume sitzen und die Schrift im Hintergrund dazu auffordert, die Blume zu pflücken.

Die Modelle der Pusteblume, samt Samen und Wiese sind im 3D-Modelierungsprogramm 3D-Studio Max entstanden. Die dort entstanden vrml-Dateien wurden mit Deep Exploration in das iv-Format konvertiert um sie unter Avango einbinden zu können. Die Szene besteht aus zwei iv-datein, zum einen die Pusteblume und Wiese, aber ohne Samen und zum anderen aus einer iv-Datei eines Samens. Da die Samen autark reagieren sollten, war es nötig, eine eigene iv-Datei zu haben. Die Pusteblume wurde so modelliert, dass der Pivotpunkt des Samens genau im Mittelpunkt der Kugel, welche die Blüte darstellt, liegt. Dies erleichterte die Anordnung der Samen auf der Blüte, da sie dadurch einfach automatisiert werden konnten. Um den Samen ein autarkes Verhalten zu geben, wurde der fp-Script Knoten eingestetzt. Jeder Samen auf der Blüte (standartmäßig ca. 128) sind in Avango durch solch einen Knoten repräsentiert, wobei alle Knoten auf ein und die selbe iv-Datei verweisen (siehe Grafik *Szenengraph*).

Szenengraph



Alle fpScript-Knoten verweisen auf die gleiche iv-Datei. Der Index n steht für die Anzahl der Samen.

3. 2. 2 Technik Eingabegerät

Das Pusten kann über verschiedene Möglichkeiten sensorisch aufgenommen werden:

Mikrofon:

Der Anwender pustet gegen ein Mikrofon und erzeugt dadurch eine sehr hohe Ausgangsspannung, besonders bei den höheren Frequenzen. Die dabei entstehende Spannung wird über den A/D-Wandler weiterverarbeitet. In der Praxis könnte man eine Elektret-Kondensator-Kapsel verwenden. Diese sind für wenige Euro im Elektronik-Fachhandel erhältlich. Empfehlenswert wäre eine Verstärkung über einen Hochpass und eine mechanische Empfindlichkeitssteuerung über einen Potentiometer.

Temperatur:

Ein Temperatursensor wird mit einem Heizdraht umwickelt, welcher durch Strom erwärmt wird. Pustet nun der Anwender gegen den Temperatursensor, kühlt sich dieser ab. Temperatursensoren erzeugen in Abhängigkeit von der Temperatur einen Widerstand, bei Abkühlung wird dieser geringer. Handelsübliche Sensoren haben einen Temperaturbereich von -55 – 150°C, den sie auf unterschiedliche Widerstandsgrößen abbilden.

Beispiel:
Philips KTY-81-1/KTY-81-2 series:
angenommener relevanter Temperaturbereich 25° – 80°C

	ca. Widerstand R bei		ΔR bei
	T1 = 25°C	T2 = 80°C	ΔT = T2 – T1 = 55°C
KTY-81-1	1 kΩ	1,5 kΩ	0,5 kΩ
KTY-81-2	2 kΩ	3 kΩ	1 kΩ

Die vorbeiströmende Luftmenge/Zeit bestimmt die Abkühlung. Daher ist bei dieser Technik mit einer verzögerten Reaktion zu rechnen.

Drehzahlmessung

Der Anwender pustet gegen ein Windrad, das sich dementsprechend dreht. Die Drehung des Windrads wird sensorisch gemessen.

Drehzahlmessung: Optisches Verfahren

An der Achse des Windrads ist eine Kodierscheibe angebracht, die sich mit dem Windrad mitdreht und so einem optischen Empfänger die Drehzahl vermittelt.

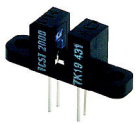
Dies geht zum Beispiel mittels eines Optoreflexkopplers (z.B. CNY 70), einem elektronischen Bauelement, das Sende-Diode und Foto-Transistor-Empfänger in einem Baustein vereint. Der Foto-Transistor misst die von der Kodierscheibe abgestrahlte Reflexion des von der Sende-Diode ausgestrahlten Lichts. Da auf der Kodierscheibe bestimmte Abschnitte besser reflektieren als andere, kommt es beim Drehen der Scheibe zu einem Wechsel der Spannungspotentiale.

CNY70 Optokoppler

Die Messbarkeit dieser Potentialdifferenzen hängt von folgenden Faktoren ab:
Möglichst geringer und konstanter Abstand Optokoppler <-> Kodierscheibe. Je weiter der Abstand, desto geringer der Signalunterschied. (< 3mm, Optimal 0.3 bis 1 mm)

- Frequenz der Wechsel Reflexion <-> keine Reflexion ist wiederum abhängig von:
- Breite der Streifen auf der Kodierscheibe
- max. Rotationsgeschwindigkeit
- Reflexionsmaterial, abwechselnd möglichst stark reflektierendes und möglichst wenig reflektierenden Materials (Lichtwellenlänge der Fotodiode beachten)
- Lichtverunreinigung durch Aussenlicht
- Schaltung (Widerstände, Transistor, Signalverstärkung?) zwischen CNY 70 und USB-A/D-Wandler

Eine Zweite Möglichkeit besteht in der Verwendung einer Lichtschranke (z.B. CNY 37). Hier besteht die Kodierscheibe aus sich abwechselnden lichtdurchlassenden und –undurchlässigen Abschnitten. Ansonsten gleicht das Prinzip dem vorab beschriebenen.



CNY37 Lichtschranke

Der Vorteil hier ist, dass dieses Verfahren in vielen Computer-Mäusen eingesetzt wird und man hier die Möglichkeit hat, kostengünstig und einfach an meist fertig einsetzbare Technik zu gelangen.

Beim Eigenbau ist ansonsten zu bedenken, dass eine Kodierscheibe für Lichtschranken schwieriger selbst herzustellen ist als eine Reflexionsscheibe.

Drehzahlmessung: Induktionsprinzip

Eine weitere Möglichkeit der Drehzahlmessung ist die über elektromagnetische Induktion. Ein zeitlich veränderliches Magnetfeld induziert in einen Leiter einen Induktionsstrom. Ebenso wird in eine Spule Strom induziert, die in einem konstanten Magnetfeld rotiert wird. Dieser Strom wird von einer Induktionsspannung begleitet. Es erfolgt also eine Umwandlung von mechanischer in elektrische Energie (Generatorprinzip).

Bei einigen CPU- und Gehäuselüftern wird über dieses Prinzip Spannung erzeugt, wenn man die Lüfter manuell andreht. Dies kann man auch durch Anpusten erreichen. Die dadurch erzeugte Spannung ist zu messen und demnach unverstärkt oder verstärkt in den A/D-Wandler weiterzuleiten.

Die Pusteblume:

Bei der Pusteblume haben wir uns für das Induktionsprinzip entschieden, da wir zur Zwischenpräsentation ein funktionierendes Gerät fertig haben wollten und hier keine eigene Schaltung konstruiert werden musste, bzw. um Ströme zu verstärken o.ä.

Die Technik in der Pusteblume besteht also aus einem handelsüblichen CPU-Lüfter der Firma Titan, einem USB-Analog/Digital-Wandler und einem digitalem Knopf.

Die Masse des CPU-Lüfters wird an der Masse des A/D-Wandlers angeschlossen, die 5V-Leitung geht in den Analog0 des A/D-Wandlers. Der Lüfter hat zwar auch ein Kabel zur Drehzahlmessung. Die hier induzierte Spannung fällt jedoch wesentlich geringer aus als in der 5V-Leitung.

Im Fuß der Blume befindet sich ein Button, der genau so verbaut ist, dass er durch das Eigengewicht der Blume ausgelöst wird und gedrückt bleibt, wenn die Blume abgestellt ist. Er wird am Digital0 des A/D-Wandlers angeschlossen.

Bearbeitung der Werte

Je nachdem welches Prinzip mit welcher Technik verwendet wird, erhält der Rechner bestimmte Werte, die wir verwenden wollen. Wichtig ist dabei zu untersuchen:

- die Fehlerbehaftung und Stetigkeit
  Es ist wichtig, dass bei gleichwertigen Manipulationen auch gleichwertige Werte erzeugt werden. Da dies nicht immer der Fall ist, sollte eine Wertmittelung vorgenommen werden.
- denWertebereich
  Zur besseren Weiterverarbeitung sollten die Daten auf einen Bereich 0..1 abgebildet werden
- die statistischen Häufigkeiten bestimmter Werte für die Anwendung gleichwertige Manipulationen erzeugen hardwareseitig unterschiedliche Daten, hier kann über einen nichtlineare Normalisierungsfunktion eine Angleichung vorgenommen werden, während man in anderen Wertenbereichen die Unterschiedsschwellen verringert.

Daraus ergeben sich folgende Schritte:

1. Wertmittelung:
   Aus einer Anzahl von Werten wird ein Mittelwert erzeugt (z.B. Minimum, Maximum, quad. Mittel, Durchschnitt...).
2. Normalisierung:
   Der durch die Hardware erzeugte Wertebereich wird auf Werte zwischen 0 und 1 abgebildet und zwar über eine

3. Normalisierungsfunktion:
   Die Abbildung linear oder nicht-linear (z.B. Quadratische Funktion oder Polynom) um bestimmte Werte zu verstärken und andere Bereiche zu desensibilisieren.

