„SLEEPWALKER“

Simon Broggi, Claire Geyer, Kai Jauslin

Interaction Design 4. Semester

Zürcher Hochschule der Künste

Retrogame real.loaded

Dozenten: Raphael Perret und Max Rheiner

21.05.2008 – 06.06.2008

IAD-Modul Bachelor Design

Frühlingssemester 2008

INHALT

RETRO GAME REAL.LOADED

Aufgabenstellung

Obwohl Spiele komplexer werden sind die dazu erforderlichen Schnitstellen relativ einfacher Natur. In der Regel handelt es sich um Gamekontroller, Gamepads, Joysticks, Maus oder Tastatur in Kombination mit einem Bildschirm. In diesem Seminar thematisieren wir die Materialisierung eines Spiels im Realraum. Der Raum wird zum Interface und Display.

Inszeniert eine Spielsituation in der ein Retrogame ohne Bildschirm im Raum aufgebaut ist.

Projektpage:

http://interaction.zhdk.ch/projects/physcomp/seminare/retro-game-fs08/

Obiges Beispiel zeigt ein Retrogame in abgewandelter Form: Anstatt zu schiessen ist das Spielziel, die abgeschossene Landschaft tatenlos zu überfliegen. Eine Verweigerungsstrategie herkömmlicher Spielprinzipien gegenüber.

ZEITPLAN:

Woche 1
Mittwoch, 21. Mai
9 -12h Thema eröffnen "Retro Game im Raum"
13 - 17h Atelierbesuch in Basel: 14:30 Tegoro Solutions, 15:15 Any Affair, Tram #8 bis Kleinhüningen

Donnerstag, 22. Mai
9 - 12h Diskussion im Plenum
Gruppen
1: Christophe, Konradin, Luisa
2: Claire, Kai, Raffaele, Simon
3: Luigi, Rosario, Soo
4: Jérôme, Johann, Patric

13 - 15h Ideenfindung
15 - 17h Diskussion im Plenum

Freitag, 23. Mai
9 - 12h Technologie Analyse
13 - 17h Technologie Bausteine 1

Woche 2
Dienstag, 27. Mai
9 -12h Technologie Bausteine 2
13-17h Weiterentwicklung der Konzeptidee

Mittwoch, 28. Mai
9 -12h Beispielaufgaben lösen
13-17h Konzeptarbeit

Donnerstag, 29. Mai
9 -10h Planung von Aufwand, Ressourcen, Zeitplan etc.
11-12h Ideen präsentieren
13-17h Umsetzung

Freitag 30. Mai 9 -17h Umsetzung

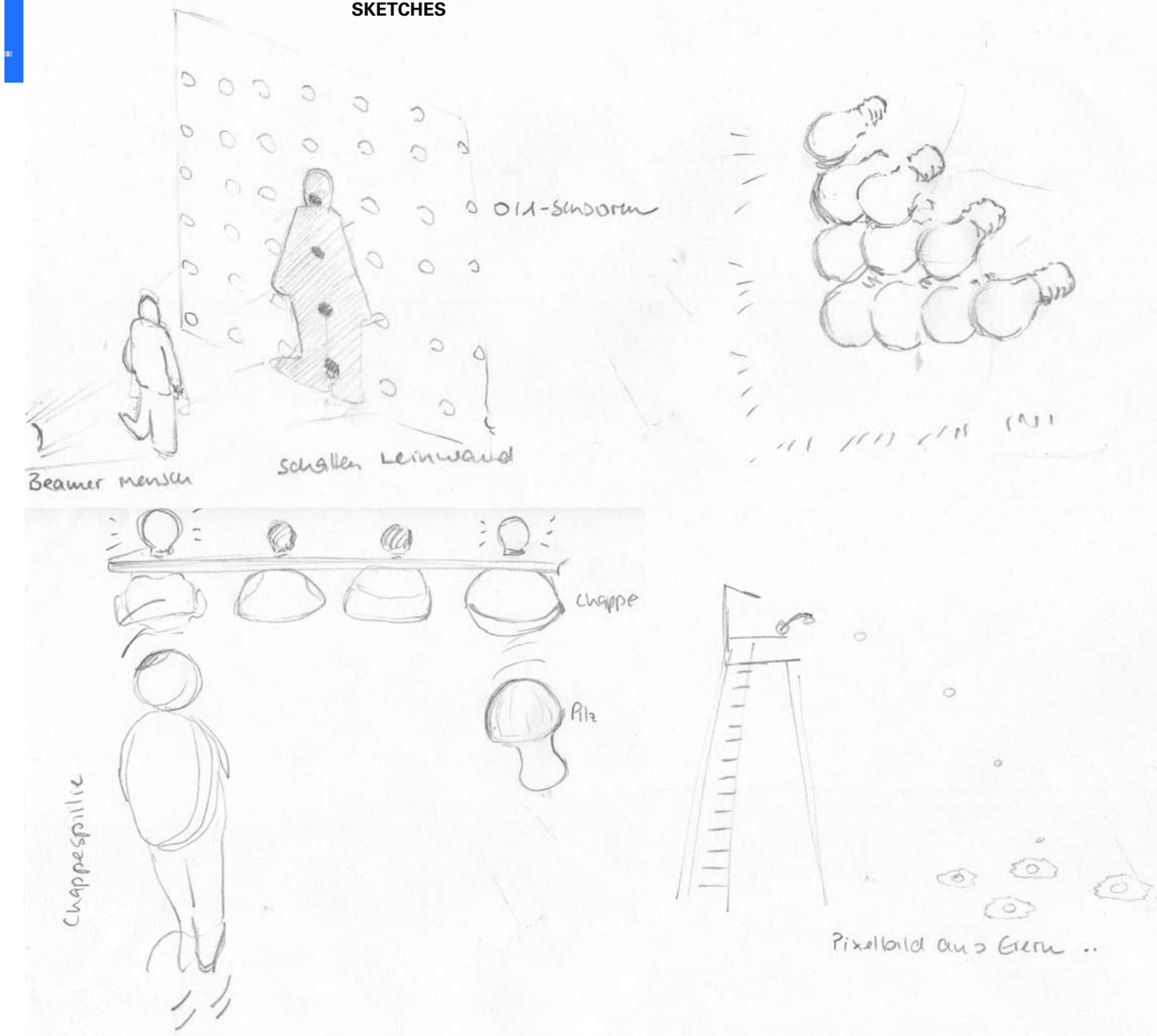
Woche 3
Dienstag, 3. Juni 9 -1 7h Umsetzung 10:00 Plan für die Zeiteinteilung der Woche vorstellen

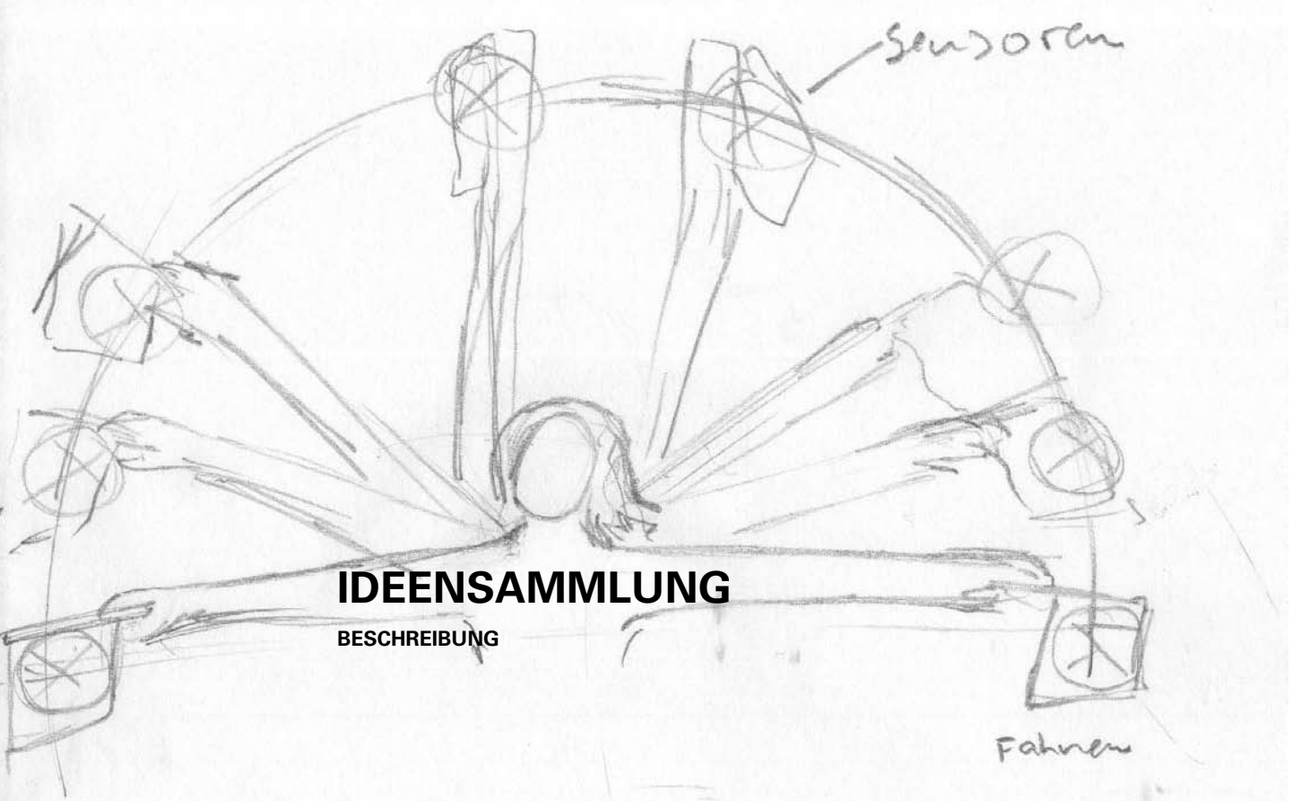
Mittwoch, 4. Juni 9 -17h Umsetzung

Donnerstag, 5. Juni 9 -17h Präsentation 14:00
Freitag, 6. Juni 9 -17h Dokumentieren, Aufräumen

IDEENSAMMLUNG 1

SKETCHES





IDEENSAMMLUNG

BESCHREIBUNG

Tetris mit Schachteln im realen Raum

Zwei-Spieler-Variante: Der obere Spieler wirft in zufälliger Reihenfolge überdimensionierte, dreidimensionale Tetristeile vom Podest. Der Untere Spieler ordnet sie schnell richtig an, so dass ganze „Zeilen“ weggelöscht werden können. Zum Löschen der Zeilen müssen die einzelnen Bauteile voneinander trennbar, also mit Magnet verbunden sein. Spielkategorie Performance.

Tetris mit Schachteln im geschlossenen Raum

Die Tetrisbausteine werden zur einen Tür reingeschoben. Ziel ist es, bei immer knapper werdendem Raum die Tele möglichst schnell und richtig anzuordnen. Spielkategorie Performance.

