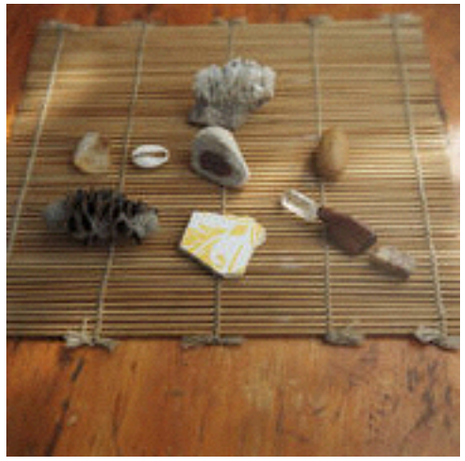
Model und System

und eine ganz konkrete Aufgabe selbst programmieren



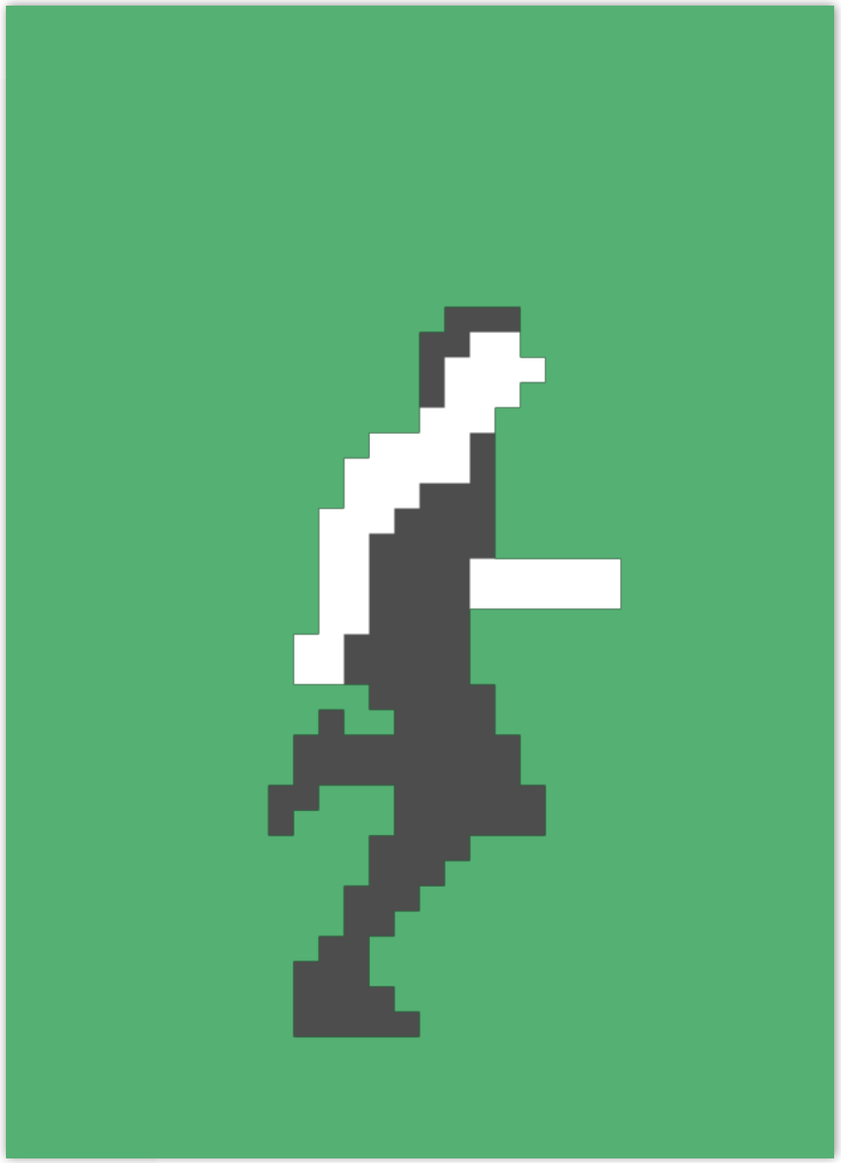

XVII
IX
XXI
Die Welt

Black Box

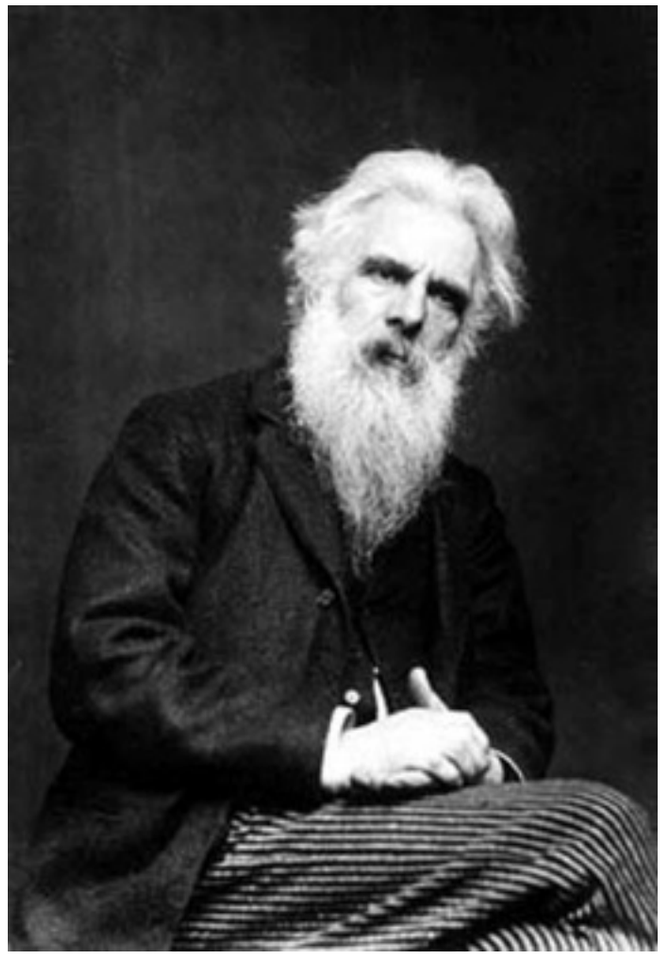
Als **Black Box** bezeichnet man in Kybernetik und Systemtheorie ein (möglicherweise sehr komplexes) System, von welchem im gegebenen Zusammenhang nur das äußere Verhalten betrachtet werden soll. Die innere Struktur mag bekannt sein; solche Kenntnis darf aber nicht benutzt werden (etwa weil ein Nachfolgemodell innen anders gebaut sein darf). Man beschränkt sich bei der Untersuchung und Beschreibung auf die Messung der Input-Output-Beziehungen (EVA-Prinzip).
Anders als ein geschlossenes System in der Thermodynamik darf so ein System auch Materie mit der Umgebung austauschen, zum Beispiel ein Fahrscheinautomat.
Das Gegenteil einer Black Box wird meistens als *White Box* (engl. *weiße Kiste*) bezeichnet; die metaphorisch konsistenteren Begriffe *Glass Box* und *Clear Box* (engl. *Glaskiste* bzw. *durchsichtige Kiste*) sind synonym, werden aber seltener verwendet.

