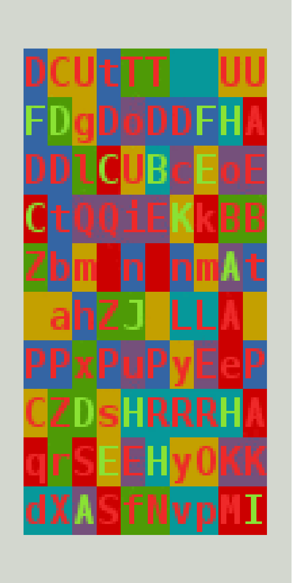
Concurrent C Programming
HPCTF – Hauronen Patronen’s Capture the Flag using ØMQ

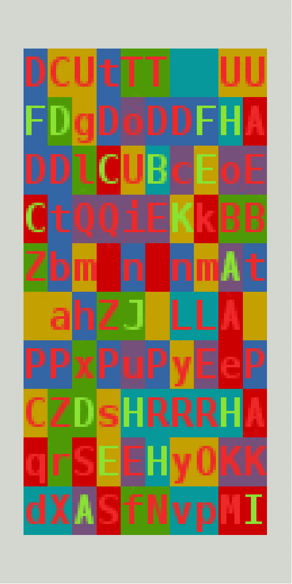


```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

ZHAW – School of Engineering

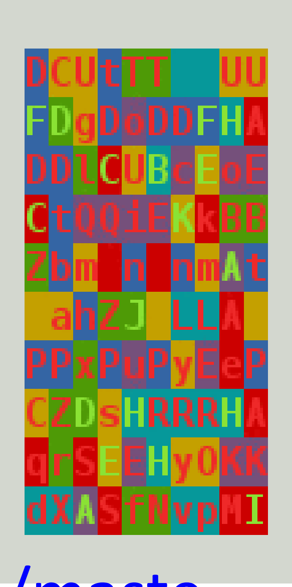
Seminar:	**Concurrent C Programming**
Dozent:	**Nico Schottelius**
Autor:	**Hauri David**
Datum:	**28.06.2015**

Inhaltsverzeichnis



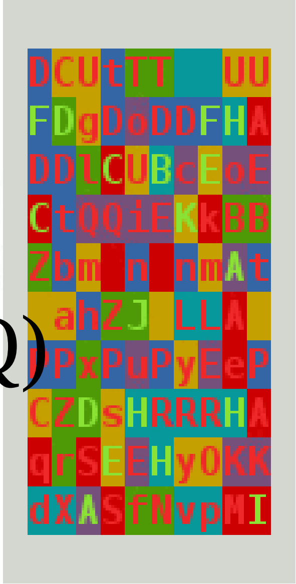
- Einleitung
- ZeroMQ
- Spielprinzip
- Max Player count
- PlayerId vs Printable char field
- Request - Reply
- Load Balancer
- Shared Key Value Map
- PlayerId vs FieldOwner
- Synchronisation Tasks
- Environment
- Frage / Demo

Einleitung



- GitHub: https://github.com/chubbson/sem_os_hpctf/
- ØMQ installation guide by David Hauri https://github.com/chubbson/sem_os_hpctf/blob/master/INSTALLZMQ.md
- Stack Overflow strtok HAD: http://stackoverflow.com/questions/2529834/strtok-wont-accept-char-str
- ØMQ: http://www.zeromq.org

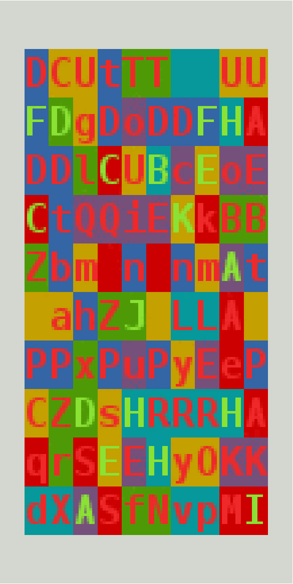
ZeroMQ, ØMQ



ØMQ (also spelled ZeroMQ, 0MQ or ZMQ) is a high-performance asynchronous messaging library aimed at use in scalable distributed or concurrent applications. It provides a message queue, but unlike message-oriented middleware, a ØMQ system can run without a dedicated message broker. The library is designed to have a familiar socket-style API.

http://en.wikipedia.org/wiki/%C3%98MQ

Spielprinzip

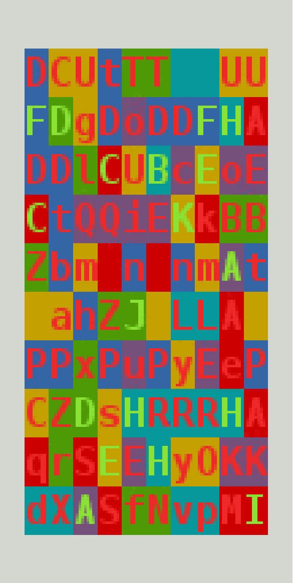