Schattenspiel mit Schattenerkennung

In einer Matrix angeordnete Lichtsensoren können grobe Schattenmuster erkennen. Spielkategorie Eingabevariante.

Glühbirnenmatrix als Ausgabedisplay

Die Möglichkeit, Glühbirnen mit Relais anzusteuern und diese Technologie als Ausgabemedium zu nutzen, hat uns fasziniert. Einsatzgebiet noch unbekannt. Spielkategorie Ausgabevariante.

Chappespiilie

Super Mario mit seiner Mütze wurde weltberühmt. Ziel unserer Spielidee wäre es, einen Knopf auszulösen, der erkennt, ob man zur richtigen Zeit mit dem Kopf gegen die richtige Kappe gesprungen ist. Spielkategorie Hochsprungspiel.

Pixelbild werfen

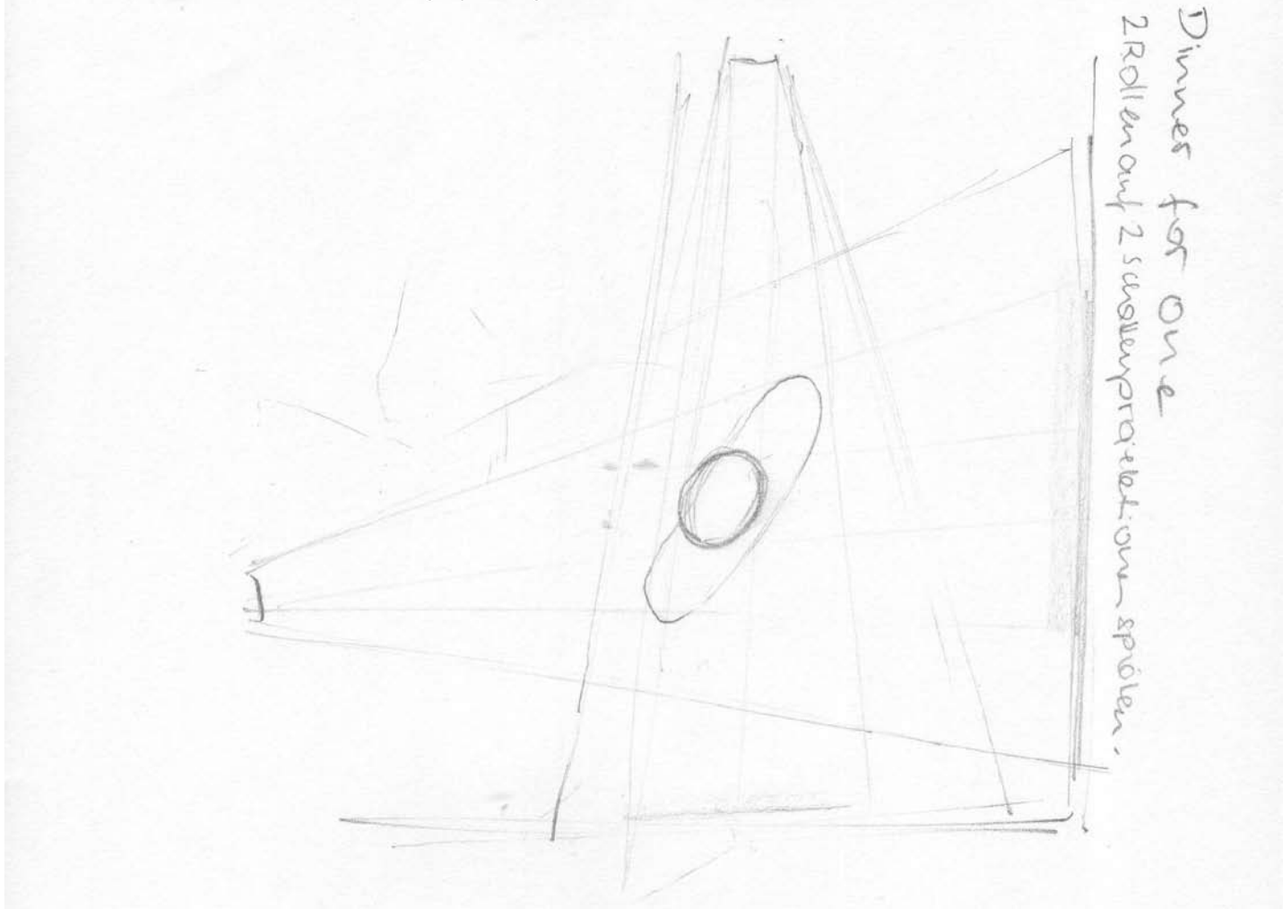
Ziel ist es, von einem Hochstand aus mit Eiern spiegeleierformige Pixel zu erzeugen. Spielkategorie Weitwurfspiel.

Fahnen winken

Mit ausgestreckten Armen den Bewegungsradius des Spielers feststellen und die Armpositionen als Eingabesmöglichkeit zu nutzen. Spielkategorie Eingabevariante.

Dinner for One

Ein Spieler spielt gleichzeitig die Rolle der alten Dame und die Rolle des Buttlers. Schattenerkennung vergibt Punkte für gelungene Darbietung. Spielkategorie Performance.







EGG EG-26 (14. Spiel)

Serie: Wide Screen

Produktnummer: EG-26

Erstausgabe: 09.10.1981

Auflage: 250’000

Inhalt > Papiere: Anleitung

Batterien: LR43

Andere: Styropor, Plastiktasche

Gewicht (Gerät): ca. 63g

Abmessungen: Gerät: ca. 112x67x12mm (LxBxH), Display: ca. 54x35mm

Anzahl Seriennummern in der G&W Datenbank: 73

Varianten/Länderausgaben:

Beschreibung: Der Wolf arbeitet im Hühnerstall und muss die von den Hühnern gelegten Eier auffangen. Achte darauf das nichts daneben fällt in dem Du den Wolf an allen vier Stellen immer rechtzeitig am richtigen Ort sein lässt. Fällt mal ein Ei runter, wird Dir ein (Hühner)-Leben abgezogen.

Rarität (standard Games*) bei Auktionen: Extrem selten

SPIELANALYSE 1

EGG

Die Herausforderung des Spiels besteht im Timing, die Geschwindigkeit wird immer schneller, Reaktion ohne zu denken wird erforderlich. Reaktionsmöglichkeiten sind oben, unten, links, rechts.

Umsetzen könnte man das Spiel mit Tischtennisbällen, die man auffangen soll.






Umsetzung:

Passanten mit Bananenschalen

Katzen retten

Dias, Sprungtechnik, (Beamer)

Haus, Projektion

Insomnie

model:

manufacturer: Ludotronic

serial: None

screen type: LCD

number of screens: 1

number of screen layers: 1

touchscreen: false

color: false

battery type: LR44 or SR44

number of batteries: 2

shape: rectangular

number of players: 1 player

release year: unknown

case colors:

overall:

graphics:

idea:

case quality:

fun factor:

SPIELANALYSE 2

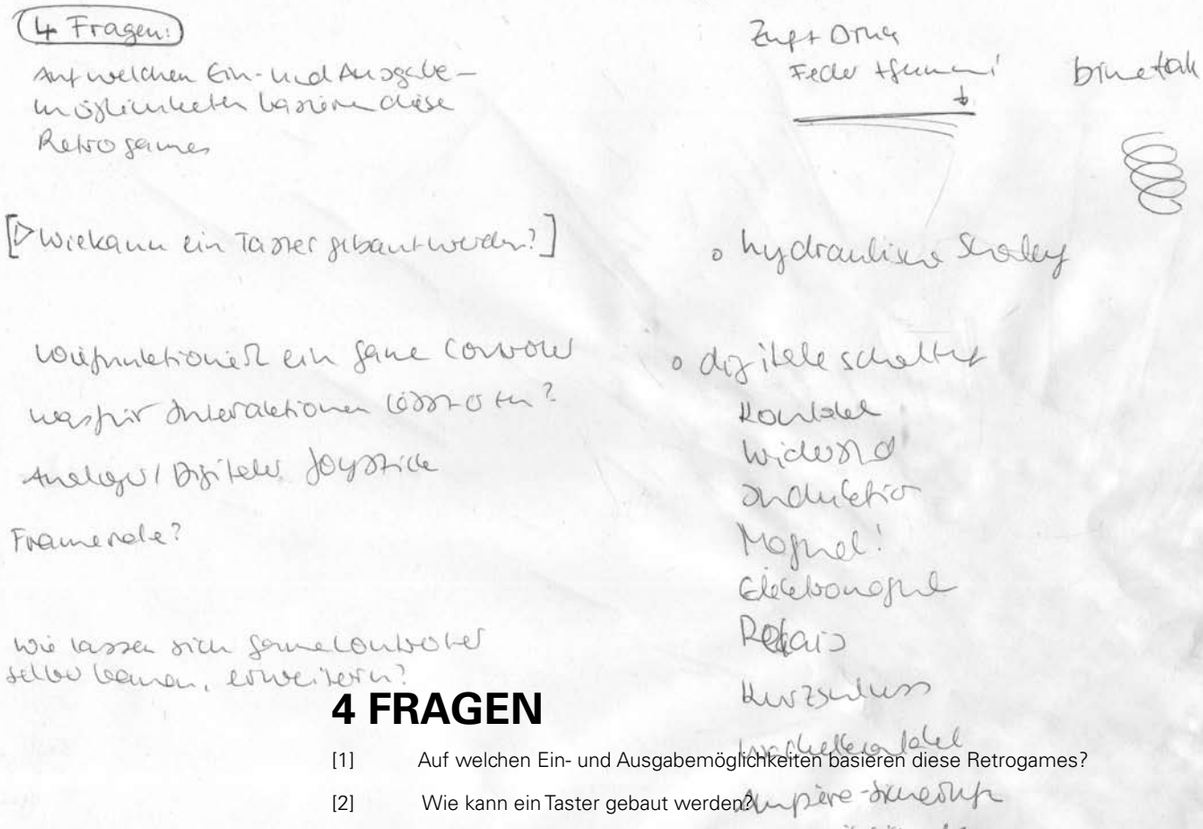
SLEEPWALKER

Charakteristisch für das Spiel ist ein ständiges Ticken, ein Rhythmus, aber auch das unerbittliche Fortschreiten des Schlafwandlers, der seine Umwelt nicht wahr nimmt und wie ferngesteuert wirkt.

Das gefährlichste am Schlafwandeln ist nicht das Umherwandern, sondern scheinbar das Erwachen in einer gefahrenvollen Position, da der erwachende Sleepwalker seine Kontrolle verliert.

Deshalb ist des das Spiel des Zieles, das Fallen des Sleepwalkers zu verhindern. Gesteuert wird nicht der Sleepwalker, sondern die Aufgabe des Spielers ist es, mit 4 Knöpfen die Hindernisse aus dem Weg zu räumen. Voraussetzung ist, dass man zur richtigen Zeit am richtigen Ort den Knopf drückt, um das Fallen zu verhindern.

