



Programmierpraktikum 2007

Einführungsveranstaltung

Mittwoch, 11 Apr 7

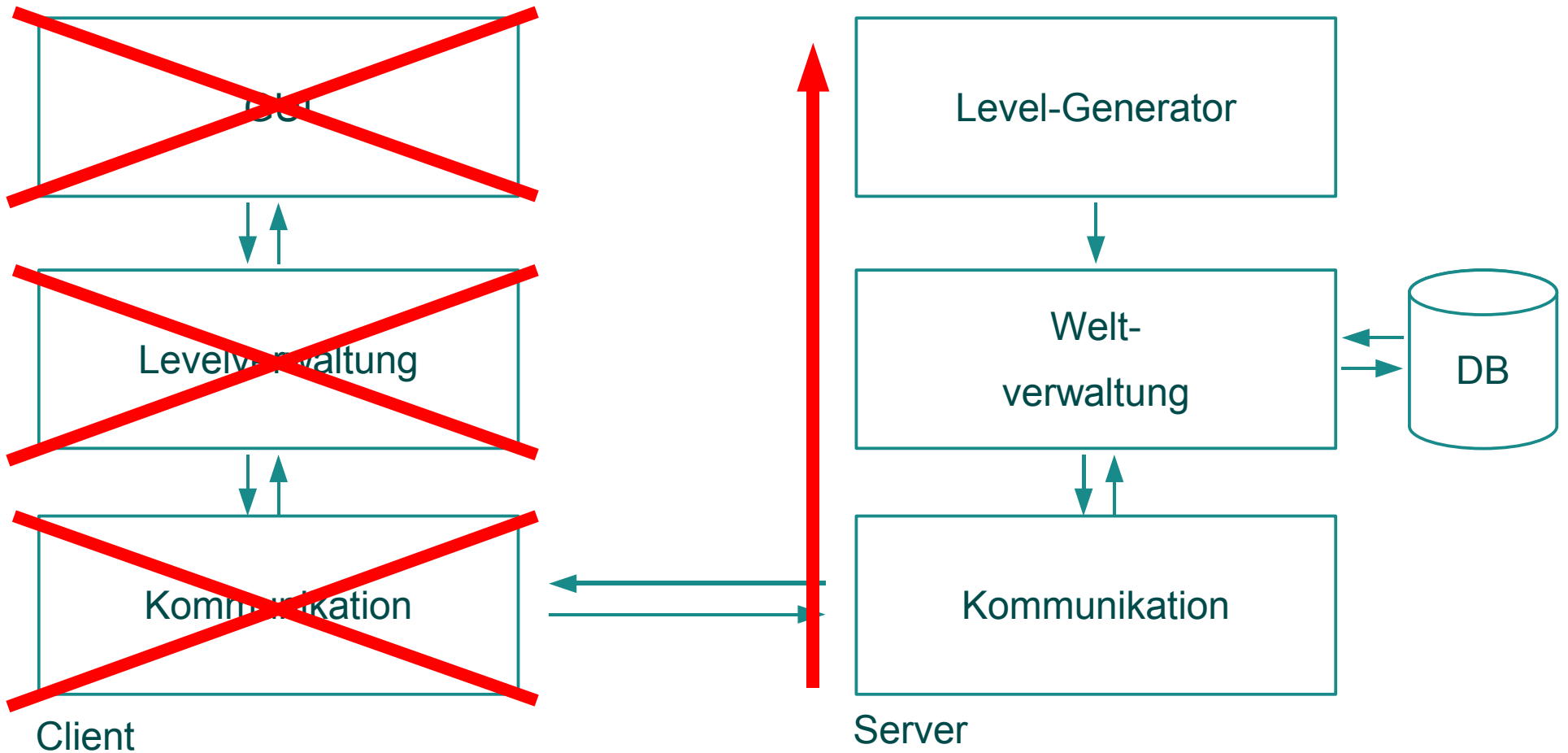
ullrich@informatik.uni-koeln.de



1. Features des Servers
 - 1.1. Kommunikation
 - 1.2. Weltverwaltung
 - 1.3. Level-Generator
2. Anforderungen an den Code
3. Gruppenbildung



Architektur





1. Features des Servers

1.1. Kommunikation



Server – Kommunikation

Aufgaben:

- Verbindet Client mit Server-Weltverwaltung
- Reicht Client-Anfragen an Server weiter
- Schickt Antworten/Level-Updates an die Clients
- Hält Kontakt mit bis zu 60 Clients gleichzeitig (Hausnummer!)