3.2.3 Anwendung Technik

Das Programm ist im Grunde strukturiert in

1) Die Eingabe und die Verarbeitung der Werte
2) Das visuelle Feedback der Pusteblume und die Reaktion der einzelnen Samen

Eingabe

Pustet der Spieler gegen das Eingabegerät, dreht sich der CPU-Lüfter und die dadurch induzierte Spannung wird im A/D-Wandler in digitale Werte umgewandelt, die über die USB-Schnittstelle an die Software vermittelt werden.

Der A/D-Wandler tastet analoge Signale mit einer Auflösung von 10 bit ab, er liefert also theoretisch Werte zwischen 0 und 2^{10}-1 = 0 bis 1023.

Die durch Anpusten induzierte Spannung wird direkt in den A/D-Wandler übertragen und nicht über eine Transistorschaltung auf eine maximale Amplitude von 5V geeicht. So liegt die gemessene Induktionsspannung zwischen 0,8 und ca. 3 V. Analog dazu erreicht man im Selbstversuch digitale Werte zwischen mindestens 160 und maximal 660.

Wie oben systematisiert, werden die Werte weiterverarbeitet. Die geschieht global in einer Callback-Funktion, die immer dann aufgerufen wird, wenn der A/D-Wandler einen Wert liefert.

Zuerst werden Unregelmäßigkeiten und Fehler ausgeglichen. Man kann beobachten, dass auch bei hohen Drehzahlen und in daraus folgenden hochwertigen Zahlenreihen immer wieder kleine Werte zwischen 180 und 200 erzeugt werden. Das gleichen wir aus mit einer einfachen Durchschnittsbildung aus 5 Werten.

```
(define (dreh-cb trigger)

  (if (< wertzaehler 5)
  ;;then
    (begin
      (set! DrehInput (+ DrehInput (fp-get-value dreh-sensor 'Value0)))
      (set! wertzaehler (+ wertzaehler 1))
     ) ;;end then
  ;;else
    (begin
      ;;nun wird die Summe durch 5 geteilt um den durchschnitt zu ermitteln
      (set! DrehInput (/ DrehInput 5))
      ;; der input wird _normalisiert_ auf werte zwischen 0 und 1 und in die
      ;; globale variable drehspeed geschrieben 0.000004*(DrehInput-160)^2)
      ;; der zaehler wird zurückgesetzt
     (set! DrehSpeed (* 0.000004 (* (- DrehInput 160)(- DrehInput 160))))
     (set! DrehInput 0.0)
     (set! wertzaehler 0)
    ) ;;end else
  ) ;;end if
) ;;end define
```

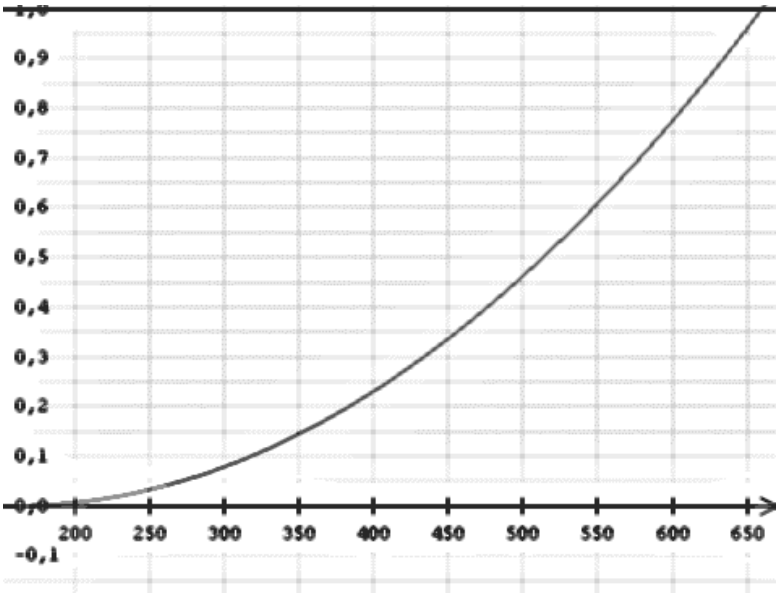
Wurde der Durchschnitt gebildet, wird er normalisiert. Hierbei ist wichtig zu beachten, dass durch technische Eigenschaften am Eingabegerät bedingt ähnliche Manipulationen unterschiedliche Werte liefern oder umgekehrt verschieden Eingabestärken ähnliche Werte liefern.

Letzteres ist bei dem hier verwendeten CPU-Lüfter der Fall: Im unteren Drehzahlbereich sind die Zahlen kaum nachvollziehbar und nehmen Werte zwischen 160 und 300 ein. Dies geschieht auch wenn der Lüfter ausbremst. Erst bei sehr hohen Drehzahlen gibt es konstant messbare Unterschiede in den Werten, die deutlich gemacht werden sollen.

(Man muss hier anmerken, dass unsere Messmethode der Drehzahlen lediglich das subjektive Empfinden beim Pusten und vor allem Hören der entstehenden Geräusche war. Dies wäre wissenschaftlich nachzuprüfen.)

Wegen dieser Eigenschaften der Wertestatistik werden die Eingangswerte über eine nicht-lineare Funktion normalisiert, also auf das Intervall [0;1] abgebildet, so dass erst Werte ab ca. 300 langsam programmtechnisch relevant werden.

$$f(x) = DrehSpeed(DrehInput) = 0.000004(DrehInput - 160)^2$$



Man errechnet so die globale Variable Drehspeed, die je nach Input Werte zwischen 0 und 1 annimmt.

Visuelles Feedback und Reaktion der Samen

Wie oben beschrieben, reagieren die Samen autark. Ermöglicht wird dies durch Verwendung von fpScript-Knoten. Jeder Samen ist ein fpScript-Knoten. fpScript-Knoten sind DCS-Knoten, denen man eigene Custom-Felder geben kann. Diesen Feldern lassen sich Funktionen in Form von Methods zuordnen, die bei Änderung des Feldinhalts – ähnlich einer Callback-Funktion eines Triggers – aufgerufen werden.

Die einzelnen Samen-fpScript-Knoten gleichen sich in ihren Feldern und Funktionen, haben jedoch auf einigen Feldern unterschiedliche Werte und dadurch individuelle Eigenschaften. Sie verweisen alle auf die gleiche Geometrie, so dass diese nur einmal geladen werden muss.

Beispiel:

```
(define samen-dcs (make-instance-by-name "fpScript"))
```

Die Custom-Felder werden deklariert. Sie sind hier strukturiert nach

```
(fp-set-value samen-dcs 'Fields
(list
```

- den spezifischen Eigenschaften für die Animation:
  ```
  "fpSFDouble Wahrscheinlichkeit"
  "fpSFDouble RotWinkel"
  "fpSFDouble PhasenWinkel"
  "fpSFDouble Gravity"
  ```
- den Feldern für die Initialisierungswerte um das Zurücksetzen zu ermöglichen:
  ```
  "fpSFDouble TimeStart"
  "fpSFDouble DrehSpeedStart"
  "fpSFVec3 Position"
  "fpSFVec3 PositionStart"
  "fpSFMatrix InitMatrix"
  ```
- Status-Feldern, die anzeigen ob sich ein Samen im Flug befindet oder schon ausgeblendet wurde:
  ```
  "fpSFBool Active"
  "fpSFBool Moving"
  ```
- und den Feldern für die Eingabe über Zeit und Benutzermanipulation, an welche später Methoden gekoppelt werden:
  ```
  "fpSFDouble TimeIn"
  "fpSFFloat InputIn"
  "fpSFBool ResetIn"
  ```

```
))
```

Die spezifischen Eigenschaften eines jeden Samens werden festgelegt:

```
(fp-set-value samen-dcs  'Name (string-append "samen-"(number->string index)))
(fp-set-value samen-dcs 'Matrix (mult-mat (make-rot-mat -90 1 0 0) (make-rot-mat x 0 1
0) (make-rot-mat z 0 0 1)
(make-trans-mat 0 0 64.98)( make-scale-mat 0.008 0.008 0.008)))
(fp-add-1value samen-dcs 'Children samen-file)


(fp-set-value samen-dcs 'Wahrscheinlichkeit (+ 10 (modulo (random) 70)))
(fp-set-value samen-dcs 'RotWinkel (modulo (random) 10))
(fp-set-value samen-dcs 'PhasenWinkel (/ (modulo (random) 63) 10))
(fp-set-value samen-dcs 'Gravity (/ (+ 4 (modulo (random) 10)) 10000))

(fp-set-value samen-dcs 'TimeStart 0)

(fp-set-value samen-dcs 'Position (make-vec3 (mat-ref (fp-get-value samen-dcs 'Matrix)
2 0)(mat-ref(fp-get-value samen-dcs 'Matrix) 2 1 )(mat-ref (fp-get-value samen-dcs
'Matrix) 2 2)))
(fp-set-value samen-dcs 'PositionStart (fp-get-value samen-dcs 'Position))
(fp-set-value samen-dcs 'InitMatrix (fp-get-value samen-dcs 'Matrix))

(fp-set-value samen-dcs 'Moving 0)
(fp-set-value samen-dcs 'Active 1)

(fp-connect-from samen-dcs 'TimeIn time-sensor 'Time)
(fp-connect-from samen-dcs 'InputIn dreh-sensor 'Value0)
(fp-connect-from samen-dcs 'ResetIn dreh-sensor 'Contact0)
```

An die letzten drei Felder warden Methoden angehängt, also Funktionen die aufgerufen werden, wenn sich der Wert des Feldes ändert.

```
(fp-set-value samen-dcs 'Methods
(list

(cons 'TimeIn
 (lambda (this field value local)
       (if (= local 0)
         (begin
```

Hier erfolgt die Animation des jeweiligen Samens. Ist die bool moving gesetzt, verändert der Samen seine Position und Rotation. Die neue Position ergibt sich aus der Berechnung von Mathematischen Funktionen für die x-, y- und z-Koordinate, was dem Betrachter ein möglichst realistisches Flugverhalten vermitteln soll.