Single Play



4 FRAGEN

[1] Auf welchen Ein- und Ausgabemöglichkeiten basieren diese Retrogames?

[2] Wie kann ein Taster gebaut werden?

[3] Wie funktioniert ein Game Controller? Was für Interaktionen lässt er zu?

[4] Wie lassen sich Game Controller selbst bauen / erweitern?

[2] Wie kann ein Taster gebaut werden?

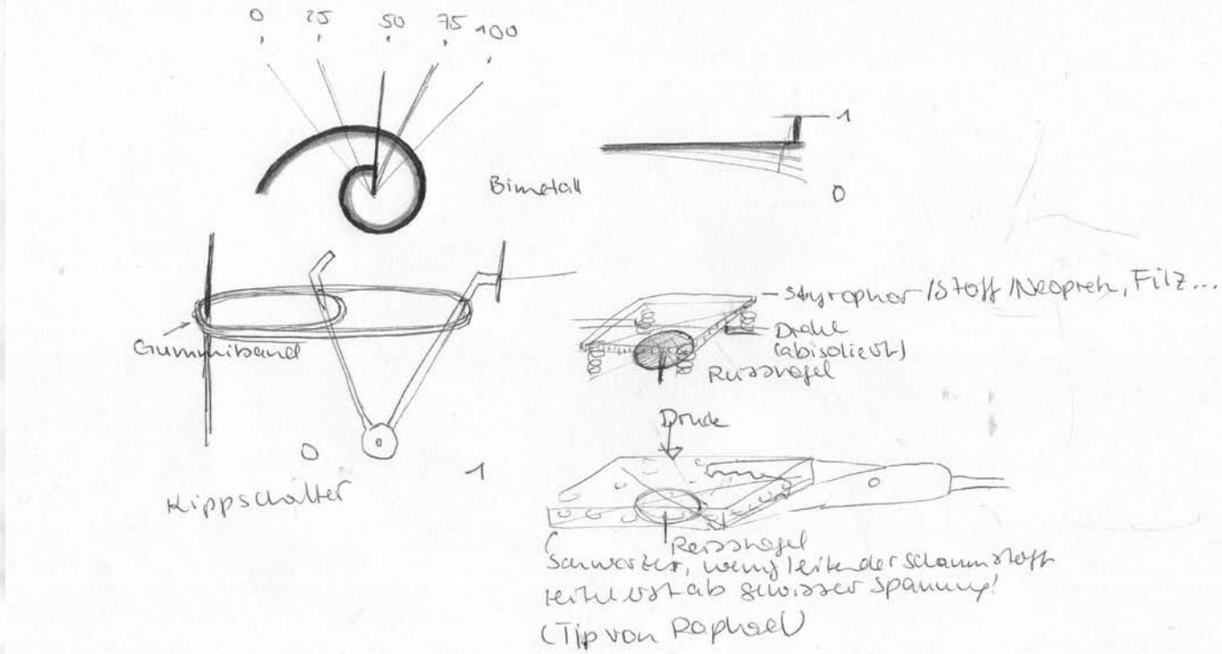
> Mechanisch

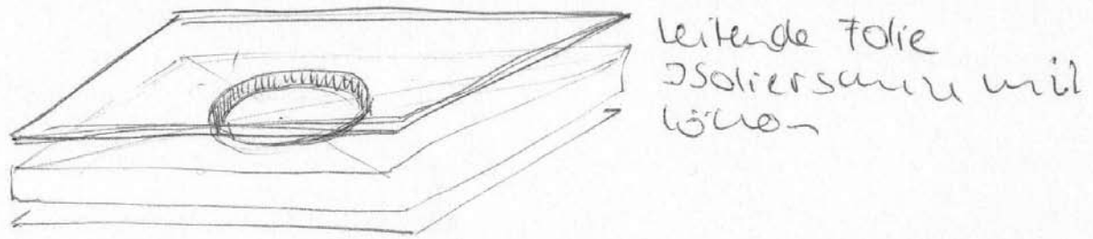
- Zug + Druck
- Feder + Gummi

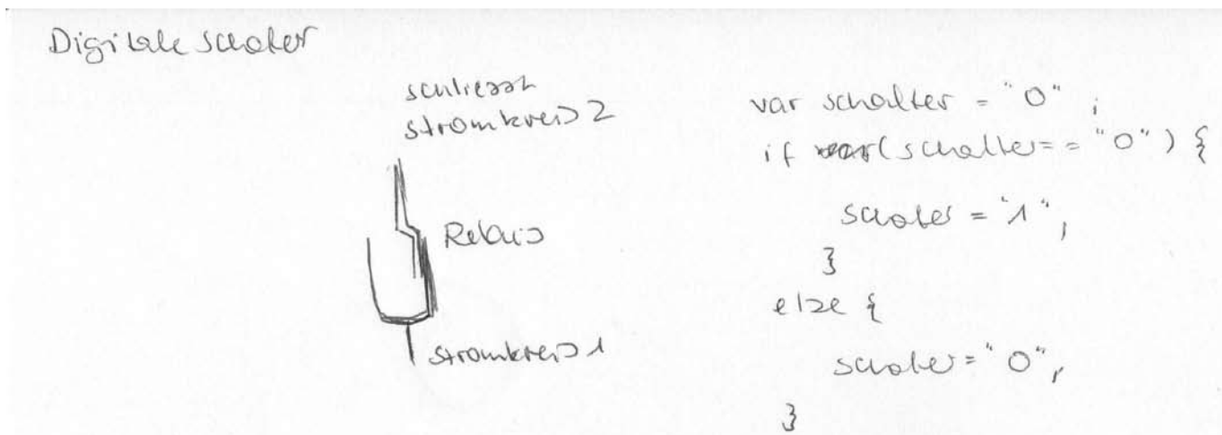
> Hydraulische Schaltung

> Digitale Schaltung

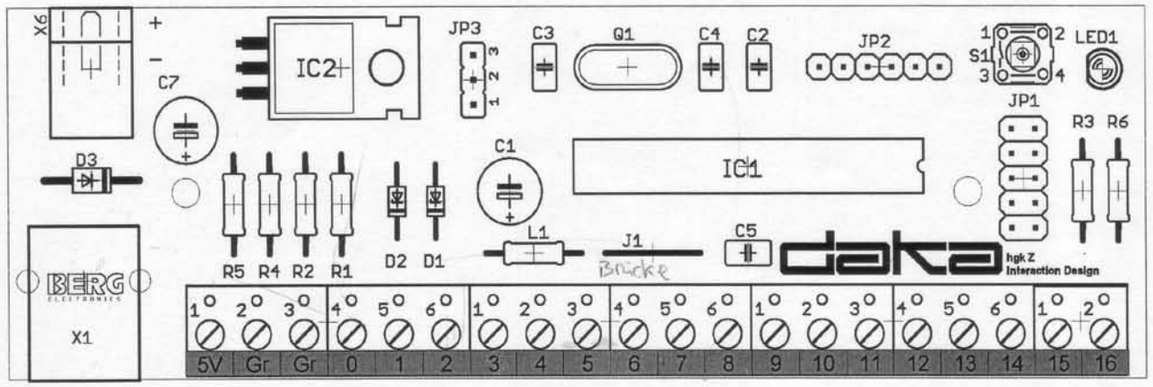
- Kontakt
- Widerstand
- Induktion
- Magnet
- Elektromagnet
- Relais
- Kurzschluss
- Wackelkontakt
- Ampère-Sicherung
- Keramiksicherung







daKa Bestückungsplan



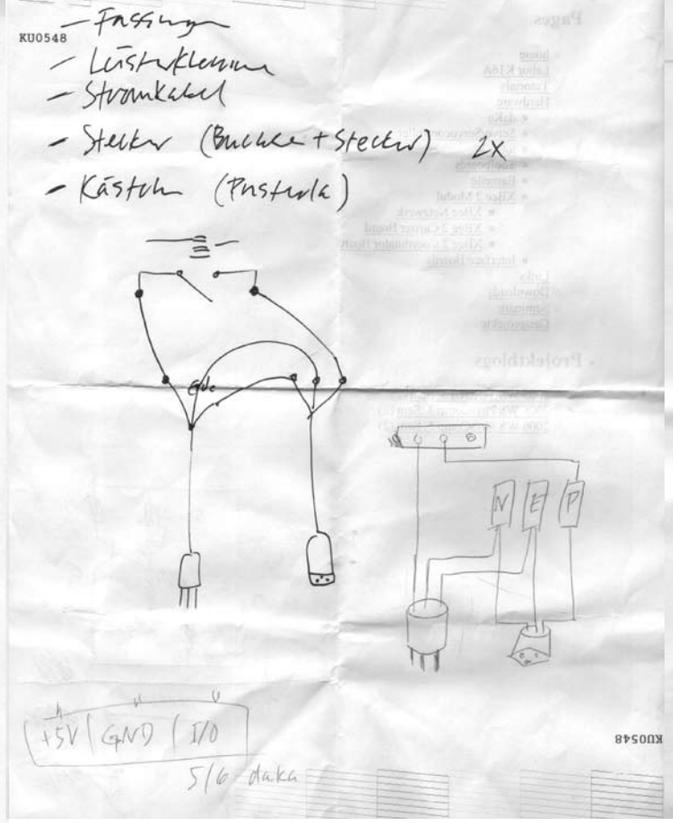
SCHALTUNG, PYTHON UND ARDUINO

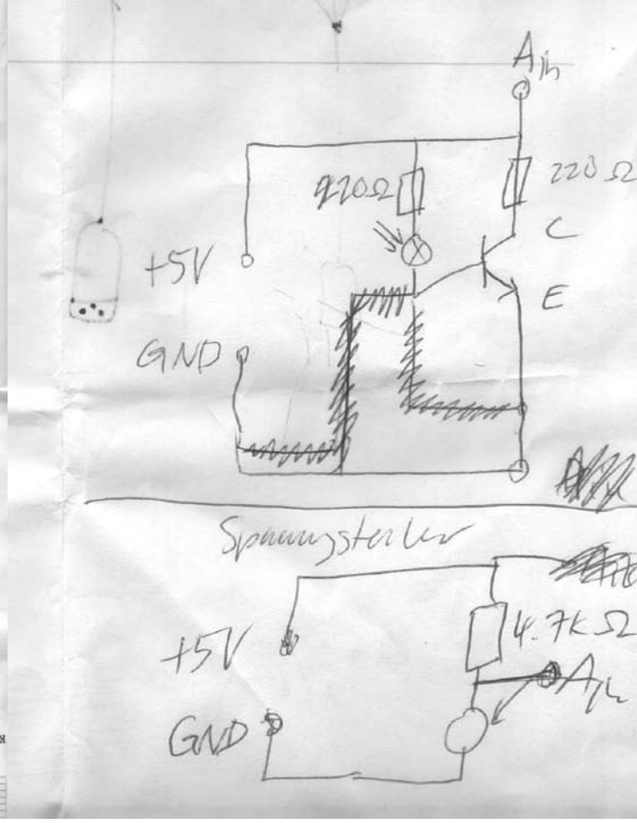
Einführung in die Elektronik

Bestückungsplan

Kurzbezeichnung	Wert
R1	1,5 Kohm
R2	1 Mohm
R3	4.7 Kohm
R4	68 Ohm
R5	68 Ohm
R6	470 Ohm
L1	10 uH
C1	10 uF
C2	100 nF
C3	22 pF
C4	22 pF
C5	100 nF
C7	10 uF
JP1	2 x 5 Pol
JP2	1 x 6 Pol
JP2	1 x 3 Pol
LED1	3 mm, orange
IC1	Atmega168-16
IC2	7805
X1	Usb Buchse
X2	Neb21R / Spannugsbuches
Q1	Quarz 16Mhz
S1	Reset-Taster

D1
D2





...

