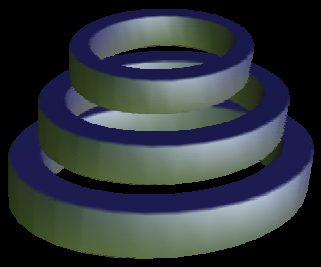
Projekt: Der Turm von Hanoi

als Java Applet...



Monika Wojtowiec
Michael Gebhard
STephan Kambor

Dauer ca. 20 min

Version 1.01

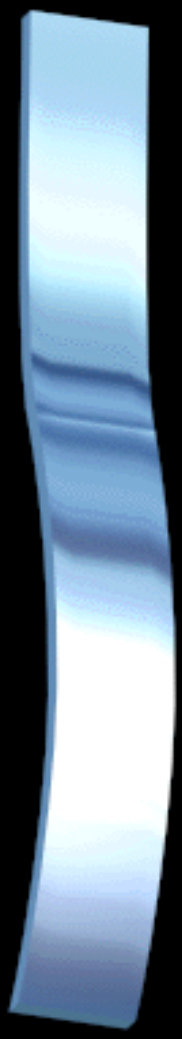
Gliederung

~> 1. Aufgabenstellung

~> 2. Das Spiel
- Geschichte
- Regeln
- Prinzip

~> 3. Die Lösung

~> 4. Die Umsetzung



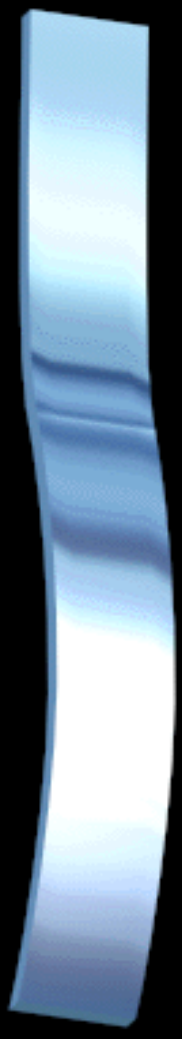
1. Aufgabenstellung

~> Entwicklung als Java Applet

~> Problem soll für Turm mit n Scheiben gelöst werden

~> graphische Darstellung

~> Ablauf der Lösung
animiert
per Vor- bzw. Zurücktasten



2. Das Spiel

Geschichte

~> Legende [1883] von französischem Mathematiker Eduard Lucas

~> in einem Tempel in der indischen Stadt Benares lagen 64 kostbare Scheiben aus Diamant zu einem Turm aufgeschichtet

~> Turm soll unter Beachtung heiliger Regeln umgeschichtet werden

~> dies klang so unwarscheinlich, dass man sagte: "Wenn es gelingt, wird der Tempel, *das Leben, das Universum und der ganze Rest* zu Staub zerfallen!"

2. Das Spiel

Die [heiligen] Regeln

~> Die Scheiben dürfen nur auf einem der drei festgelegten Plätze liegen.

~> Man darf immer nur eine Scheibe umlegen.

~> Man darf eine größere Scheibe nicht auf eine kleinere legen.

2. Das Spiel

Prinzip

Beispiel mit drei Scheiben ~> Ausgangssituation

2. Das Spiel

Prinzip

Beispiel mit drei Scheiben

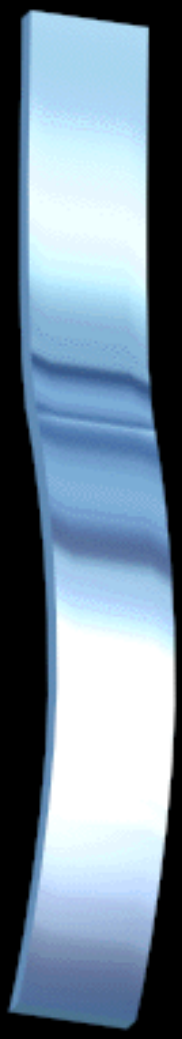
Die Scheiben müssen mit Hilfe des Zwischenlagers [rechts] und unter beachtung der Regeln von Platz 1 [links] auf Platz 2 [mitte] gebracht werden.