Server – Kommunikation II

Konkreter:

- Verschickt und empfängt Message-Klassen
- Kernklassen sind für Sie implementiert
- Client/Server-Paper erklärt C/S-Kommunikation in Java
- Internet-Kommunikation in Java sehr komfortabel realisierbar



1. Features des Servers

1.2. Weltverwaltung



Server – Weltverwaltung

Aufgaben:

- › Hält aktuell besuchte Levels im Speicher
- › Führt Buch über die eingeloggten Spieler
- › Hält Verbindung zur Datenbank (--> JDBC/SQL-Tutorium)
- › Bewegt Monster (KI)
- › Berechnet Spieler-Aktionen
- › Berechnet Kämpfe (--> Paper)



Spieler-Aktion: Beispiel

Spieler-Kommando kommt herein:

```
user#12: take item at (1, 10, 12)
```

Konsequenzen:

- Item wird identifiziert (Item Nr. 123, „Strohhut“)
- Item wird aus Level entfernt
- Item wird in Inventar des Spielers Nr. 12, „Sebastian“, aufgenommen
- Update-Message an alle „Item ist weg, bitte Level neu zeichnen!“



Spieler-Aktionen

Typen von Spieler-Aktionen:

- › Laufen (in acht Richtungen)
- › Monster angreifen
- › Item-Aktionen (*take, use, drop*)
- › Chatnachrichten

Weltverwaltung ist umfangreiche Komponente, aber algorithmisch und vom Lernaufwand her relativ simpel.



Datenbank

Sehr kleine Datenbank - Zwei Tabellen:

- Spieler
- Items in Spieler-Inventar

Später evtl. noch Zusatztabelle für Monstertypen.

Auswertung mit SQL-Abfragen, z.B.:

```
SELECT name, vorname FROM personen WHERE plz=50969;
```



1. Features des Servers

1.3. Levelgenerator





Labyrinth-Generator

Aufgaben

- › Erzeugt Levelstrukturen aus Zufallskeim
- › Setzt Ausgänge nach oben und unten
- › Verteilt Items
- › Verteilt Monster

Ziel: Labyrinth muss vollständig und zusammen hängend sein; also keine Inseln!

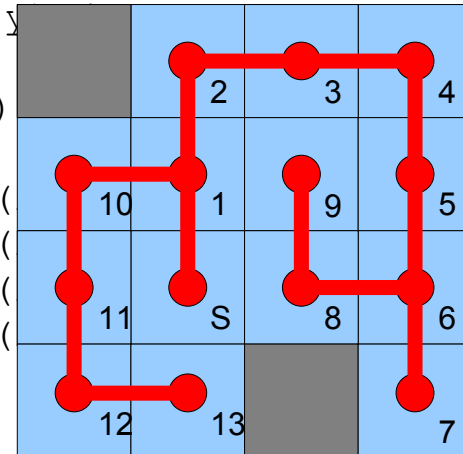
Hinweis: Wir verwenden den Begriff Labyrinth – fälschlicherweise – synonym mit Irrgarten



FloodFill-Algorithmus

Kern des Labyrinth-Generators: FloodFill-Algorithmus

```
private void floodFill(int x, int y)
    // Fels wird weg gehackt
    map[x][y].setType(Map.Clear)
    // Rekursiv Wege schlagen
    if (map[x-1][y].isRock()) floodFill(x-1, y)
    if (map[x][y+1].isRock()) floodFill(x, y+1)
    if (map[x+1][y].isRock()) floodFill(x+1, y)
    if (map[x][y-1].isRock()) floodFill(x, y-1)
}
```