Serielle Verbindung statt USB

analoger Servo, Potentiometer + Motor, Leitung mit Takt, bewegt sich an richtige Position.

Controller Board, Humanoide Roboter

180°, 300°, in 1 Richtung endlos, liefert genaue Position zurück

www.robotis.com

Sensorplatte Berührungslos Schaltung machen, nah genug ran

3-achsiger Beschleunigungssensor xyz

RFID: wer bist Du?

Zigbee: meshnetworksensoren 50m

networkfähig bis 100K Bit / sec.

Licht / Wärmesensoren

Epoxyharz-Kupferschicht, wird weggeätzt

Widerstände haben Wert, sind nicht polarisiert

Widerstand kann man messen

I/O Module

9V –> 5V für Chip

USB-Kabel = Speisung || Batterie, Jumper leitet Strom um.

Quarz gibt Takt vor, MHz-Zahl, wie schnell

...

ECLYPSE, PYTHON UND ARDUINO

Einführung in die Elektronik

Das Dakaboard wird mit Arduino-Code beladen. Das funktioniert auch als Stand-Alone-Lösung. Das Dakaboard kann aber auch mit dem Pythoncode im Eclypse kommunizieren, wie im Beispiel: Das Board gibt 40 und 41 im Wechsel aus.

Arduino – 0011 Alpha

Upload to I/O Board

sketch_080527a

```
// ----------------------------------------------------------
//  Blink Test 2
// ----------------------------------------------------------
// Blinken steuerbar per schalter

int pinOut=7;
int pinIn=6;

void setup()
{
   pinMode(pinOut, OUTPUT);
   pinMode(pinIn, INPUT);
}

void loop()
{
  if(!digitalRead(pinIn))
   {
     digitalWrite(pinOut, HIGH);
     delay(500);
     digitalWrite(pinOut, LOW);
     delay(500);
   }
   else
     digitalWrite(pinOut, LOW);
}
```

Done uploading.

```
Couldn't determine program size:    text          data        bss         dec
hex         filename

17
```

```
byte message1 = 40;
byte message2 = 41;


void setup(){
  Serial.begin(9600);
  Serial.println("Start");
  delay(100);
}

void loop(){
  Serial.print(message1, BYTE);
  delay(500);
  Serial.print(message2, BYTE);
  delay(500);
}
```

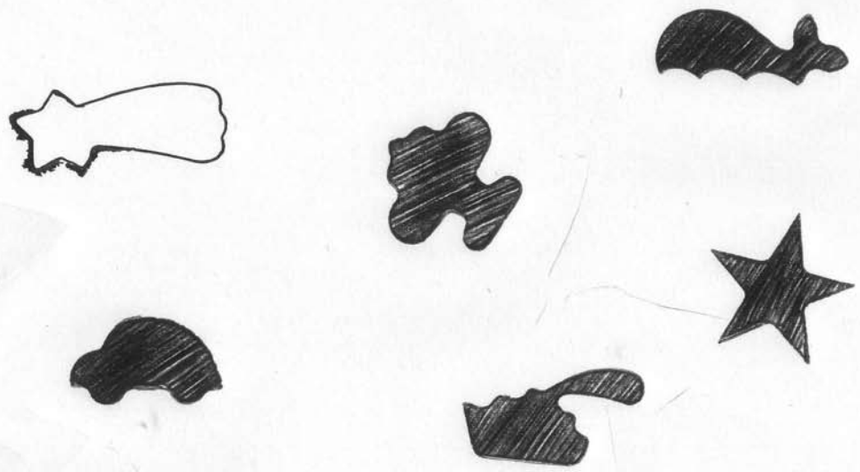
pySerialTest.py

```
ser = serial.Serial('/dev/tty.usbserial-FTDXTWB5', 9600)
# minimale konfiguration der seriellen schnittstelle fuer details siehe die dokumentation von pyserial
# auf mac laesst sich der namen der schnittstelle mit folgendem terminalbefehl herausfinden: ls /dev/tty.*
# auf pc sind die seriellen ports von 0 bis x numeriert. genaue angaben sind in der system konfiguration

ser.open()                                          # serielle kommunikation ini

pygame.init()
screen = pygame.display.set_mode((100, 100))        # fenster erzeugen
clock = pygame.time.Clock()                         # clock objekt erzeugen

while 1:
    clock.tick(33)                                  # max 33 fps

    for event in pygame.event.get():                # event queue durchlaufen
        if (event.type == pygame.QUIT) or (event.type == pygame.KEYDOWN):      # wird das fenster geschlossen o
            ser.close()                             # serielle schnittstelle frei geben
            sys.exit(0)                             # python skript beenden

    message = ord(ser.read(1))                      # ankommende ascii daten lesen
                                                    # mit der funktion ord() wird daraus ein int
    print 'message received'                                    # auf konsole ausgeben

    if message == 40:
        print 40
    elif message == 41:
        print 41

    print '----------------'
```

Problems | Console

<terminated> /Users/itz/Documents/workspace/retrogame/src/pySerialTest.py

```
40
----------------
message received
41
----------------
message received
40
----------------
message received
41
```