2. Das Spiel

Prinzip

Beispiel mit drei Scheiben ~> Endsituation

copyleft - all rights reversed

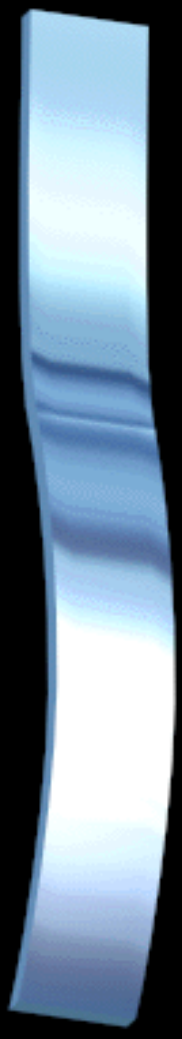


3. Die Lösung

~> Beweis das es möglich ist n Scheiben umzustapeln

~> Berechnung der Anzahl der Züge

~> Erklärung am besten mit dem Applet



3. Die Lösung

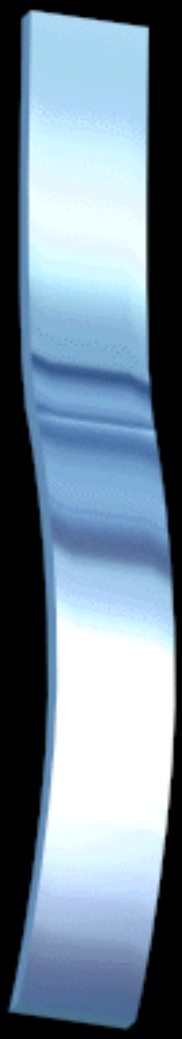
~> gleiches Prinzip egal wie viele Scheiben

~> wenn drei Scheiben dann auch vier

~> wenn vier Scheiben dann auch fünf usw.

~> Beweis durch vollständige Induktion

copyleft - all rights reversed

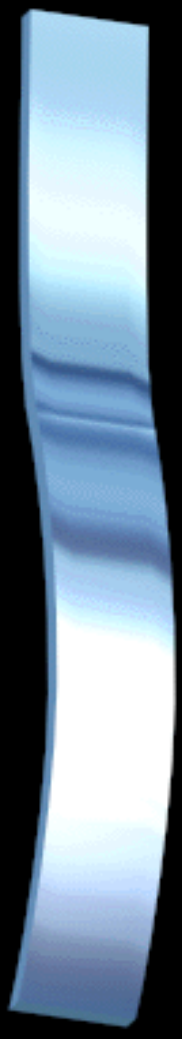


3. Die Lösung

Für 3 Scheiben 7 Züge.
Für 4 Scheiben 7+1+7=15 Züge.
Für 5 Scheiben 15+1+15=31 Züge.

Daraus ergibt sich die Formel:

$$\text{Züge zur Lösung} = 2^{\text{Scheibenanzahl}} - 1$$



3. Die Lösung

~> Beispiel mit 64 Scheiben

$$\sim> 2^{64} - 1 = 18.446.744.073.709.551.615$$

~> 18 Trillionen 446 Billiarden 744 Billionen 73 Milliarden 709 Millionen 551 Tausend 615 Züge

~> In einem Menschenleben nich zu schaffen

~> grob gerundet 585 Milliarden Jahre

4. Die Umsetzung

Erzeugung eines Strings mit der Lösung.

```
String Loesung(int anzahl,String quelle,String lager,String ziel)
{
 String loesung=new String();
 loesung = quelle+ziel;
 if (anzahl>1)
  loesung = Loesung(anzahl-1,quelle,ziel,lager)+loesung+Loesung(anzahl-1,lager,quelle,ziel);
 return loesung;
}
```


4. Die Umsetzung

Die Lösung wird durch angeben des Quell und des Zielstiftes des entsprechenden Schrittes berechnet.

Die Nummerierung beginnt mit Null.

01021201202101 ist in diesem Fall die Lösung für drei Scheiben.

Für das Programm heißt dieses, dass zuerst vom ersten auf den zweiten Stift, dann vom ersten zum dritten usw. geschoben wird.

So ein Schritt wird in der Methode *void demo (boolean go)* ausgeführt.