Animation

Animation (von lat. *animare*, „zum Leben erwecken“; *animus*, „Geist, Seele“) ist im engeren Sinne jede Technik, bei der durch das Erstellen und Anzeigen von Einzelbildern für den Betrachter ein bewegtes Bild geschaffen wird.



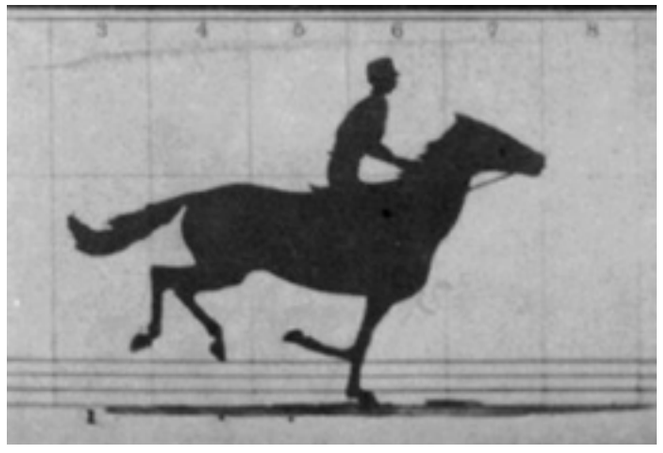
Eadweard Muybridge

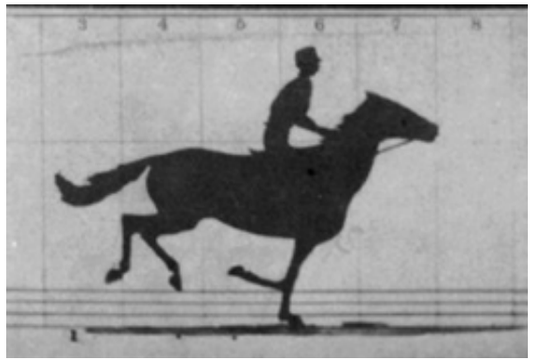


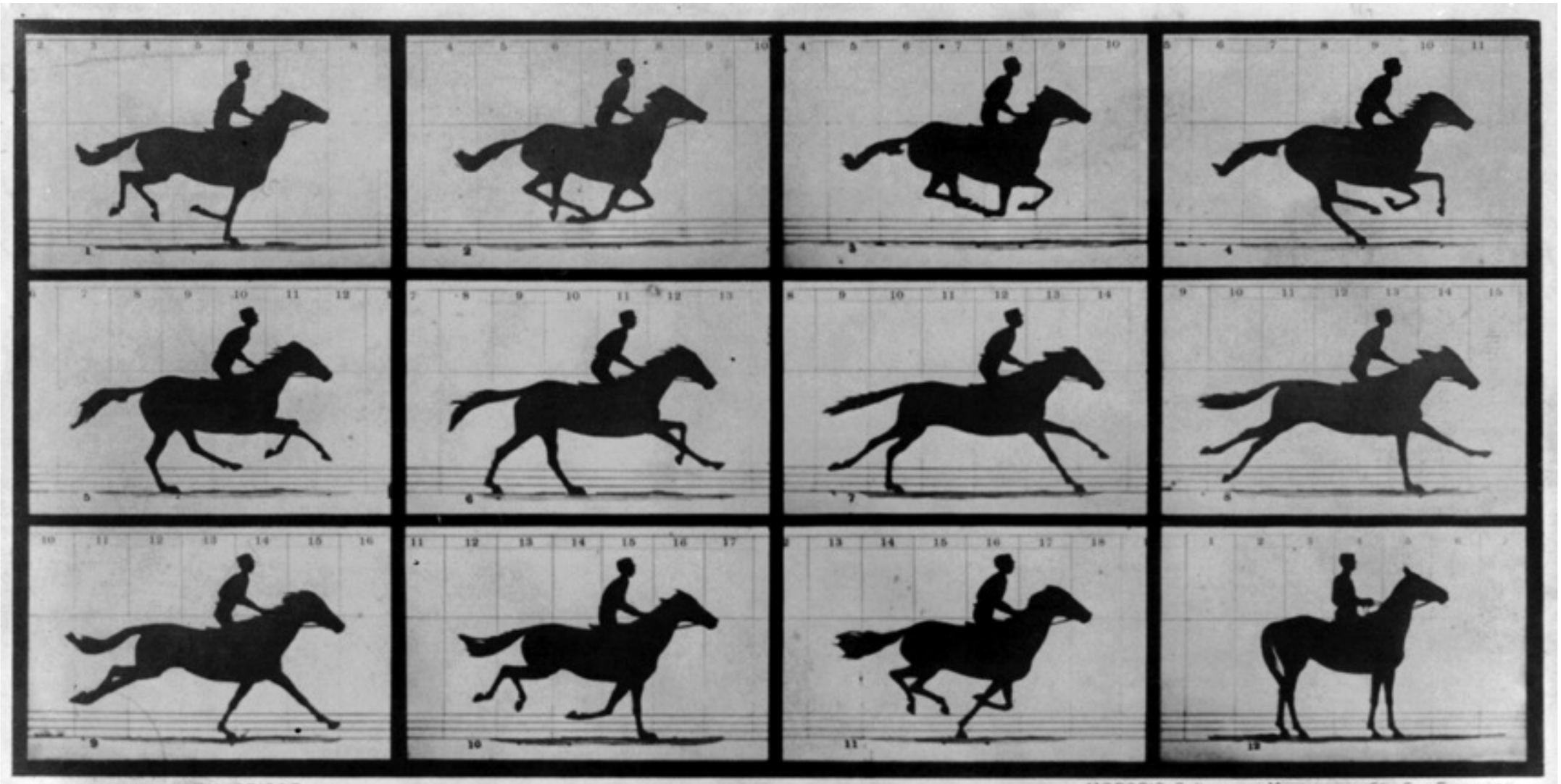
(* 9. April 1830 in Kingston upon Thames; † 8. Mai 1904 ebenda; eigentlich *Edward James Muggeridge*) war ein britischer Fotograf und Pionier der Fototechnik.
Er gilt – neben Étienne-Jules Marey und Ottomar Anschütz – aufgrund seiner Reihenfotografien und Serienaufnahmen mit Studien des menschlichen und des tierischen Bewegungsablaufs als einer der bedeutendsten frühen Vertreter der Chronofotografie.

