EINFÜHRUNGSPRAKTIKUM: STRATEGIESPIELE

MICHAEL WILLIG, CORNELIA WULF

- individuelles Arbeiten, keine Gruppen

- plagiieren verboten

- Code nicht veröffentlichen (GitHub)

- Pflicht: Einschreibung in Prüfung

- ihr bekommt: Quellcode-Archiv

- ihr liefert: einen Computer-Gegner

- E-Mail:
  ads-strategiespiele@groups.tu-dresden.de

**TECHNISCHE
UNIVERSITÄT
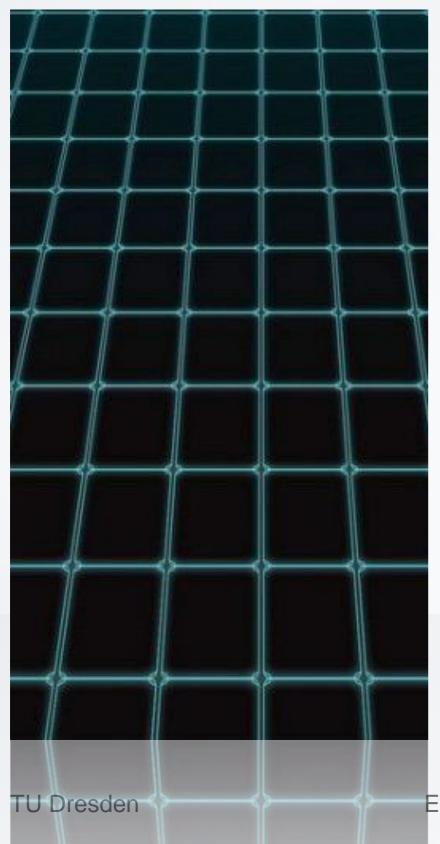DRESDEN**

**Mittwoch, 01.04.2020, 13: Uhr
Per E-Mail an
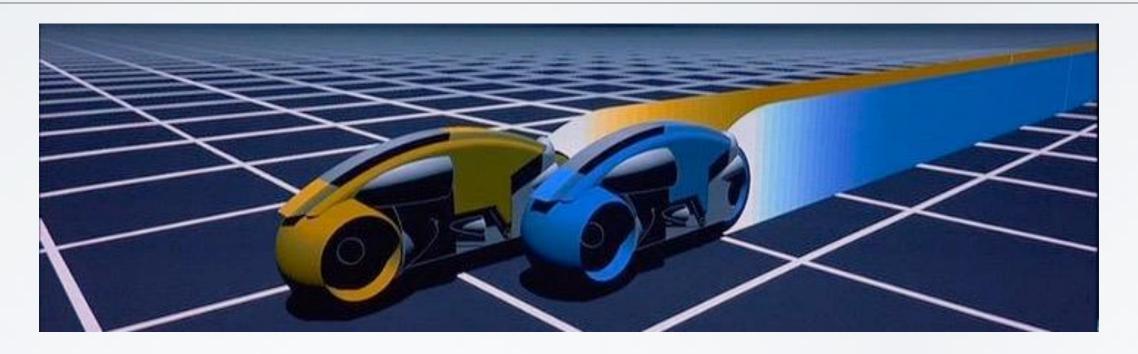ads-strategiespiele@groups.tu-dresden.de**

PROGRAMMIEREN

GAME GRID

- Entwicklung unter Linux in C oder C++
- Referenz: PCs im ZIH
- Kommandozeile
- Tar-Archiv zu jedem Spiel auf Webseite
- bauen mit `make`
- `README lesen`