4. Die Umsetzung

In einzelnen Schritten wird der Lösungsstring generiert:

Aufruf mit
(3,“0“,“2“,“1“):
loesung=“01“
loesung=(2,“0“,“1“,“2“)+01+(2,“2“,“0“,“1“)
(2,“0“,“1“,“2“):
loesung=“02“
loesung=(1,“0“,“2“,“1“)+02+(1,“1“,“0“,“2“)
(1,“0“,“2“,“1“):
loesung=“01“
return “01“
(1,“1“,“0“,“2“):
loesung=“01“
return “12“
(2,“0“,“1“,“2“):
return “010212“

(2,“2“,“0“,“1“):
loesung=“21“
loesung=(1,“2“,“1“,“0“)+02+(1,“0“,“2“,“1“)
(1,“2“,“1“,“0“):
loesung=“20“
return “20“
(1,“0“,“2“,“1“):
loesung=“01“
return “01“
(2,“0“,“1“,“2“):
return “202101“
(3,“0“,“2“,“1“):
return “01021201202101“

4. Die Umsetzung

void demo(boolean go) führt damit immer einen Schritt aus, wobei *go* aussagt in welche Richtung, *true* für vorwärts *false* für rückwärts.

Die Buttons [Go] und [Back] rufen jeweils diese Methode einmal auf, falls der Turm nicht am Ende ist.

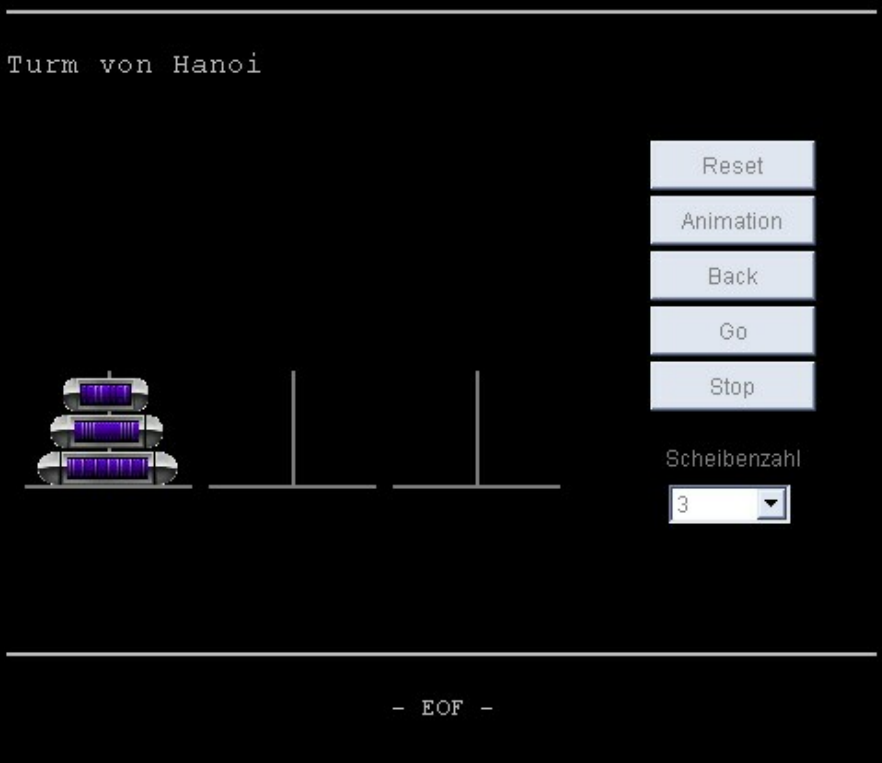
Der Button [Animation] erstellt, falls nicht schon durch vormaliges drücken geschehen, einen Thread der die Methode *demo* in vorwärts Richtung immer wieder aufruft. Er setzt die Laufvariable solange auf *true*, bis entweder durch [Reset] oder [Stop] die Laufvariable *false* gesetzt wird.

4. Die Umsetzung

Der Button [Reset] setzt den Turm in die Ausgangsposition zurück und aktualisiert die beteiligten Variablen. Die Anzahl der Scheiben wird auf Grundlage der Auswahl generiert.

Die Scheiben selbst werden durch entsprechende Größenanpassung des Grundbildes in paint() erzeugt.

Vielen Dank für die Aufmerksamkeit ...



copyleft - all rights reversed