Da diese Funktion mit dem Feld 'TimeIn verknüpft ist, wird sie in jeder Frame aufgerufen.

```
       ))
))

(cons 'InputIn
 (lambda (this field value local)
       (if (= local 0)
        (begin
```

Diese Funktion wird immer dann aufgerufen, wenn ein Input über den CPU-Lüfter erfolgt.

Zunächst wird der Samen animiert, er wackelt etwas hin und her, damit auch visuelles Feedback geboten wird, falls noch kein Samen wegfliegt.

Außerdem wird für den jeweiligen Samen geprüft, ob eine Zufallszahl unter einer bestimmten Schranke liegt. Je größer die spezifische 'Wahrscheinlichkeit des Samens und je stärker der global berechnete Input Drehspeed, desto höher die Schranke und desto wahrscheinlicher ist es, dass der Samen losfliegt.

Ist dies der Fall, erhält das Feld 'Moving den Wert 1 und zur Gesamtpunktzahl werden Punkte in Abhängigkeit von der Wegfliegwahrscheinlichkeit und dem aktuellen Drehspeed addiert.

Fliegt der Samen noch nicht los, wird seine Wegfliegwahrscheinlichkeit erhöht.

```
       ))
))

(cons 'ResetIn
 (lambda (this field value local)
       (if (= local 0)
        (begin
```

Diese Funktion wird aufgerufen, wenn die Blume wieder abgestellt wird und somit der Reset-Button ausgelöst wird. Hier werden alle Felder des Samens auf die beim Programm-Start initialisierten Werte zurückgesetzt. Da jeder Samen das für sich selbst übernimmt, kann man sich eine globale Reset-Funktion sparen.

```
       ))
))

)) ; end list/methods
```

4. Windmühle

4.1 Konzept Eingabegerät

Eine Analogie zur Anwendung sollte auch bei dem zweiten Gerät geschaffen werden. Das Windrad hat eine sehr hohe Mappingfunktion, weil es jeder als Spielzeug kennt und es mit Pusten in Verbindung bringt. Ausserdem dreht es sich auf Grund seiner Konstruktion sehr leicht. Ideen zur Umsetzung und Schnittmuster des Windrades wurden mit Hilfe von Kinderbastelbüchern recherchiert.


Recherche Windrad

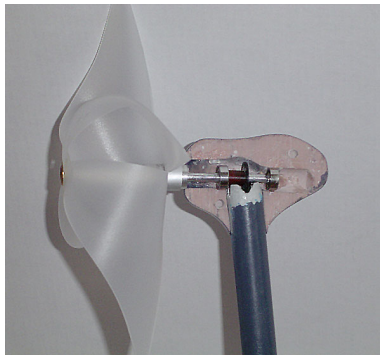

Recherche Ventilator

4.1.1 Umsetzung

Die Technik des Eingabegerätes ist aus einer funkgesteuerten Maus entnommen. Durch das Entfernen der Hülle konnte Volumen eingespart werden. Die meisten Teile der Maus sind in den Standfuß integriert. Daraus ergibt sich die Größe des Fußes. Die Taster und das Scrollrad sind von den beiden Platinen abgelötet und mit Kabeln verbunden worden. So konnten sie an der Oberfläche des Fußes angeordnet werden. Ein optisches Sensorpärchen, welches die y-Achse des Mauszeigers abgreift, wurde ebenfalls abgelötet und mit Kabel verbunden. Dadurch ist es möglich, die Sensorik durch das Griffrohr des Gerätes zu führen. Im Kopf wird die Achse des Windrades gelagert. Das Windrad und die Achse sind miteinander verbunden, damit die Achse vom Windrad angetrieben wird. Kugellager sorgen für ein gleichmäßiges und sehr leichtes Rotieren. Die Codierscheibe, welche die Werte der y-Achse der Maus abgreift, ist auch auf der Achse befestigt und dreht sich zwischen den Sensoren. So werden Werte erzeugt und an den Rechner weitergegeben.

Der Standfuß ist aus Polyurethanschaum gearbeitet, der Griff aus PVC-Rohr (Durchmesser 16mm) und das Windrad aus PP-Folie. Die Achse besteht aus einem vier Millimeter starken Aluminiumstab.


Bearbeitung des Standfußes


Details

Vorderansicht Seitenansicht von rechts

4.2 Anwendung

Edutainment

Nachdem wir uns neu orientiert hatten, sollte nun ein zweites Szenario entstehen, bei dem das Pusten als Input erhalten bleibt, jedoch die Charakteristik der Anwendung eine andere ist. Die reine Spielstruktur sollte einer lehrreichen Unterhaltungsanwendung weichen. Hierbei konnten wir die Vorteile der VR voll und ganz ausnutzen, da wir nicht an die Einschränkungen der realen Welt gebunden waren. Wind spielt bei einigen Sachverhalten mit technischem Zusammenhang eine Rolle z.B. aus Windkraft Strom erzeugen, Blowmoldingverfahren bzw. Glasblasverfahren oder der Wellenerzeugung. Wir entschieden uns dafür, die Abläufe und Zusammenhänge beim Mahlen von Getreide in einer Windmühle zu visualisieren und zu erklären. Diese Anwendung ist auf Kinder im Alter von 5-10 Jahren zugeschnitten. Wir haben uns von Sendungen wie *Sendung mit der Maus, Es war einmal das Leben* oder *Löwenzahn* inspirieren lassen. Zur Recherche haben wir uns Kinder- und Erzählbücher angesehen, welche sich damit beschäftigen, Kindern in kindgerechter und ansprechender Form Wissen zu vermitteln.

Intro der Anwendung, Würfel mit Bildern von Windmühlen

4.2.1 Szene

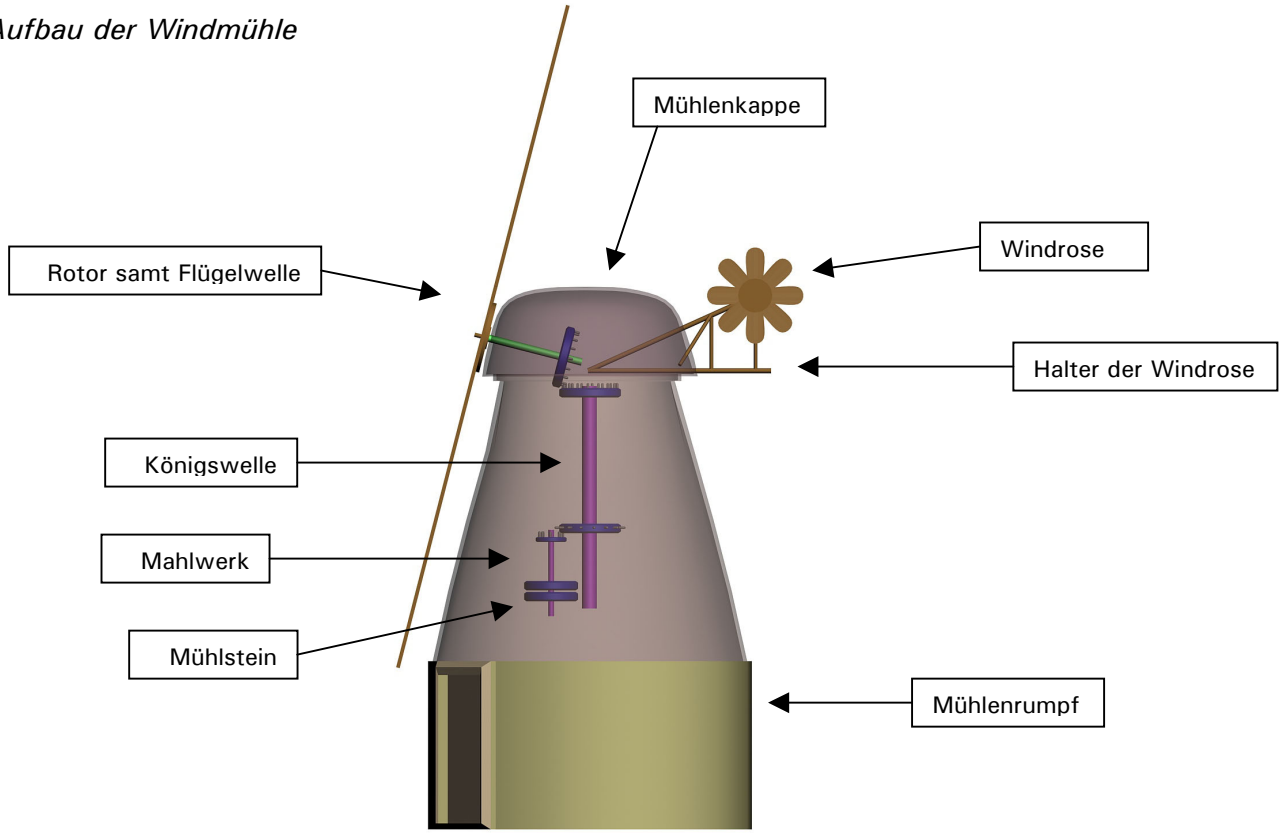
Um die Anwendung kindgerecht zu halten, haben wir uns eine kleine Geschichte ausgedacht, um so wichtige Zusatzinformationen zu vermitteln und den Kindern etwas Unterhaltung zu bieten. Nach dem gleichen Prinzip wie es die oben erwähnten Fernsehsendungen tun. Dies führte dazu, dass die Anwendung aus einer sequentiellen Abfolge von audiovisuellen Szenen besteht, durch die sich der Benutzer klickt. Am unteren linken Bildschirmrand sind bei jeder einzelnen Szene die möglichen Aktionen angezeigt, so dass der Benutzer immer sehen kann welche Optionen ihm gerade zur Verfügung stehen. Dies sind die Möglichkeiten *weiter* oder *beenden*; in zwei Szenen erfolgt der Input des Pustens bzw. das Ausrichten der Windrichtung mit Hilfe eines Scrollrades. Dieses etwas unausgewogene Verhältnis ist Folge des Konzeptes, eine Geschichte um die rein technischen Zusammenhänge zu erzählen, damit der Unterhaltungswert der Anwendung nicht zu kurz kommt.