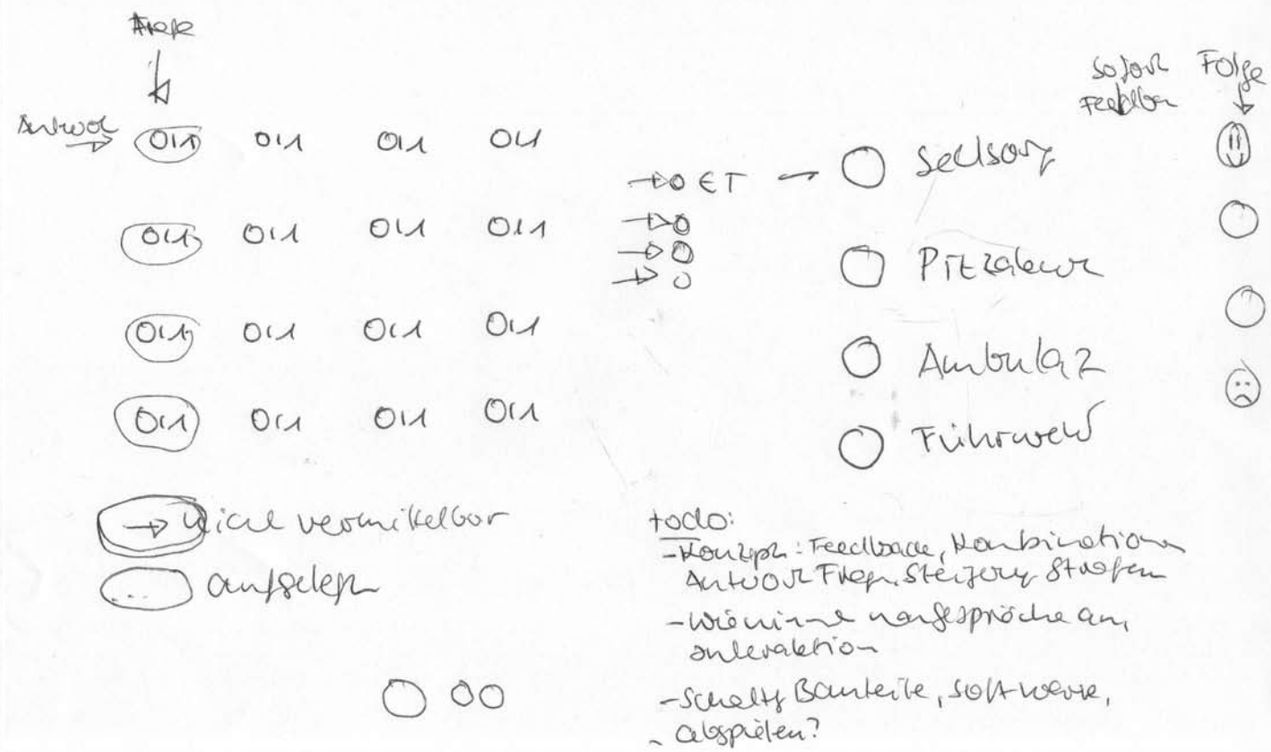
EIERKARTON UND SCHALTZENTRALE

IDEENSAMMLUNG

Telefonzentrale

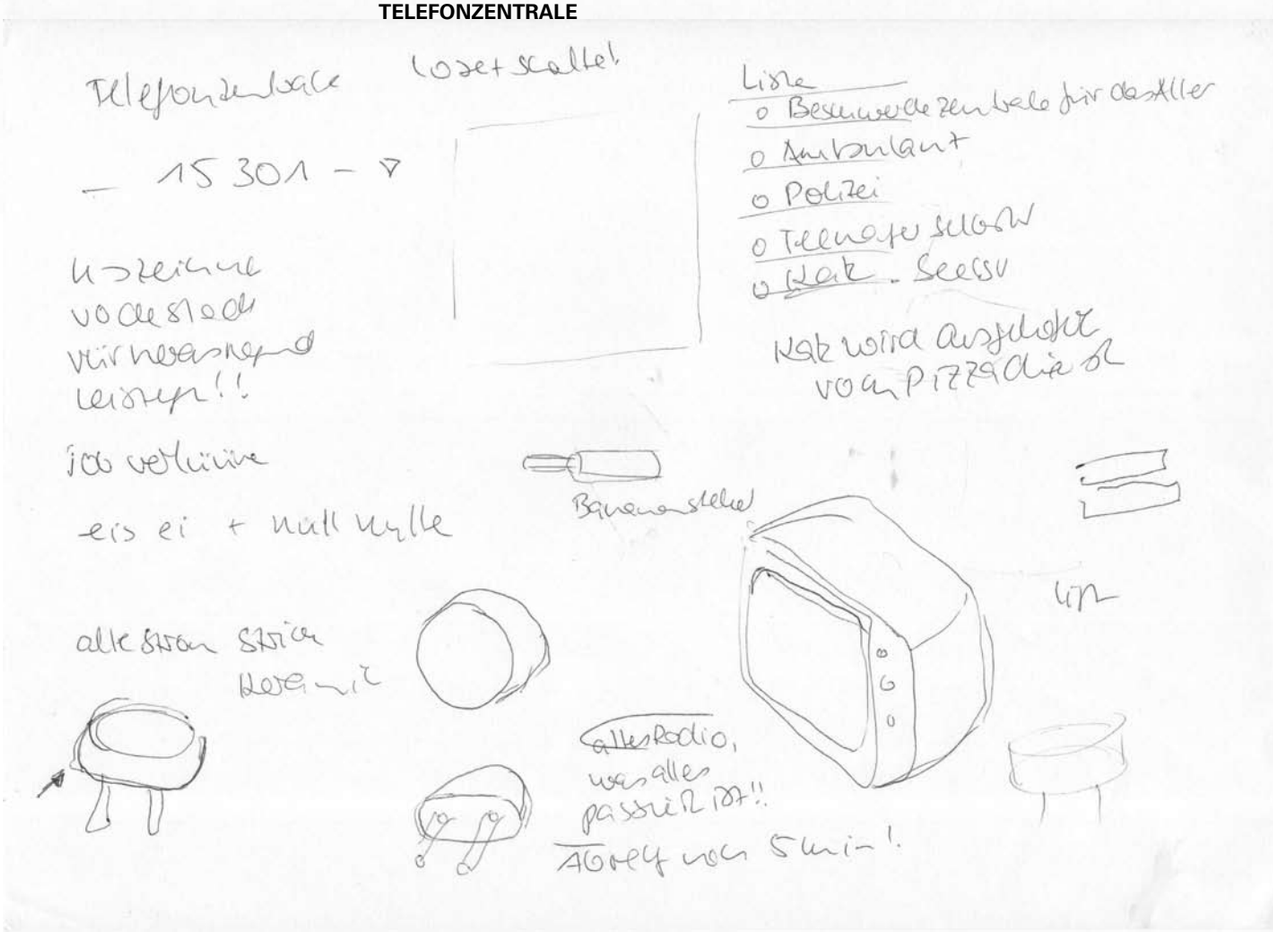
Mit Bananensteckern müssen ankommende Anrufe richtig verbunden werden. Wird ein Anruf falsch verbunden, hört man im Radio Katastrophennachrichten. Eingehende Anrufe werden mit einer blinkenden Lampe signalisiert. Die Anrufe können nur der Reihenfolge nach abgearbeitet, d.h. verbunden werden. Es gibt nicht für alle Anrufer auch den richtigen Ansprechpartner. Beispiel:Katze wird vom Pizzadienst ausgeliefert. Nach einigen Runden gibt es bei Erfolg eine Auszeichnung von der Stadt.

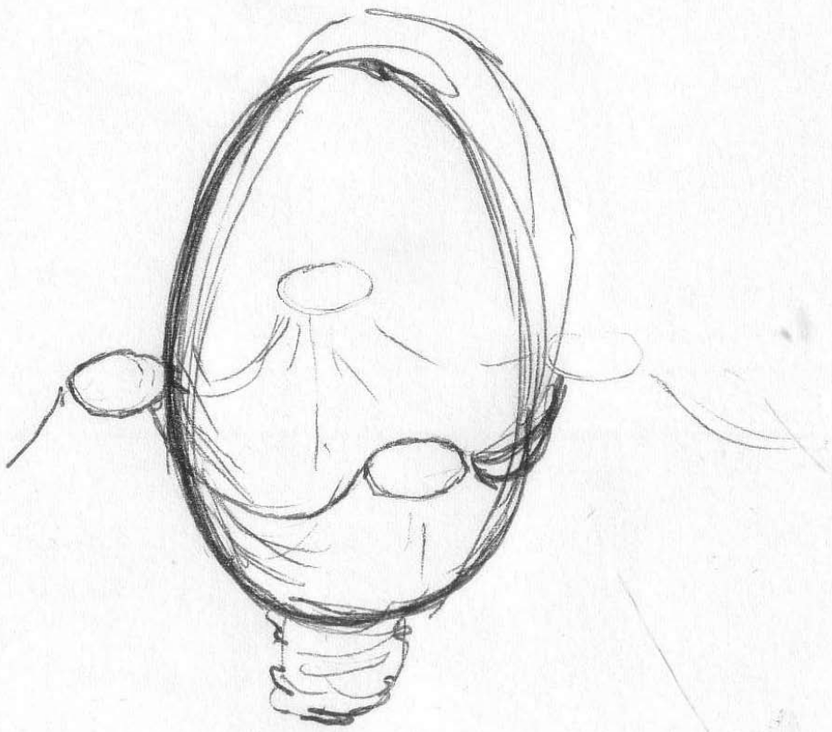
Anrufer	Ansprechpartner
Pizzabestellung	Pizzakurier
ET	Seelsorger
Katze vermisst	Beschwerdezentrale
...	....



IDEE 1

TELEFONZENTRALE





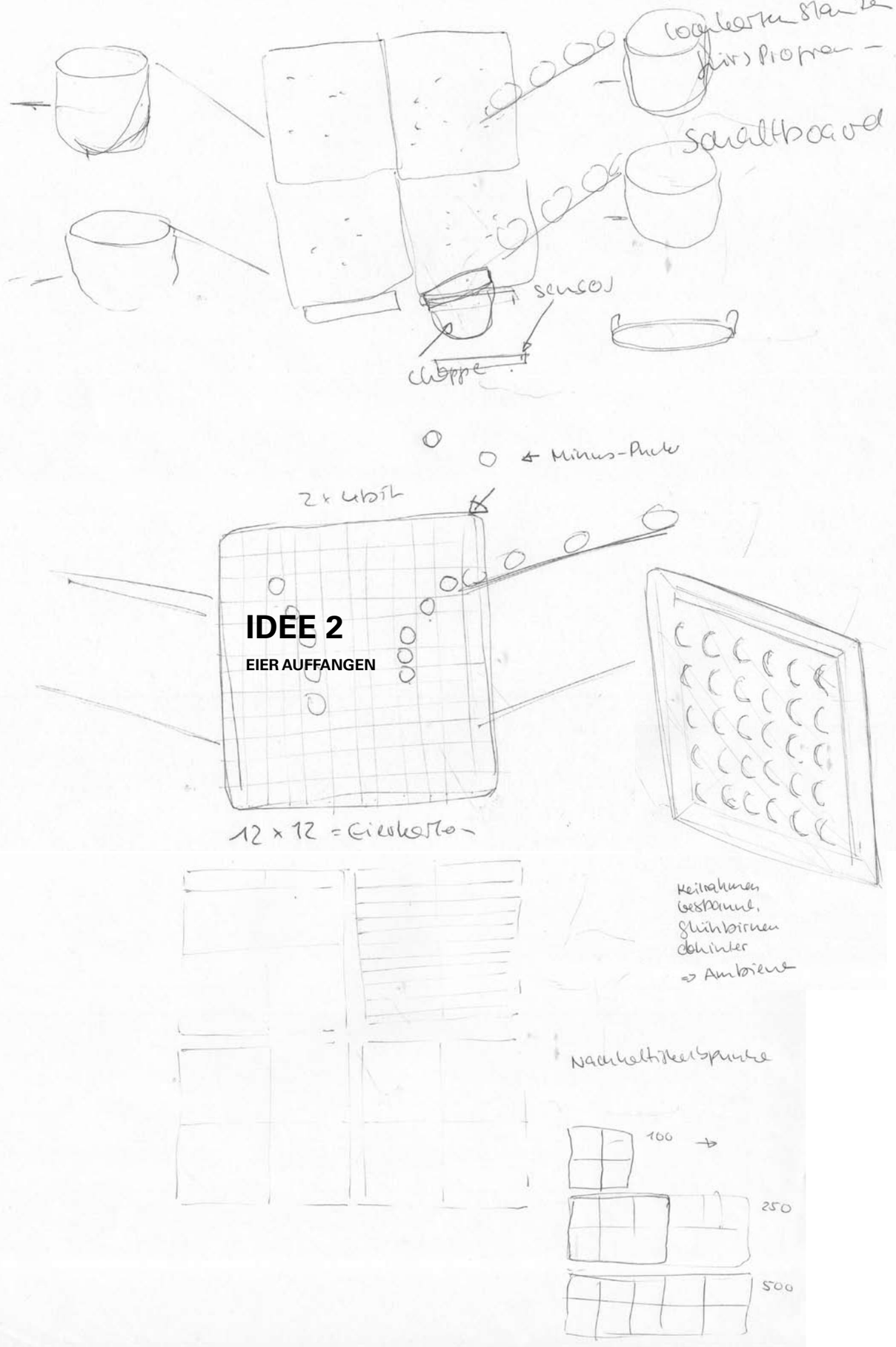
EIERKARTON UND SCHALTZENTRALE

IDEENSAMMLUNG

Eier auffangen

Ausgehend von dem bereits analysierten Egg-Spiel könnte man das Auffangen der Eier statt einem Wolf zu überlassen, selbst in die Hand nehmen. Mit einer Kappe (vgl. Idee Kappespiilie) werden die Eier aufgefangen. Heruntergefallene ergeben Minuspunkte. Der Wolf soll nicht die Funktion der Empfängnisverhütung behalten, das Spielziel soll weniger absurd werden. Statt Display könnte man einen Eierkarton nehmen !!! Auch der Eierkarton war so eine Idee, die immer wieder auftauchte. Glühbirnen in Eierform könnten in eine Eierkartonmatrix geschraubt werden ...

Die herunterfallenden Eier werden mit aufleuchtenden Glühbirnen symbolisiert. Belohnungsvariante: 4-er mit Bio-Eiern, 6-er mit Bodenhaltungseiern, 10-er mit Legebatterie-Eiern, Nachhaltigkeitspunkte ...



