# László Moholy-Nagy

## Projekt-Dokumentation

Luise Fiedler und Martin Stelter

„Generatives Gestalten zum Bauhaus“,
Orientierungsmodul Interface Design,

WS 2009/10
Prof. Steffi Hußlein
Interaktive Werkstatt Markus Walthert

# Das Thema





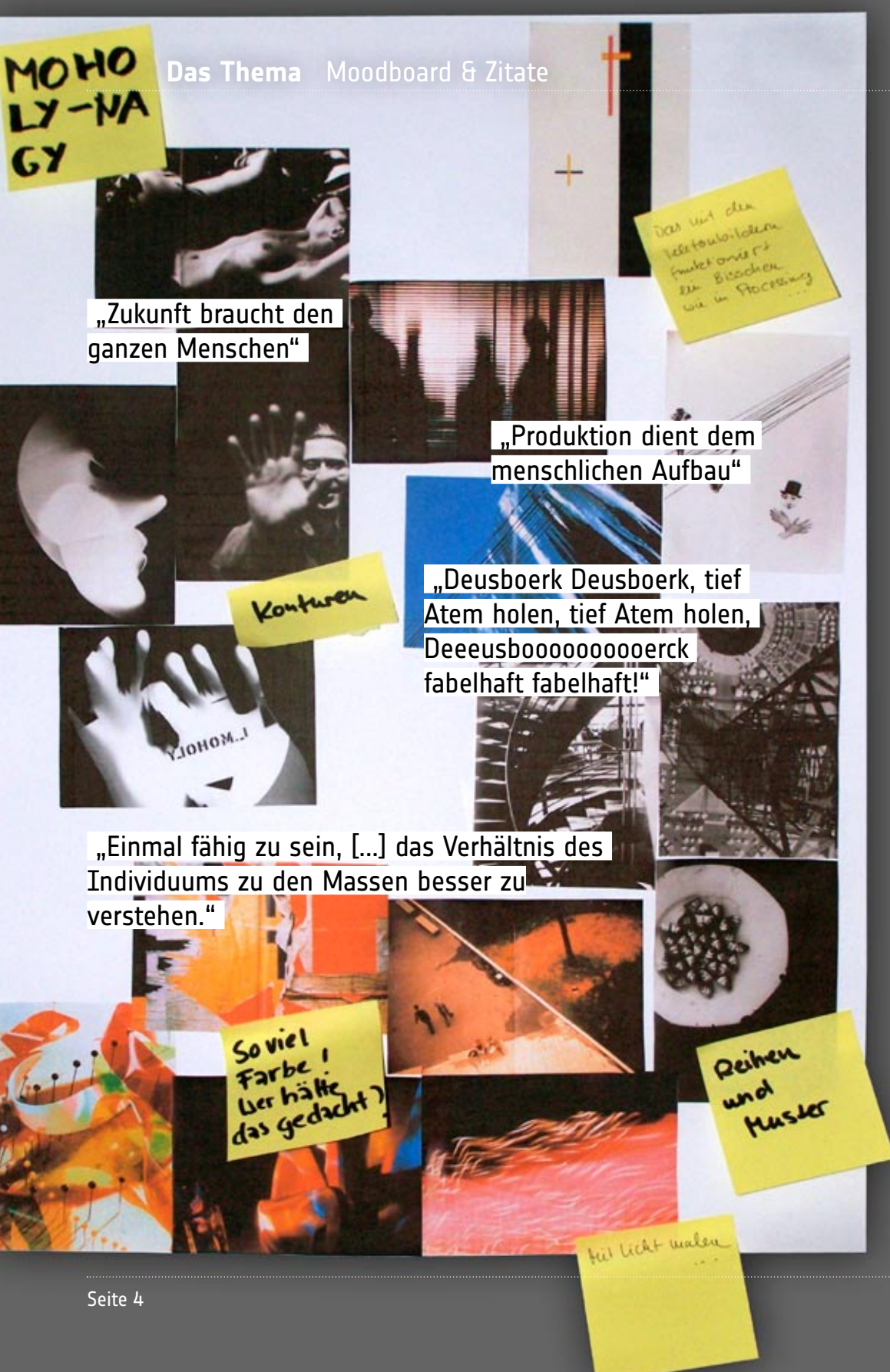
László Moholy-Nagy ist uns vorallem durch seine vielseitigen Fotogramme bekannt. Darüber hinaus malte, fotografierte und filmte, baute und konstruierte er. Moholy-Nagys Werke bestechen durch überraschende Blickwinkel und Rhytmen. Mit simplen Gestaltungsparametern erzeugte er Bilder und Skulpturen, welche durch Klarheit fesseln und dennoch nie ganz durchdrungen werden können.