Leben

Muybridge besuchte die Lateinschule in Kingston upon Thames und fand eine erste Anstellung bei der *London Printing and Publishing Co.*, für die er 1852 als Vertreter in die USA übersiedelte. Es ist nicht geklärt, wann und wo er seine fotografische Ausbildung erhielt. Anschließend fand er eine Beschäftigung bei dem Landschaftsfotografen Carleton E. Watkins. Später machte er sich selbständig und arbeitete für Thomas Houseworth und das Kriegsministerium der Vereinigten Staaten (*U. S. War Department*).
1867 machte Muybridge eine aufsehenerregende Fotoserie aus dem Yosemite-Nationalpark und wirkte als Dokumentarfotograf in Alaska.
1872 wurde er von Leland Stanford (Eisenbahn-Tycoon und kalifornischer Gouverneur) engagiert, um die exakte Beinstellung eines galoppierenden Pferdes zu bestimmen. Damit begründete er die Serienfotografie mit komplexen Aufbauten, bestehend aus 12, 24 und schließlich 36 sukzessive auslösenden Fotoapparaten. So wurde erstmals der sichtbare Beweis erbracht, dass sich beim galoppierenden Pferd zeitweise alle vier Beine in der Luft befinden. Bei seinen Serienaufnahmen von Trabern und Galoppern berührten die Pferde einzelne, quer zur Pferderennbahn gespannte Zugdrähte, wodurch sich die elektrisch betriebenen Hochgeschwindigkeitsverschlüsse der nebeneinander aufgestellten Kameras kurzzeitig öffneten. 1879 erfand Muybridge das Zoopraxiskop zur Präsentation seiner Reihenaufnahmen, das die in Einzelbilder zerlegte Bewegung einem Kinofilm ähnlich synthetisierte. Im Jahr 1881 veröffentlichte Muybridge seine berühmten Serienaufnahmen unter dem Titel *The attitudes of animals in motion* in Form von Albuminpapierabzügen.
1874 ermordete Muybridge den Liebhaber seiner Frau, den Theaterkritiker Harry Larkins,[1] er wurde jedoch wegen „entschuldbaren Mordes“ freigesprochen (Larkins-Affaire). Anschließend begab er sich auf eine Foto-Expedition nach Mittelamerika, wo er den Isthmus von Panama dokumentierte und chemische und verschlusstechnische Experimente durchführte. 1876 kehrte er nach San Francisco zurück. Am 8. Mai 1904 starb er im Alter von 74 Jahren in Kingston upon Thames.







Copyright, 1878, by MUYBRIDGE.

MORSE'S Gallery, 417 Montgomery St., San Francisco.

THE HORSE IN MOTION.

Illustrated by

MUYBRIDGE.

AUTOMATIC ELECTRO-PHOTOGRAPH.

"SALLIE GARDNER," owned by LELAND STANFORD; running at a 1.40 gait over the Palo Alto track, 19th June, 1878.

The negatives of these photographs were made at intervals of twenty-seven inches of distance, and about the twenty-fifth part of a second of time; they illustrate consecutive positions assumed in each twenty-seven inches of progress during a single stride of the mare. The vertical lines were twenty-seven inches apart; the horizontal lines represent elevations of four inches each. The exposure of each negative was less than the two-thousandth part of a second.

Simulation

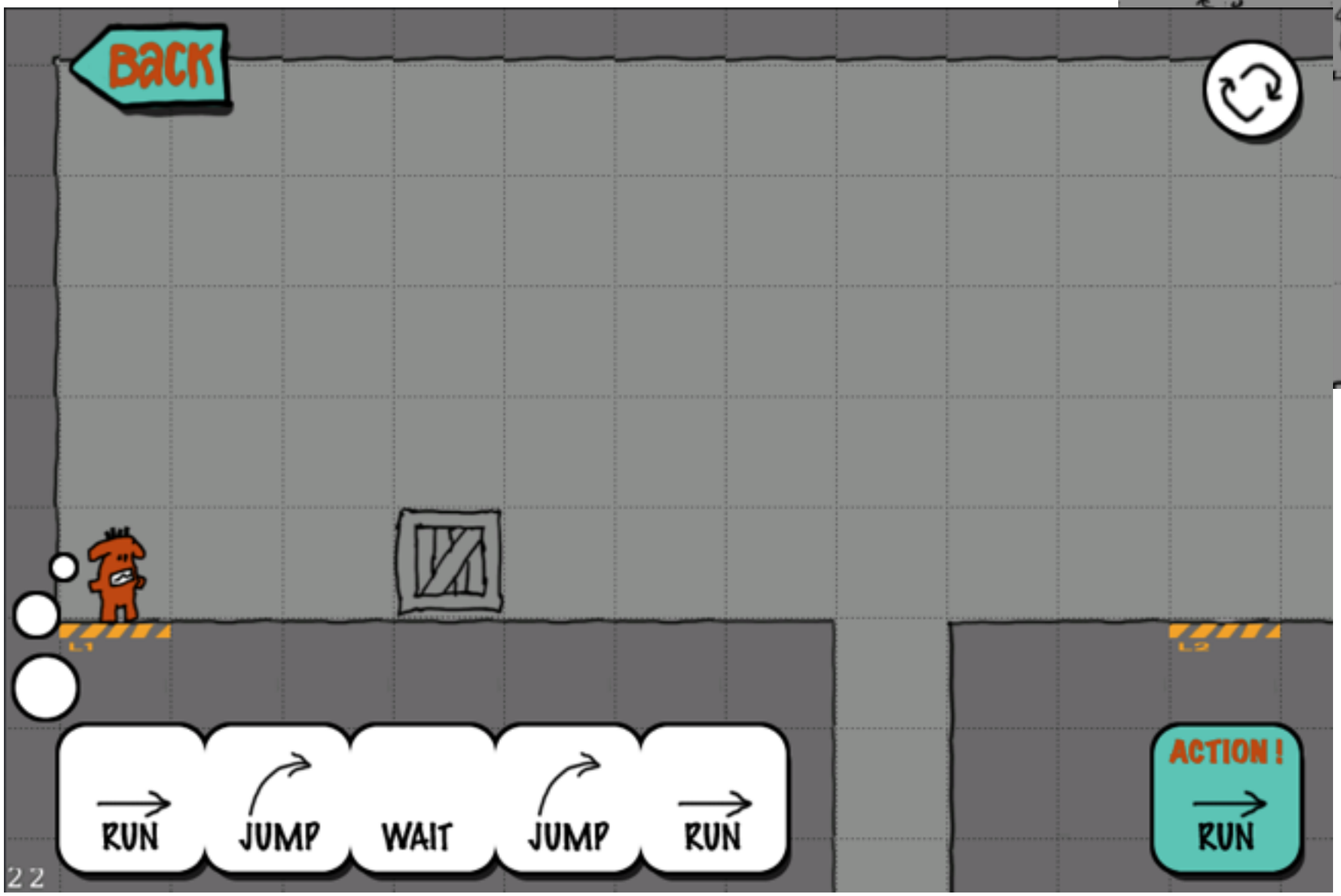
(von lateinisch *simulātio* „Schein, Verstellung, Täuschung“)

