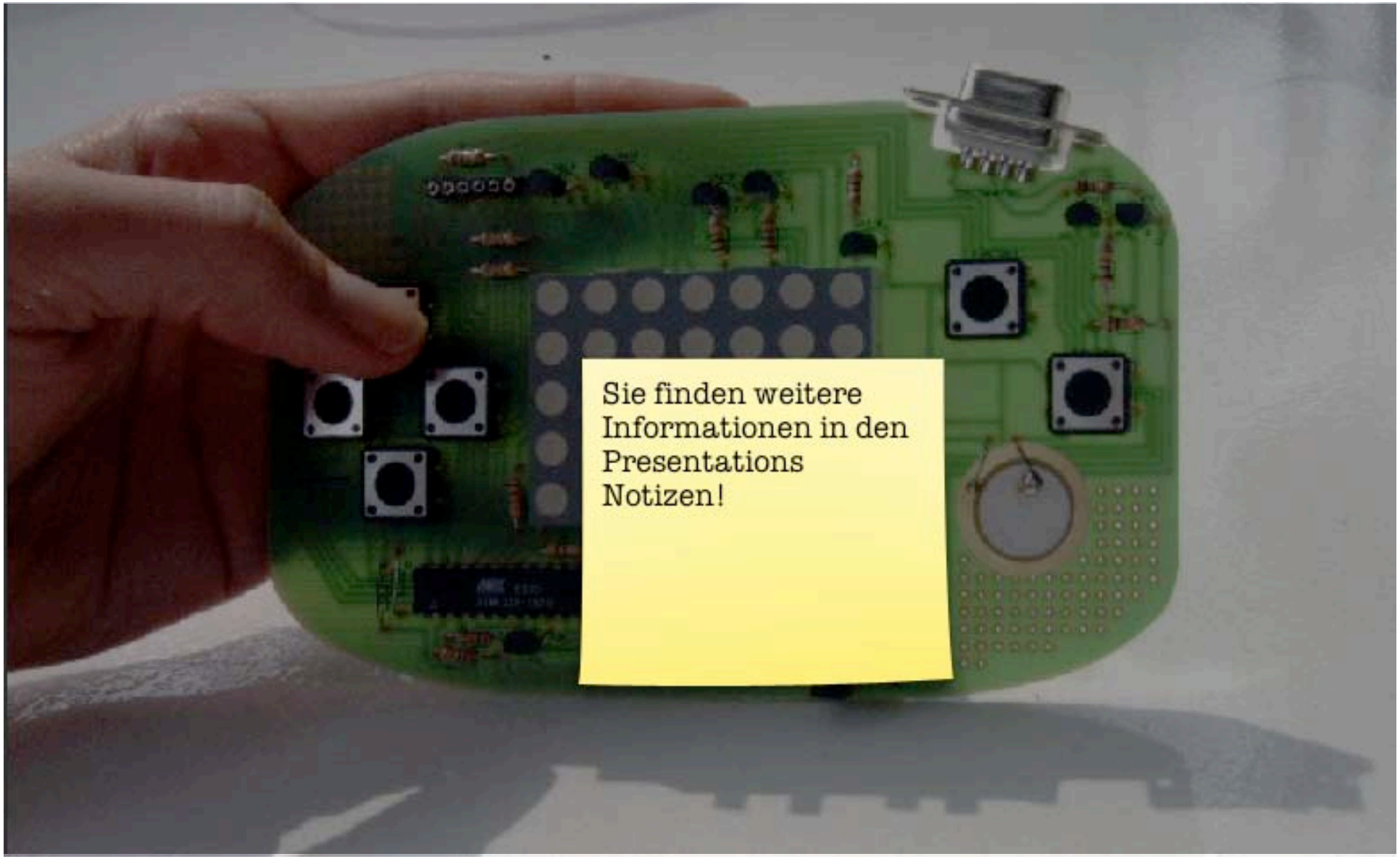
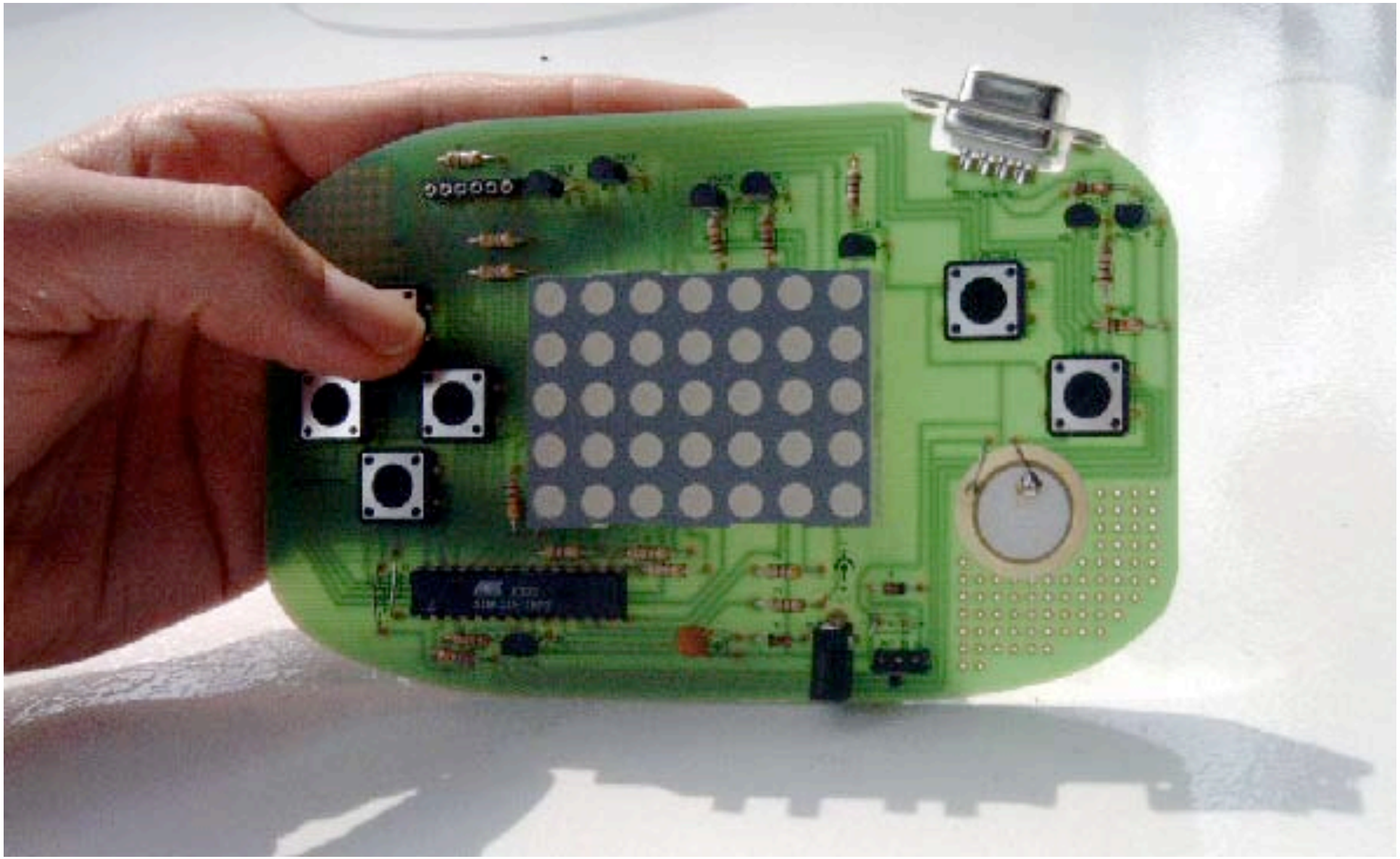