Unser Ziel ist Aspekte aus Moholy-Nagys Werken aufzugreifen und darzustellen. Dies soll auf einfache und plakative Weise geschehen.

*1895* geboren in Bacsborsod, Ungarn
*1923–28* Bauhaus, Vorkurs & Formmeister d. Metallwerkstatt
*1934* Amsterdam
*1935* London
*1937* Chicago „New Bauhaus“
*1946* Tod an Leukämie

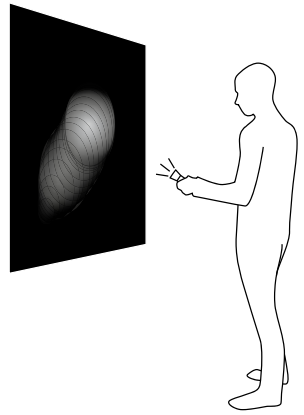
**Selbstportrait, 1926**

Fotogramme *Licht-Raum-Modulator* Fotocollage *»Ein Lichtspiel« (1930)* nichtgegenständlich *Grafikdesigner* Filmkünstler *Fotograf* schwarz-weiß-grau *Gleichgewicht* Licht-Schatten *Perspektive* Telefonbilder *Geometrie* Typographie *Farbe-Fläche-Raum* Muster *Raster* Überlagerung *Verfremdung* Interpretation







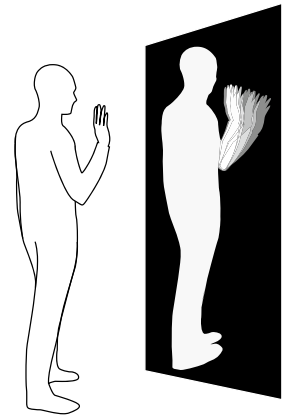



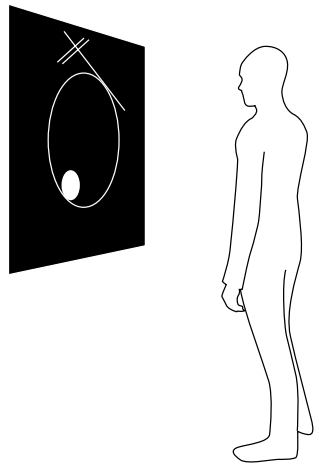
## 1. Mit Licht malen

Moholy-Nagy hat sich viel mit Licht und Lichtwirkung auseinandergesetzt. Ähnlich wie auf dem Foto „Pink Traffic Abstraction“ soll man mit einem Lichtschweif malen können.

## 2. Interaktives Fotogramm

Was wäre wenn die Menschen intuitiv ihr eigenes Fotogramm erstellen könnten?





## 3. Rhythmus und Komposition

In vielen Werken Moholy-Nagys findet man Kompositionen aus einfachen geometrischen Formen. Diese wollen wir im Raum erlebbar machen.