IDEE 2

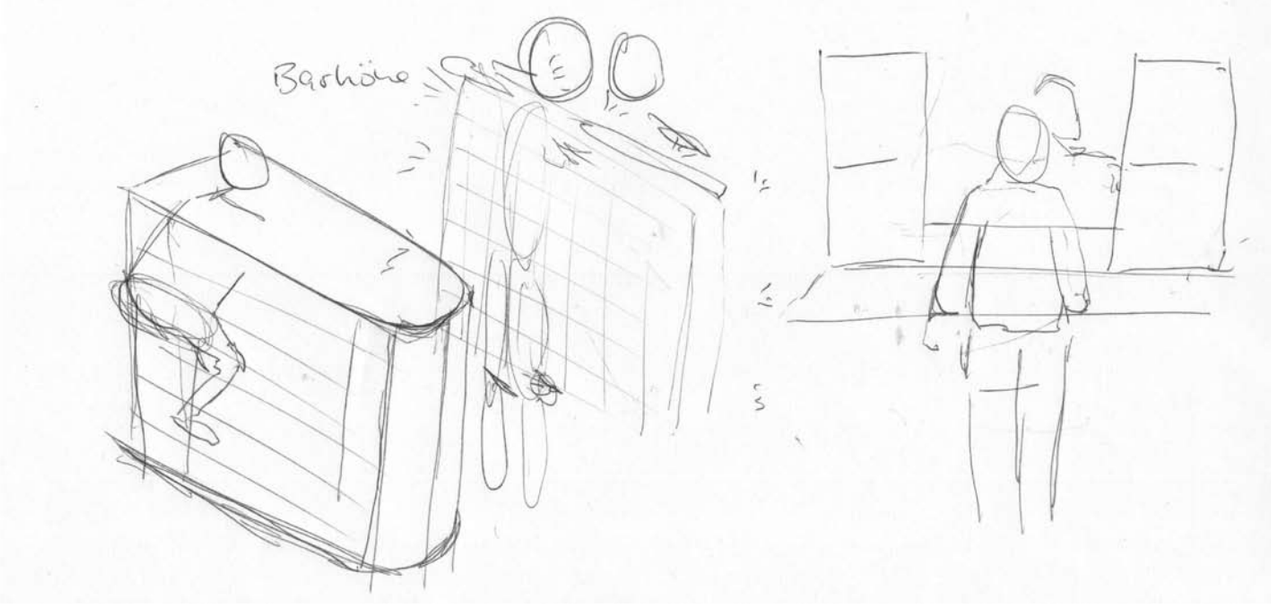
EIER AUFFANGEN

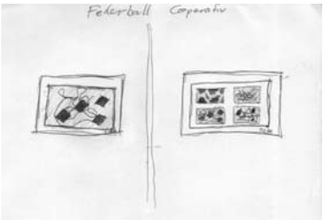
WEITERE SPIELIDEEN

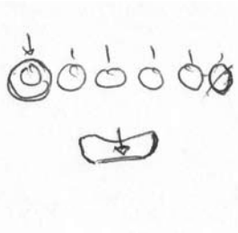
SCHALTERSITUATION KOOPERATIV

Schaltersituation kooperativ

Ursprungsidee war ein dreidimensionales Random-Bild, das der eine Spieler beschreiben muss. Der andere versucht, so schnell wie möglich zu entscheiden, welches der angebotenen Bilder gemeint sein könnten. Um den Bildschirm zu ersetzen, wurde die Graphik reduziert und mit einfacher Punktmatrix darstellbar gemacht. Ideen wären der bespannte Keilrahmen mit Glühbirnen dahiter, der ein angenehmes Ambient-Licht verstrahlt, oder die Bar mit Buttons, die das Kommunizieren fördert, oder jede schalterartige Situation, die es erschwert, sich gegenseitig in die Karten zu sehen. Eingaben könnten auch graduell erfolgen mit eher richtig, weniger richtig, und die Versuche werden gezählt.

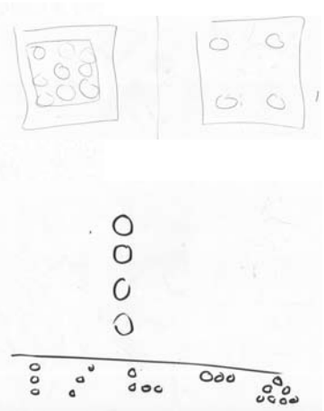


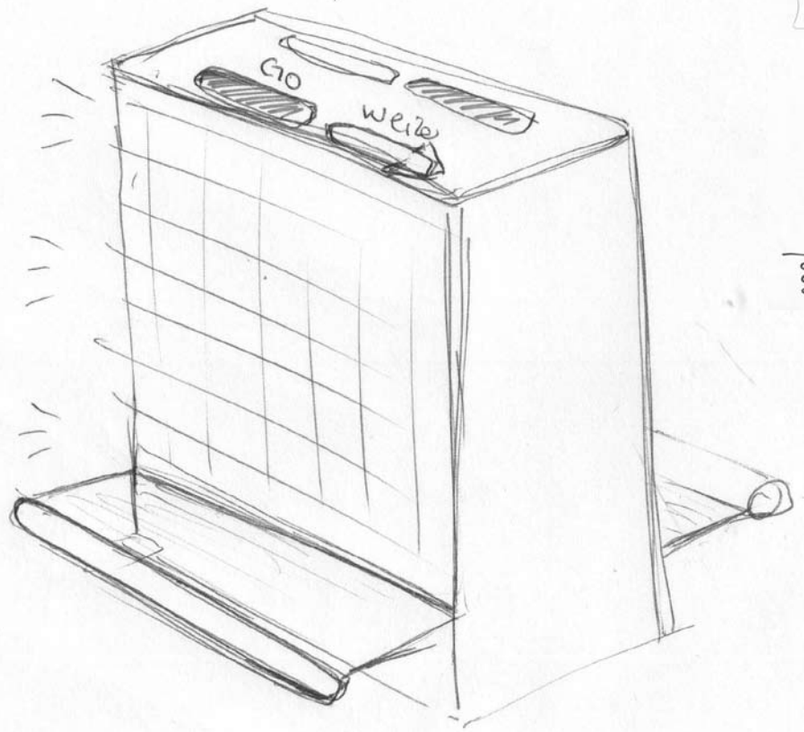




IDEE 3

SCHALTERSITUATION KOOPERATIV











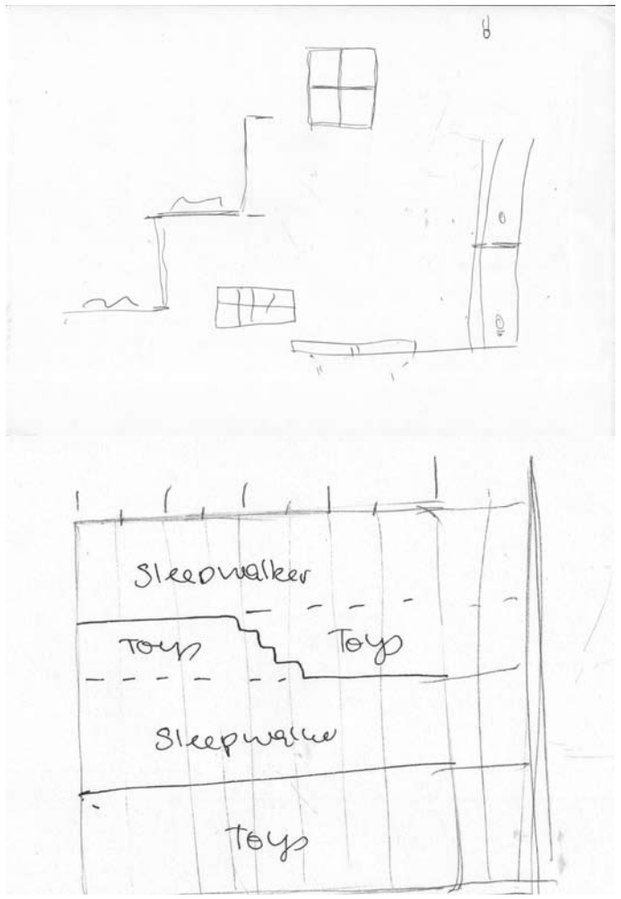
SPIELKONZEPT

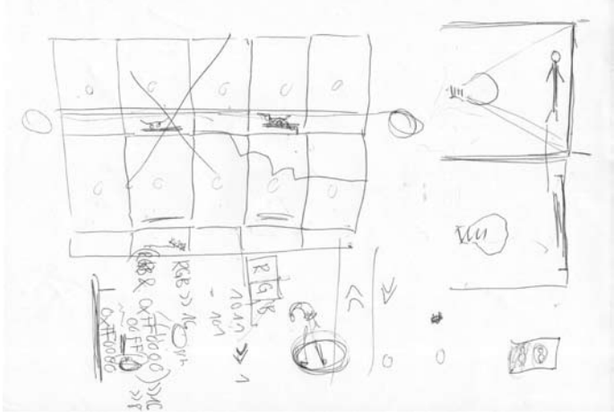
DER SLEEPWALKER

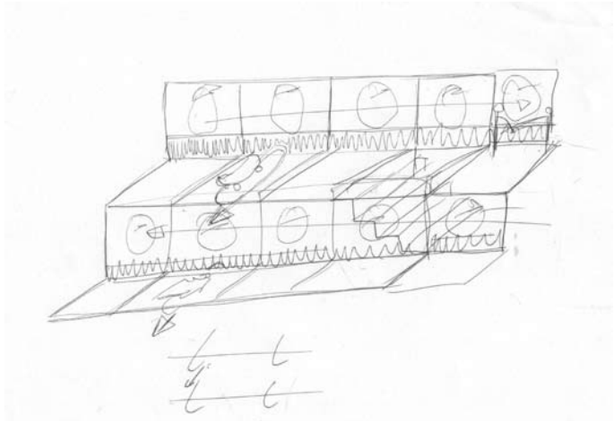
Der Sleepwalker

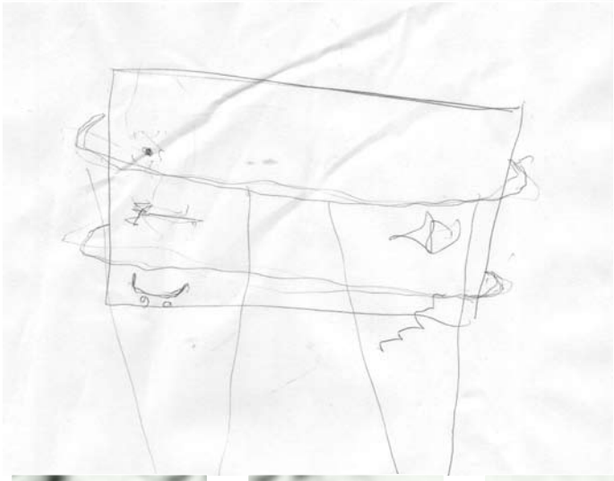
Das Sleepwalker-Spiel haben wir vor lauter Matrix und Eierkartons aus den Augen verloren. Charmant am Sleepwalker sind die Vektorgraphiken, die wir imitieren könnten, also weg von den Pixeln hin zu Schattenfiguren, die auf eine Leinwand projiziert werden.

Wie läuft der Sleepwalker? Von links nach rechts, immer ein Frame weiter. Das könnte man mit Schachteln nachempfinden, in denen Glühbirnen montiert werden, die mit Relais geschaltet werden. Der Sleepwalker läuft unten wieder zurück. So sind wir auf 2x5 Frames gekommen. Wie die Spielzeuge, die im Weg liegen, aus dem Weg geräumt werden können, ist die Frage. Man könnte sie mit einem Baustellenhelm mit Lampe wegleuchten. Durch das Licht verschwindet die Schattenprojektion. Oder Gegenstände wegnehmen.



