Mignon Game Kit Workshop

Programmieren



Mignon Game Kit Workshop

Programmieren

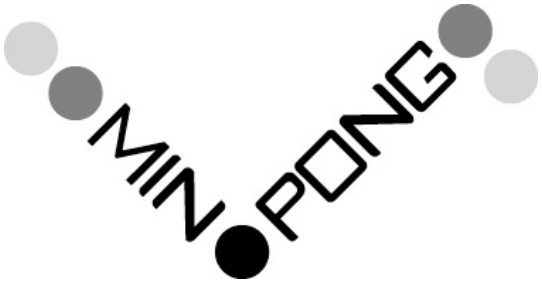
www.ralphbaer.com Ralph Baer

http://timeline.computerspielemuseum.de

Game History

man

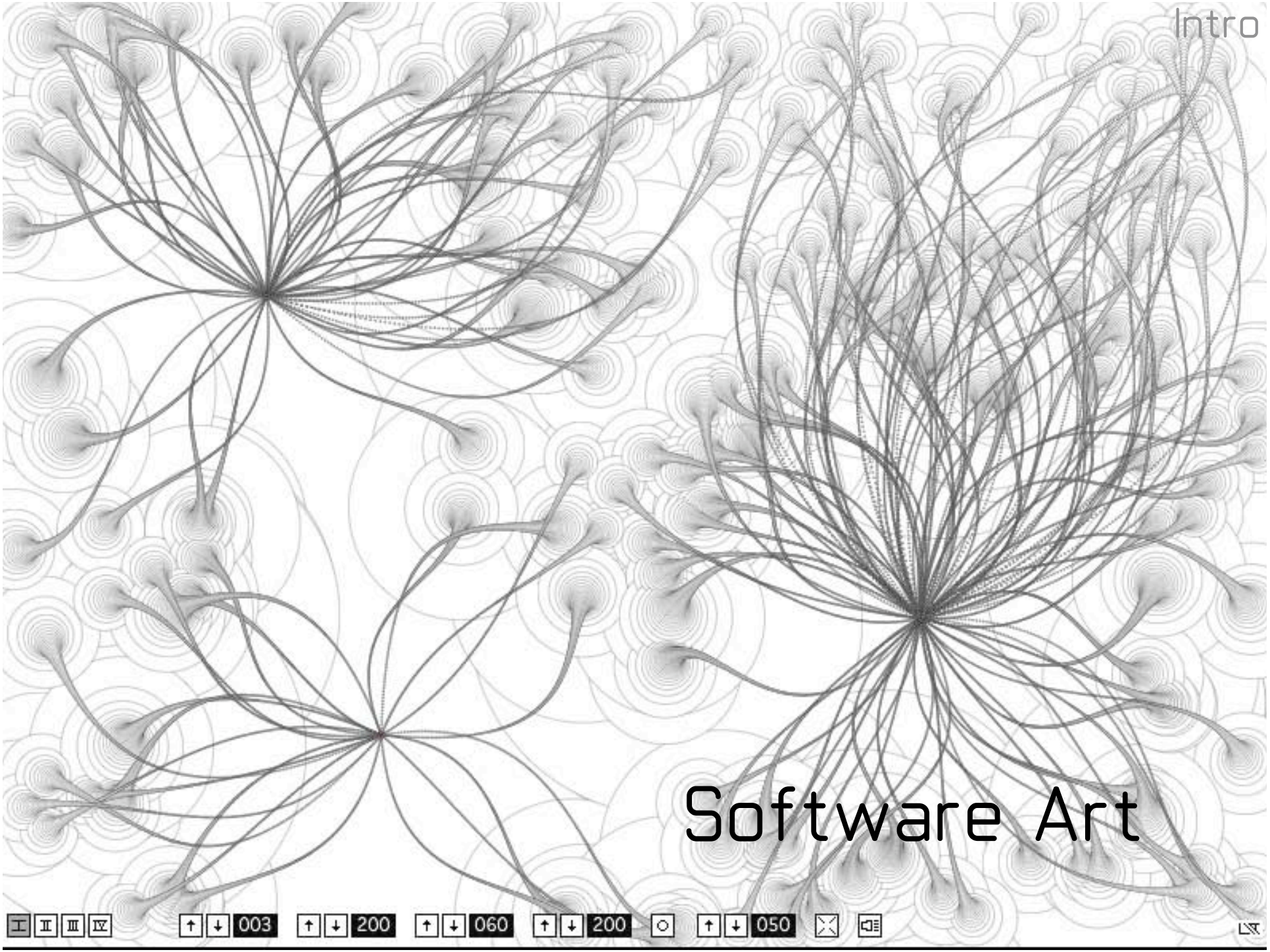
MAZE DIVER


MIN PONG

GAME

Freie Software

Software Art

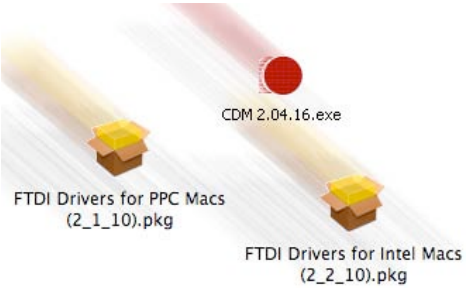


Processing

www.processing.org

Weitere Open Source Projekte im kreativen Bereich

Installation



1. FTDI-Treiber installieren

Damit MigProg sich mit Ihrem Computer verbinden lässt muss zuvor ein Treiber der Firma FTDI installiert werden.

Windows:
Du findest den Treiber auf der FTDI-Homepage bei “Drivers” und dann unter “VCP” (Virtual COM Port Drivers) dort gibt es im ersten Abschnitt rechts auch eine Exe-Datei, die sich besonders einfach installieren lässt. “Also available as a setup executable” Die Exe-Datei dann einfach per Doppelklick starten und den Anweisungen folgen. www.ftdichip.com

Mac:
Der Arduino-Download enthält bereits die FTDI-Treiber. Wähle den geeigneten Ordner je nach dem welchen Prozessor dein Computer hat PPC oder Intel. Sie finden diese Information im Apfelmenü des Finders bei “Über diesen Mac”. Die Installation einfach per Doppelklick starten!

2. Arduino installieren

1. Lade die Arduino Software für Dein System per Download auf deinen Rechner. Das kann Windows, Mac oder Linux sein. www.arduino.cc

- Endpacke die Software und lege den Ordner zu Deinen anderen Programmordnern.

- Es bietet sich an das Icon der Arduino Software in die Schnellstartleiste zu ziehen, um das Programm zukünftig von dort starten zu können.

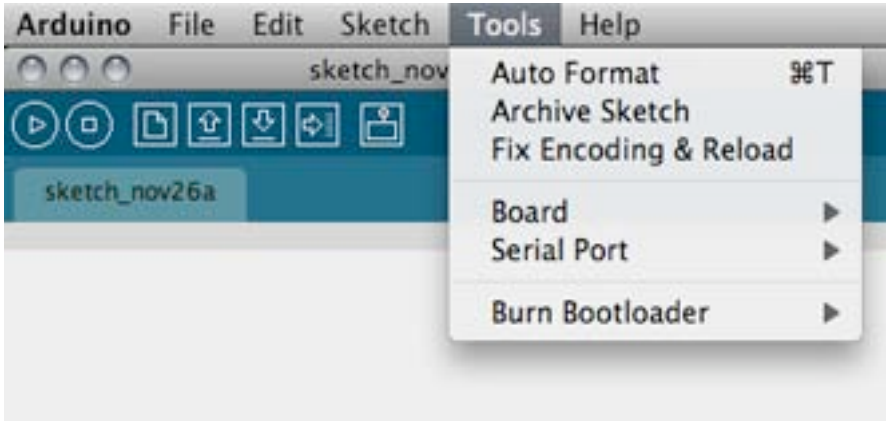
- Das Mignon Game Kit ist aktuell mit der Version Arduino 0017 getestet. Sollte es bei neueren Versionen zu Fehlern kommen, verwende bitte die getestet Version. Ältere Versionen der Software befinden sich auf der Download-Seite unter “Previous IDE Versions”.

3. Einbinden der Game-Kit-Bibliothek

- Lade die Game-Kit-Library von der Mignon Game Kit Homepage und entpacke die Datei, so das Du den Ordner “gamekit” erhältst.

- Starte die Arduino Software und beende sie dann gleich wieder. Das Programm erstellt beim ersten Start eine Ordner mit dem Namen “Arduino” im “Eigene Dateien” bzw. “Dokumente” Ordner des System. Dieser “Arduino”-Ordner ist der vorgesehen Ort für alle Programme die auf das Game Kit hochgeladen werden. Und darin werden auch zusätzliche Libraries installiert.

- Hierzu muss einfach ein neuer Ordner mit dem Namen “libraries” erstellt werden, in den der Game Kit Library-Ordner geschoben wird.

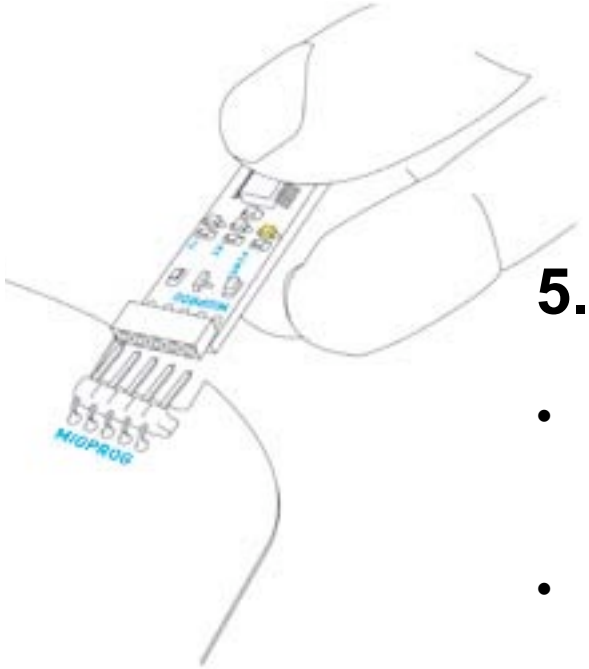


4. Board anwählen

Vor dem ersten Hochladen muss man noch bei der Arduino Software anwählen, zu welchem Arduino Board das Game Kit kompatibel ist.

Diese Einstellungen wird unter dem Menüpunkt “Tools” vorgenommen.

Das Mignon Game Kit ist immer kompatibel zu den normalen Arduino Boards. Diese unterscheiden sich durch die eingebauten Mikrocontroller. Bis jetzt gibt es ATmega8, ATmega168, und den aktuellen ATmega328. Bei guter Beleuchtung kannst Du die Typenbezeichnung auf Deinem Mikrocontroller lesen.

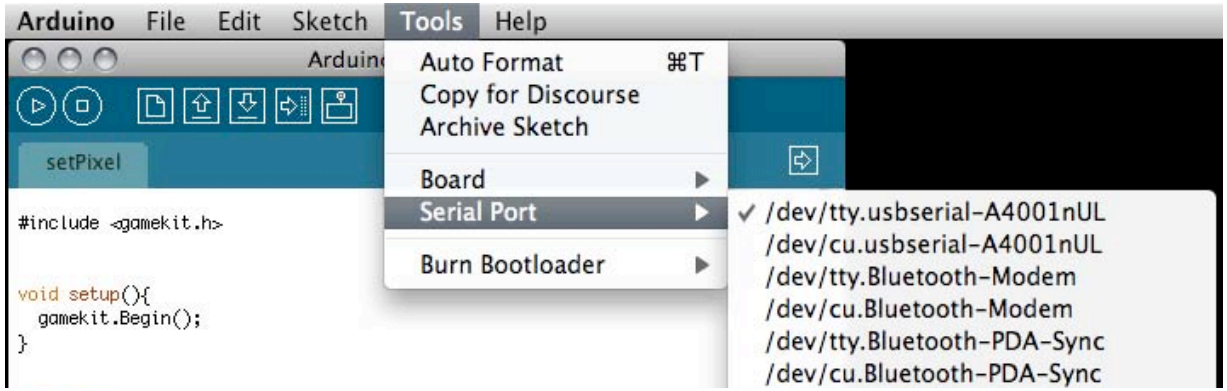


5. Verkabeln

- MigProg wir mit einem normalen USB-Kabel an deinen Rechner angeschlossen.