# Konzept 1
## Mit Licht malen



## Was:

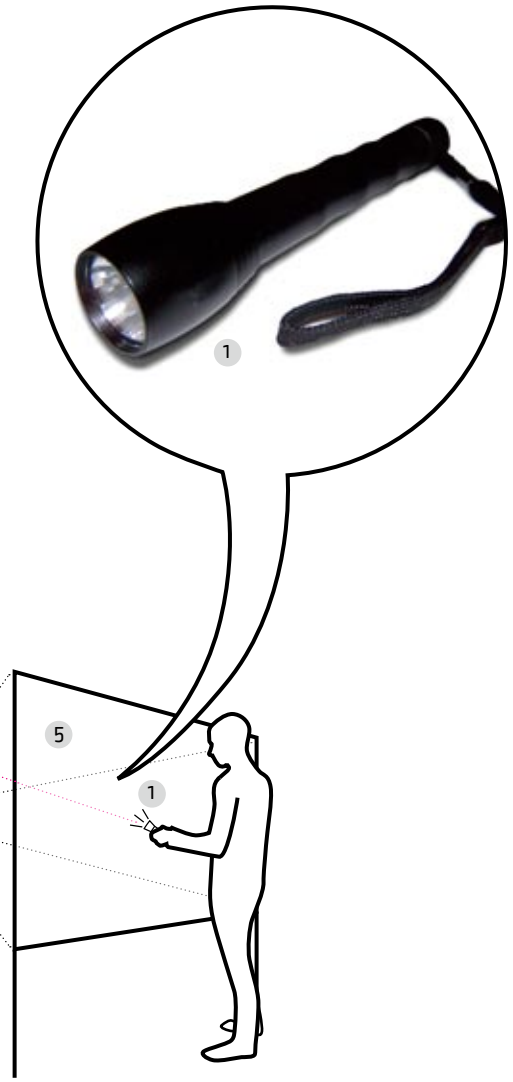
Inspiriert von einer Fotografie Moholy-Nagys enstand die Idee, den Nutzer mit Licht malen zu lassen.

**Pink Traffic Abstraktion, Fotografie, 1937–1940**

## Wie:

Als Interaktionswerkzeug dient eine Taschenlampe (1), in der sich eine Infrarot-Glühbirne befindet. Die Wii (2), die sich für den Nutzer unsichtbar hinter der Installation befindet, verarbeitet das Infrarotsignal und sendet es via Bluetooth an den Rechner (3). Dieser deutet das Signal als Maus. Im Processing-Script ist festgelegt, dass der Lichtpunkt immer dort sein soll, wo sich auch die Maus befindet.
Das so entstandene Bild wird mit dem Beamer (4) auf der Rückprojektionsfolie (5) abgebildet.

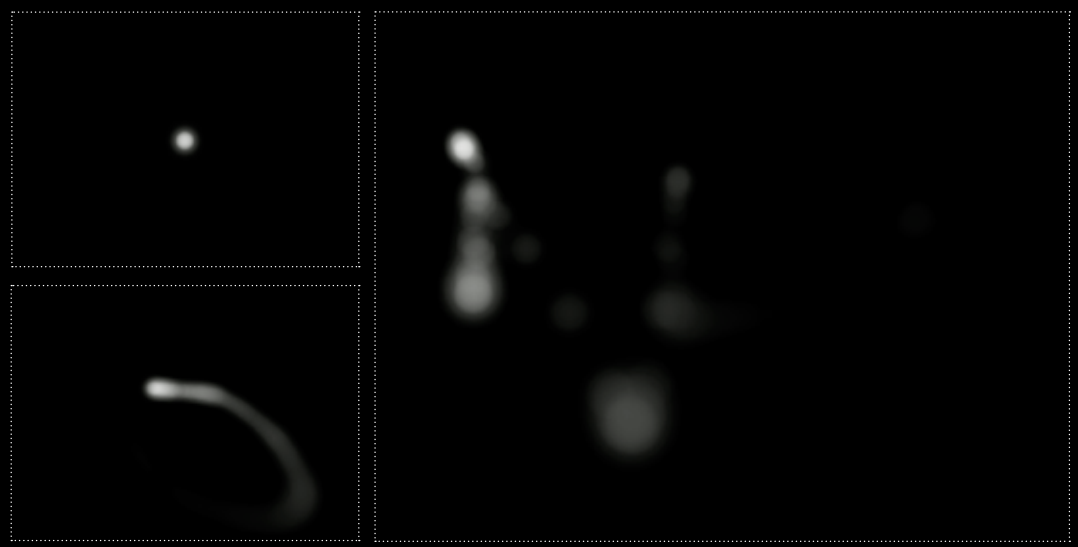
Der Benutzer kann nun scheinbar mit der Taschenlampe einen Lichtschweif erzeugen.