Rendering Windmühle

Die ganze Anwendung kann man in drei Teile unterteilen. Der erste Teil dient dazu, den Kindern zu erklären, woher eigentlich das Korn kommt das in der Windmühle zermahlen wird. Dies geschieht durch einen Würfel auf dessen Seitenflächen verschiedene Bilder von Getreidefeldern, Bauern bei der Ernte und Anfahrt des Getreides zur Windmühle gemappt werden. Die entsprechenden Informationstexte und Audiosamples werden simultan eingeblendet. Der Benutzer hat dann die Möglichkeit immer eine Szene weiter zu klicken. Geschieht dies, dreht sich der Würfel um 90 Grad und die nächste Seitenfläche mit dem nächsten Bild kommt zum Vorschein. Die Informationstexte und Audiosamples werden dem neuen Bild angepasst. Sind wir nun in unsere Geschichte soweit vorgedrungen, dass das Korn die Windmühle erreicht hat, wechseln wir zum zweiten Teil der Anwendung. Der Würfel verschwindet und an dessen Stelle ist nun ein 3D-Modell einer Windmühle zu sehen. Die Schaltflächen *weiter* und *beenden* bleiben erhalten. Die Anwendung wurde wiederum als vrml-Datei in 3D-Studio Max modelliert und anschließend mit Deep Exploration ins iv-Format überführt, zur Einbindung in Avango. Auch hier mussten wieder Teile der Windmühle als einzelne iv-Dateien produziert werden, um sie später in Avango einzeln rotieren zu lassen. Die Windmühle besteht insgesamt aus acht Bestandteilen (siehe Grafik *Aufbau der Windmühle*), wobei zwei Bestandteile, nämlich die Kappe und der Mühlenrumpf in zweifacher Ausführung vorliegen. Hierbei existiert von beiden Bestandteilen jeweils noch eine iv-Datei mit einer Transparenz von 0.8, um so an angemessener Stelle innerhalb der Anwendung einen Blick ins Innere der Windmühle zu ermöglichen. Anhand dieses Modells wird die prinzipielle Bauart mit den dahinter stehenden Gedanken erklärt und veranschaulicht. Mittels Kamerafahrten um das Modell werden fortlaufend die Bestandteile der Windmühle fokussiert, deren Funktionsweise und Zweck gerade durch die Texte erklärt wird. Diese Kamerafahrten wurden mit dem fp-Interpolator Knoten realisiert (siehe *Technik*). Schließlich kommt man an einen Punkt in der Geschichte an dem man zum ersten Mal mit dem Windmühlenmodell interagiert. Es gilt die Windrichtung ein-

zustellen, dies geschieht mit Hilfe des Scrollrades. Dann muss der Benutzer den Wind erzeugen, durch Hineinpusten in das Eingabegerät. Die Kappe der Windmühle wird sich sobald „Wind weht“ in Windrichtung ausrichten, dies geschieht, da zuvor die Windrose vom Wind erfasst wurde und durch ihre Rotation die Kappe in Bewegung versetzt. Ist nun die Mühle nach dem Wind ausgerichtet, wird ein Blick in das Innere der Mühle geworfen und auch dort werden wieder alle Bestandteile bezüglich ihrer Funktion und ihres Zwecks erklärt. Dann ist der Benutzer wieder an der Reihe der Windmühle erneut Wind zuzuführen. So wird die Drehung des Windrades über Achse und Zahnräder auf die Mühlsteine übertragen, um damit das Korn zu mahlen. Damit ist der zweite Teil der Anwendung beendet. Im dritten Teil, indem wieder ein Bilderwürfel in Erscheinung tritt, wird die Geschichte zu Ende erzählt. Ist die letzte Szene vorüber, startet die Anwendung neu. Der Würfel durchläuft einen Loop und zeigt Bilder von Windmühlen, bis jemand anderes durch Drücken eines Buttons die Anwendung startet. Das Szenario selbst wird auch geloopt, da wir davon ausgegangen sind, dass es möglicherweise in einem Science Center, einem Museum oder etwas ähnlichem ausgestellt sein könnte. Also ein Ort an dem potenzielle Benutzer nacheinander in Erscheinung tretenund die Anwendung ausprobieren.

Aufbau der Windmühle



Die aufgeführten Einzelteile liegen in der Anwendung alle als einzelne iv-Dateien vor. Dies ist eine Notwendigkeit, da wir einzelne Teile wie z.B. die Windrose rotieren lassen müssen. Würde die Windmühle nur aus einer iv-Datei bestehen, wäre dies nicht möglich.

4.2.2 Technik Eingabegerät

Hier sollten ein Optoreflexkoppler und eine reflektierende Codierscheibe zum Einsatz kommen. Es bietet sich jedoch eine simple und komfortable Lösung an: Wir nehmen eine handelsübliche Funkmaus von Logitech (http://www.logitech.com/index.cfm?page=products/details&CRID=3&CONTENTID=4983&countryid=7&languageid=4) und verbauen deren Technik. Dies ist keine optische Maus, sondern eine mechanische Kugel überträgt das Rollen auf dem Tisch auf zwei kleine Rädchen. Die Drehung dieser Rädchen wird nach dem Lichtschranken-Prinzip übertragen. Wir haben also diese Lichtschranken-Sensoren (LED und Foto-Transistor) von der Maus-Platine entfernt und über Kabel nach außen verlegt. So können wir die eigentliche Maus im Fuß der Windmühle unterbringen und die Drehzahlsensorik im Kopf des Geräts. So wird die Licht durchlässige Codierscheibe mit der Achse des Windrads verbunden. Dreht sich also das Windrad, so dreht sich die Codierscheibe durch die Lichtschranke. Das gleicht in unserem Gerät einer vertikalen Bewegung der Maus auf dem Tisch.
Analog zur Lichtschranke wurden auch zwei Buttons und das Mausrad nach außen gelegt und neu mit der Mausplatine verlötet.
Insgesamt stehen auf der Maus folgende Eingabemöglichkeiten zur Verfügung:

Maus	Werte	Windmühle
Linker Mausknopf	0,1	Gelber Button
Mittlerer Mausknopf	0,1	Roter Button
Rechter Mausknopf	0,1	-
Bewegen in X-Richtung	-128..0..127	-
Bewegen in Y-Richtung	-128..0..127	Windrad
Mausrad	-128..0..127	Windrichtung

4.2.3 Technik Anwendung

Der Programmablauf ist strukturiert in hauptsächlich drei Szenen:

1) Intro
   Der allein rotierende Würfel mit vier verschiedenen Mühlenbildern mit dem Start des Programms als einzige Interaktionsmöglichkeit
2) Informationswürfel
   Die jeweils ersten und letzten drei Takes, in denen der Würfel zu sehen ist und die zum Betrachter stehende Seite ein Bild zeigt. Dazu wird ein Text gesprochen und abgebildet. Durch Drücken des gelben Knopfes kann der Benutzer zum nächsten Take weitergehen.
3) Windmühle
   Das 3D-Modell einer Windmühle wird gezeigt. Die Kamera fährt in jedem Take bestimmte Bahnen ab um Teile der Mühle detailliert zu zeigen. In dieser dritten Szene erhält der Benutzer zweimal die Möglichkeit durch Pusten zu interagieren.