- Die rote Power LED des MigProg sollte jetzt leuchten. (Da das Game Kit mehr Strom benötigt als kleinere USB Geräte, „fragt“ MigProg den Computer ob es sich die benötigten 5 Volt 100mA beziehen „darf“. Erst nach der Zustimmung der Systen-Software leuchtet die rote Power LED des MigProg)

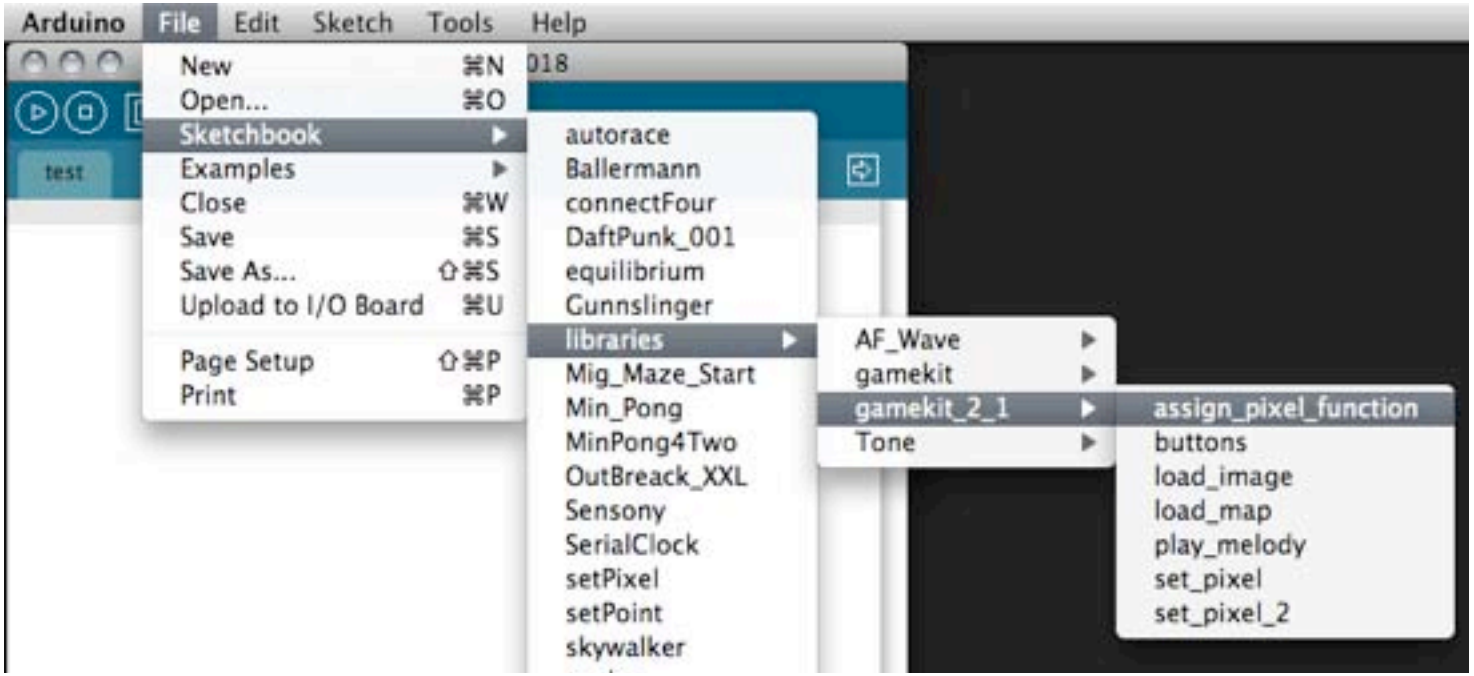
1. Wenn Du jetzt MigProg an der MigProg-Buchse des Game Kit anschließt, sollten die beiden anderen LEDs der Kommunikationsleitungen Tx und Rx kurz klinken.



6. Serial Port wählen

Windows:
Im “Geräte Manager” findest Du unter “Anschlüsse” hinter dem Eintrag “USB Serial Port” in Klammern an welchem Serial Port das MigProg angeschlossen ist. Zu dem “Geräte Manager” galangt man zum Beispiel mit einem Rechten-Maus-Klick auf “Arbeitsplatz” über den Menüpunkt “Verwalten”.

Mac:
Wähle einfach immer den obersten Eintrag “dev/tty/usbserial-…”



7. Load Example

Lade das erste Beispiel „load_image“ in die Arduino Software, das Du unter „Sketchbook / libraries / gamekit“ findest!

7. Upload to Game Kit

Klicke auf die „Upload“ Taste (Pfeil nach rechts), die TX und RX LEDs des MigProg müssten jetzt blinken, wenn die Daten zum Game Kit übertragen werden. Zusätzlich hörst Du den Datenstrom über den Lautsprecher des Game Kits.

Das „load_image“ Beispiel zeigt einen Smilie!

Kommentare

Mit Kommentaren machen wir unseren Programmcode für Andere verständlich. Damit der Computer weis, dass er die Kommentare nicht berücksichtigen soll, werden sie mit „//“ oder „/*“ und „*/“ gekennzeichnet.

```
// Kommentar
```

```
/*
Langer Kommentar
über mehrere Zeilen
*/
```

; semicolon

Jede Progammzeile endet mit einem Semikolon!

Das gild für viele Programmiersprachen wie C, Java usw. aber nicht für alle.

```
Befehl(Wert);

x = 1;
```

Data Types

- boolean
- char
- byte
- int
- unsigned int
- long
- unsigned long
- float
- double
- string
- array
- void

Variablen

„Eine Variable ist ein Speicherplatz“

Bevor eine Variable verwendet werden kann muss man diese initialisieren. Das heist die Variable wird angemeldet, in dem man ihr einen **Namen**, **Datenformat** und evt. auch einen **Wert** zuweist.

int
Integer (Ganze Zahlen, keine Komma Zahlen)
Bei Arduino von -32768 bis +32768.

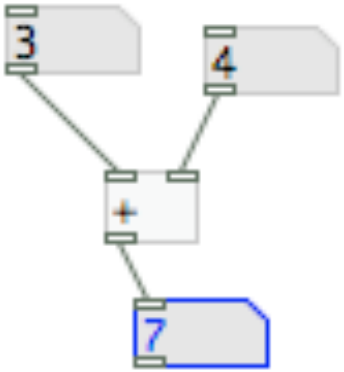
```
int x;
```

```
int x = 5;
```

Funktion

Zum Programmieren schreibt man Befehle, die der Computer erkennen kann untereinander.

Ein wichtiges Element ist dabei die so genannte Funktion.




```
x = plus(3,4);
```

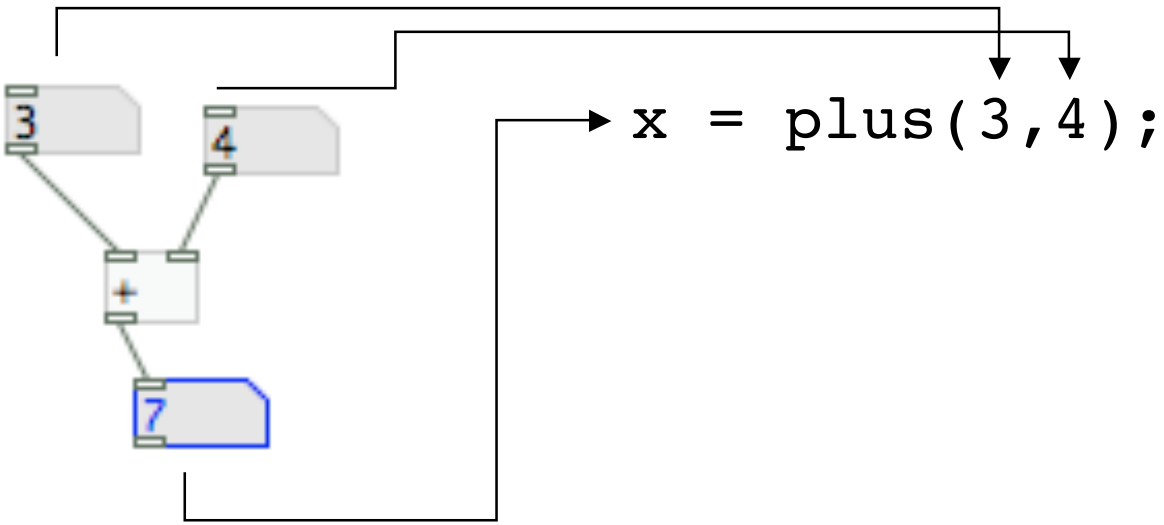
Pure Data

c++

Funktion

Zum Programmieren schreibt man Befehle, die der Computer erkennen kann untereinander.

Ein wichtiges Element ist dabei die so genannte Funktion.



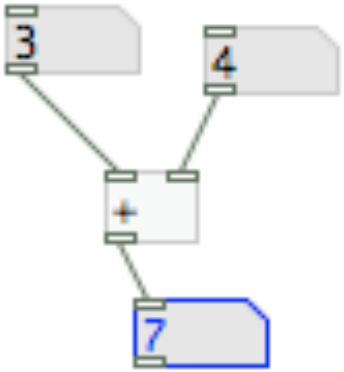
Pure Data

c++

Funktion

Zum Programmieren schreibt man Befehle, die der Computer erkennen kann untereinander.

Ein wichtiges Element ist dabei die so genannte Funktion.



Pure Data

```
x = plus(3,4);

int plus(int a, int b){
  int c = a + b;
  return c;
}
```

c++

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return

Klammern, Syntax, Einrückungsstils

Die verschiedenen Klammertypen sind ein wichtiges Elemt in der logischen Struktur der Programmiersprachen. Das Einrücken von ineinader verschachtelten Bereichen gestaltet den Code übersichtlicher.