# Umsetzung

**Auf der Suche nach dem richtigen Lichtpunkt**

**Schließlich wirkte doch ein einfacher weißer Punkt mit Schein nach außen am realistischsten**





**Wenn gerade niemand interagiert, pulsiert in der Mitte ein Lichtpunkt. So merkt der Nutzer, dass hier etwas passiert.**
**Fährt man mit der eingeschalteten Taschenlampe über die Fläche, entsteht der eindruck eines Lichtschweifes. Je weiter man mit der Lampe nach oben geht, desto kleiner wird der Schweif. Dies erzeugt räumliche Tiefe.**





# Script

```
PImage b;

 */
int c = 0;
int d = 0;

boolean moovin;

void setup(){
  size (1024,768);
  background (0);
  smooth();
  frameRate(20);
  imageMode(CENTER);
  noCursor();



}

void draw(){

    noCursor();
```

**Der Lichtpunkt wird als Bild geladen.**

```
  if (pmouseX == mouseX){
    d++;
    if (d>=40) {
      moovin = true;
      d=0;
    }
  }
  else{
    moovin = false;
    d=0;
  }
```

**Wenn sich die Maus nicht bewegt, pulsiert in der Mitte ein Lichtpunkt.**

```
  if (moovin){
    c++;
    if (c >= 131){
      c=0;
    }
    background(0,15);
    b =loadImage("lichtkreis"+c+".png");
    image(b, width/2, height/2,138,138);

  }

}
```

**Der pulsierende Lichtkreis besteht aus 131 Bildern. Diese werden nacheinander geladen.**

```
void mouseDragged(){
    noCursor();
  fill (0,7);
  rect(0,0,width,height);
  b =loadImage("lichtkreis50.png");
  image(b, mouseX, mouseY,(mouseY/3+10),(mouseY/3+10));
}
```

**Wenn die Maus bewegt wird, bewegt sich auch der Lichtkreis. Seine Größe richtet sich nach der y-Position der Maus.**

# Konzept 2
## Interaktives Fotogramm

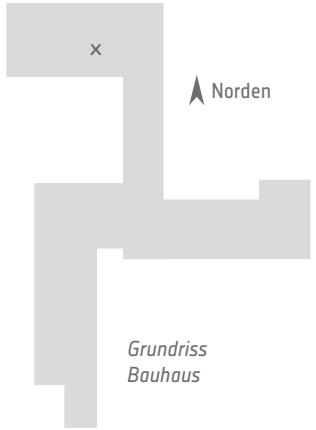


## Was:

Der Benutzer steht zwischen Lichtquelle und Fotopapier. Er „schützt“ einen Teil des Fotopapieres vor dem Licht, wodurch diese Stelle unbelichtet, sprich weiß, bleibt. Der Rest des Fotopapieres wird mit der Zeit vom Licht geschwärzt. Nach einer gewissen Dauer beginnt der Vorgang vom Neuen.

## Wo:

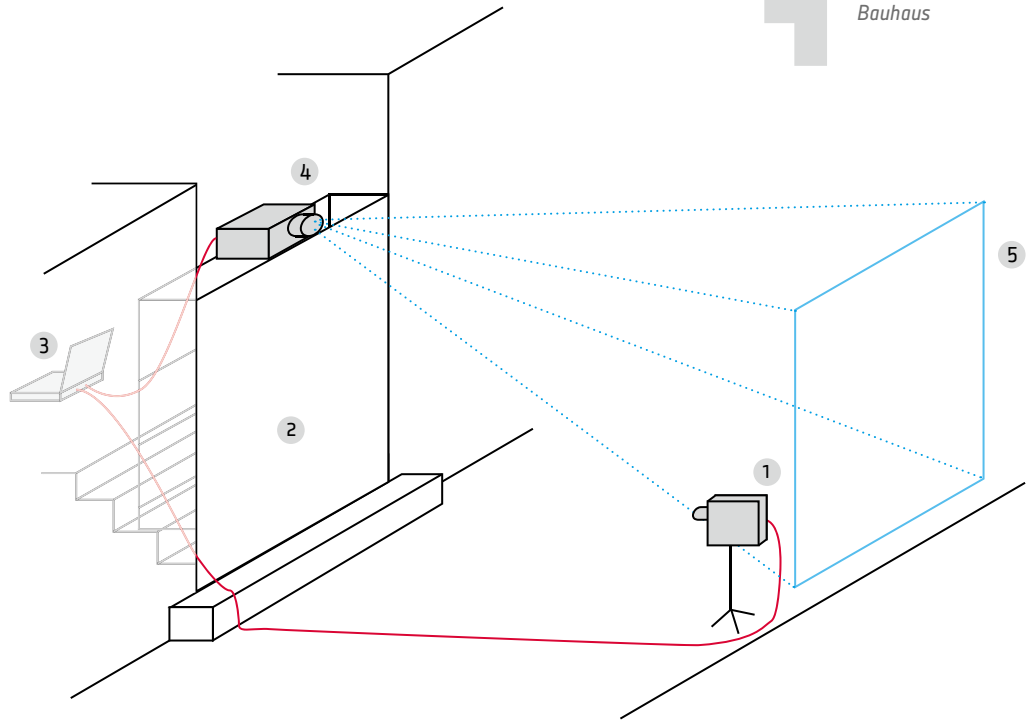
Der Aufbau soll im Gang des Erdgeschosses des Nordflügels im Bauhaus stattfinden. Die Lichtquelle des interaktiven Fotogramms (Leuchtkasten) wird in einen Treppenaufgang eingelassen.