Hier der Ablauf der einzelnen Takes:

Szene	Take	Was passiert	Gesprochener Text	Überschrift	Aktionen
1	1	Würfel dreht sich, 4 verschiedene Bilder von Mühlen sind zu sehen	-	Die Virtuelle Windmühle	Gelb: Start der Präsentation
2	2	Kornfeld ist zu sehen	Das ist ein Kornfeld. Hier wird Getreide angebaut.	Das Kornfeld	Gelb: Weiter Rot: Neustart
	3	Getreidernte	Zur Erntezeit mäht der Bauer das Feld ab.	Die Getreideernte	Gelb: Weiter Rot: Neustart
	4	Bauer bringt Korn	Der Bauer bringt das Getreide zur Mühle. Hier wird es zu Mehl gemahlen. Das schafft der Müller nicht alleine. Der Wind wird ihm dabei helfen!	Die Windmühle	Gelb: Weiter Rot: Neustart
3	5	3D-Szene1: Totale	Das ist unsere Windmühle. Mal schauen wie sie funktioniert.	Wie funktioniert die Windmühle?	Gelb: Weiter Rot: Neustart
	6	3D-Szene1: Centering auf Windrad	Der Wind kann die großen Flügel des Windrades antreiben. Doch dazu muss das Windrad in Windrichtung stehen.	Das Windrad muss in Windrichtung stehen	Gelb: Weiter Rot: Neustart
	7	3D-Szene1: Centering auf Windrose	Unsere Mühle hat eine Windrose. Die Windrose dreht die Mühlenhaube mit dem Windrad immer in Windrichtung.	Das geschieht über die Windrose	Gelb: Weiter Rot: Neustart

	8	3D-Szene1: Totale, Interaktion	Mit dem Rädchen kannst du einstellen aus welcher Richtung der Wind weht. Puste dann kräftig hinein und schau was passiert.	Jetzt bist du dran: Du machst den Wind!	Rädchen: Windrichtung Pusten: Wind erzeugen Gelb: Weiter Rot: Neustart
	9	3D-Szene2: Totale	Wenn der Wind weht, dreht sich das Windrad. Doch was passiert dabei in der Mühle?	Die Mühle von Innen	Gelb: Weiter Rot: Neustart
	10	3D-Szene2: Zoom auf die Achsen und Kamerafahrt bis zu den Mühlsteinen	Das Drehen des Windrads wird über Zahnräder auf die Mühlsteine übertragen.	Die Zahnräder bewegen die Mühlsteine	Gelb: Weiter Rot: Neustart
	11	3D-Szene2: Zoom auf die Mühlsteine	Die rauen Mühlsteine reiben aufeinander. Sie mahlen so das Getreide zu Mehl	Die Mühlsteine mahlen das Mehl	Gelb: Weiter Rot: Neustart
	12	3D-Szene2: Ansicht von hinten, Interaktion	Puste hinein und schau was passiert	Jetzt bist du dran: Du machst den Wind!	Pusten: Wind erzeugen Gelb: Weiter Rot: Neustart
2	13	Mehl rieselt in die Säcke	Ein Stockwerk tiefer fällt das gemahlene Mehl in Mehlsäcke.	Bei den Mehlsäcken	Gelb: Weiter Rot: Neustart
	14	Bäckerbild	Die Mehlsäcke werden zum Bäcker gebracht. Damit backt der Bäcker Brot und Kuchen.	Beim Bäcker	Gelb: Weiter Rot: Neustart
	15	Kinder mit Brot	MMHM LECKER!!	Beim Bäcker	Gelb: Weiter Rot: Neustart

Im Scheme-Code wurde dies dadurch realisiert, dass in einem Time-Sensor-Callback über eine case-Anweisung festgestellt wird, welche Szene (current_scene) und welcher Take (current_take) aktuell sind und die entsprechenden Befehle ausgeführt werden. (z.B. Sichtbarmachen von bestimmten Geometrien, Setzen der Keyframes für die Kamerafahrten, setzen der Variable listen-to-input, die Interaktion verhindert, wenn gerade ein Szenenwechsel vollzogen wird, etc.)
Dies geschieht auch im Input-Sensor-Callback, wenn der Benutzer die entsprechende Taste zum Szenenwechsel betätigt.

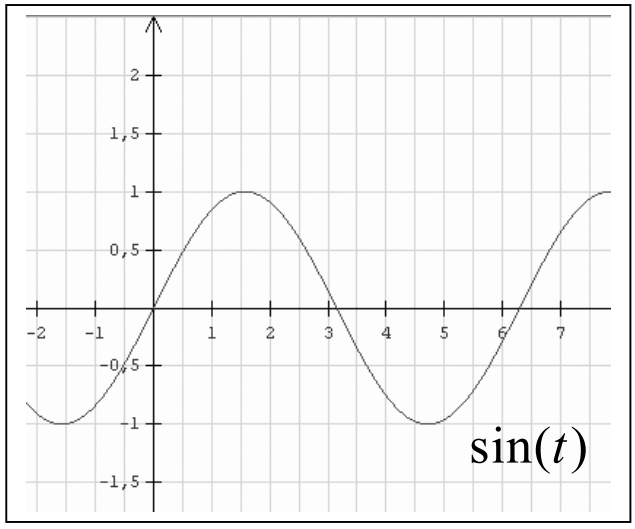
Vor allem das Setzen von einigen globalen Variablen und Flags wäre besser beim Input-Sensor Callback untergebracht, da dies nicht in jeder frame geschehen muss. Das wurde auch teilweise umgesetzt, trotzdem ist der Code Redundanz behaftet was seinen Grund in der Entstehung des Programms hat.

Zu Beginn der Programmierung standen die bequemen Animationspfade über das fpInterpolPack noch nicht zur Verfügung. Die Animation des Würfels wurde manuell im Time-Sensor-Callback ausgeführt und da diese Animationstechnik auch für die Mühlenkamerafahrten verwendet werden sollte, wurde der komplette Programmablauf im Time-Sensor-Callback geschrieben. Später kamen die Interpolationskamerafahrten hinzu. Eine Neustrukturierung des Programms fand nicht statt. Es handelt sich also um eine gewachsene Struktur, denn um eine systematisch konstruierte.

Im folgenden eine Beschreibung der zwei verschiedenen Animationsarten

1) Die Slow-In Slow-Out Animation
2) Kamerafahrten mit dem fpInterpolPack

1) SiSo-Animation



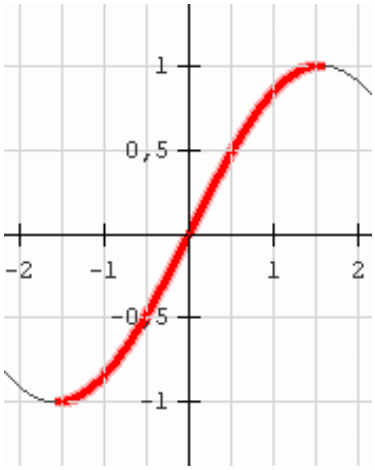
Das Drehen des Würfels wird manuell animiert. Wir bilden den Rotationswinkel, mit dem der Würfel frameweise transformiert wird, auf den Ausschnitt einer Sinus-Schwingung ab und erreichen so eine visuell ansprechende Bewegung, die langsam beginnt, dann schneller wird und zum Ende wieder bis zum Stillstand abbremst.

Die Abbildung rechts zeigt eine normale Sinus-Schwingung

$f(t) = A\sin(\omega t + \varphi) + B$ mit

$A = 1;\ \ T = 2\pi \Rightarrow \omega = 1;\ \ \varphi = 0;\ \ B = 0$

Für unsere Animation ist der Abschnitt im Intervall $\left[-\frac{\pi}{2};+\frac{\pi}{2}\right]$ relevant (hier rot-gefärbt).



Um ihn zu normalisieren, halbieren wir die Amplitude auf $A = 0,5$, verschieben die Schwingung um $\varphi = -\frac{\pi}{2}$ auf der x-Achse und um $B = 0,5$ auf der y-Achse.

Die Kreisfrequenz erhält den Parameter t_{Dreh}, mit dem wir den Abschnitt auf der x-Achse festlegen können.

$$f(t) = A\sin(\omega t + \varphi) + B$$

$$t_{Dreh} = \frac{T}{2} \quad \rightarrow \quad T = 2t_{Dreh}$$

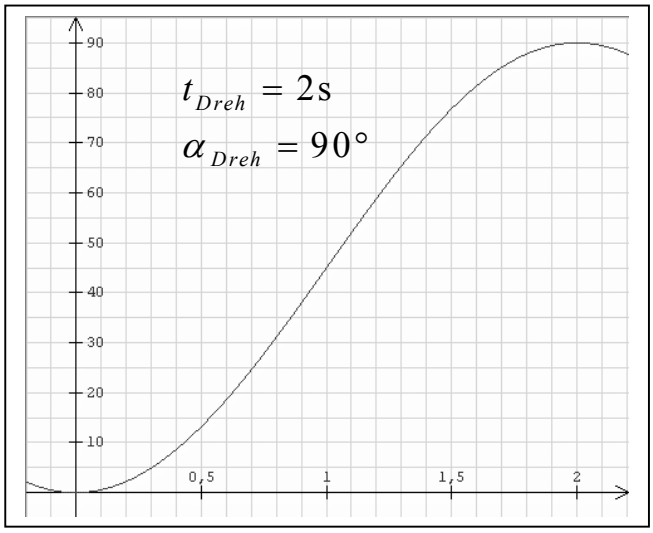
$$\omega = \frac{2\pi}{T} = \frac{2\pi}{2t_{Dreh}} = \frac{\pi}{t_{Dreh}}$$

$$\Rightarrow f(t) = \alpha_{Dreh}\left(0.5\sin\left(\frac{\pi t}{t_{Dreh}} - \frac{\pi}{2}\right) + 0.5\right)$$

für $0 \le t \le t_{Dreh}$

Das ganze multiplizieren wir mit α_{Dreh} und legen so den Wertebereich auf der y-Achse fest.