SPIELKONZEPT

DER SLEEPWALKER

Der Sleepwalker

Wir hatten verschiedene Lösungsansätze durchgespielt:

Jeder Sleepwalker eine Schachtel mit Glühbirne dahinter

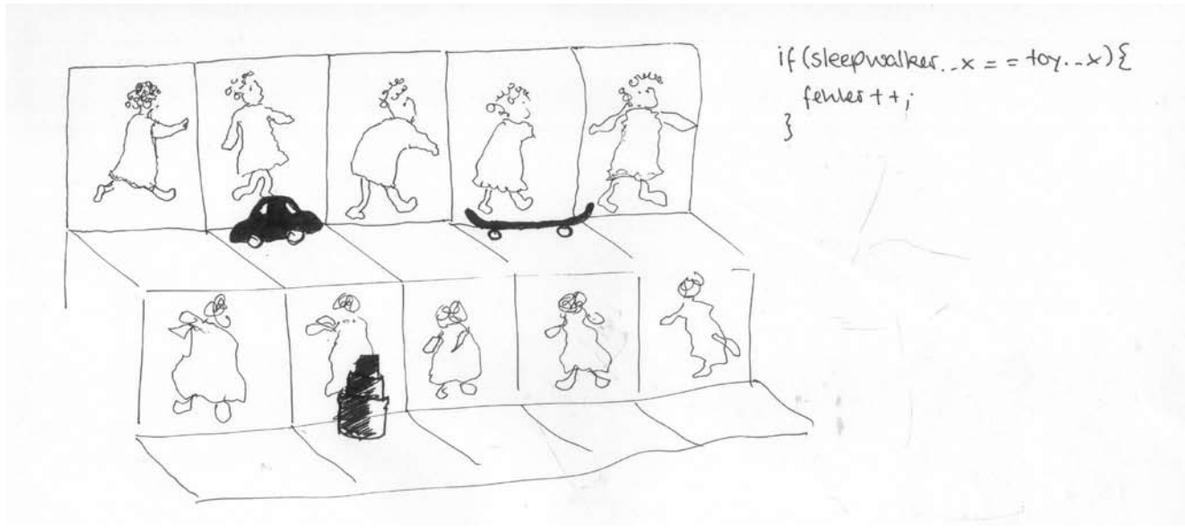
In einer Schachtel 2 Keyframes, die abwechselnd beleuchtet werden können und zu interessanten Schattenüberlagerungen führen.

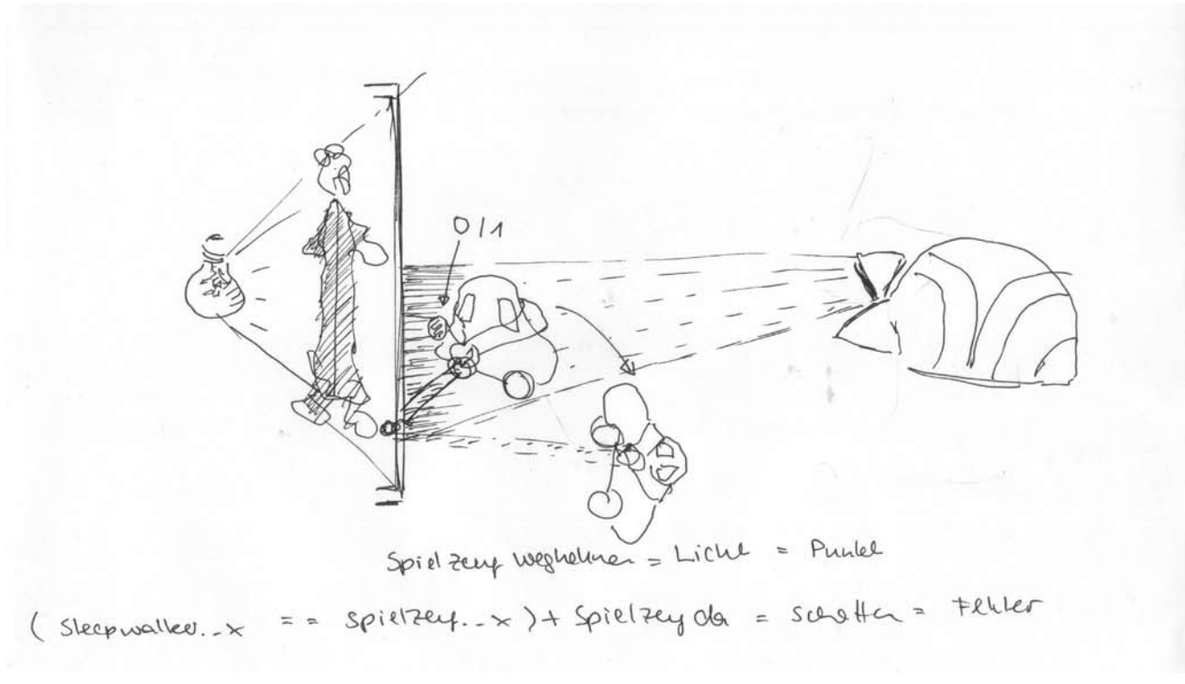
Die Spielzeuge als Schattenprojektionen, als reale Gegenstände, als gebastelte Gegenstände, ...

Die Spielzeuge zum Wegnehmen, zum Anleuchten mit Lichtsensoren, Leuchten mit Helmlampe oder Taschenlampe, ...

Die Sleepwalker in 2x5 Frames angeordnet, auf zwei Ebenen mit Stufen und zu guter Letzt als zimmerhoher Turm.

Zwischendurch sind wir auch bei Lichtinstallationen mit scheuen Tieren, die auf Bewegung reagieren gelandet und bei Fighter-Games mit Stein-Schere-Papier-Prinzip um auszuhandeln, wer das nächste Kästchen belegen darf.





ENDRESULTAT

DER SLEEPWALKER

Spielbeschreibung

Im Spiel „Sleepwalker“ bewegt sich eine Schlafwandler auf einem fixen Weg über den Bildschirm. Auf diesem Weg liegen vier Hinternisse im Weg, die den Schlafwandler zum umfallen bringen, falls sie nicht aus dem Weg geräumt werden. Der Spieler hat vier Knöpfe, mit denen er je ein Hinterniss aus dem weg räumen kann, solange er den entsprechenden Knopf gedrückt hält. Er kann aber nur ein Knopf aufs Mal drücken. Der Schlafwandler bewegt sich immer schneller, so dass die Knöpfe immer schneller nacheinander gedrückt werden müssen. Durch die simple Spielmechanik ist das Spiel sehr schnell erlernbar und durch den immer schneller werdenden Rhythmus wird der Spieler in eine Art Trance versetzt.

Unser Spiel vs Original

Wir haben die Spielmechanik vom Sleepwalker fast eins zu eins kopiert. Der grösste Unterschied besteht darin, dass es bei uns nur drei und nicht wie im Originalspiel vier Objekte (bzw Eingabemöglichkeiten) gibt. Da das Spiel auf ein total anderes Medium transportiert wurde, sind die Ein- und Ausgaben bei uns völlig anders: Die Eingabe passiert via Taschenlampe, mit der auf ein Objekt geleuchtet wird statt mit Knöpfen.

Da der Spieler nur eine Taschenlampe hat, kann er nur ein Objekt gleichzeitig anleuchten, dieser Teil der Spiellogik musste im Originalspiel in der Software abgehandelt werden. Wir haben ihn durch die Beschränkung auf eine Lampe pro Spieler mittels Hardware implementiert.

Diese Eingabemethode hat eine Änderung auf einer narrativen Ebene zur Folge: Wo beim Originalspiel Objekte durch das Drücken der entsprechenden Taste aus dem Weg geräumt wurden, werden sie durch das Anleuchten präsenter als vorher. Aus diesem Grund haben wir uns dafür entschieden, die Objekte nicht als Hinternisse darzustellen sondern als Sachen die die Spielfigur braucht, um ihren Weg fortzusetzen. So macht das Einblenden der Objekte wider Sinn, denn ohne diese kann die Spielfigur nicht weitergehen.

Die Ausgabe ist nicht so weit vom Original entfernt. Sie besteht aus Lichtern hinter einer Schablone und einer Projektionsfläche, während das Original mit entsprechend geformten lcd Elementen arbeitet.