```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

- Mehrere Spieler
- Ein Server
- n*n Felder
- Capture The Flag
- Own all Flags -> Winner
- TCP/IP -> ØMQ

Max Player count

1 Terminal character = 1 Field (Owner)



```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

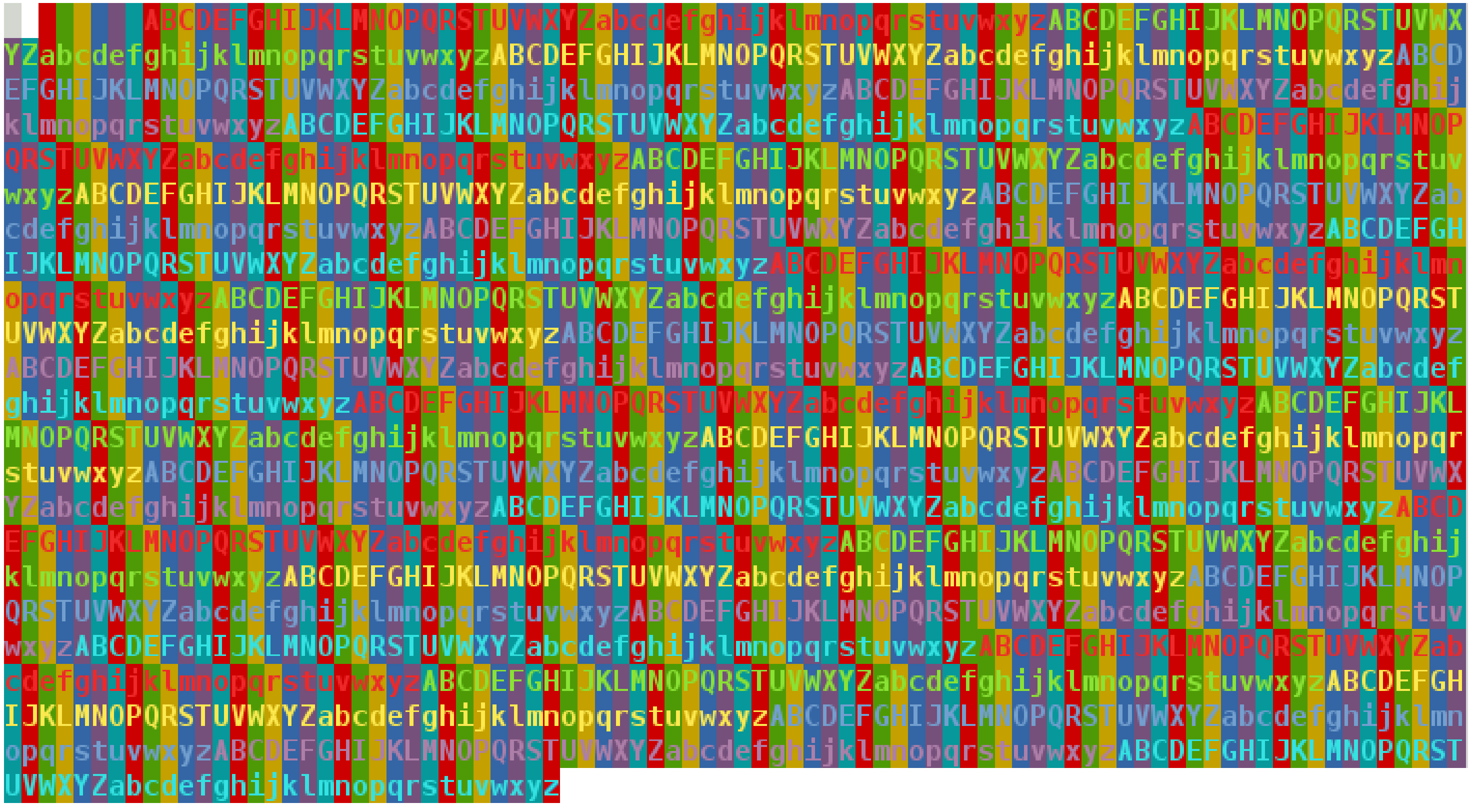
$$l = 26 \text{ [Letters a..z]}$$

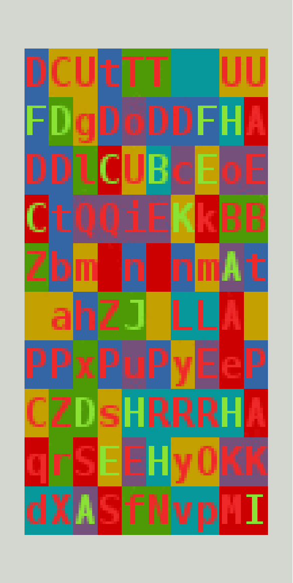
$$c = 6 \quad \text{[Colors]}$$

$$p_{max} = c + (c^2 * l * 2)$$

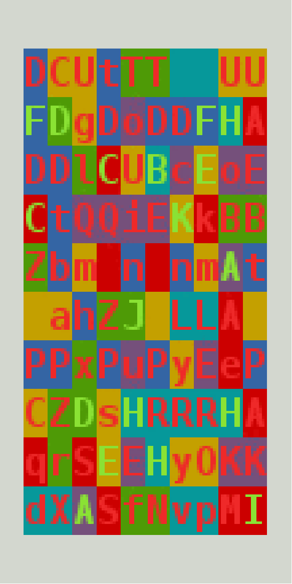
$$p_{max} = 6 + 36 * 26 * 2 = 1878$$

Max Player count



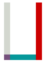


PlayerId => Printable char Field



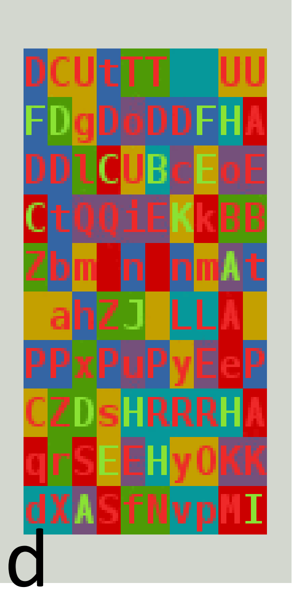
```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

```
 const int shftidx = 0x41;                                          // lowest ascii tablechar
 const int alphcnt = 26; const int lettercnt = (alphcnt*2);         // a-z // a-Z
 const int coldigcnt = 6; const int fldcolcnt = coldigcnt*coldigcnt; // diferent colors // color combination
 const int plrcnt = coldigcnt + (fldcolcnt * lettercnt);            // max palyer
 idx = idx%plrcnt; int n = 0; int letidx, digbg, digfg;             // mod given idx with maxplrcntconst
switch(idx){
  case -1:                 // fill Buffer with ‘Border’