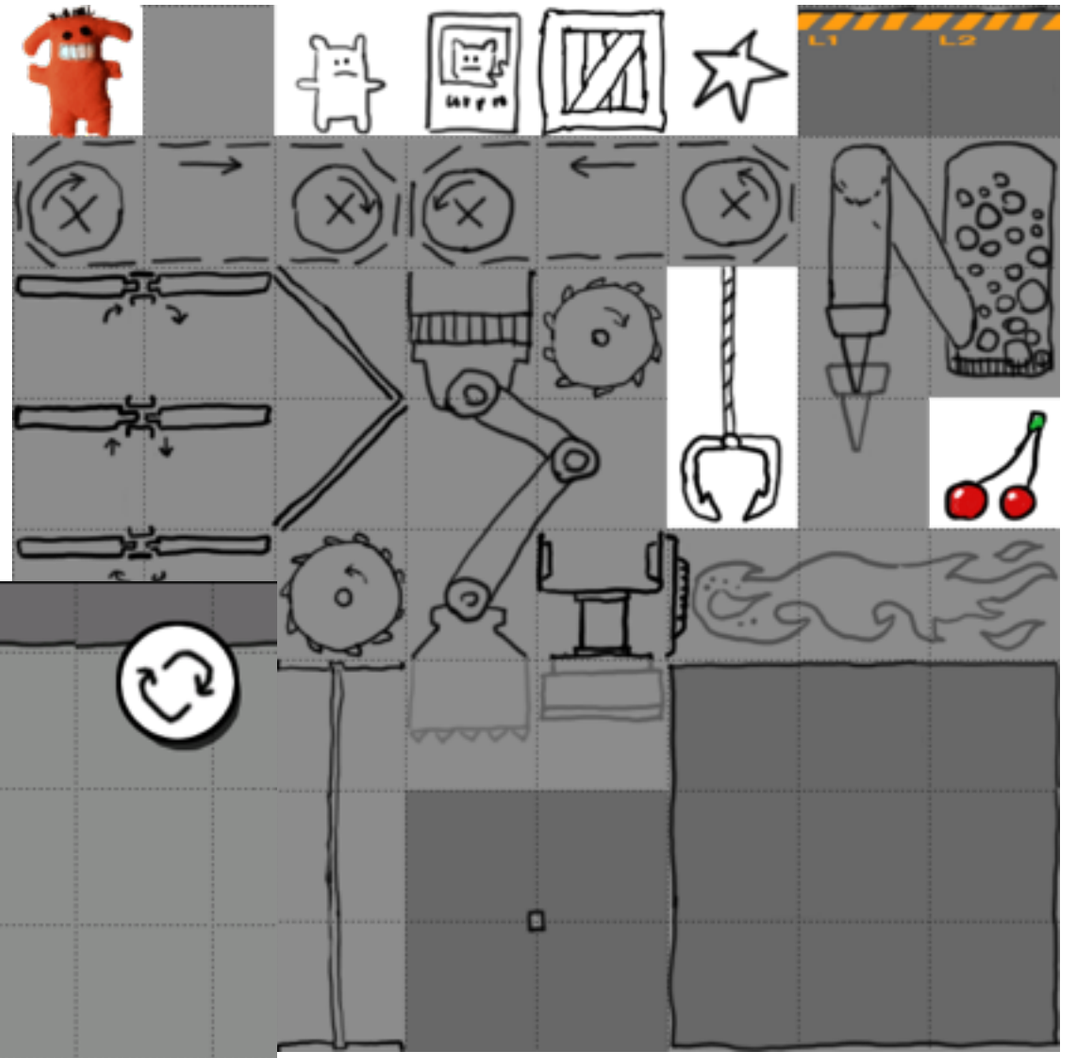
Die **Simulation** oder **Simulierung** ist eine Vorgehensweise zur Analyse von Systemen, die für die theoretische oder formelmäßige Behandlung zu komplex sind. Dies ist überwiegend bei *dynamisch*em Systemverhalten gegeben. Bei der Simulation werden Experimente an einem Modell durchgeführt, um Erkenntnisse über das reale System zu gewinnen. Im Zusammenhang mit Simulation spricht man von dem zu simulierenden System und von einem Simulator als *Implementierung* oder *Realisierung* eines Simulationsmodells. Letzteres stellt eine Abstraktion des zu simulierenden Systems dar (Struktur, Funktion, Verhalten). Der Ablauf des Simulators mit konkreten Werten (Parametrierung) wird als Simulationsexperiment bezeichnet. Dessen Ergebnisse können dann interpretiert und auf das zu simulierende System übertragen werden.