- interaktiver Tastatur-Spieler

- euer Computer-Gegner
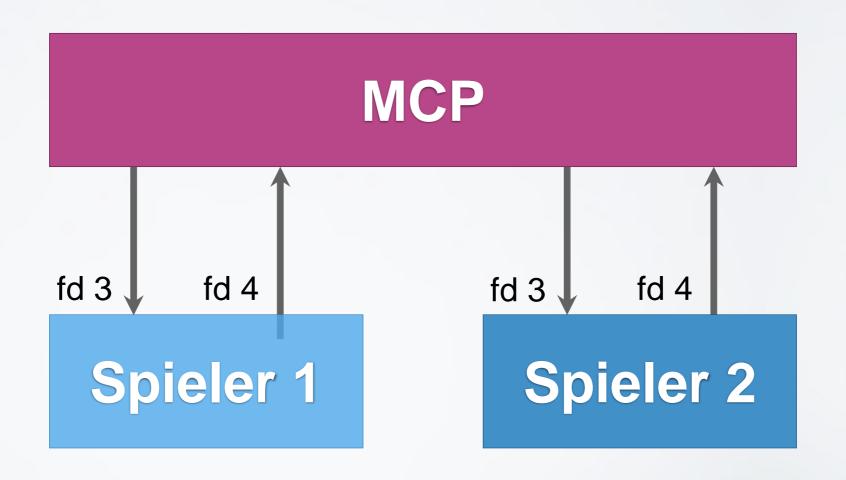 - diese Datei ist von euch auszufüllen

- Master Control Program

- verbindet zwei Spieler
  - überträgt Spielfeld
  - empfängt Zug
  - siehe Tastatur-Spieler
- überwacht Regeln und Ressourcen

make all	baut alles
make demo	zwei Tastatur-Spieler
make run	euer Gegner und Tastatur
make fight	euer Gegner zweimal
make clean	gebaute Dateien löschen
make help	Anleitung

- Linux-Installation in VirtualBox

- gezielte Übergabe eines Spielzustands an Spieler

- dynamische Arrays

- Zufallszahlen

BESTEHEN

- euer Code muss **fehlerfrei kompilieren**

  - „schöner" Code bringt Bonuspunkte

- Computer-Gegner muss **gültig spielen**

  - Regelverstöße führen zum Nicht-Bestehen

- **nicht-triviale** Strategie

  - klare Gewinnchancen nutzen

  - dem Gegner keine Steilvorlagen liefern

- Tipp: reicht **Zwischenstände** ein

- **konsistente** Formatierung

- **sprechende** Bezeichner

- Gültigkeitsbereich minimieren

- **erläuternde** Kommentare

- **klare** Code-Struktur

- kein Spaghetti- oder Lasagne-Code

- **DRY:** Don't Repeat Yourself

- Spieler und MCP kommunizieren über UNIX-Pipes
  - wie das geht, seht ihr im Tastatur-Spieler
  - ihr dürft von dort abschreiben
- MCP-Code darf nicht benutzt werden
  - ist deshalb auch teilweise in Assembler
- erlaubt: C-Bibliothek, C++/STL, pthreads
  - andere Bibliotheken auf Anfrage