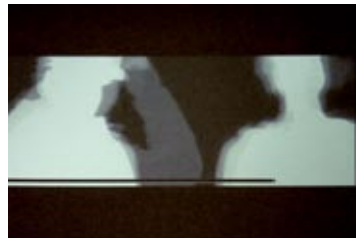



*Grundriss Bauhaus*

## Wie:

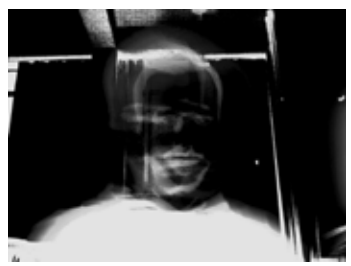
Über eine Kamera (1) wird der Bereich vor dem Leuchtkasten (2) gefilmt. Im Rechner (3) wird mittels der Software Processing das Bild verarbeitet und über den Beamer (4) auf die gegenüberliegende Wand projeziert (5)





## Version 1:
Das Videobild wird in schwarz-weiß umgewandelt

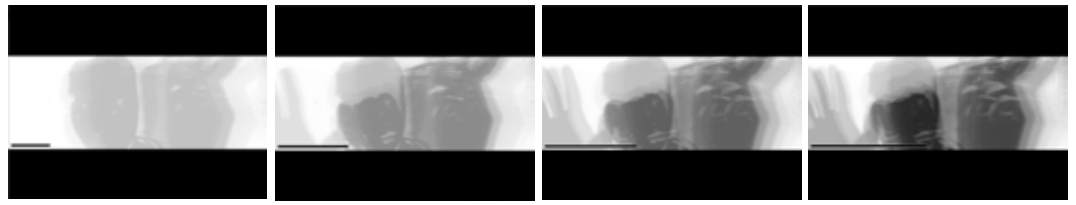
## Version 2:
Die Bilder überlagern sich. Der Gesamteindruk wird weicher.

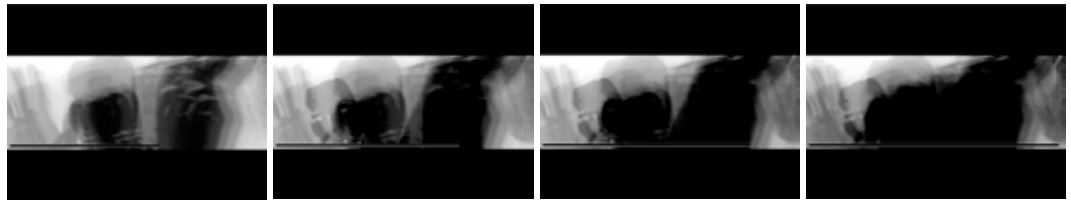


## Version 3:
Das Bild ist invertiert: Bereiche des „Fotopapieres“, die später von Personen bedeckt sind, bleiben hell.









```
/**
 * Interactive Photogram (Moholy-Nagy) by Martin Stelter
 * based on Frame Differencing by Golan Levin.
 */

import processing.video.*;

int myCount = 0;
int myTime = 100; // interval
int numPixels;
int[] previousFrame;
Capture video;
int thresh = 50; //treshold


void setup() {
  noCursor();
  size(1024, 768);
  video = new Capture(this, width, height, 24);
  frameRate(24);
  numPixels = video.width * video.height;

  // Create an array to store the previously captured frame
  previousFrame = new int[numPixels];
  loadPixels();
  background(255);
  colorMode(RGB, 255,255,255,500);
}
```