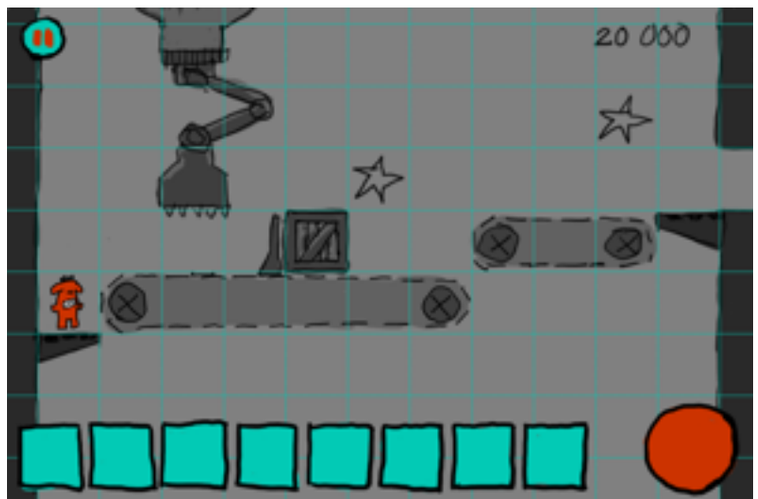
Deswegen ist der erste Schritt einer Simulation stets die Modellfindung. Wird ein neues Modell entwickelt, spricht man von Modellierung. Ist ein vorhandenes Modell geeignet, um Aussagen über die zu lösende Problemstellung zu machen, müssen lediglich die Parameter des Modells eingestellt werden. Das Modell, respektive die Simulationsergebnisse können dann für Rückschlüsse auf das Problem und seine Lösung genutzt werden. Daran können sich – sofern stochastische Prozesse simuliert wurden – statistische Auswertungen anschließen.

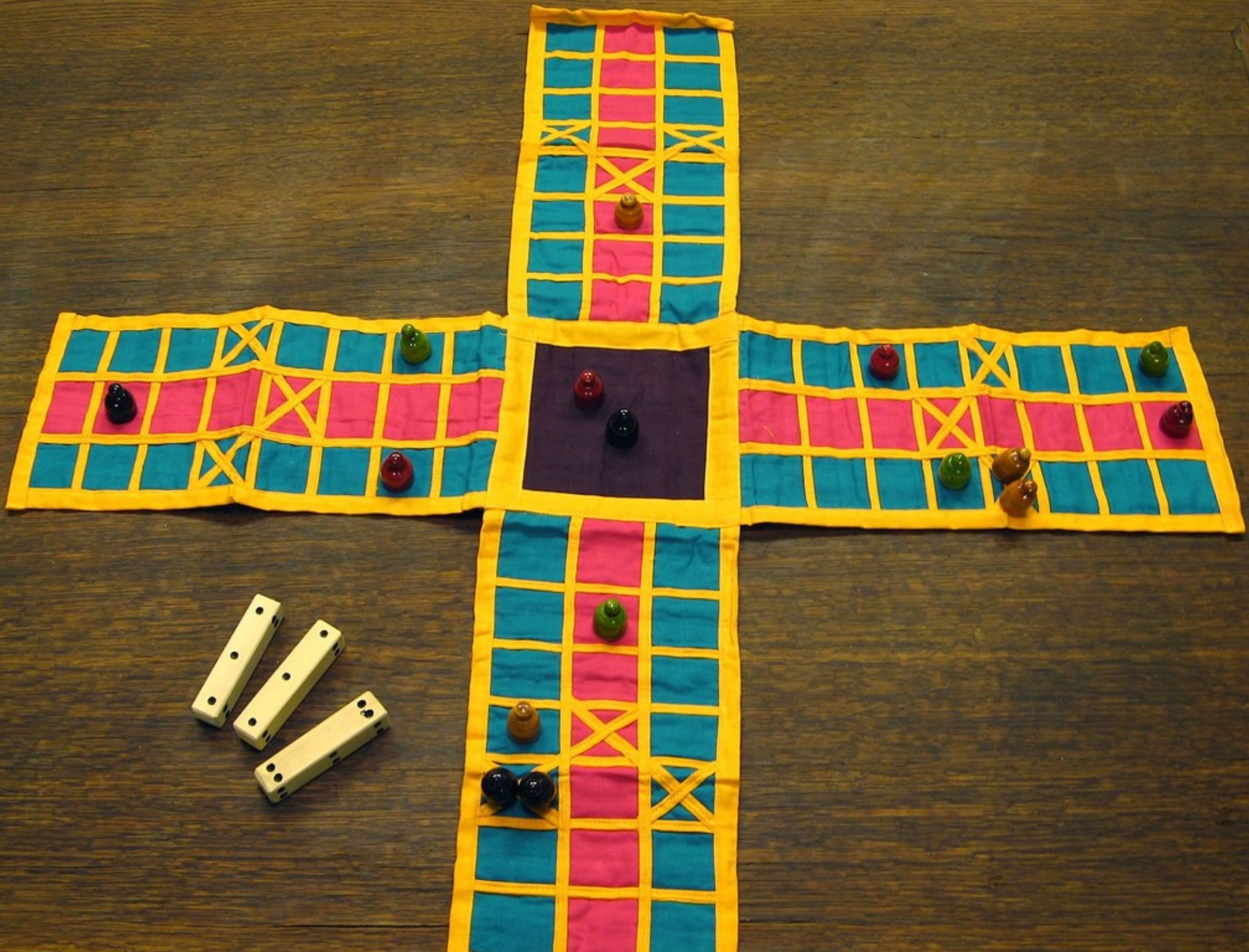
Die Methode der Simulation wird für viele Problemstellungen der Praxis eingesetzt. Bekannte Felder des Einsatzes von Simulationen sind die Strömungs-, Verkehrs-, Wetter- und Klimasimulation.


787
THALES


BACK
RUN
JUMP
WAIT
JUMP
RUN
ACTION !
RUN
L1
L2


L1
L2


20 000



Arten der Simulation

Statische Simulation
In der statischen Simulation spielt die Zeit keine Rolle. Das Modell ist statisch, d. h., es betrachtet nur einen Zeitpunkt, ist also quasi eine Momentaufnahme.