- konzentriert euch zuerst darauf, **gültige Züge** zu finden

- Donnerstag/Freitag:
Termin zur individuellen **Besprechung** eures Designs

- danach Zeit für **fortgeschrittene** Ideen

- **Mittwoch: Abgabe per E-Mail**

- nutzt E-Mail bei Fragen

DOS & DON'TS

```
if((buffer[j-3]=='b' && buffer[j-6]=='-' && j!=4 &&
j!=8 && j!=12&&j!=16&&j!=20&&j!=24&&j!=28&&j!=32) ||
(buffer[j-4]=='b'&&buffer[j-8]=='-
'&&j!=1&&j!=5&&j!=9&&j!=13&&j!=17&&j!=21&&j!=25&&j!=29)
||
(buffer[j+1]=='W'&&buffer[j+5]=='b'&&buffer[j+10]=='-
'&&j!=4 && j!=8 &&
j!=12&&j!=16&&j!=20&&j!=24&&j!=28&&j!=32) ||
(buffer[j+1]=='W'&&buffer[j+4]=='b'&&buffer[j+8]=='-
'&&j!=1&&j!=5&&j!=9&&j!=13&&j!=17&&j!=21&&j!=25&&j!=29)
)
```

```
if ( !possibilities[piece_number][5] && right_side )
{                            // wenn kein jump gesetzt ist und die figur im rechten bereich ist
    if ( ( possibilities[piece_number][2] & choice1 ) == choice1 ) {possibilities[piece_number][3] += 20; possibilities[piece_number][4] = choice1; possibilities[piece_number][5] = 1;} else {
        if ( ( possibilities[piece_number][2] & choice2 ) == choice2 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice2; possibilities[piece_number][5] = 1;} else {
            if ( ( possibilities[piece_number][2] & choice3 ) == choice3 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice3; possibilities[piece_number][5] = 0;} else {
                if ( ( possibilities[piece_number][2] & choice4 ) == choice4 ) {possibilities[piece_number][3] += 10; possibilities[piece_number][4] = choice4; possibilities[piece_number][5] = 0;} else {
                    if ( ( possibilities[piece_number][2] & choice7 ) == choice7 ) {possibilities[piece_number][3] += 8; possibilities[piece_number][4] = choice7; possibilities[piece_number][5] = 1;} else {
                        if ( ( possibilities[piece_number][2] & choice8 ) == choice8 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice8; possibilities[piece_number][5] = 1;} else {
                            if ( ( possibilities[piece_number][2] & choice5 ) == choice5 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice5; possibilities[piece_number][5] = 0;} else {
                                if ( ( possibilities[piece_number][2] & choice6 ) == choice6 ) {possibilities[piece_number][3] += 3; possibilities[piece_number][4] = choice6; possibilities[piece_number][5] = 0;} else {
                                    if ( ( possibilities[piece_number][2] & choice9 ) == choice9 ) {possibilities[piece_number][3] -= 3; possibilities[piece_number][4] = choice9; possibilities[piece_number][5] = 0;} else {
                                        if ( ( possibilities[piece_number][2] & choice10 ) == choice10 ) {possibilities[piece_number][3] -= 5; possibilities[piece_number][4] = choice10; possibilities[piece_number][5] = 0;} else {possibilities
                                        }        // wird der bevorzugte zug gesetzt, jump entsprechend auf 1 oder 0 gesetzt und die priorität wird erhöht, bzw. erniedrigt
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

if ( !possibilities[piece_number][5] && left_side )
{                            // wenn kein jump gesetzt ist und die figur im linken bereich ist
    if ( ( possibilities[piece_number][2] & choice2 ) == choice2 ) {possibilities[piece_number][3] += 20; possibilities[piece_number][4] = choice2; possibilities[piece_number][5] = 1;} else {
        if ( ( possibilities[piece_number][2] & choice1 ) == choice1 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice1; possibilities[piece_number][5] = 1;} else {
            if ( ( possibilities[piece_number][2] & choice4 ) == choice4 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice4; possibilities[piece_number][5] = 0;} else {
                if ( ( possibilities[piece_number][2] & choice3 ) == choice3 ) {possibilities[piece_number][3] += 10; possibilities[piece_number][4] = choice3; possibilities[piece_number][5] = 0;} else {
                    if ( ( possibilities[piece_number][2] & choice8 ) == choice8 ) {possibilities[piece_number][3] += 8; possibilities[piece_number][4] = choice8; possibilities[piece_number][5] = 1;} else {
                        if ( ( possibilities[piece_number][2] & choice7 ) == choice7 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice7; possibilities[piece_number][5] = 1;} else {
                            if ( ( possibilities[piece_number][2] & choice6 ) == choice6 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice6; possibilities[piece_number][5] = 0;} else {
                                if ( ( possibilities[piece_number][2] & choice5 ) == choice5 ) {possibilities[piece_number][3] += 3; possibilities[piece_number][4] = choice5; possibilities[piece_number][5] = 0;} else {
                                    if ( ( possibilities[piece_number][2] & choice10 ) == choice10 ) {possibilities[piece_number][3] -= 3; possibilities[piece_number][4] = choice10; possibilities[piece_number][5] = 0;} else {