Hier das Beispiel aus dem Programm für die Windmühle. Die Funktion nimmt im Intervall [0;2] Werte von 0 bis 90 auf der y-Achse ein, und zwar mit den Eigenschaften, die für uns wichtig waren: slow in, slow out.



In unserem Programm wird die jeweilige Rotationsmatrix in der Time-Sensor-Callbackfunktion errechnet.

Die Zeit wird auf 0 gesetzt und die Funktion zur Errechnung des Winkels wird solange ausgeführt, bis die Zeit > t_{Dreh} erreicht.

```
(define (time-cb trigger)

 (set! t (- (fp-get-value time-sensor 'Time)
time_reset))

 (if (< t t_dreh)
    (begin
     ;;frameweise Rotation des Wuerfels
     (set! winkel
      (* alpha_dreh (+ 0.5 (* 0.5 (sin (- (/ (* pi t)
t_dreh) (/ pi 2))))))
     )
     (fp-set-value dcs-node 'Matrix (make-rot-mat
winkel 0 0 1))
    )
 )
)

(define time-trigger (make-instance-by-name
"fpTriggerCB"))
(fp-set-value time-trigger 'Callback time-cb)
(fp-connect-from time-trigger 'Input time-sensor
'Time)
```

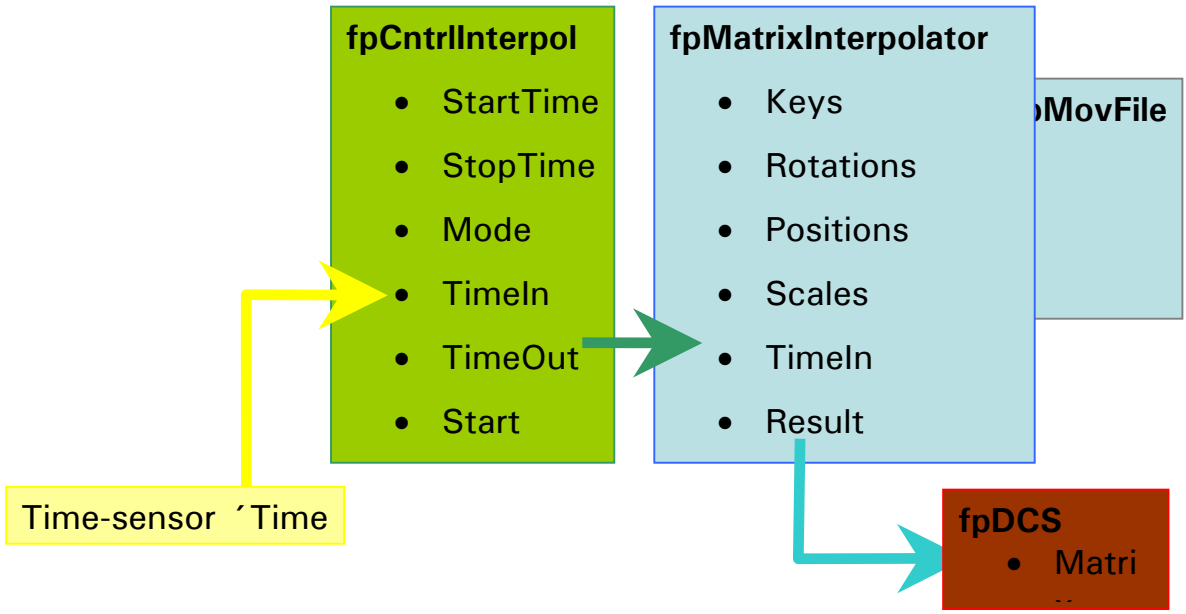
Animation starten mit

```
(set! t_dreh 2)
(set! alpha_dreh 90)
(set! time_reset (fp-get-value time-sensor 'Time))
```

2) Animation mit dem fpInterpolPack

Über den fpMatrixInterpolator-Knoten aus dem fpInterpolPack lassen sich Objekte entlang von Pfaden bewegen und transformieren. Der User definiert für Keyframes die jeweilige Position, Rotation und Skalierung. Daraufhin berechnet der Knoten frameweise die Transformationsmatrizen für die Übergänge zwischen den Keyframes. Die Eigenschaften der Keyframes werden direkt in den Feldern der Knoten eingetragen (Listen). Alternativ lassen sie sich über den fpMovFile-Knoten aus .mov-Files auslesen, welche von üblichen Render-programmen wie 3D-Studio MAX generiert werden können. Für eine Steuerung der Transformationspfade steht der Knoten fpCntrlInterpol zur Verfügung. Über ihn kann man festlegen wann zwischen welchen Keyframes interpoliert werden soll.

Folgende Grafik soll die Zusammenhänge erläutern:



fpMatrixInterpolator

```
(define matrix-intpol-node (make-instance-by-name
"fpMatrixInterpolator"))
```

`fpMFDouble Keys`	Die Keyframes (Zeitpunkte in Sekunden) zwischen denen interpoliert werden soll `(fp-set-value matrix-intpol-node 'Keys (list 0 3 4))`

fpMFQuat Rotations	Die Rotationen zu den jeweiligen Keyframes `(fp-set-value matrix-intpol-node 'Rotations (list (make-quat 90 1 0 0)(make-quat 180 0 1 0)(make-quat 45 1 0 0)))`
fpMFVec3 Positions	Die Positionen zu den jeweiligen Keyframes `(fp-set-value matrix-intpol-node 'Positions (list (make-vec3 0 -4 0)(make-vec3 -1 -5 0)(make-vec3 -0.3 -5 -0.5)))`
fpMFVec3 Scales	Die Skalierungen zu den jeweiligen Keyframes `(fp-set-value matrix-intpol-node 'Scales (list (make-vec3 1 1 1)(make-vec3 0.8 0.8 0.8)(make-vec3 0.01 0.01 0.01)))`
fpSFDouble TimeIn	Das TimeIn Feld wird mit dem Time-Sensor verknüpft, in diesem Beispiel indirect über den fpCntrlInterpol-Knoten `(fp-connect-from matrix-intpol-node 'TimeIn control-intpol-node 'TimeOut)`
fpSFMatrix Result	In diesem Feld werden frameweise die interpolierten Transformationsmatrizen ausgegeben. Es wird mit dem zu transformierenden DCS-Knoten verknüpft. `(fp-connect-from dcs-node 'Matrix matrix-intpol-node 'Result)`

fpCntrlInterpol

```
(define control-intpol-node (make-instance-by-name
"fpCntrlInterpol"))
```

fpSFDouble StartTime	Die Start-Keyframe `(fp-set-value control-intpol-node 'StartTime 0.0)`
fpSFDouble StopTime	Die Stop-Keyframe `(fp-set-value control-intpol-node 'StartTime 3.0)`
fpSFString Mode	Der Abspielmodus (einmal, Endlosschleife etc.) `(fp-set-value control-intpol-node 'Mode once)` möglich sind: `once, once_reverse, loop, loop_reverse, swing, swing_once`
fpSFDouble TimeIn	Das TimeIn Feld wird mit dem Time-Sensor verknüpft `(fp-connect-from control-intpol-node 'TimeIn time-sensor 'Time)`
fpSFDouble TimeOut	Das TimeOut Feld steuert den fpMatrixInterpolator-Knoten und liefert diesem die entsprechende Start- und Stop-Time, damit zwischen den richtigen Keyframes interpoliert wird `(fp-connect-from matrix-intpol-node 'TimeIn control-intpol-node 'TimeOut)`

fpSFInt Start	Durch folgenden Aufruf wird die Interpolation zwischen den vorher festgelegten Werten von StartTime und StopTime gestartet. `(-> (-> control-intpol-node 'Start) 'touch)`
fpSFInt OnceDone	Wird = 1, wenn die festgelegte StopTime erreicht wurde. `(if (= 1 fp-get-value control-intpol-node 'OnceDone))(display 'movement finished!'))`
fpSFInt ResetLastTime	Nachdem alle Keyframes abgearbeitet wurden, muss der fpCntrlInterpol-Knoten mit diesem Befehl wieder zurückgesetzt warden, damit der Ablauf von vorne beginnen kann, z.B. beim Neustart des Programmablaufs `(fp-set-value control-intpol-node 'ResetLastTime 1)`

Das Beispiel aus dem Windmühlen-Programm. Die Kamera wird über Interpolationsknoten bewegt:

1) Initialisierung

```
(define camera (make-instance-by-name "fpDCS"))
(fp-set-value camera 'Name "camera")

(fp-remove-1value scene-root 'Children av-viewer)

(define interpol  (make-instance-by-name "fpMatrixInterpolator"))
(define intcontrol (make-instance-by-name "fpCntrlInterpol"))

(fp-set-value camera 'Children (list av-viewer interpol intcontrol))
(av-add camera)

(fp-set-value intcontrol 'Mode "once")
(fp-connect-from intcontrol 'TimeIn time-sensor 'Time)

(fp-connect-from interpol 'TimeIn  intcontrol 'TimeOut)

 (fp-set-value interpol 'Keys  (list 0 1 2 4 6 8 10 12 14 16 17 18 20 22 24 25))
(fp-set-value interpol 'Positions  (list  (make-vec3 0 -4 0)
                                          (make-vec3  -1 -5 0)
                                          (make-vec3 -0.3 -5 -0.5)
                                          (make-vec3 -0.3 -3 0.2)
                                          (make-vec3 3 -0.5 0.2)
                                          (make-vec3 3 -4 -0.5)
                                          (make-vec3 5 -1 -0.5)
                                          (make-vec3 2 -1 -0.2)
                                          (make-vec3 0.3 -0.3 -0.2)
                                          (make-vec3 0.3 -0.3 -1.2)
                                          (make-vec3 0.3 0 -1.2)
                                          (make-vec3 0.3 0 -1.6)
                                          (make-vec3 0.1 0 -1.6)
                                          (make-vec3 -0.5 1.3 -1.0)
                                          (make-vec3  -1 -5 0)
                                          (make-vec3 0 -4 0)
                                          ))
(define null-rot  (make-quat 0 0 1 0))
(define eins-rot  (make-quat 225 0 0 1))
(define zwei-rot  (make-quat 45 0 0 1))
(define drei-rot  (make-quat 90 0 0 1))
```

```
(define null-scl  (make-vec3 1 1 1))
(define eins-scl  (make-vec3 0.8 0.8 0.8))

(fp-set-value interpol 'Rotations (list null-rot null-rot null-rot null-rot drei-rot
zwei-rot drei-rot drei-rot drei-rot drei-rot drei-rot drei-rot drei-rot eins-rot null-
rot null-rot))
(fp-set-value interpol 'Scales (list null-scl null-scl null-scl null-scl null-scl
null-scl null-scl null-scl null-scl null-scl null-scl null-scl eins-scl eins-scl null-
scl null-scl))

(fp-connect-from camera 'Matrix interpol 'Result)
```

erste Kamerafahrt:

```
(fp-set-value intcontrol 'StartTime 0.0)
(fp-set-value intcontrol 'StopTime 1.0)
(fp-set-value intcontrol 'OnceDone 0)

(if (= 0 (fp-get-value intcontrol 'OnceDone))
   (-> (-> intcontrol 'Start) 'touch)

   ;else
```

sobald die Kamerafahrt abgeschlossen ist (OnceDone = 1):

```
   (begin
       (set! current_scene 1)
       (set! change_scene 1)
       (fp-set-value szenensound 'Play 1)
   )
)

…
```

Nach dem Ende aller Kamerafahrten:

```
(fp-set-value intcontrol 'ResetLastTime 1)
```

5. Evaluation und Ausblick

Am Ende des Projektes war es uns sehr wichtig, nicht nur unsere Entwürfe zu präsentieren, sondern auch Probleme aufzuzeigen und das Feedback auszuwerten, wobei der Rundgang die Grundlage der Evaluation darstellt. Was für uns sehr erfreulich war, ist die Tatsache, dass viele Besucher unser Pusteblumenspiel in der Virtuellen Vitrine sehr positiv bewertet haben. Die Virtuelle Windmühle war aus technischen Gründen nur zwei Stunden am Rundgang zu sehen und konnte nur ohne Sound vorgeführt werden. Bei der Benutzung der Eingabegeräte gab es für viele Besucher kleinere Probleme. Zum einen ließ sich die Pusteblume zu schwer anpusten, da der CPU-Lüfter doch ein gewisses Maß an Wind braucht um anzulaufen und somit brauchbare Werte zu liefern. Zum anderen haben einige Leute es vermisst, dass das „Pflücken" des Eingabegerätes nicht grafisch gemappt war in der Form, dass die virtuelle Blume auch von ihrer Wiese gepflückt wird. Der Hintergrund der Szene war etwas zu dunkel gestaltet, so das teilweise schlecht zu erkennen war, wo die einzelnen Samen hinflogen nachdem sie von der Blume gepustet worden waren. Ansonsten ist uns aufgefallen, dass neue Eingabegeräte immer einen kleinen Lern- und Übungsprozess erfordern, besonders wenn sie aufgrund des untypischen Inputs etwas seltsam aussahen. Sie entsprachen somit einfach nicht der Vorstellung der meisten Leute von einem Eingabegerät. Nach einer kurzen Anleitung unsererseits war es ihnen aber doch relativ leicht möglich die Szenarien und deren Eingabegeräte auszuprobieren. Eine weitere Erkenntnis ist wiederum, dass Kabelverbindungen von Eingabegerät zum Rechner als störend empfunden werden. Besonders wenn auch noch eine Schatterbrille im Spiel ist, wie es ja bei der virtuellen Vitrine notwendig ist. Anderseits zeigte sich, dass das Fehlen des Kabels die Information genommen hat, ein funktionsfähiges Eingabegerät vor sich zu haben. Die Hygiene ist ein weiteres Problem, weil viele Leute in die Eingabegräte hineinpusten und dies nie ganz ohne die Abgabe von Speichelresten passiert. Vor allem das Eingabegerät der virtuellen Pusteblume ist davon betroffen, da man dort sehr nah mit dem Mund zum Lüfter gehen muss um eine Reaktion zu erzielen.

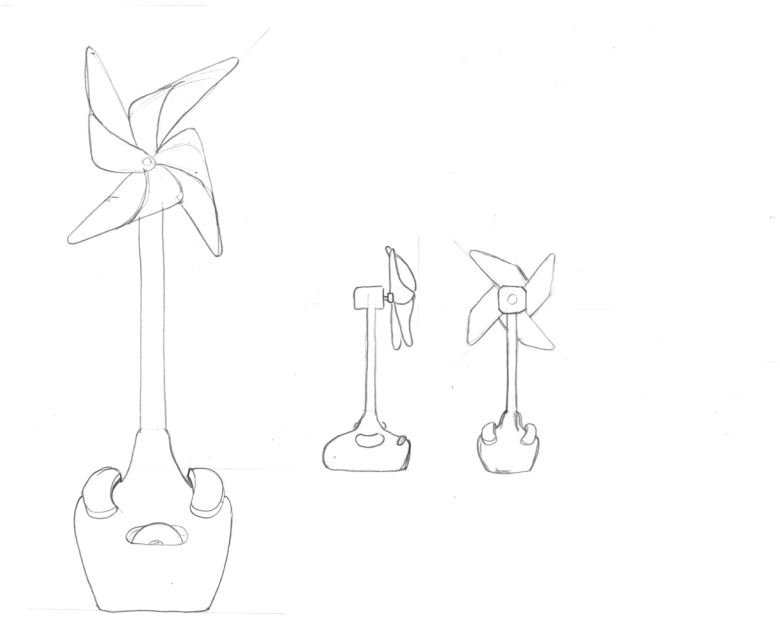
Bei der Virtuellen Windmühle ist erst beim Rundgang aufgefallen, dass das Pusten nur zweimal in der Anwendung zum Tragen kommt, obwohl es eigentlich der Hauptinput ist. Die Ursache dafür bildet die Geschichte die wir um die Hauptinteraktion herum gebaut haben, um so einleitende Informationen zu vermitteln und den Unterhaltungsfaktor nicht zu kurz kommen zu lassen. Das Windrad ist daher vielleicht etwas zu zentral am Eingabegerät angeordnet. Der intuitive Griff zur Maus war bei allen Besuchern erkennbar, die die Anwendung nicht kannten. Das Scrollrad mit dem die Windrichtung eingestellt werden kann wurde oft als zu klein bemängelt. Darüber hinaus war die grafische Repräsentation