Monte-Carlo-Simulation
Fußt die Simulation auf Zufallszahlen und/oder Stochastik (Wahrscheinlichkeitsmathematik), so spricht man wegen der begrifflichen Nähe zum Glücksspiel von Monte-Carlo-Simulation. Diese Methode hat besonders in der Physik wichtige Anwendungen gefunden, und zwei Bücher ein-und-desselben Autors gehören zu den meistzitierten Veröffentlichungen in dieser Wissenschaftssparte.[1]

Dynamische Simulation
Für die Modelle der dynamischen Simulation spielt die Zeit immer eine wesentliche Rolle. Die dynamische Simulation betrachtet Prozesse bzw. Abläufe.

Kontinuierliche Simulation
Bei der kontinuierlichen Simulation werden stetige Prozesse abgebildet. Diese Art der Simulation nutzt Differentialgleichungen zur Darstellung physikalischer oder biologischer Gesetzmäßigkeiten, welche dem zu simulierenden Prozess zugrunde liegen.

Diskrete Simulation
Die diskrete Simulation benutzt die Zeit, um nach statistisch oder zufällig bemessenen Zeitintervallen bestimmte Ereignisse hervorzurufen, welche ihrerseits den (nächsten) Systemzustand bestimmen.
Auch als Ablaufsimulation oder ereignisgesteuerte Simulation bezeichnet, findet die diskrete Simulation im Produktions- und logistischen Bereich ihre hauptsächliche Anwendung. Der weit überwiegende Teil der Praxisprobleme liegt in diesem Bereich. Die Modelle dieser Simulation sind im Gegensatz zu den kontinuierlichen gut mit standardisierten Elementen (z. B. Zufallszahlen, Warteschlangen, Wahrscheinlichkeitsverteilungen usw.) darstellbar. Einen weiteren leistungsfähigen Ansatz zur Entwicklung diskreter, ereignisgesteuerter Modelle bietet die Petri-Netz-Theorie.
Die Stärke der diskreten Simulation liegt darin, dass sie den *Zufall* bzw. die *Wahrscheinlichkeit* in das Modell mit einbezieht und bei genügend häufiger Durchrechnung eine Aussage über die zu erwartende Wahrscheinlichkeit der verschiedenen Systemzustände liefert. Das Anwendungsfeld für diese Art der Simulation ist daher entsprechend groß:

- Arbeitsabläufe in der Produktion (alle Automobilhersteller sind große Simulationsanwender)
- Prozesse der Logistik (Supply-Chains, Container-Umschlag usw.)
- Abläufe mit großem Personen- oder Güter-Aufkommen (Flughäfen, Großbahnhöfe, aber auch Autobahn-Mautstellen, öffentliche Verkehrssysteme, Post-Verteilzentralen, Verschiebebahnhöfe usw.)

Hybride Simulation
Von *hybrider Simulation* spricht man dann, wenn das Modell sowohl Eigenschaften der kontinuierlichen als auch der diskreten Simulation aufweist. Derartige Modelle finden sich beispielsweise in medizinischen Simulationen - insbesondere zu Ausbildungszwecken - wieder, bei denen die zu simulierende Biologie nicht hinreichend bekannt ist, um ein ausreichend detailliertes, kontinuierliches Modell erstellen zu können.

System Dynamics
Unter Systemdynamik wird die Simulation

- komplexer,
- zeitdiskreter,
- nicht linearer,
- dynamischer und
- *rückgekoppelter*

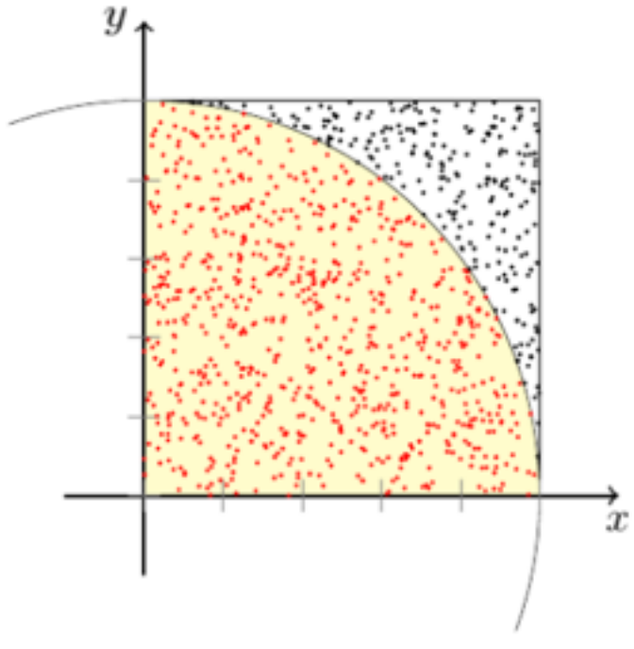
Systeme verstanden. Im Wesentlichen werden unter solchen Simulatoren

- das Rückkopplungsverhalten sozioökonomischer Systeme („Industrial Dynamics“),
- die Entwicklung von Ballungszentren („Urban Dynamics“) und
- Weltmodelle, wie z. B. für den Club of Rome („World Dynamics“)

subsumiert. Die Arbeitsweisen und Werkzeuge entsprechen nahezu zur Gänze denen der Regelungstechnik bzw. der Kybernetik.

Multi Agenten Simulation
Die Multi-Agenten-Simulation, die als Spezialfall der diskreten Simulation gesehen werden kann, erlaubt, emergente Phänomene und dynamische Wechselwirkungen zu modellieren.

Monte-Carlo-Simulation



Viertelkreis, dessen Fläche durch die Monte-Carlo-Methode angenähert wird. Pi ist genau das Vierfache der Wahrscheinlichkeit, mit der ein Punkt in den Kreis fällt. Damit lässt sich eine Näherung von Pi bestimmen.

Monte-Carlo-Simulation oder **Monte-Carlo-Studie**, auch **MC-Simulation**, ist ein Verfahren aus der Stochastik, bei dem sehr häufig durchgeführte Zufallsexperimente die Basis darstellen. Es wird dabei versucht, mit Hilfe der Wahrscheinlichkeitstheorie analytisch nicht oder nur aufwendig lösbare Probleme numerisch zu lösen. Als Grundlage ist vor allem das Gesetz der großen Zahlen zu sehen. Die Zufallsexperimente können entweder – etwa durch Würfeln – real durchgeführt werden oder durch Erzeugung von geeigneten Zufallszahlen. Computergenerierte Vorgänge können den Prozess in ausreichend häufigen Zufallsereignissen simulieren.
Zu den Pionieren der Monte-Carlo-Methode in den 1940er Jahren gehören Stanislaw Ulam, Nicholas Metropolis und John von Neumann.