                                        if ( ( possibilities[piece_number][2] & choice9 ) == choice9 ) {possibilities[piece_number][3] -= 5; possibilities[piece_number][4] = choice9; possibilities[piece_number][5] = 0;} else {possibilities[pi
                                        }        // wird der bevorzugte zug gesetzt, jump entsprechend auf 1 oder 0 gesetzt und die priorität wird erhöht, bzw. erniedrigt
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

if ( !possibilities[piece_number][5] && left_left_side )
{                            // wenn kein jump gesetzt ist und die figur im ganz linken bereich ist
    if ( ( possibilities[piece_number][2] & choice2 ) == choice2 ) {possibilities[piece_number][3] += 20; possibilities[piece_number][4] = choice2; possibilities[piece_number][5] = 1;} else {
        if ( ( possibilities[piece_number][2] & choice4 ) == choice4 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice4; possibilities[piece_number][5] = 0;} else {
            if ( ( possibilities[piece_number][2] & choice8 ) == choice8 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice8; possibilities[piece_number][5] = 1;} else {
                if ( ( possibilities[piece_number][2] & choice6 ) == choice6 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice6; possibilities[piece_number][5] = 0;} else {
                    if ( ( possibilities[piece_number][2] & choice10 ) == choice10 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice10; possibilities[piece_number][5] = 0;} else {
                        if ( ( possibilities[piece_number][2] & choice12 ) == choice12 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice12; possibilities[piece_number][5] = 1;} else {possibilities[piece_number][4
                        }        // wird der bevorzugte zug gesetzt, jump entsprechend auf 1 oder 0 gesetzt und die priorität wird erhöht, bzw. erniedrigt
                    }
                }
            }
        }
    }
}

if ( !possibilities[piece_number][5] && right_right_side )
{                            // wenn kein jump gesetzt ist und die figur im ganz rechten bereich ist
    if ( ( possibilities[piece_number][2] & choice1 ) == choice1 ) {possibilities[piece_number][3] += 20; possibilities[piece_number][4] = choice1; possibilities[piece_number][5] = 1;} else {
        if ( ( possibilities[piece_number][2] & choice3 ) == choice3 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice3; possibilities[piece_number][5] = 0;} else {
            if ( ( possibilities[piece_number][2] & choice7 ) == choice7 ) {possibilities[piece_number][3] += 15; possibilities[piece_number][4] = choice7; possibilities[piece_number][5] = 1;} else {
                if ( ( possibilities[piece_number][2] & choice5 ) == choice5 ) {possibilities[piece_number][3] += 10; possibilities[piece_number][4] = choice5; possibilities[piece_number][5] = 0;} else {
                    if ( ( possibilities[piece_number][2] & choice9 ) == choice9 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice9; possibilities[piece_number][5] = 0;} else {
                        if ( ( possibilities[piece_number][2] & choice11 ) == choice11 ) {possibilities[piece_number][3] += 5; possibilities[piece_number][4] = choice11; possibilities[piece_number][5] = 1;} else {possibilities[piece_number][4
                        }        // wird der bevorzugte zug gesetzt, jump entsprechend auf 1 oder 0 gesetzt und die priorität wird erhöht, bzw. erniedrigt
                    }
                }
            }
        }
    }
}
```

```c
#include <stdio.h>

int i,j,m,l,n,k,a,c,t,s,w,v;


int main(void)
{
    /* ... */
    for (i = 0; i < 8; i++) {
        /* ... */
        for (j = 0; j < 8; j++) {
            /* ... */
            for (i = 5; i > 0; i--) {
                /* ... */
            }
        }
    }
}
```

```c
#include <stdio.h>



int main(void)
{
    /* ... */
    for (int i = 0; i < 8; i++) {
        /* ... */
        for (int j = 0; j < 8; j++) {
            /* ... */
            for (int i = 5; i > 0; i--) {
                /* ... */
            }
        }
    }
}
```

```
for (int i; i < 8; i++) {
    /* ... */
}



bool check_for_something()
{
    bool found;
    if ( /* some condition */ )
        found = true;
    return found;
}
```

```
int get_element(i)
{
    return matrix[i];
}

get_element(-1);



int get_element(vector<int> &v, i)
{
    if (v.size())
        return v[i];
    else
        return v[0];
}
```

```
void output_move()
{
    char *buffer = malloc(1024);
    snprintf(buffer, 1024, /* format string */);
    write(output_to_mcp, buffer, 1024);
}
```

```
void analyze()
{
    Move *move = new Move();
    /* analyze move */
    delete move;
}
```

```
void analyze()
{
    Move move;
    /* analyze move */


}
```
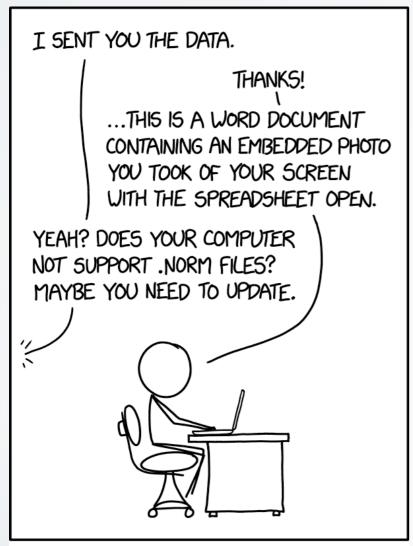
- **Immer** an
  Ads-strategiespiele@groups.tu-dresden.de
  schreiben/antworten

