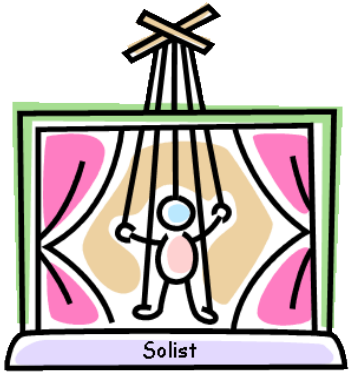


Solist

eine Entwicklungsumgebung für
Miniprogrammierwelt-Simulatoren

Dietrich Boles
Universität Oldenburg

MWS 2010 10.05.2010

Gliederung

- Miniprogrammierwelten
  - Beispiele
  - Charakteristika
  - Vorteile
- Solist
  - Aufbau
  - Demo
  - Potential
- Fazit und Ausblick

Miniprogrammierwelten

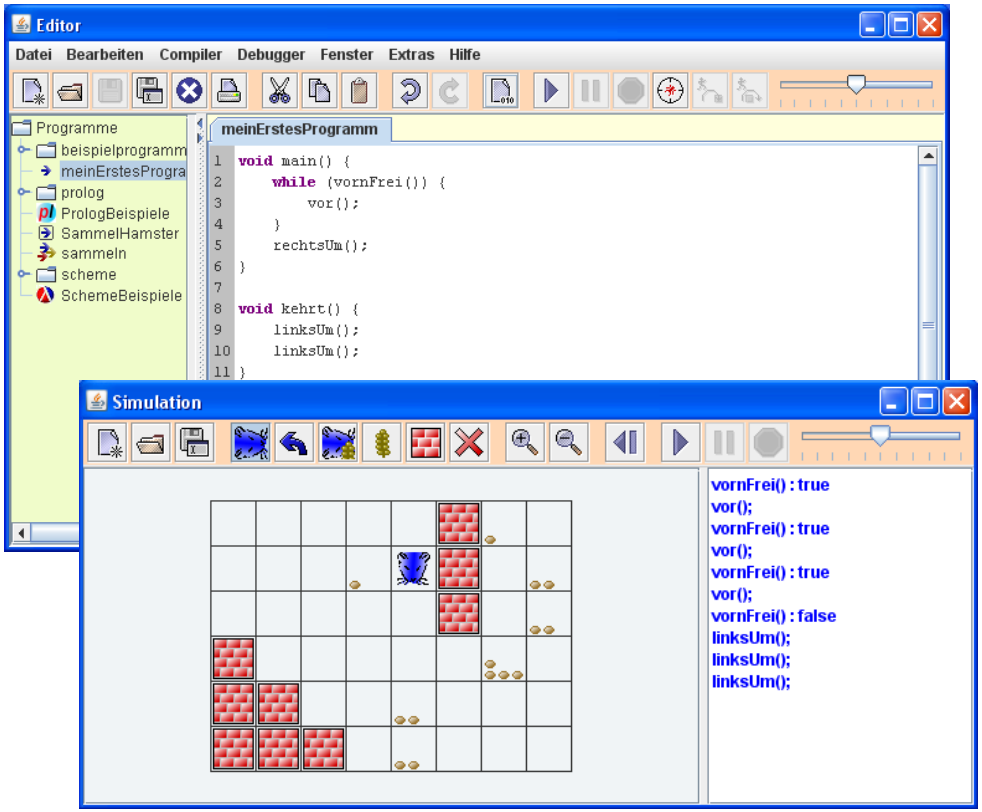
- „micro worlds“
- Lernumgebungen für Programmieranfänger
- Fokus ist die Algorithmik
- Seit 70er Jahren

- Bestandteile:
  - (didaktisches) Modell
  - Programmiersprache
  - Simulator

Miniprogrammierwelten

- Beispiele:
  - Hamster-Modell

```
void vor();
void linksUm();
void gib();
void nimm();
boolean vornFrei()
boolean kornDa()
boolean maulLeer()
```



  - Kara, der Marienkäfer
  - Robot Karol
  - Turtle-Graphics

Miniprogrammierwelten

- Charakteristika:
  - Virtuelle Welt / Bühne
  - Virtueller Akteur (Hamster, Marienkäfer, …)
  - Virtuelle Requisiten (Mauern, Kleeblätter, …)
  - Einige wenige vordefinierte Befehle

  - Einsatz:
    - Lehrer: Aufgaben stellen
    - Schüler: Programm entwickeln (Steuerung des Akteurs)