Mac:
() Runde Klammer SHIFT + 8 und 9
{} Geschweifte Klammer ALT + 8 und 9
[] Eckige Klammer ALT + 5 und 6

```
digitalWrite(ledPin, HIGH);

if(x > 5){
  y = y + 1;
  x = 0
}

int myList[] = {12, 7, 15, 4, 1, 8}
```

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return

Klammern, Syntax, Einrückungsstils

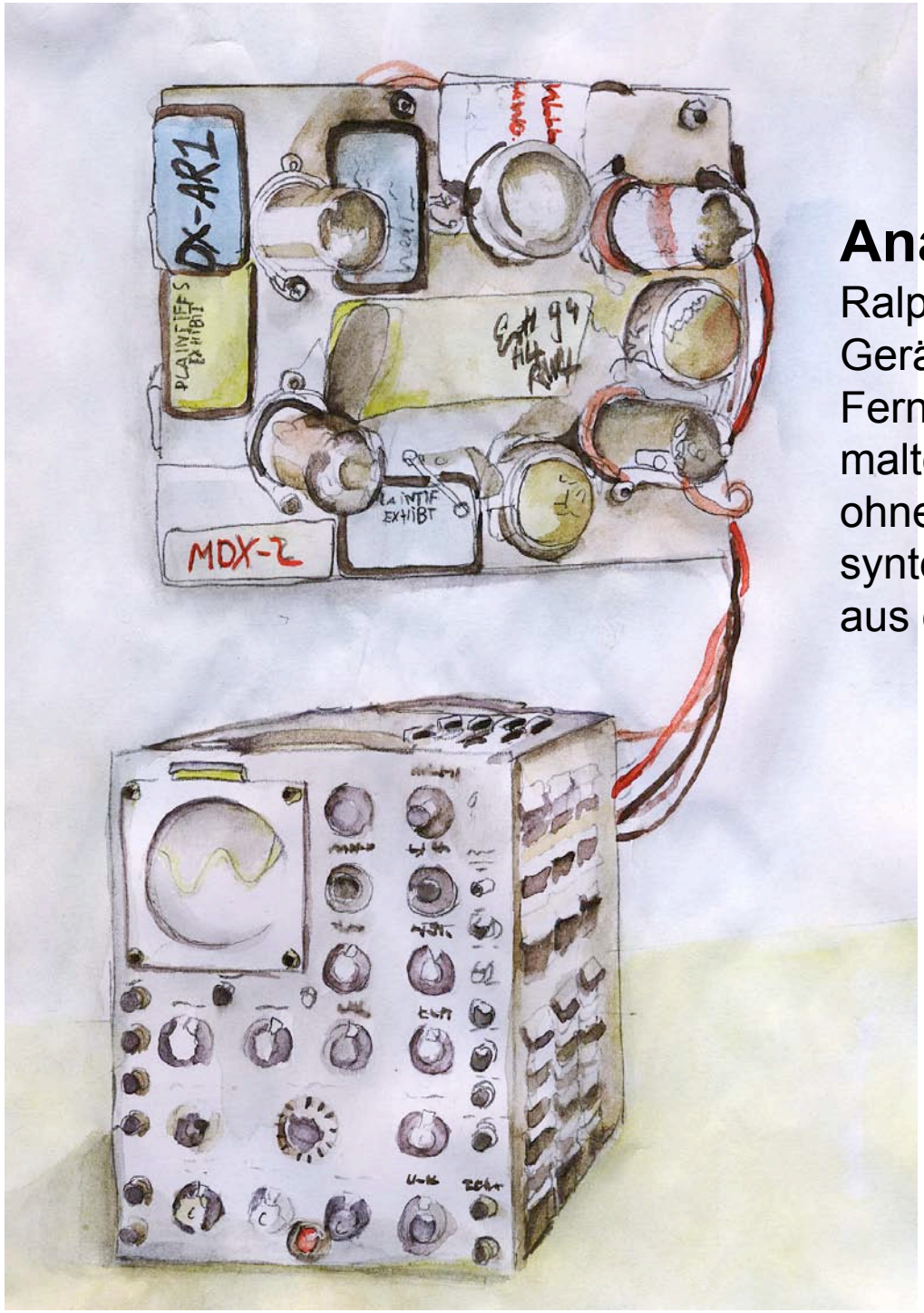
Die verschiedenen Klammertypen sind ein wichtiges Elemt in der logischen Struktur der Programmiersprachen. Das Einrücken von ineinader verschachtelten Bereichen gestaltet den Code übersichtlicher.

Mac:
() Runde Klammer SHIFT + 8 und 9
{} Geschweifte Klammer ALT + 8 und 9
[] Eckige Klammer ALT + 5 und 6

```
digitalWrite(ledPin, HIGH);
```

```
if(x > 5){
   y = y + 1;
   x = 0
}
```

```
int myList[] = {12, 7, 15, 4, 1, 8}
```



Analoges Signal

Ralph Baer kam 1966 auf die Idee, ein Gerät zu bauen, das jeder an seinen Fernsehr anschliesen kann. Dieses Gerät malte mit analogen Schaltkreisen, also ohne einen Computer, einfache syntetische Formen auf den Bildschirm aus denen sich Spiele entwickelten.

Beispiel #1

Analoges Signal

```
void setup(){
// Funktionen zur Voreinstellung
}
```

```
void loop(){
// Funktionen des laufenden
// Programms
}
```

Mikrocontroller Pin

```
pinMode(pin, mode);
```

Hier können nur Konstanten eigetragen werden:

INPUT = Eingangs
OUTPUT = Ausgang

Mikrocontroller Pin

```
digitalWrite(pin, value);
```

Hier können nur Konstanten eigetragen werden:

HIGH = 5 V
LOW = GND

Beispiel #1

Analoges Signal

Warte eine Sekunde

```
delay(1000);
```

```
delayMicroseconds(1);
```

Warte eine Microsekunde

Beispiel #1

Analoges Signal

pinMode(pin, mode)
pin: the pin number
mode: INPUT or OUTPUT

digitalWrite(pin, value)
pin: the pin number
value: HIGH or LOW

delay(ms)
ms: milliseconds

delayMicroseconds(ms)
ms: microseconds

```
void setup(){
  // set pin 9 as output

}

void loop(){
  // turn the power on and off



}
```

Beispiel #1

Analoges Signal

pinMode(pin, mode)
pin: the pin number
mode: INPUT or OUTPUT

digitalWrite(pin, value)
pin: the pin number
value: HIGH or LOW

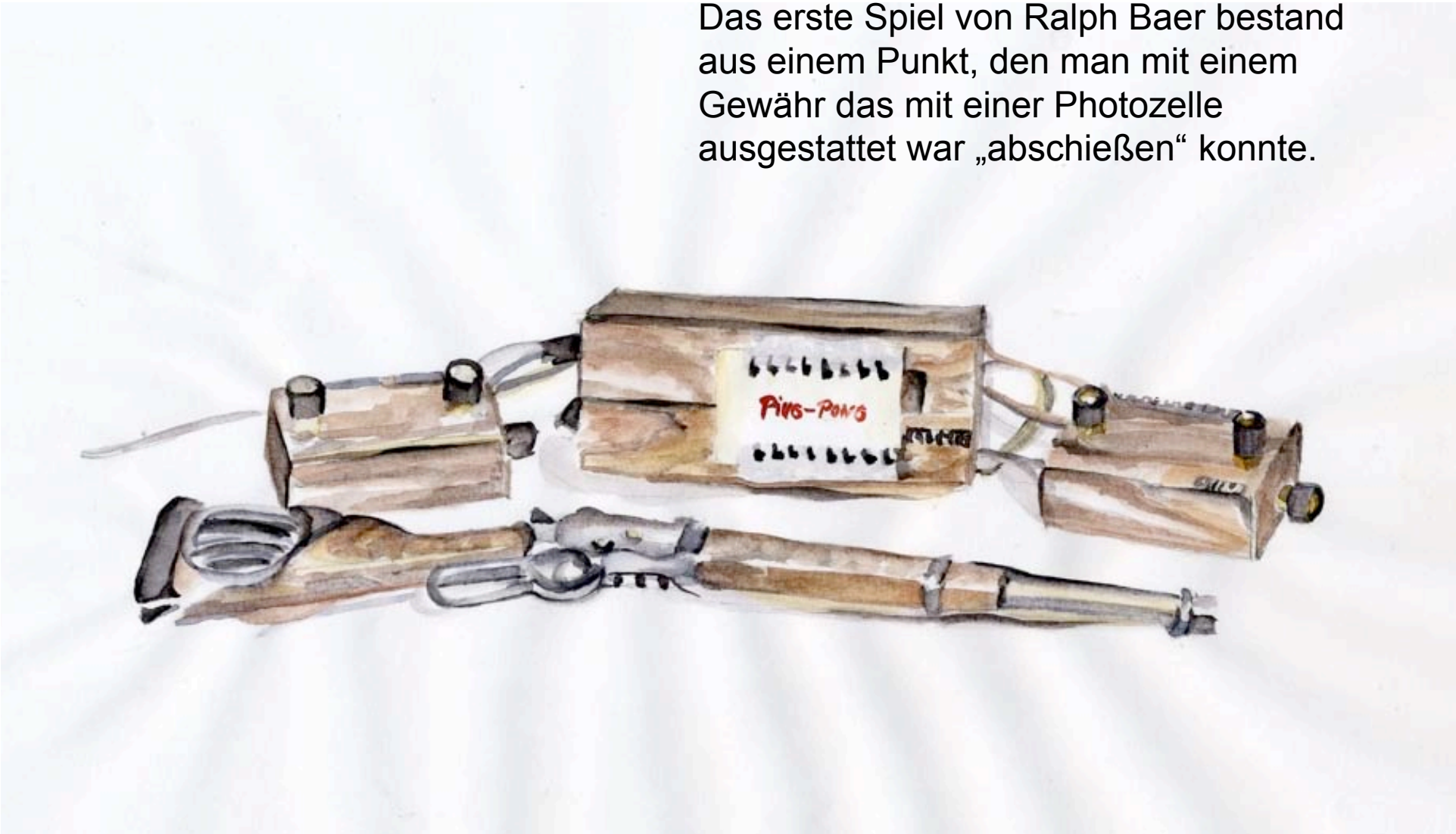
delay(ms)
ms: milliseconds

delayMicroseconds(ms)
ms: microseconds

```
void setup(){
  // set pin 9 as output
  pinMode(9, OUTPUT);
}

void loop(){
  // turn the power on and off
  digitalWrite(9, HIGH);
  delay(1);
  digitalWrite(9, LOW);
  delay(1);
}
```

Ein Punkt

Das erste Spiel von Ralph Baer bestand aus einem Punkt, den man mit einem Gewähr das mit einer Photozelle ausgestattet war „abschießen“ konnte.



```
#include <gamekit_2_1.h>
```

Die vorgefertigte Libery “gamekit.h” enthält viele hilfreiche Funktionen die das Programmieren des Mignon Game Kits erleichtern. Mit dem “include” Befehl wird die Game Kit Library importiert. Diesen Befehl setzt man am besten ganz an den Anfang des Programmcodes.