Siehe auch: Paper



FloodFill-Algorithmus III – Randomized FloodFill (RFF)

- Übergang von festgelegter Richtungsreihenfolge zu bei jedem Schritt neu ausgewürfelten (Pseudo-Zufallsfolge)

```

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X                                     X X                               X X
X XXXXXXXXXXXX XXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX X X X XXX XXXXX X X
X      X X  X  X  X  X      X      X  X X  X  X X  X  X  X X
XXXXXXXX X X XXX X XXXXXXX XXX X XXX XXX XXXXXXX X XXX X XXX X
X  X  X X  X X  X  X X X  X  X  X  X  X  X  X X X  X  X
X X X XXX XXX X XXX X X X X XXXXX XXX XXX X X XXX X XXXXXXX X
X X  X  X  X X X  X  X  X      X X  X X  X X  X  X  X X
X XXXXX XXX XXX X XXXXXXXXXXX XXXXXXX X XXX XXX X X X X X X X
X X      X X  X  X  X  X      X X      X X  X X  X  X  X
X XXXXXXX X X XXX X X X XXX XXXXXXX X XXXXXXX XXXXX XXXXXXXXXX
X X      X X X      X  X  X  X X  X      X  X X  X  X  X
X X XXX X X XXXXXXXXXXXXXXXXXXX X XXX X XXXXX X X X X XXX X X X X
X X X  X                                     X  X  X  X  X  X  X  X  X X X
X X X XXXXXXX XXXXXXXXXXX XXX X XXXXX X XXXXXXXXXXX XXX XXXXX XXX
X  X  X  X X  X  X X X X  X  X X  X  X  X  X X X X  X  X
X X X X X X X X X X X X X X X X X X X X X X X X X X X X
XXX X XXX XXX X X XXXXXXXXXXX X XXX X XXX X XXXXXXXXXXX X XXX X X
X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  
```

Zum Herstellen von Ausweichmöglichkeiten zufällig weitere Löcher in den Fels schlagen.



Erzeugter Irrgarten zu komplex?





Mehrere Strukturebenen

- › Zerlegen in mehrere (hier 3x3) Bereiche
- › Anlegen von Räumen
- › Verbinden der Räume mittels RFF
- › Füllen ausgewählter Räume mit Irrgärten

```

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX X X   X  X X
XXX          XXX          XXXXXXXXXXXX  X XXXX X X X
XXX          XXX          XXXXXXXXXXXX  X      X  X
XXX          XXX          XXXXXXXXXXXX  XXXXXXXX X X
XXX          XXX          XXXXXXXXXXXX  X X   X   X X
XXX          XXX          XXXXXXXXXXXX  XXXXXXXXXXXX
XXX          XXX          XXXXXXXXXXXX  XXXXXXXXXXXX
XXX          XXXXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XX          XX          XXXXXXXXXXXX          XXX
XX
XX          XX          XXXXXXXXXXXX          XXX
XX          XX          XXXXXXXXXXXX          XXX
XX          XX          XXXXXXXXXXXXXXXX XXXXXXXXXXXX
XXXXXX XXXXXXXXXXXX          XXXXXXXXXXXX          XX
  
```

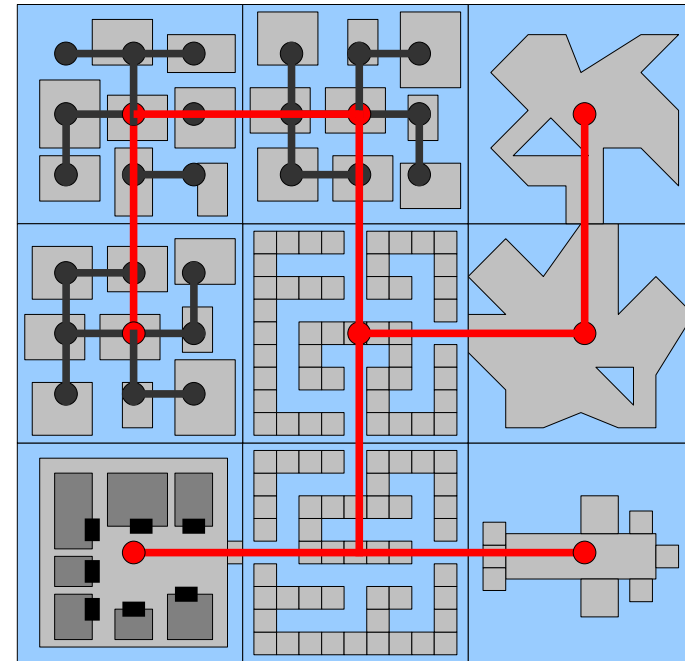
```

XXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XX
XXX XXXXXXXXXXXX XXXX XX
XXX XXXXXXXXXXXX XXXX XX
XXX XXXXXXXX XXXX YY
  