- Processing
- Arduino
- openFrameworks
- Unity3D
- VVVV
- Flash
- Cocos2D

- Java
- Javascript
- C
- C++
- C#
- Objective-C
- Swift

- openGL
- DirectX
- GL ES
- WebGL
- GLSL
- openCL

- Box2D
- ChipMunk
- Kinekt
- openCV

Objektorientierte Programmierung

Die **objektorientierte Programmierung** (kurz **OOP**) ist ein auf dem Konzept der Objektorientierung basierendes Programmierparadigma. Die Grundidee besteht darin, die Architektur einer Software an den Grundstrukturen desjenigen Bereichs der Wirklichkeit auszurichten, der die gegebene Anwendung betrifft. Ein Modell dieser Strukturen wird in der Entwurfsphase aufgestellt. Es enthält Informationen über die auftretenden Objekte und deren Abstraktionen, ihre Typen. Die Umsetzung dieser Denkweise erfordert die Einführung verschiedener Konzepte, insbesondere Klassen, Vererbung, Polymorphie und spätes Binden.

Konzepte

Im Vergleich mit anderen Programmiermethoden verwendet die objektorientierte Programmierung neue, andere Begriffe.
Die einzelnen Bausteine, aus denen ein objektorientiertes Programm während seiner Abarbeitung besteht, werden als Objekte bezeichnet. Die Objekte werden dabei in der Regel auf Basis der folgenden Paradigmen konzipiert:

Abstraktion
Jedes Objekt im System kann als ein abstraktes Modell eines *Akteurs* betrachtet werden, der Aufträge erledigen, seinen Zustand berichten und ändern und mit den anderen Objekten im System kommunizieren kann, ohne offenlegen zu müssen, wie diese Fähigkeiten implementiert sind (vgl. abstrakter Datentyp (ADT)). Solche Abstraktionen sind entweder Klassen (in der klassenbasierten Objektorientierung) oder Prototypen (in der prototypbasierten Programmierung).

Klasse
Die Datenstruktur eines Objekts wird durch die Attribute (auch *Eigenschaften*) seiner Klassendefinition festgelegt. Das Verhalten des Objekts wird von den Methoden der Klasse bestimmt. Klassen können von anderen Klassen *abgeleitet* werden (Vererbung). Dabei erbt die Klasse die Datenstruktur (*Attribute*) und die Methoden von der *vererbenden* Klasse (Basisklasse).

Prototyp
Objekte werden durch das Klonen bereits existierender Objekte erzeugt und können anderen Objekten als Prototypen dienen und damit ihre eigenen Methoden zur Wiederverwendung zur Verfügung stellen, wobei die neuen Objekte nur die Unterschiede zu ihrem Prototyp definieren müssen. Änderungen am Prototyp werden dynamisch auch an den von ihm abgeleiteten Objekten wirksam.

Datenkapselung
Als Datenkapselung bezeichnet man in der Programmierung das Verbergen von Implementierungsdetails. Auf die interne Datenstruktur kann nicht direkt zugegriffen werden, sondern nur über definierte Schnittstellen. Objekte können den internen Zustand anderer Objekte nicht in unerwarteter Weise lesen oder ändern. Ein Objekt hat eine Schnittstelle, die darüber bestimmt, auf welche Weise mit dem Objekt interagiert werden kann. Dies verhindert das Umgehen von Invarianten des Programms.

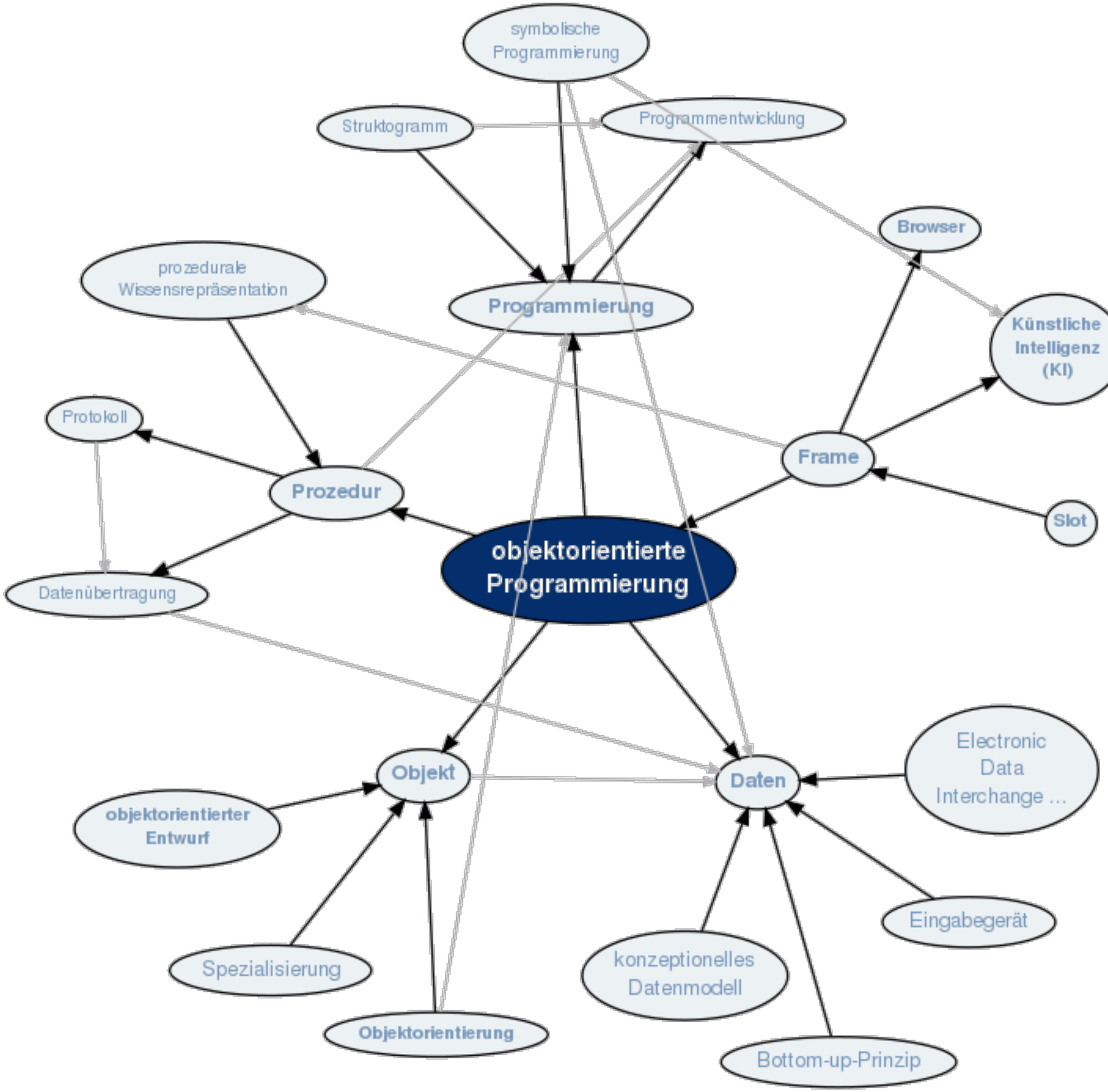
Feedback
Verschiedene Objekte kommunizieren über einen Nachricht-Antwort-Mechanismus, der zu Veränderungen in den Objekten führt und neue Nachrichtenaufrufe erzeugt. Dafür steht die Kopplung als Index für den Grad des Feedbacks.