(Der Ordner „gamekit_2_1“ mit der Library muss zuvor in den Ordner “Dokumente / Arduino / libraries” der Arduinosoftware kopiert werden.)

```
gamekit.Begin();
```

Mit der „Begin“ Funktion aus der „gamekit“- Library werden die Ein und Ausgänge des Mikrocontrollers eingerichtet. Diese Funktion wird im „Setup“ aufgerufen.

Beispiel #2

Ein Punkt

```
gamekit.set_pixel(r,c,v);
```

Wenn man einen Punkt auf das Display des Mignon Game Kits platzieren möchte, verwendet man am einfachsten den “set_pixel”-Befehl, der „gamekit“-Library. Mit den Koordinaten „r“ und „c“ wird der Punkt positioniert. Mit dem dritten Wert „v“ kann eine von 16 Helligkeitsstufen 0 - 15 eingestellt werden. Höhere Werte wie zum Beispiel „v“ = 16 lassen den Punkt Blinken.

Beispiel #2

Ein Punkt

#include <library.h>
library: library name
No semicolon terminator!

gamekit.Begin()

gamekit.set_pixel(r , c, v)
r: row
c: column
v: value

```
//One Dot

// include Game Kit library


void setup(){
  // start Game Kit library

}

void loop({
  // set pixel

}
```

Beispiel #2

Ein Punkt

#include <library.h>
library: library name
No semicolon terminator!

gamekit.Begin()

gamekit.set_pixel(r , c, v)
r: row
c: column
v: value

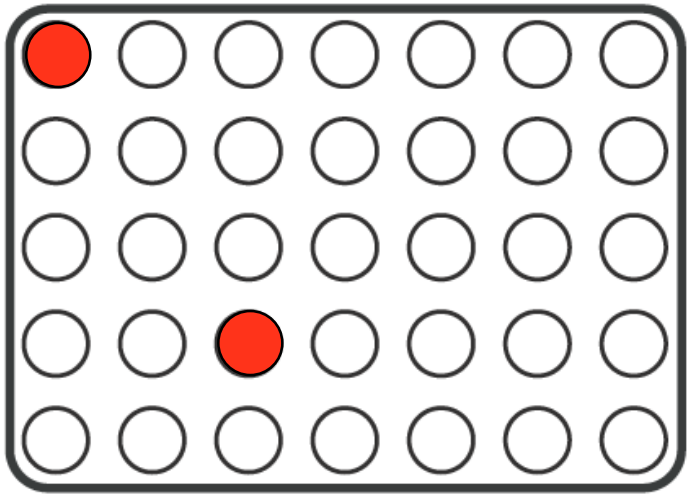
```
//One Dot

// include Game Kit library
#include <gamekit_2_1.h>

void setup(){
  // start Game Kit library
  gamekit.Begin();
}

void loop(){
  // set pixel
  gamekit.set_pixel(2,3,15);
}
```

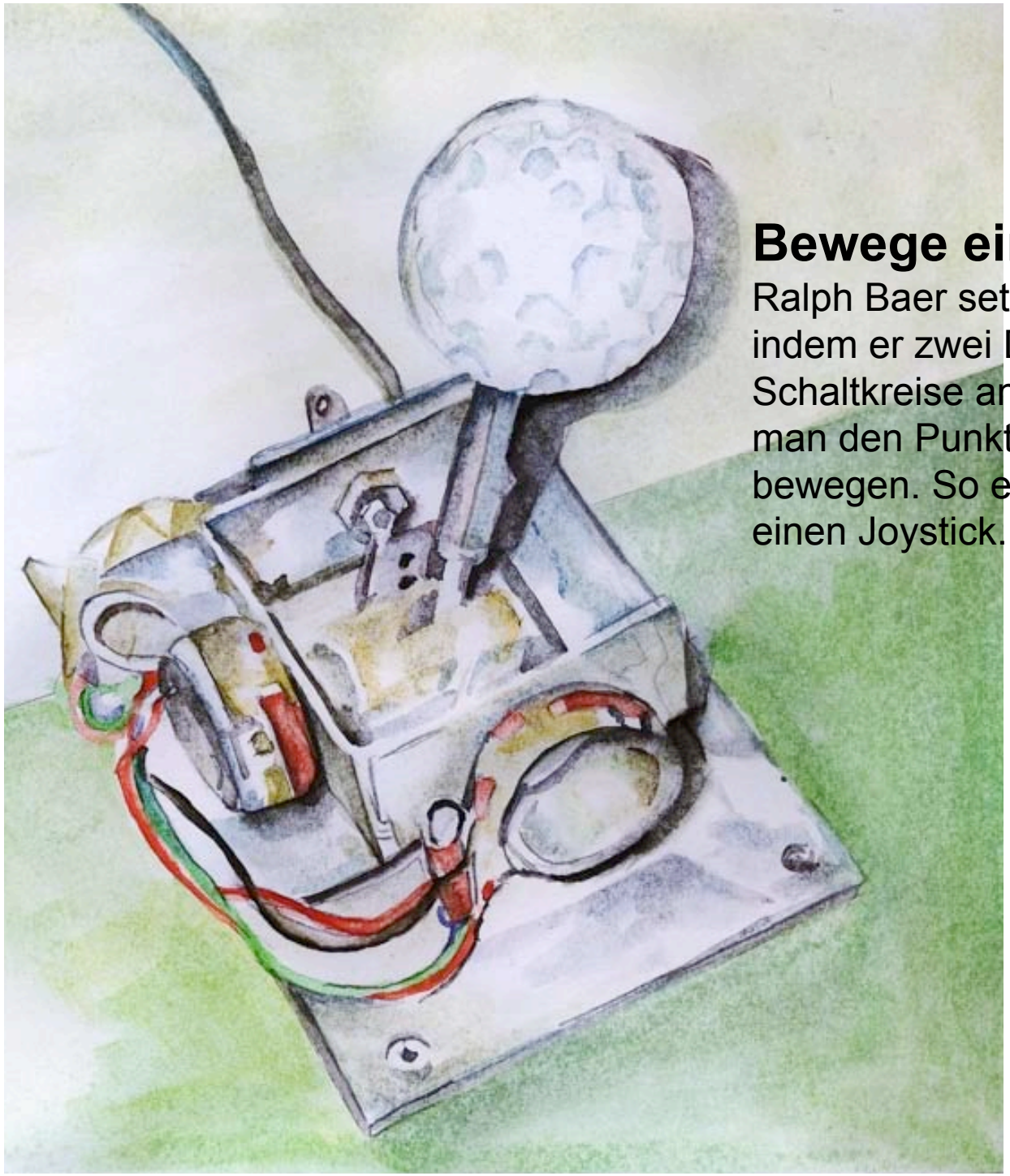
Beispiel #2

Ein Punkt

r = 0 und c = 0

r = 3 und c = 2





Bewege einen Punkt!

Ralph Baer setzte seine Experimente fort indem er zwei Drehregler seine Schaltkreise anschlos. Mit diesen konnte man den Punkt auf dem Bildschirm bewegen. So entstandt eine Skitzze für einen Joystick.

Name der Variable

```
int var = val;
```

Der Wert, den man der Variable zuweisen möchte
(Bei Arduino von -32768 bis +32768)

Beispiel #3

Bewege einen Punkt!

```
if(gamekit.button_pressed(butt_UP))
```

Konstanten für die Buttons:

butt_UP
butt_DOWN
butt_LEFT
butt_RIGHT
butt_FUNCA
butt_FUNCB

Beispiel #3

Bewege einen Punkt!

int var = val
var: name
val: value

if (conditions) {code}

boolean gamekit.button_pressed(button)
button: butt_UP, butt_DOWN, butt_LEFT, butt_RIGHT, butt_FUNCA, butt_FUNCB

```
// Move one Dot

//include gamekit library

// initialize two variables for rows and columns
// and assign them values of the center

void setup(){
  // start the gamekit library
}

void loop(){
  // put a dot one the screen according to the
  // values of the two variables

  // if the button UP is pressed,
  // erase the current dot and reduce
  // value of the rows variable

  // if the button DOWN is pressed,
  // erase the current dot and increase
  // value of the rows variable

  // if the button LEFT is pressed,
  // erase the current dot and reduce
  // value of the column variable

  // if the button RIGHT is pressed,
  // erase the current dot and increase
  // value of the column variable
}
```

Beispiel #3

Bewege einen Punkt!