## Version 4 (final):
Jeder Pixel des Frames wird verarbeitet: Liegt sein Farbwert über einem festgelegten Schwellenwert, wird an seiner Stelle das Bild um 5% dunkler. Ist der Farbwert des Pixels unter der Schwelle, bleibt das Bild an dem Punkt unverändert. Mit der Zeit schwärzt sich das Beamerbild an den Stellen, an denen die Kamera Licht einfängt. Dort wo Menschen oder Objekte stehen, bleibt das Beamerbild hell. Nach einem bestimmten Intervall wird die gesammte Bildfläche weiß gefärbt und alles beginnt von vorn.

```
void draw() {
  if (myCount < myTime) {
    if (video.available()) {
      video.read(); // Read the new frame from the camera
      video.loadPixels(); // Make its pixels[] array available
      int movementSum = 0; // Amount of movement in the frame
      int curPixel = 0;
      for(int x = 0; x < video.height; x++){
        for(int y = 0; y < video.width; y++){
          int i = curPixel;
          curPixel++;
          color currColor = video.pixels[i];
          color prevColor = previousFrame[i];
          // Extract the red, green, and blue components from current pixel
          int currR = (currColor >> 16) & 0xFF; // Like red(), but faster
          int currG = (currColor >> 8) & 0xFF;
          int currB = currColor & 0xFF;
          // Extract red, green, and blue components from previous pixel
          int prevR = (prevColor >> 16) & 0xFF;
          int prevG = (prevColor >> 8) & 0xFF;
          int prevB = prevColor & 0xFF;
          // Compute the difference of the red, green, and blue values
          int diffR = abs(currR - prevR);
          int diffG = abs(currG - prevG);
          int diffB = abs(currB - prevB);
          // Add these differences to the running tally
          movementSum += diffR + diffG + diffB;
          if((currR > thresh) || (currR < (0 - thresh)) || (currG > thresh) ||
            (currG < (0 - thresh)) || (currB > thresh) || (currB < (0 - thresh))){
            pixels[i] = color(0, 5);  //pixel becomes darker
          }
          else {
            pixels[i] = color(0,0);  //pixel stays unchanged
          }

          // Save the current color into the 'previous' buffer
          previousFrame[i] = currColor;
        }
      }

      updatePixels();

    }
    // timebar
    fill(0,200);
    stroke(255,200);
    rect(10,height-20, (width-20)*myCount/myTime, 10);
    noStroke();
  }
  if (myCount >= myTime+2) {  // time is up, refresh the screen
    println(myCount);
    fill(255);
    rect(0,0,width,height);
    myCount=0;
  }
```

# Konzept 3
## Rhythmus und Komposition

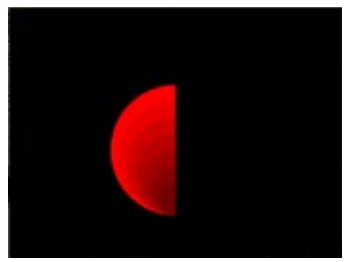
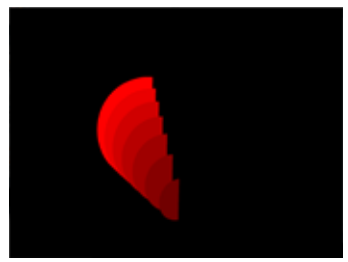
## Was:

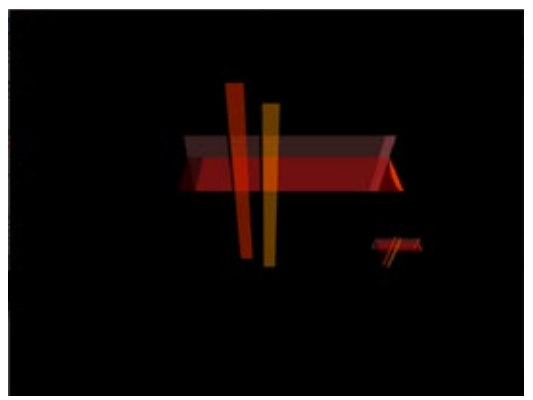
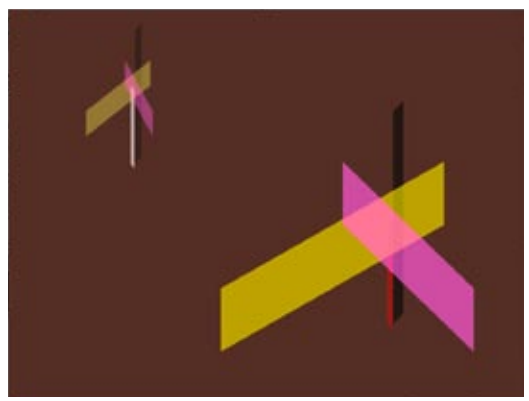
Die geometrisch-abstrakten Gleichgewichts-Kompositionen Moholy-Nagys sollen interaktiv erlebar werden. Wir wählten verschiedene einfache Beispiele, die dem Benutzer die Möglichkeit geben sollten in die visuelle Welt Lāszlōs einzutauchen. Wir entschieden und später aufgrund fehlender Zeit das Konzept nicht weiter zu verfolgen. Die hier gezeigten Beispiele stellen den Stand unser Arbeit dar.



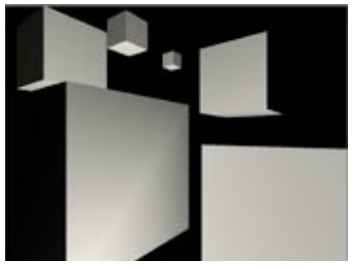











## Typographie:


```
PFont[] myFont = new PFont[1];
String[] myZitat = new String [3];
int myZahl;

void setup () {
  size (649,480);
  smooth();
  frameRate(30);
  myZitat[0]="Zukunft braucht den ganzen Menschen";
  myZitat[1]="Produktion dient dem menschlichen Aufbau";
  myZitat[2]="xx";
  myFont[0] = loadFont ("NeutraTextTF-BookAlt-32.vlw");
  textFont(myFont[0]);
  textAlign(LEFT);
  fill(0);
}
```

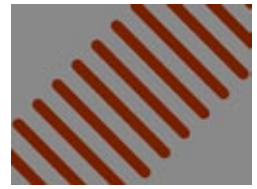
```
void draw () {
  background(220);
  fill(0);
  text (myZitat[0], width*0.5, height*0.35, 300,500);

  noStroke();
  fill(220);
  rect(0,0,mouseX,height);
  fill(0);
  text (myZitat[1], width*0.2, height*0.7, 300,500);
  fill(220);
  noStroke();
  rect(mouseX, height*0.5, width,height);
  stroke(0);
  strokeWeight(20);
  line(mouseX,0,mouseX,height);
}
```

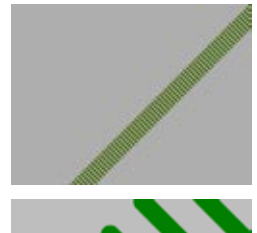
## Stop-Motion-Film mit 6 Portraitbildern:



## Übungsaufgaben

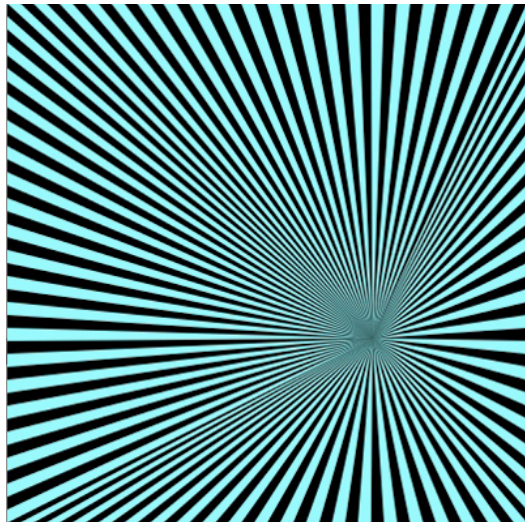
### Strahlen 1:





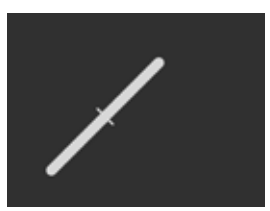
```
void setup () {
  size (640, 480);
  smooth();
  frameRate(30);
  noCursor();
}

void draw () {
  //oben dunkel, unten hell
  background((int)255*mouseY/height);

  // unten dunkel, oben hell
  int varFarbe = 255 - (int)255*mouseY/height;
  stroke(varFarbe,abs(255 - varFarbe*2),0);

  for (int i=0; i<250; i++) {

    float varBreite = 5*abs(width - mouseX*2);
    // Abstand der Linien voneinander
    float varVarianz = i*0.9*abs(width - mouseX*2);
    float varLaenge = 2*abs(width - mouseX*2);
    strokeWeight(0.5*abs(width - mouseX*2)+1);

    //Linien;
    line(mouseX + varVarianz-varLaenge,
         mouseY - varVarianz-varLaenge,
         mouseX + varVarianz+varLaenge,
         mouseY - varVarianz+varLaenge);

    line(mouseX - varVarianz-varLaenge,
         mouseY + varVarianz-varLaenge,
         mouseX - varVarianz+varLaenge,
         mouseY + varVarianz+varLaenge);
  }
}
```

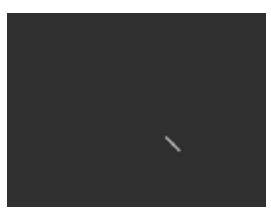
## Strahlen 2:



```
void setup(){
  size (400,400);
  smooth ();
  frameRate(30);
  cursor(MOVE);
}

void draw(){
  dreieckeStrahlen();


}
void dreieckeStrahlen(){
  background (0);

  int myColorR=mouseX/(3/2);
  int myColorG=mouseY/(3/2);

  for(int i=0;i<400; i=i+20){


    noStroke ();
    fill (80,myColorR,myColorG);
    triangle (mouseX, mouseY, 400,0+i,400,10+i);
    triangle (mouseX, mouseY, 20+i,400,10+i,400);
    triangle (mouseX, mouseY, 0,20+i,0,10+i);
    triangle (mouseX, mouseY, 0+i,0,10+i,0);
  }


}
```

## Strahlen 3:



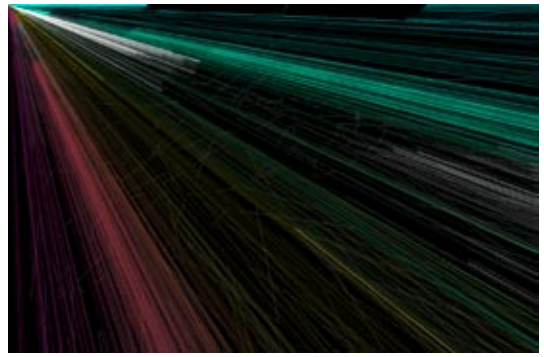


```
void setup () {
  size(640,480);
  smooth();
  noCursor();
}

void draw () {
  background(50);
  stroke(220);

  //Linie horizontal
  float varLaenge = 3000/(abs(width - mouseX*2)+1);
  strokeWeight(0.3*varLaenge);
  line (mouseX-varLaenge,
        mouseY-varLaenge,
        mouseX+varLaenge,
        mouseY+varLaenge);

  //Linie vertikal
  float varLaenge2 = (abs(width - mouseY*2)+1);
  strokeWeight(0.2*varLaenge2);
  line (mouseX+varLaenge2,
        mouseY-varLaenge2,
        mouseX-varLaenge2,
        mouseY+varLaenge2);
}
```





## Strahlen 4:



```
void setup () {
  size(1300, 1000);
  background(0);
  frameRate (30);
}

void draw () {
  //if (mousePressed) {
  int meineMausY = mouseY;
  int meineMausX = mouseX;
  int ry = round(meineMausY *256 / 200);
  int rx = round(meineMausX *256 / 200);
  stroke(ry, rx, abs(rx - ry), 80);
  int i=0;
  while (i<1000) {
    drawHalfWay(i*pmouseX, i* pmouseY);
    i=i+1;
  }
}

void drawHalfWay (int wertX, int wertY) {
  int halbWertX = getDurchschnitt (mouseX, wertX);
  int halbWertY = getDurchschnitt (mouseY, wertY);
  line (wertX, wertY, halbWertX, halbWertY);
}

int getDurchschnitt (int wertA, int wertB) {
  int myHalfWay = ((wertA + wertB) /2);
  return myHalfWay;
}
```

Fachbereich
**Design** Dessau
Department
of Design