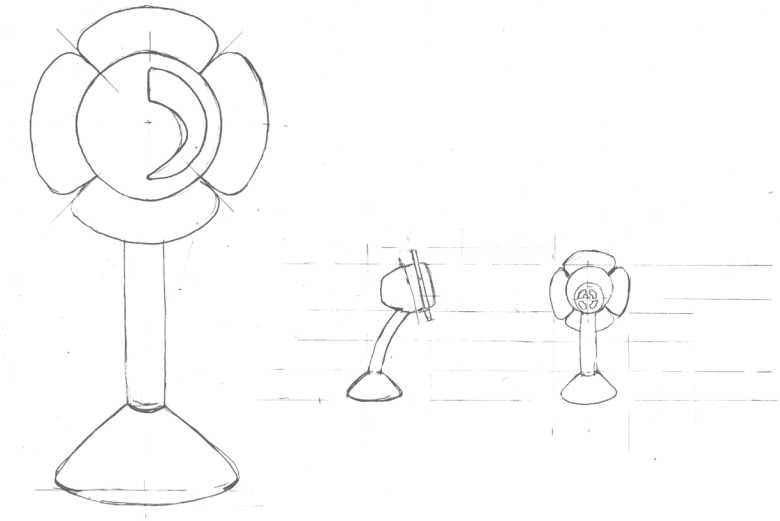
der Windrichtung nicht ganz eindeutig umgesetzt. Warum haben wir es nicht besser gemacht, wenn wir ja doch die Fehler entdeckt haben? Nun, wir haben gemerkt, dass das Vorhaben, eine museumstaugliche Anwendung zu erstellen, sehr komplex und schwierig ist. Da wir nur drei Wochen Zeit hatten, ist die Virtuelle Windmühle nur als Zwischenergebnis zu verstehen. Um wirklich eine Anwendung zu programmieren, die die Anforderung erfüllt, lehrreich und gut benutzbar zu sein, bedarf es einer Testreihe, um Fehler zu entdecken. Weiter wäre es sinnvoll, einen Pädagogen zu Rate zu ziehen, der Tipps darüber geben könnte, wie man Informationen für Kinder und Jugendliche bis etwa 10 Jahre attraktiv gestaltet. Wir haben uns bei diesem Gesichtspunkt beholfen indem wir uns an der Vorlage von Kinderbüchern orientierten die auf kindgerechte Art und Weise Wissen vermitteln. Wir hatten beim Rundgang das Glück auch zwei Familien mit Kindern im Alter der angestrebten Zielgruppe als unsere Gäste begrüßen zu können. Nachdem sie die Virtuelle Windmühle ausprobiert hatten, äußerten sich die Elterteile jeweils sehr positiv über das Szenario. Eine Mutter begrüßte es auch solche Tendenzen beim Rundgang zu entdecken. Sie war der Meinung, dass dadurch nicht nur den Kindern ansprechend Wissen vermittelt werden kann sondern die Kleinen so auch an die Computertechnik herangeführt werden. Sinnvoll wäre es natürlich auch, wenn sich Erwachsene von der Anwendung angesprochen fühlten. Die Frage, warum wir uns ausgerechnet für eine Windmühle entschieden haben und nicht etwa für einen Windpark, um eine Lernanwendung zu gestalten, lässt sich damit begründen, dass es nicht nur um einen Produktionsvorgang geht, der erläutert werden soll, sondern auch um die Abhängigkeit des Menschen von Rohstoffen. Heute wird Mehl mit Hilfe von Strom erzeugt, der oftmals aus Atomenergie gewonnen wird. Es ergibt sich also die Frage, warum nicht wieder wie vor 200 Jahren auf den Wind zurückgegriffen wird. Große Windparks sind die Grundlage für Energie, die ökologisch ist. Die Erläuterung von Windkraftanlagen in einer Virtuellen Vitrine könnte also die Fortsetzung von der Mühlenanwendung sein und ebenfalls in einem Museum oder Science-Center ausgestellt werden.

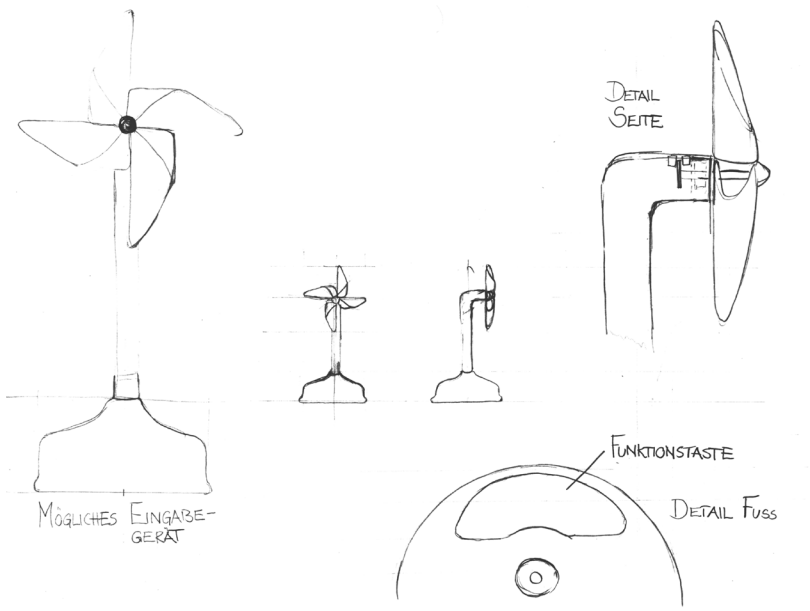
Das Ziel, Pusten als Input zu verwenden haben wir erreicht. Es wäre auch gut möglich, dies nicht nur in spielerischen Anwendungen ein zusetzten, sondern auch in der Industrie oder der Medizin anzuwenden. Die Technik könnte eingesetzt werden um Produktionsprozesse wie Glasblas- oder Blow-Moldingverfahren aus der Kunststoffindustrie zu simulieren.

In der Medizin könnte man zum Beispiel das Lungenvolumen testen, was heute auch schon ähnlich funktioniert, aber sehr aufwendig und teuer ist. Es wäre denkbar, mit veränderter Technik günstigere Methoden zu entwickeln.

6. Skizzen und Bilder

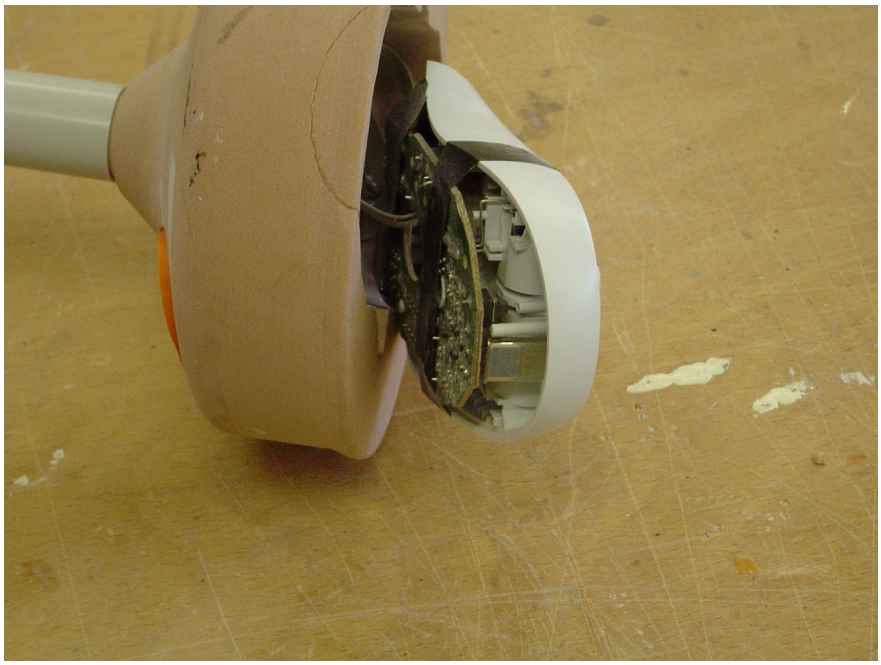




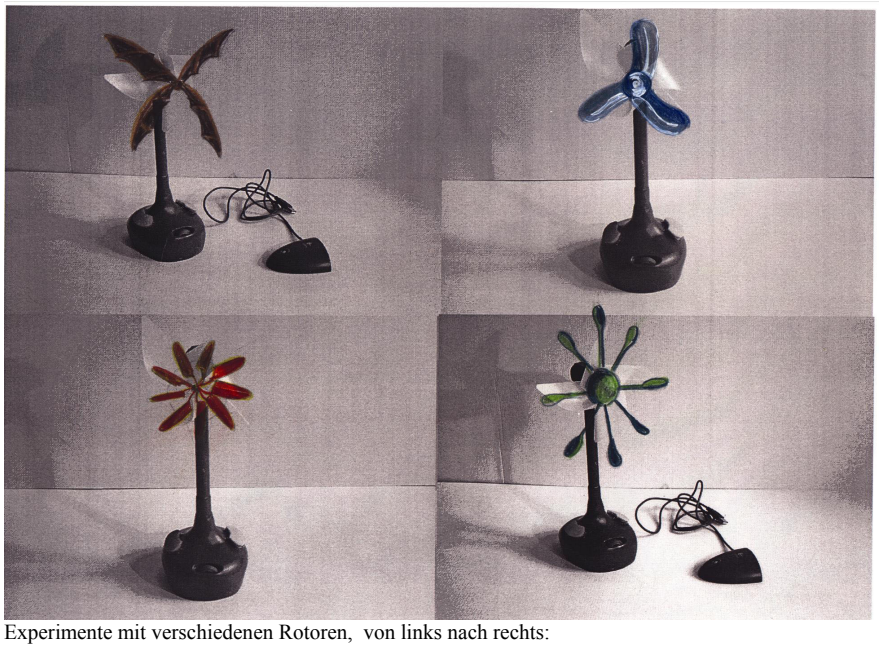


Testperson beim Pusten in erstes Funktionsmodell

Vorderansicht Pusteblume, bevor die Technik integriert wurde



Die Maus wird in den Fuß eingebaut



Experimente mit verschiedenen Rotoren, von links nach rechts:
Fledermausflügel, Ventilator, bionisches Windrad, Rotor aus Plastiklöffeln

7. Quellen

Der Guckkasten. Die Mühle, hg. v. de Ruiter, Hamburg 1993

Lemke, Stefan: basteln, lachen, selbermachen, Stuttgart und Wien 1989.

Links:
www.windmill.de
www.loewenzahn.de
http://www.logitech.com/index.cfm?page=products/details&CRID=3&CONTENTID=4983&countryid=7&languageid=4
www.conrad.de
www.reichelt.de
www.avango.org