int var = val
var: name
val: value

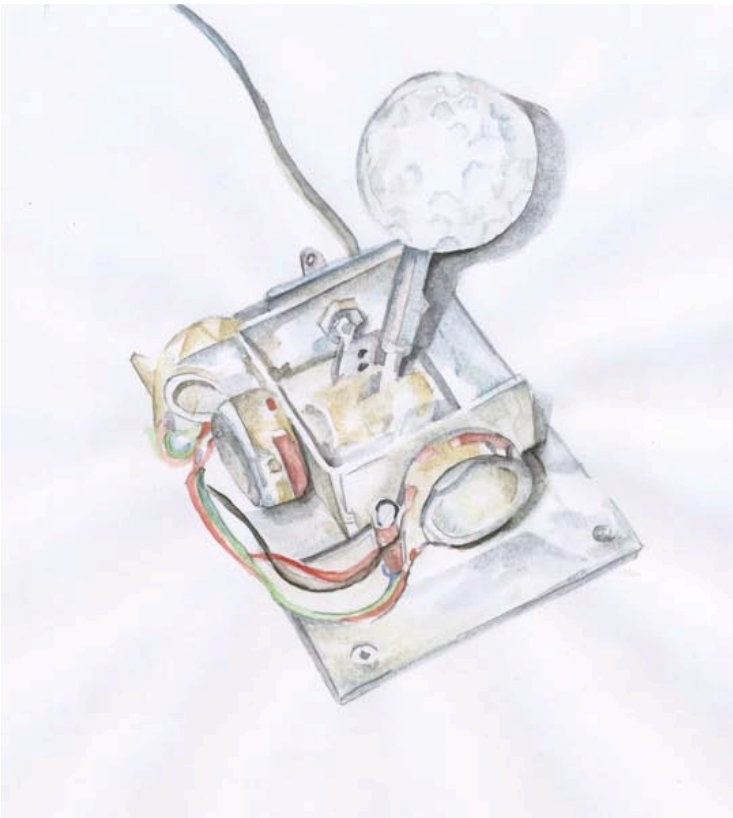
if (conditions) {code}

boolean gamekit.button_pressed(button)
button: butt_UP, butt_DOWN, butt_LEFT, butt_RIGHT, butt_FUNCA, butt_FUNCB

```
// Move one Dot

#include <gamekit_2_1.h>

int row = 2;
int column = 3;

void setup(){
  gamekit.Begin();
}

void loop(){
  gamekit.set_pixel(row,column,15);

  if(gamekit.button_pressed(butt_UP)){
    gamekit.set_pixel(row,column,0);
    row = row - 1;
  }

  if(gamekit.button_pressed(butt_DOWN)){
    gamekit.set_pixel(row,column,0);
    row = row + 1;
  }

  if(gamekit.button_pressed(butt_LEFT)){
    gamekit.set_pixel(row,column,0);
    column = column - 1;
  }

  if(gamekit.button_pressed(butt_RIGHT)){
    gamekit.set_pixel(row,column,0);
    column + column + 1;
  }
}
```

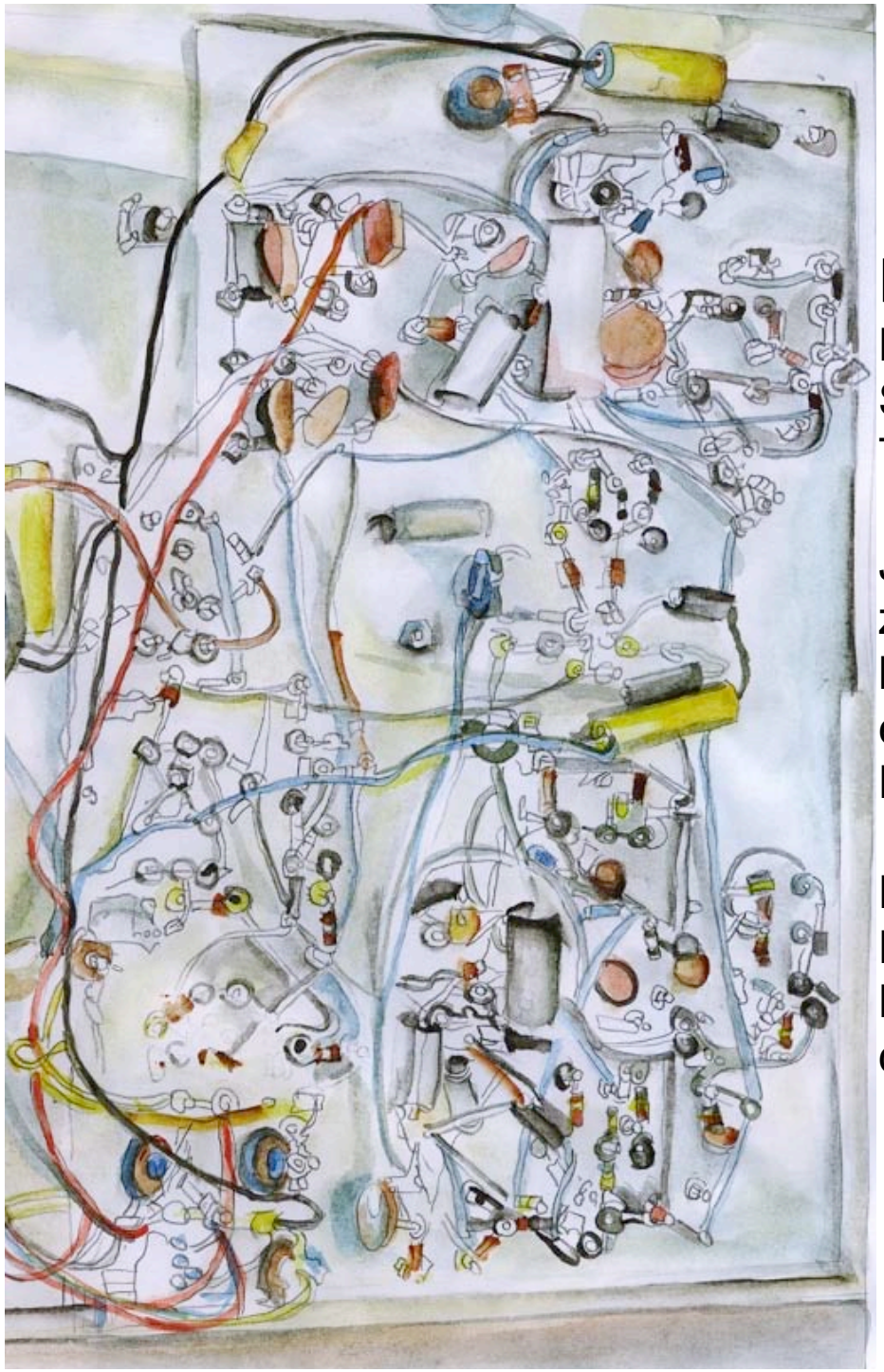
Beispiel #3

Bewege einen Punkt!



Diese Beispiel läst sich leiht zu einem Malprogramm umbauen. Hierzu reicht es aus die Zeilen zu entfernen, in denen der Punkt gelöscht wird.

In einer weiteren Ausbaustufe könnte das Malen und Löschen von Punkten mit den Funktionstasten gesteuert werden.



Das Fangen Spiel

Der nächste große Schritt in den Schaltkreisen von Ralph Baer und seinem Team war die Collision Detection.

Jetzt konnte man erkennen wann sich zwei Punkte auf dem Bildschirm berührten. Das erst Spiel das sich daraus entwickelte war, ganz einfach zwei Punkte, die einander „jagen“.

Da das Game Kit nur auf einer Seite vier Navigationstasten hat, kann sich in diesem Beispiel der zweite Spieler ausschlißlich diagonal bewegen.