Persistenz
Objektvariablen existieren, solange die Objekte vorhanden sind und „verfallen“ nicht nach Abarbeitung einer Methode.

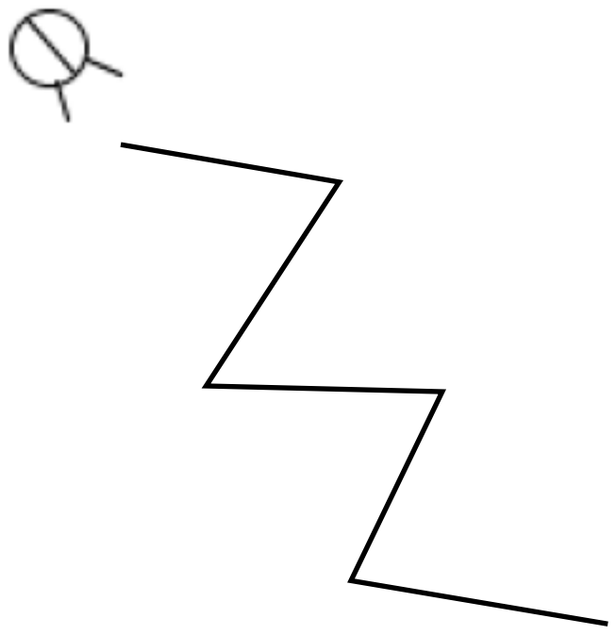
Polymorphie
Fähigkeit eines Bezeichners, abhängig von seiner Verwendung unterschiedliche Datentypen anzunehmen. Verschiedene Objekte können auf die gleiche Nachricht unterschiedlich reagieren. Wird die Zuordnung einer Nachricht zur Reaktion auf die Nachricht erst zur Laufzeit aufgelöst, wird dies auch späte Bindung genannt.

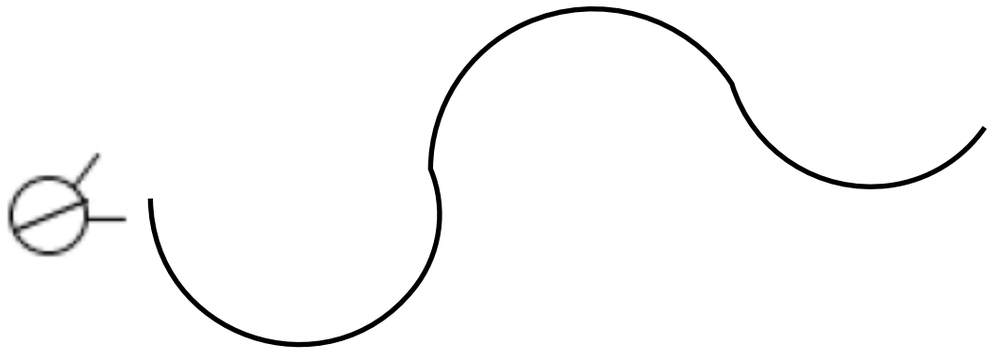
Vererbung
Vererbung heißt vereinfacht, dass eine abgeleitete Klasse die Methoden und Attribute der Basisklasse ebenfalls besitzt, also „erbt“. Somit kann die abgeleitete Klasse auch darauf zugreifen. Neue Arten von Objekten können auf der Basis bereits vorhandener Objektdefinitionen festgelegt werden. Es können neue Bestandteile hinzugenommen werden oder vorhandene überlagert werden.


MICHAEL
JACKSON
EARTH SONG


symbolische Programmierung
Struktogramm
Programmentwicklung
Browser
prozedurale Wissensrepräsentation
Programmierung
Künstliche Intelligenz (KI)
Protokoll
Frame
Prozedur
Slot
objektorientierte Programmierung
Datenübertragung
Objekt
Daten
Electronic Data Interchange ...
objektorientierter Entwurf
Eingabegerät
Spezialisierung
konzeptionelles Datenmodell
Objektorientierung
Bottom-up-Prinzip

Aufgabe





main class

```
ArrayList<Kaefer> liste = new ArrayList<Kaefer>();

void setup() {

  size(1200, 800);
  for(int i=0; i<10; i++) liste.add(new meinKaefer());

}

void draw() {

  background(255);
  for (Kaefer k : liste) k.draw();
}
```

Kaefer class

```
class Kaefer {

  PVector position;
  PVector direction;
  float speed;

  Kaefer() {
    position = new PVector(random(width), random(height));
    direction = new PVector(0, 1);
    speed = 0;
  }

  void draw() {

    fill(100);
    ellipse(position.x, position.y, 20, 20);
  }
}
```

meinKaefer class

```
class meinKaefer extends Kaefer {

   // …
  meinKaefer() {

      super();
      // …
   }

   void draw() {

        // …
   }

  // …
}
```