```

Weitere Möglichkeiten

- › Dörfer/Häuser
- › Kathedralen/Tempel
- › Klosteranlagen um Kreuzgang herum
- › Sümpfe/Wälder
- › Höhlen (sog. Ant Caves)

Teilbereiche werden mittels RFF verbunden
(hier rot).





Ein-/Ausgänge

- › Werden als Items dargestellt
- › Können zufällig placiert werden (da Level zusammen hängend)
- › Bei Wegen nach unten: Index merken, um Nummer des Ziellevels zu berechnen

Items → Liste zu implementierender Items: s. Pflichtenheft

Monster → dito.



2. Code-Anforderungen



Codeklau

Zum Übernehmen fremden Quellcodes:

- Wird sehr ernst genommen und geahndet
- Minimum: Herabsetzung der Note
- Maximum: Disqualifizierung

➔ Bücher, Webseiten, etc, können als Vorbilder für Algorithmen, Verhaltensweisen, etc, benutzt werden. (Als Quelle angeben!)





Java-Packages

Bitte folgende Package-Bezeichner verwenden.

Client GUI: `package pp07.gruppenn.client.gui;`

Client Level: `package pp07.gruppenn.client.level;`

Client Komm: `package pp07.gruppenn.client.comm;`

Server LabGen: `package pp07.gruppenn.server.generator;`

Server WV: `package pp07.gruppenn.server.world;`

Server Komm: `package pp07.gruppenn.server.comm;`

Tools: `package pp07.gruppenn.tools;`

Mit *nn* = Gruppennummer



Gestaltung des Quellcode

Ziele sind

- › Übersichtlichkeit,
- › Erweiterbarkeit und
- › Wartungsfreundlichkeit.

Faustregeln:

- › „Sprechende“ Variablennamen, z.B. *currentMonster* statt *m*
- › Maximale Länge einer Methode: Ein Bildschirm
- › Pro Quellcodezeile nur eine Anweisung
- › Mit Tabulatoren strukturieren



```
... C k; main(){ Display*e=  
XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))  
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,  
0,0,WhitePixel(e,0),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}  
K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_F/ K+d/K*E*_; B=  
sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_D*_F*_; l=W*E*B*_; for(o+=(I=D*W+E  
*T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]_L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s  
]= 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ c=W/K *4E2+ze2; C= 2E2+4e2/ K  
*D; N=1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*1); T=X*X+ 1*1+M *M;  
XDrawString(e,,k ,20,380,f,17); D=v/1*15; i+=(B *1-M*r -X*Z)*_; for(; !XPending(e); u *=CS!=N){  
XEvent z; ...
```

M+= H*_;