Beispiel #4

Das Fangen Spiel

Zuweisendes Gleich

Vergleichendes Gleich

```
var1 = var2

if(var1 == var2)

if(var1 = var2)
```

Vergliechendes Gleich bei Bolean Wariablen

Beispiel #4

Das Fangen Spiel

Boolean Verknüpfung

&& = und

|| = oder

! = ist nicht

```
if( (var1 == var2)&&(var3 == var4) )
```

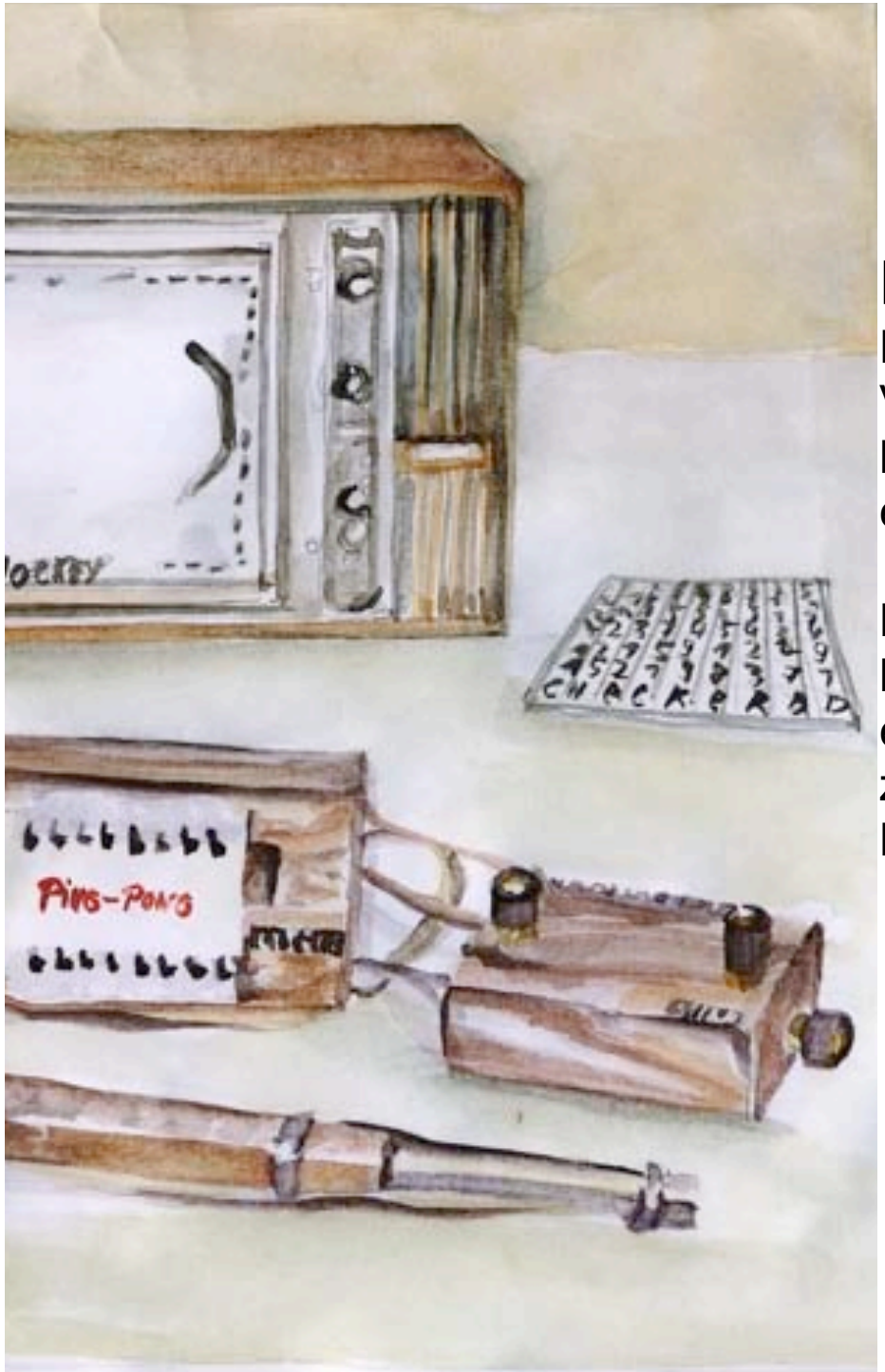
1. Kondition

1. Kondition

Beispiel #4

Das Fangen Spiel

```
// Chasing Game

#include <gamekit_2_1.h>

int row1 = 1; // Dot1
int column1 = 1;

int row2 = 3; // Dot2
int column2 = 5;

void setup(){
  gamekit.Begin();

}

void loop(){

  gamekit.set_pixel(row1,column1,15);
  gamekit.set_pixel(row2,column2,18);


  if(gamekit.button_pressed(butt_UP)){
    gamekit.set_pixel(row1,column1,0);
    row1--;
  }

  if(gamekit.button_pressed(butt_DOWN)){
    gamekit.set_pixel(row1,column1,0);
    row1++;
  }

  if(gamekit.button_pressed(butt_LEFT)){
    gamekit.set_pixel(row1,column1,0);
    column1--;
  }

  if(gamekit.button_pressed(butt_RIGHT)){
    gamekit.set_pixel(row1,column1,0);
    column1++;
  }

  if(gamekit.button_pressed(butt_FUNCA)){
    gamekit.set_pixel(row2,column2,0);
    column2--;
    row2--;
  }

  if(gamekit.button_pressed(butt_FUNCB)){
    gamekit.set_pixel(row2,column2,0);
    column2++;
    row2++;
  }

//collision detection
  if((row1 == row2)&&(column1==column2)){

    // blink for 2 seconds
    gamekit.set_pixel(row1,column1,20);
    delay(2000);

    // go back to start positions
    gamekit.set_pixel(row1,column1, 0);
    row1 = 1;
    column1 = 1;
    row2 = 3;
    column2 = 5;
  }
}
```



Mignon Pong

Das Fangen Spiel gab eine erste Vorahnung von dem was Videospiele sein könnten, doch es fehlte immer noch eine entschidenede Komponente.

Kurze Zeit später kam der Durchbruch mit hilfe eines Schaltkreises, der es erlaubte einen Punkt auf dem Bildschrim zu zaubern, der sich autonom bewegte – der Ball!

Beispiel #5

Mignon Pong

```
// Mignon Pong

#include <gamekit_2_1.h>

int row1 = 2; // Dot1
int column1 = 0;

int row2 = 2; // Dot2
int column2 = 6;

int row3 = 2; // Dot 3
int column3 = 3;
int rowM = 1;
int columnM = 1;
int balltime = 0;

void setup(){
  gamekit.Begin();
  //set the random seed to a noise value which is measured
on the analogue input pin 5 of the extensions port
  randomSeed(analogRead(5));
}

void loop(){

  gamekit.set_pixel(row1,column1,15);
  gamekit.set_pixel(row2,column2,15);
  gamekit.set_pixel(row3,column3,15);


  if(gamekit.button_pressed(butt_UP)){
    gamekit.set_pixel(row1,column1,0);
    row1--;
  }

  if(gamekit.button_pressed(butt_DOWN)){
    gamekit.set_pixel(row1,column1,0);
    row1++;
  }

  if(gamekit.button_pressed(butt_FUNCA)){
    gamekit.set_pixel(row2,column2,0);
    row2--;
  }

  if(gamekit.button_pressed(butt_FUNCB)){
    gamekit.set_pixel(row2,column2,0);
    row2++;
  }

  // move ball
  if(gamekit.get_systemcounter()> 100+balltime){
    balltime = gamekit.get_systemcounter();

    // collision detection 1
    if(((row1 ==
row3+rowM)&&(column1==column3+columnM))||((row2 ==
row3+rowM)&&(column2==column3+columnM))){
      columnM = columnM*-1;
      rowM = random(3)-1; // find a new angle for balls path
      if(row3==4) rowM = random(2)-1;
      if(row3==0) rowM = random(2);
    }

    gamekit.set_pixel(row3, column3, 0); // turn old dot off
    column3 = column3 + columnM; // move the dot further

    // bounce at top and bottom
    row3 = row3 + rowM;
    if(row3 +rowM <= -1){
      rowM = 1;
    }
    if(row3 +rowM >= 5){
      rowM = -1;
    }
  }
}
```

Intro Image

Ralph Baers Spielkonsole „Odyssey“ kam dann endlich 1972 als Produkt der Firma Magnavox auf den Markt. Leider bescherte sie ihm keinen kommerziellen Erfolg, was unter anderem daran lag, dass die Spiele auf dem Bildschirm zu abstakt blieben.

Gestalte ein Bild, das beim starten eines Games erscheint!

```
uint8_t image[5][7] PROGMEM={
0 ,4 ,7 ,0 ,4 ,7 ,0 ,
0 ,7 ,15,0 ,7 ,15,0 ,
0 ,0 ,0 ,0 ,0 ,0 ,0 ,
0 ,8 ,0 ,0 ,0 ,8 ,0 ,
0 ,0 ,12,15,12,0 ,0 ,
};
```

Definition eines Bildes
Die Variable „image“ steht für den Bildname, über den das Bild dann mit dem Befehl „gamekit.load_image(Bildname);“ aufgerufen werden kann.

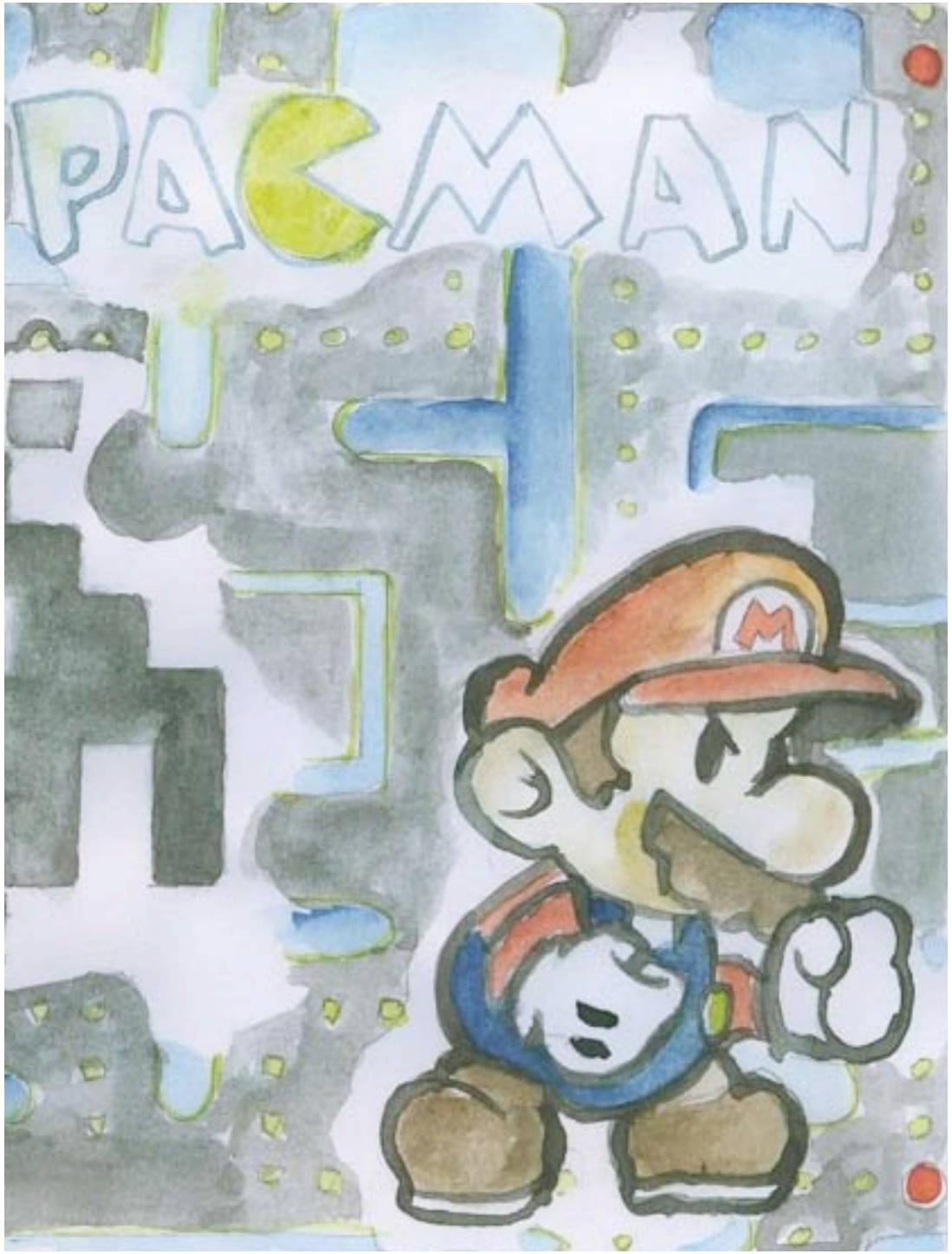
Beispiel #6

Intro Image

Uint8_t var [r] [c] PROGMEM = value
var: name
r: rows of the image
c: columns of the image
val: value

boolean gamekit.load_image(name)
name: image name

```
// Intro Image

#include <gamekit.h>
#include <avr/pgmspace.h>

uint8_t myman[5][7] PROGMEM = {
  0 ,0 ,0 ,15,0 ,0 ,0 ,
  0 ,3 ,3 ,3 ,3 ,3 ,0 ,
  0 ,0 ,0 ,3 ,0 ,0 ,0 ,
  0 ,0 ,3 ,0 ,3 ,0 ,0 ,
  0 ,0 ,3 ,0 ,3 ,0 ,0 ,
};

void setup(){
  gamekit.Begin();
  // some more setup code of my game ...

  gamekit.load_image(myman);
  delay(4000);
}

void loop(){
  // the code of my game ... (can by any game)
}
```



Intro Animation

Wie wichtig diese symbolischen Darstellungen bei Computerspielen sind zeigen die Erfolge von „Sapce Invaders“ und „Pac Man“ bei dem 1980 zum ersten Mal der digitalen Spielfigur einen eigener Namen gegeb wurde. Mit „Mario“ wurde 1981 der erste „Character“ etabliert.