- Spielnamen im Betreff hilft der Zuordnung

- Fehler-/Problembeschreibung
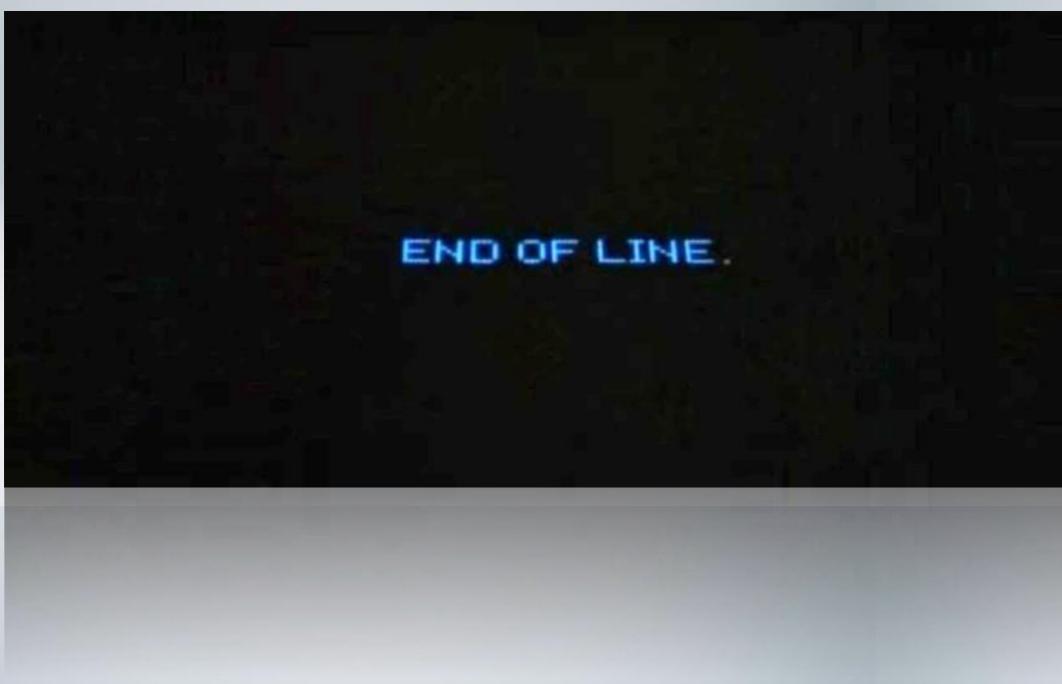
- ggf. (eigenen!) Code anhängen (nur my_player.cc)

Quelle: https://xkcd.com/2116/

- Keine Bildschirmfotos per Email

- Keine Fotos von Bildschirmen per Email

- Pro-Tipp: Text kann man kopieren & einfügen. Auch auf/von der Shell.

TECHNISCHE UNIVERSITÄT DRESDEN

Matrikeln	Spiel	Betreuer	Mitarbeiter
…003 – …340	Mühle	Max Friedrich	Martin Hohmann
…362 – …575	Haliotis	Ferdinand Thießen	Cornelia Wulf
…591 – …814	Isolation	Robert Ufer	Michael Willig
…820 – …996	Back-gammon	Manuel Thieme	Gökhan Akgün Julian Haase