K=D*m-B*T-H*E;



```
#include <math.h>
#include <sys/time.h>
#include <X11/Xlib.h>
#include <X11/keysym.h>

double L, o, P
, _dt, T, Z, D=1, d,
s[999], E, h= 8, I,
J, K, v[999], M, m, O
, n[999], j=33e-3, i=
1E3, z, t, u, v, W, S=
74.5, l=221, X=7.26,
a, B, A=32.2, c, F, H;
int N, q, C, y, p, U;
Window z; char f[52]

; GC k; main(){ Display*o=
XOpenDisplay( 0 ); z=RootWindow(o,0); for (XSetForeground(e,k=XCCreateGC( e,z,0,0),BlackPixel(o,0))
; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
0,0,WhitePixel(o,0) ),KeyPressMask); for(XMapWindow(o,e,z); ; T=sin(O){ struct timeval G( 0,d,t*1e6)
; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_P; z=E*K; W=cos(O); m=K*W; H=K*T; O+=D*_F/ K+d/K*_; B=
sin(j); a=B*T*D-E*W; XClearWindow(o,z); t=T*E+ D*B*W; j+=d*_D*_F*E; P=W*E*B-T*D; for (o+=(I=D+W+E
*T*B,E*d/K *B+v*B/K*_D)*_; pCy: ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
]=0|K <fabs(W*T*_I-E +D*P) |fabs(D=t *D+Z *T-a *_E)> R)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
*D; N-1E4&& XDrawLine(o ,z,k,N ,U,q,C); N=q; D=C; } ++p; } L+=_ (X*t +P*M+m*1); T=X*X+ 1*1+m *_M;
XDrawString(e,z,k ,20,380,f,17); D=v/1*15; l+= (B *1-l*_z -X*Z)*_; for(; XPending(e); u *=CS!N){
XEvent z; XNextEvent(o ,&z);
++*( (N=XLookupKeysym
(&z._xkey,0))-IT?
N-L? UP-N?& E:&
J:& u: &h); --*(
DN -N? N-DT ?N==
RT?&u: & W:&h:&J
); } m=15*F/1;
c+=(I=m/ 1,1*H
+I*M+a*X)*_; H
=a*_z+v*_X-F*1+(
E=.1+X*4.9/1,t
=I*m/32-I*T/24
)/S; K=F*M+(
h* 1e4/1-(T+
E*5*T*B)/3e2
)/S-X*d-B*A;
a=2.63 /1*d;
X+=( d*1-T/S
*(.19*E +a
*.64+J/1e3
)-M* v +A*
Z)*_; l +=
K *_; W=d;
sprintf(f,
"%5d %3d"
"%7d",p ,l
/1.7, (C=9E3+
O*57.3)*0550, (int)l); d+=T*(.45-14/1*
X-a*130-J*_ .14)*_/125e2+F*_v; P=(T*(47
*I-m* 52+E*94 *D-t*_ .38+u*_ .21*E) /1e2+W*
179*v)/2312; select(p=0,0,0,0,6G); v--=(
W*F-T*(.63*m-I*_ .086+m*E*19-D*25-.11*u
)/107e2)*_; D=cos(o); E=sin(o); } }
```

Carl Banks: „A Flight Simulator“,
The International Obfuscated C Code Contest,
Winning Entry, 1998



Kommentare

- Keine Kommentare als Selbstzweck

```
// Dies ist ein Getter  
public int getID() {  
    return ID;  
}
```

- Kommentare sollten Zusammenhänge verständlich machen

```
i++; // i wird um eins erhöht ← Schlecht!  
i++; // Index auf nächstes Item setzen ← Gut!
```

- Kommentare müssen zum aktuellen Stand des Quelltextes passen



Umlaute

Unterschiede zwischen Umlautcodierung in *Windows*- und *UNIX*-Systemen!

Bitte verwenden Sie keine Umlaute in

- Dateinamen
- Bezeichnen (Methoden-, Variablennamen, etc)
- **Kommentaren (vorläufig!)**

Linux-Anwender: Bitte UTF-8-Codierung verwenden!

Siehe auch: WWW-FAQ



3. Gruppenbildung



Gruppenbildung

- › Gruppen (max. 20) mit fünf bis sieben Teilnehmern
- › Bitte Formblatt ausfüllen mit
 - › Gruppenname
 - › Teilnehmern
 - › Kontaktperson (inkl. EMail-Adresse, die auch gelesen wird)

Danach bitte noch nicht nach Hause gehen: Zugangsdaten für Server!

`progprak.scale.uni-koeln.de`



Vielen Dank.

ullrich@informatik.uni-koeln.de