Miniprogrammierwelten

- Vorteile:
  - Schnelle Erfolgserlebnisse
  - Komplexitätsreduktion
  - Graphische Repräsentation
  - Visualisierung von Anweisungen
  - Selbstständiges Lernen möglich

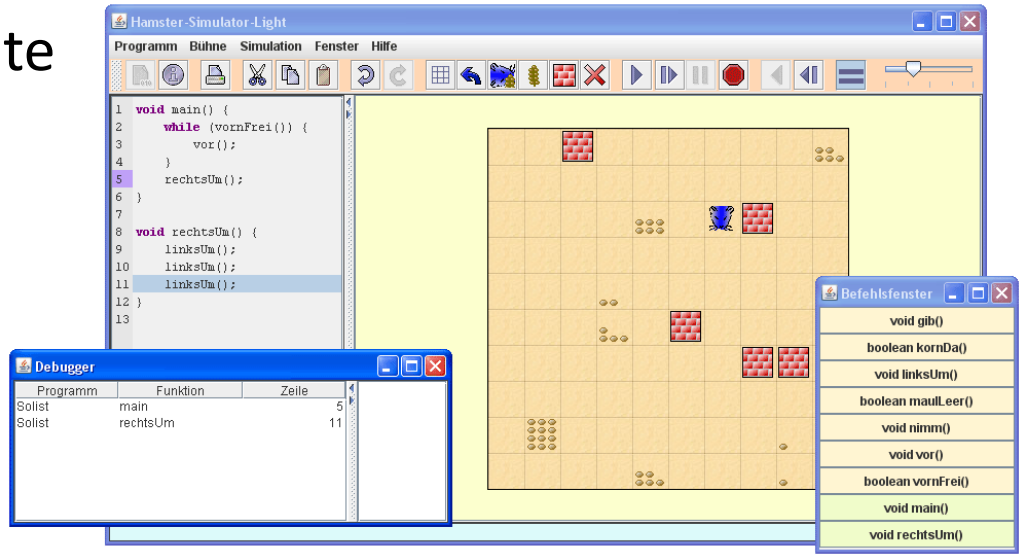
Solist

- Werkzeug zur Entwicklung neuer MPWs
- `http://www.programmierkurs-java.de/solist`

- Theater-Metapher
- Bestandteile:
  - Graphisch-interaktive Entwicklungsumgebung
  - Vordefinierte Klassenbibliothek (Theater-API)
- Einsatz-Szenario:
  - Lehrer entwickeln neue MPWs bzw. passen existierende MPWs an
  - Lehrer generieren Simulatoren
  - Schüler nutzen Simulatoren