Du kannst Dein Introimage leicht zu einer Animation erweitern, indem Du mehrere Bilder getrennt durch den „delay“-Befehl untereinander setzt.

Zusätzlich kannst Du ein Game Logo entwerfen. Auch das Gehäuse und Folien für das Display bieten weitere Freiräume.

void Begin();
Initialises (i.e. starts) the Gamekit.

void set_pixel(row, column, value);
Allocates the pixel value in the row and the column.
Values between 0 and 15 are levels of brightness, higher values make pixels blink, or the like.

uint8_t get_pixel(row, column);
Reads the value of the pixel in the *row* and the *column*.

void load_image(Bild);
Loads the image onto the display (see the example *load_image*).

void load_map(uint8_t *, uint8_t, uint8_t, uint8_t, uint8_t);
Loads a detail of a large image to the display (see example *load_map*).

void assign_pixelfunction(uint8_t, uint8_t(*)(uint8_t, uint8_t, uint32_t));
Allocates a value of pixels (has to be higher than 15) to a function.

uint32_t get_systemcounter();
Reads the system counter. The system counter rises by one when the display is redrawn.

boolean button_pressed(button);
Checks if the key *button* is pressed.

button may be:
butt_UP (Up)
butt_DOWN (Down)
butt_LEFT (Left)
butt_RIGHT (Right)
butt_FUNCA (function A)
butt_FUNCB (function B)

void wait_button_pressed(button);
Waits untill the key *button* is pressed.

void wait_button_released(button);
Waits untill the key *button* is released.

void set_button_timing(first, common);
Changes the speed of the repetition of the keys (see example *button_pressed*).

uint8_t get_buttons();
Reads all keys at the same time.

void play_tone(frequency, duration, volume);
Plays a sound of the pitch *frequency* over a given *duration* (in milliseconds)
Sound *volume* can be LOUD for loud and SILENT for silent.

void play_melody; (for the advanced)
Plays the melody (see example play_melody).

uint16_t get_current_melody_event();
Reads which *play_melody* is playing currently.

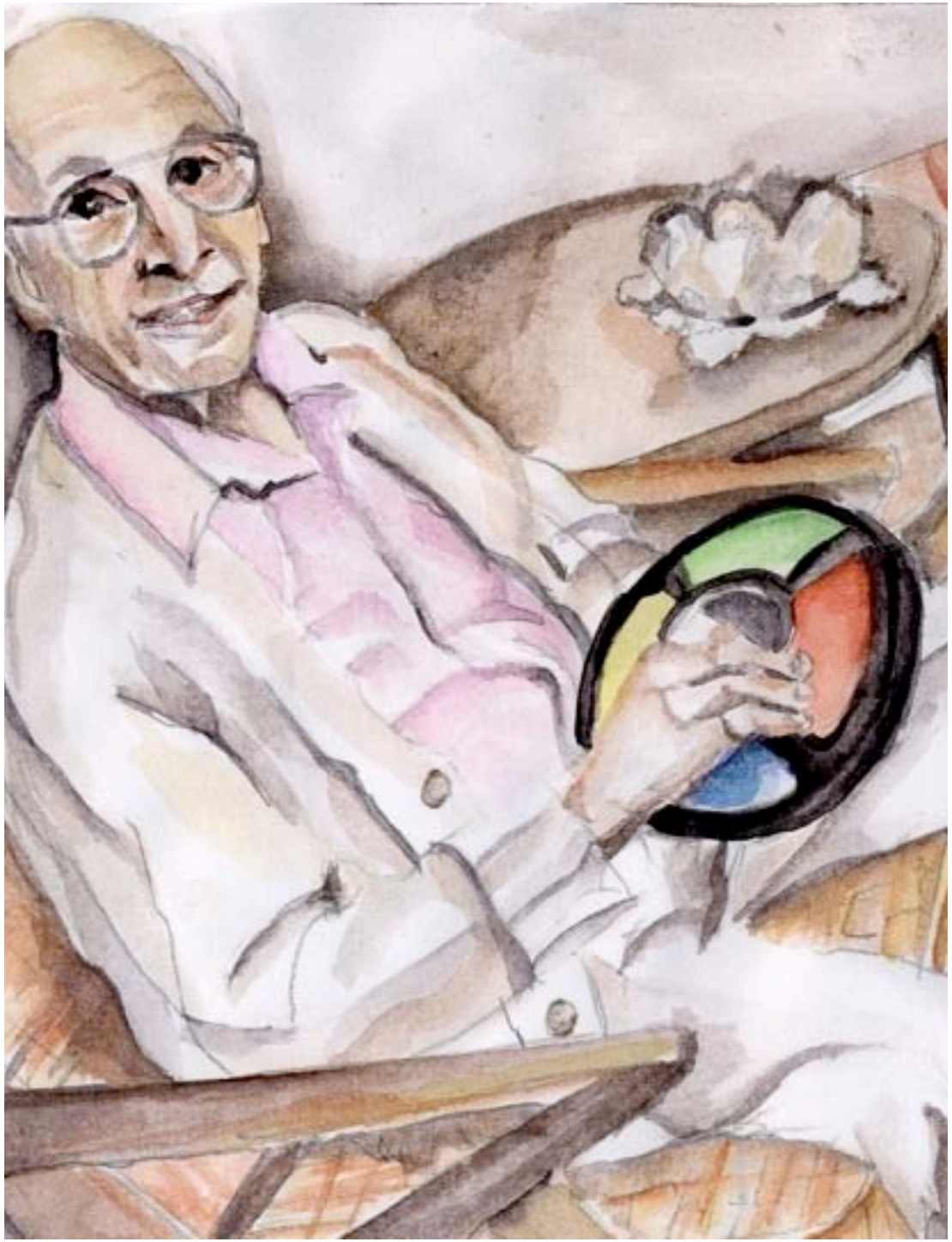
void set_current_melody_event(number);
Jumps to the place *number* of the melody.

Noch in Bearbeitung...

Die Folgenen Slides befinden sich noch in Bearbeitung.

Für Anregungen Korekturen und neue Beiträge bin ich sehr dankbar.

Info@olafval.de



Simon 6

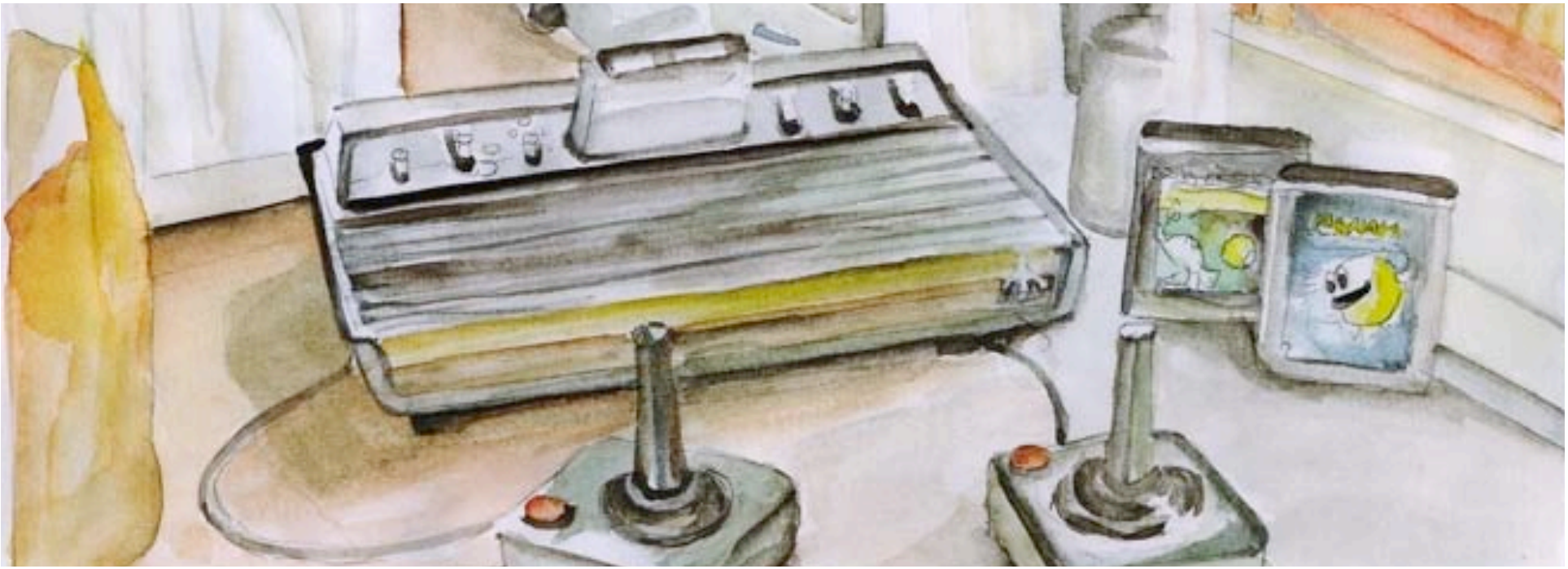
Text....

```
gamekit.play_tone(f,d,LOUD);
```

Spielt einen Ton der Tonhöhe “f” (Frequenz) über die Zeit von “d” (Duration in Millisekunden) ab.
Lautstärke kann „LOUD“ für laut und „SILENT“ für leise sein.

Noise Generator

Text....



Intro Melody

Text....