TASCHENLAMPE

DER SLEEPWALKER

FLUGOBJEKTE

DER SLEEPWALKER

ANLEUCHTEN

DER SLEEPWALKER

SLEEPWALKER LÄUFT WEITER

DER SLEEPWALKER

FLUGI

DER SLEEPWALKER

RAKETE

DER SLEEPWALKER

LICHTSENSOR

DER SLEEPWALKER

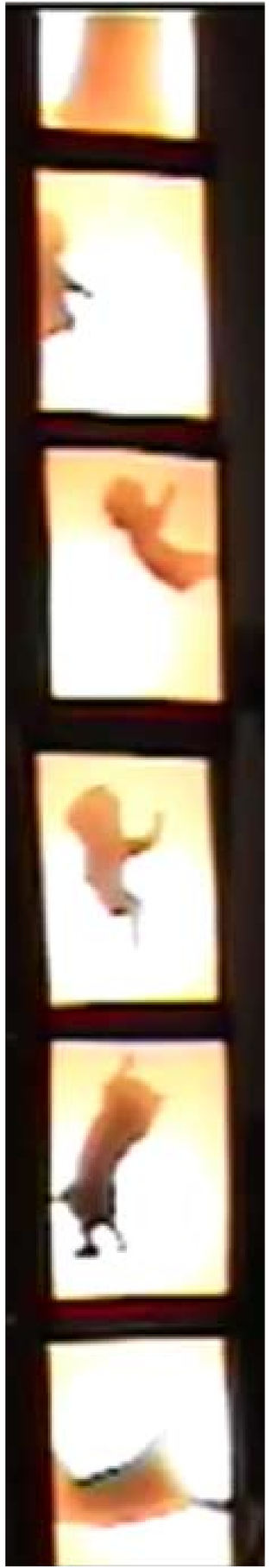
UFO

DER SLEEPWALKER

CODE

DER SLEEPWALKER

```
/**
 * Advanced Physical Computing
 * FS 2008
 *
 */

// used pins
/*
folgende Pins werden für den gebraucht um die Lichter zu steuern
der erste Pin im array ist das unterste Licht, der letzte das Oberste
*/
int pins[] = { 10, 11, 12, 14, 0, 1, 3, 4, 5, 6 }; // 10, 11, 12, 14
int numPins = 10;///anzahl der verwendeten ausgabe pins

// game state
/*
state = Zustand des programmes. folgende Zustände sind möglich:
0 = kalibrierung: die Sensorwerte werden ausgelesen um die Raumhelligkeit zu bestimmen
1 = warten auf spielstart (spielstart ist wenn ein sensor angeleuchtet wird).
2 = spielen bis der sensor wo die spielfigur ist nicht angeleuchtet wird, dann runterfall anima-
tion und in zustand 3 wechseln.
3 = spielfigur ist soeben runtergefallen, ein leben (lifes) wird abgezogen, wenn noch leben
vorhanden wird die Spielfigur mit resetWalkers() an die ausgangsposition und der zustand auf
2 gesetzt, sonst ist gameover und der zustand auf 1 gesetzt
*/
int state=0;
int lifes=3;///anzahl leben pro spiel

// walkers positions
int numWalkers = 1;///anzahl spielfiguren pro leben
int walkers[3];///positionen der spielfiguren (da jezt nur eine spielfigur pro leben vorhanden ist
wird nur walkers[0] gebraucht)

// sensor value storage
int numSensors = 3;///anzahl sensoren
int normalValues[3];///werte der sensoren bei normaler raumbelichtung (nicht angeleuchtet)
int normalizedSensorValues[3];///werte über die herausgefunden wird ob ein sensor angeleuchtet
wurde oder nicht, wenn der Wert kleiner ist als 1 dann ist er heller als normal, wenn er grösser ist
as 1 bekommt er weniger Licht
int sensorPositions[] = { 7, 2, 5 };///positionen der sensoren
int sensorValues[3];///ausgelesener Wert der sensoren

int speed = 1500;///verzögerung in einem zyklus, je kleiner desto schneller läuft das spiel (also
eigentlich ein delay)

void setup()
{
  //Serial.begin(9600);
  ///wird für serielles debugging gebraucht
  randomSeed(analogRead(1));///zufallswert wid duch ausgelesener sensorwert inizialisiert

  // setup output pin modes
  ///die output pins werden gesetzt
  for (int i=0; i<numPins; i++) {
    pinMode(pins[i], OUTPUT);
  }

  // initial game state (calibration)
  ///Zustand des programmes wird auf Kalibrierung gesetzt
  state = 0;

  resetWalkers();
}

void resetWalkers() {
  // initial position of walkers: all disabled (-1)
  ///setzt die position der spielfigur(walker[0]) auf -2 damit die spielfigur nicht gleich auf der position
0 erscheint wenn das spiel gestartet wird
```



```
  for (int i=0; i<numWalkers; i++) {
    walkers[i] = -2*(i+1);
  }
}

// simple animation for rewarding
void upDown() {
  int delayTime = 100; ///anzahl millisekunden die gewartet wird zwischen den animationsschrit-
ten

    // up
    ///schleiffe die die lampen von unten nach oben nacheinander aufleuchten lässt
    for (int i=0; i<numPins; i++) {
      digitalWrite(pins[i], HIGH);
      delay(delayTime);
      //digitalWrite(pins[(i-1) % numPins], LOW);
      digitalWrite(pins[i], LOW);
    }

    // down
    ///schleiffe die die lampen von oben nach unten nacheinander aufleuchten lässt
    for (int i=numPins-1; i>=0; i--) {
      digitalWrite(pins[i], HIGH);
      delay(delayTime);
      //digitalWrite(pins[(i-1) % numPins], LOW);
      digitalWrite(pins[i], LOW);
    }

}

/*
klaibrierung der sensoren, füllt den array normalValues[] mit werten die die sensoren bei normaler
beleuchtung haben
*/
void calibration() {
  delay(2000);///warten
  AllOn();///alle lampen einschalten
  delay(500);///warten

  // reset normal values
  ///alle normalValues auf 0 setzten
  for (int i=0; i<numSensors; i++) {
    normalValues[i] = 0;
  }

  // read sensor values when light on for calibration
  ///5 mal alle sensoren auslesen und durchschnittswert in normalValues speichern, dazwischen
etwas warten.
  ///so wird ein durchschnitt errechnet falls die raumbeleuchtung nicht konstant ist
  int count = 0;
  for (count = 0; count < 5; count++) {

    for (int i=1; i<=numSensors; i++) {
      ///folgender code ist gleich
      ///normalValues[i-1] = (analogRead(i) + normalValues[i-1]) /2;
      ///wobei sichergesttellt ist das immer abgerundet wird
      ///so wird ein durchschnitt errrechnet
      normalValues[i-1] = (analogRead(i) + normalValues[i-1]) >> 1;
    }
    delay(150);
  }

/* serielle ausgaben für debugging
  //Serial.print(„NORMAL VALUES=[„);
  for (int i=0; i<numSensors; i++) {
    //Serial.print(normalValues[i]);
    if (i<numSensors-1) {
      //Serial.print(„,“);
    }
  }
  //Serial.print(„]“);
  //Serial.println();
*/

  // signal that we’re ready!///alles blinken lassen um zu zeigen das die kalibrierung beendet ist
```

```
  AllOff();
  delay(1000);
  AllOn();
  delay(4000);
  AllOff();
}

// turn all lights on ///alle lichter anschalten
void AllOn() {
  for (int i=0; i<numPins; i++) {
    digitalWrite(pins[i], HIGH);
  }
}

// turn all lights off  ///alle lichter ausschalten
void AllOff() {
  for (int i=0; i<numPins; i++) {
    digitalWrite(pins[i], LOW);
  }
}

// write sensor values to serial out for monitoring
///serielle ausgaben für debugging
void printSensorValues() {
  Serial.print(„ANALOG IN=“);
  for (int i=0; i<numSensors; i++) {
    Serial.print(sensorValues[i]);
    if (i<numSensors-1) {
      Serial.print(„,“);
    }
  }
  Serial.println();
}

///animation um den erfolgreichne spieler zu belohnen
///bonbons für die augen ;)
///der speed wirt hir auch etwas verkleinert um die nächste runde etwas schwieriger (schneller)
zu machen
void successAnimation() {
  AllOff();
  for (int i=0; i<numPins; i++) {
    digitalWrite(pins[i], HIGH);
    delay(180);
  }
  delay(500);
  for (int i=0; i<numPins; i++) {
    digitalWrite(pins[i], LOW);
    delay(180);
  }

  speed = round(speed*0.8);
}

// step upwards
void moveUp() {
  for (int i=0; i<numWalkers; i++) {
    ///lampe an der position der spielfigur ausschalten
    if (walkers[i] >= 0) {
      digitalWrite(pins[walkers[i]], LOW);
    }

    // move it up ///position der spielfigur um eins erhöhen
    walkers[i] += 1;

    // enable walker ///lampe an der position der neuen spielfigur einschalten
    if (walkers[i] >= 0) {
      digitalWrite(pins[walkers[i]], HIGH);
    }

    // disable walker
    ///wenn die spielfigur die oberste position erreicht hat wird der spieler belohnt
     ///und die spielfigur wird unter auf -3 gesetzt damit sie nach 3 zyklen wider ganz unten er-
scheint
    if (walkers[i] >= numPins) {
      successAnimation();
      walkers[i] = -3;
```

```
    }
  }

}

///gibt true zurück wenn ein sensor angeleuchtet wird und false wenn er nicht angeleuchtet wird
boolean isActive(int sensorNum) {
  return ( normalizedSensorValues[sensorNum] < 0.7 ); ///0.7 wird als schwellwert verwendet. die lampe mit der die sensoren angeleuchtet werden muss so hell sein das normalizedSensorValues kleiner als 0.7 ist
}

///wenn sich die spielfigur auf der position eines sensores befindet der nicht angeleuchtet wird gibt diese funktion die entsprechende position zurück
///sonst wird -1 zurückgegeben
int checkCollision() {
  // get active sensors

  for (int i=0; i<numWalkers; i++) {
for (int j=0; j<numSensors; j++) {
  if (sensorPositions[j] == walkers[i]) {
    // walker i is on position for sensor j
    if (isActive(j)) { /*
      Serial.print(„yeah, you saved her on position „);
      Serial.print(walkers[i]);
      Serial.print(„, sensor „);
      Serial.print(j);
      Serial.println();
      */
    } else {
      /*
      Serial.print(„oh oh... she will fall down „);
      Serial.print(walkers[i]);
      Serial.print(„, sensor „);
      Serial.print(j);
      Serial.println();
      */
      return sensorPositions[j];
        }
      }
    }

  }

  return -1;
}


///animation fürs runterfallen der spielfigur (ein visueller adrenalin-kick)
void fallDown(int pos) {
  AllOff();

  for (int i=0; i<10; i++) {
    digitalWrite(pins[pos], HIGH);
    delay(250);
    digitalWrite(pins[pos], LOW);
    delay(250);
  }
}


///die hauptschleiffe des programmes
void loop()
{
  // STATE 0: calibration phase, after turning on
  ///zustand 0: kalibrierung
  if (state == 0) {
    calibration();
    state = 1; ///zustand auf 1 setzen: warten auf spielstart
  }


  // read sensor values from analog in
  ///sensoren auslesen und den normalizedSensorValues array mit werten füllen
  ///wenn normalizedSensorValues[sensorposition] == 1 dann ist der sensor genau auf der während der kalibrierung gemessenen raumbelichtung
```

```
 ///je nähre bei 0 der wert ist desto warscheinlicher ist er angeleuchtet. wenn er über 1 ist ist erhält
der sensor weniger licht als die während der kalibrierung berechnete raumbeleuchtung
 for (int i=0; i<numSensors; i++) {
   sensorValues[i] = analogRead(i+1);
   if (normalValues[i] > 0) {
     normalizedSensorValues[i] = sensorValues[i] / normalValues[i];
   } else {
     normalizedSensorValues[i] = 0;
   }
 }

 // STATE 1: waiting for game to start
 ///zustand 1: warten auf spielstart
 ///spiel wird gestartet wenn einer der sensoren angeleuchtet wird
 if (state == 1) {
   AllOff();
   if (normalizedSensorValues[0] < 0.6 ||
       normalizedSensorValues[1] < 0.6 ||
       normalizedSensorValues[2] < 0.6) {
         // start game
         state = 2;///ein sensor wurde angeleuchtet, zustand auf 2 setzen: spielen
         //Serial.println(„---> START GAME! <---“);
       }
 }

 else
 ///zustand 2: spielen
 if (state == 2) {
    moveUp();///position des spielers um eins nach oben setzen, entsprechende lampen aus / ein
schalten

     int pos = checkCollision();///prüft ob die spielfigur an einer sensorposition ist die nicht ange-
leuchtet wird
   if (pos >= 0) {///die spielfigur ist an einer position wo ein sensor ist der nicht angeleuchtet wurde,
pos ist diese position
      // walker collided with object: life is lost
      //Serial.println(„life lost on position „);
      //Serial.print(pos);
      //Serial.println(„: falling down...“);

      // do falling down here...
      fallDown(pos);///runterfallanimation mit start von der position wo deie spielfigur war
      state = 3;///status auf 3 setzen
    }
    delay(speed);///verzögerung des spieles, je kleiner speed desto schneller läuft das spiel
}
else
///zustand 3: spielfigur ist auf dem boden aufgeschlagen
if (state == 3) {
   lifes--;///leben wird um eins reduziert
   state=2;///status wird auf 2 gesetzt: spielen
   resetWalkers();///positionen der spielfiguren wird neu gesetzt

   if (lifes <= 0) {///wenn keine leben mehr vorhanden sind
      state = 1;///zustand auf 1 setzen: warten auf spielstart
      lifes = 3;///leben auf 3 setzen
    }
   }
}
```


NYC