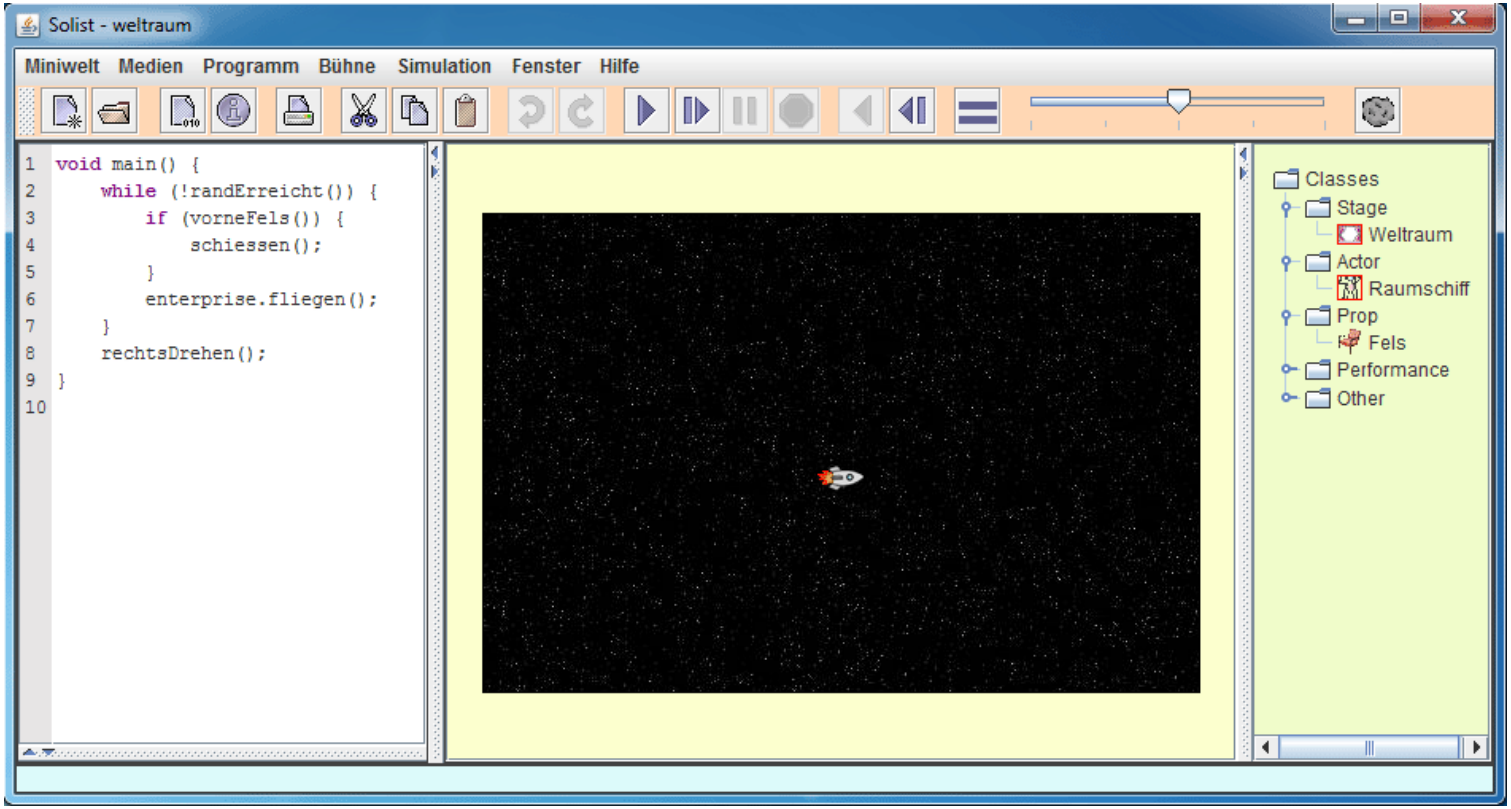
Solist

- Generierte Simulatoren:
  - „Bühnen-Gestalter“
  - Interaktiver Interpreter
  - Editor
  - Compiler
  - Simulationskomponente
  - Debugger



Solist

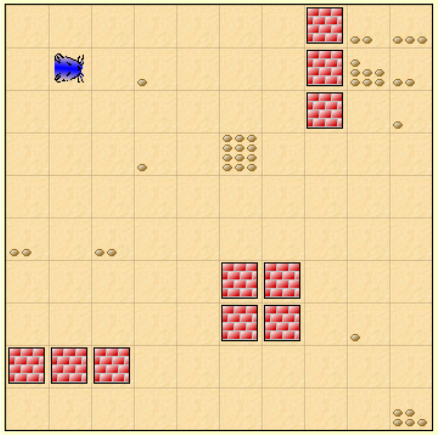
- Demo


Solist - weltraum
Miniwelt Medien Programm Bühne Simulation Fenster Hilfe
1 void main() {
2 while (!randErreicht()) {
3 if (vorneFels()) {
4 schiessen();
5 }
6 enterprise.fliegen();
7 }
8 rechtsDrehen();
9 }
10
Classes
Stage
Weltraum
Actor
Raumschiff
Prop
Fels
Performance
Other

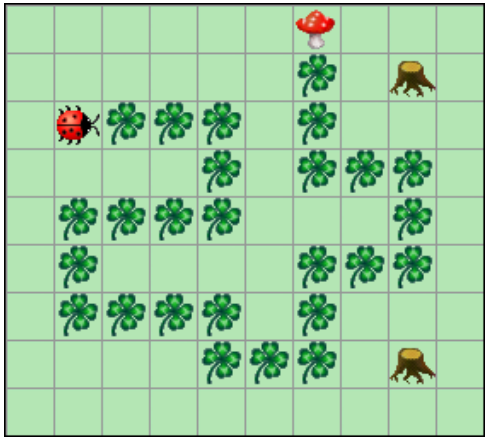
Solist

- Fertige „Plays“ (klassische MPWs):

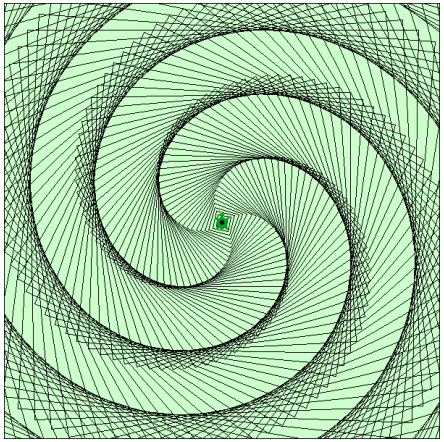
Hamster



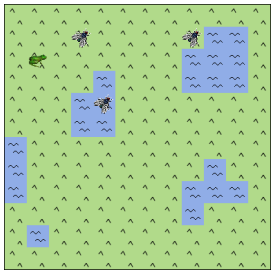
Kara



Turtle-Graphics



Frosch

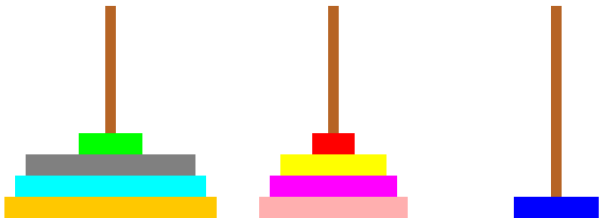


Geisterstunde

Solist

- Weiterführende MPWs
- → Algorithmenvisualisierung

Türme von Hanoi



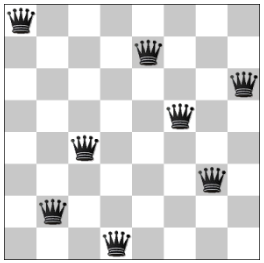
Befehle:

```
int anzahlScheiben()

boolean istLeer(int turm)

int obersteScheibe(int turm)

void verschiebe(int von, int nach)
```

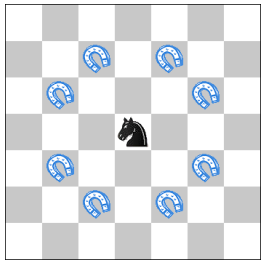
Solist

- Rekursion / Backtracking

Damen



Springer



Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

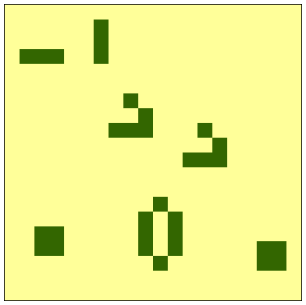
Befehle:

```
boolean istBelegt(int r, int s)

int welcheZahl(int r, int s)

void loeschen(int r, int s)

void setzen(int r, int s, int zahl)
```

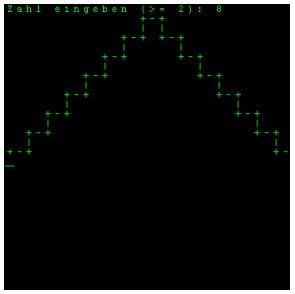
Solist

- Spiele / Simulationen

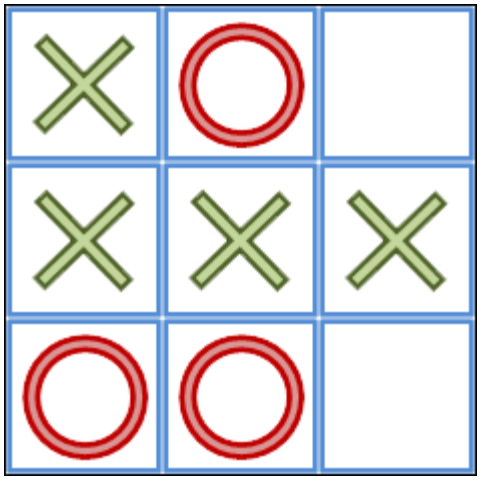
Game of Life



Terminal



TicTacToe



Befehle:

```
void ziehen(Spielzug zug)

Spielzug warten(int spieler)
```

Solist

- Optimierungsprobleme

Früchte

Fazit und Ausblick

- Solist: einfache Entwicklung neuer MPWs
  - → Anpassung an spezifische Anforderungen
  - → Abwechslung im Unterricht
  - → unterschiedliche Schwierigkeitsgrade
- Zukunft:
  - Unterstützung anderer Programmiersprachen (Python, Ruby, Scratch)
- Weitere Theater-Werkzeuge:
  - Objekt-Theater (→ Greenfoot mit Theater-API)
  - Threadnocchio (→ parallele Programmierung)

Ende

Danke für Ihre Aufmerksamkeit!

Anmerkungen, Fragen?