    sprintf(buf, "\x1B[%d;3%d;4%dm%c\x1B[0m", 0, 7, 7, ' '); break;
  case 0:                  // fill Buffer with ‘empfy’
    sprintf(buf, "\x1B[0m%c", ' '); break;
  case 1 ... 6:            // fill Buffer with uni color
    sprintf(buf,  "\x1B[%d;3%d;4%dm%c\x1B[0m", 0, idx, idx, ' '); break;
  case 7 ... 1878:
    idx -= coldigcnt; idx -= 1;             // idx – 1..6 base colors; // idx 2 zero base
    letidx = idx % lettercnt;               // letidx based to 0..51 -> a..Z
    letidx += letidx >= alphcnt ? 6 : 0;    // idx > (int)’z’ => skip 6 characters to shift to A..Z range
    letidx += shftidx;                      // shift 0x41 to (int)‘a’ character in ascii table
    digfg = (idx/lettercnt)%coldigcnt + 1;  // Foreground Color 1 .. 6 => 1Red, 2Green, 3Yellow, 4Blue, 5Magenta, 6Cyan
    digbg = (idx - ( idx/(lettercnt*coldigcnt) ) )%coldigcnt + 1;// BgCol 1 .. 6 => 1Red, 2Green, …, 4Blue, 5Magenta, 6Cyan
    sprintf(buf, "\x1B[1;3%d;4%dm%c\x1B[0m", digfg, digbg, (char)letidx);  // "\x1B[1;31;43mC"  =>
    break;          // 1 Bold on; ForeClr 30-37 Black,Red…Cyan,White; BackClr 40-47 Black,Red…Cyan,White m Character
default: break; }
```



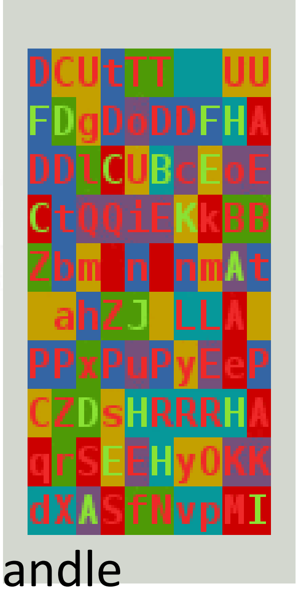
```
C
```

Request – Reply (Client)



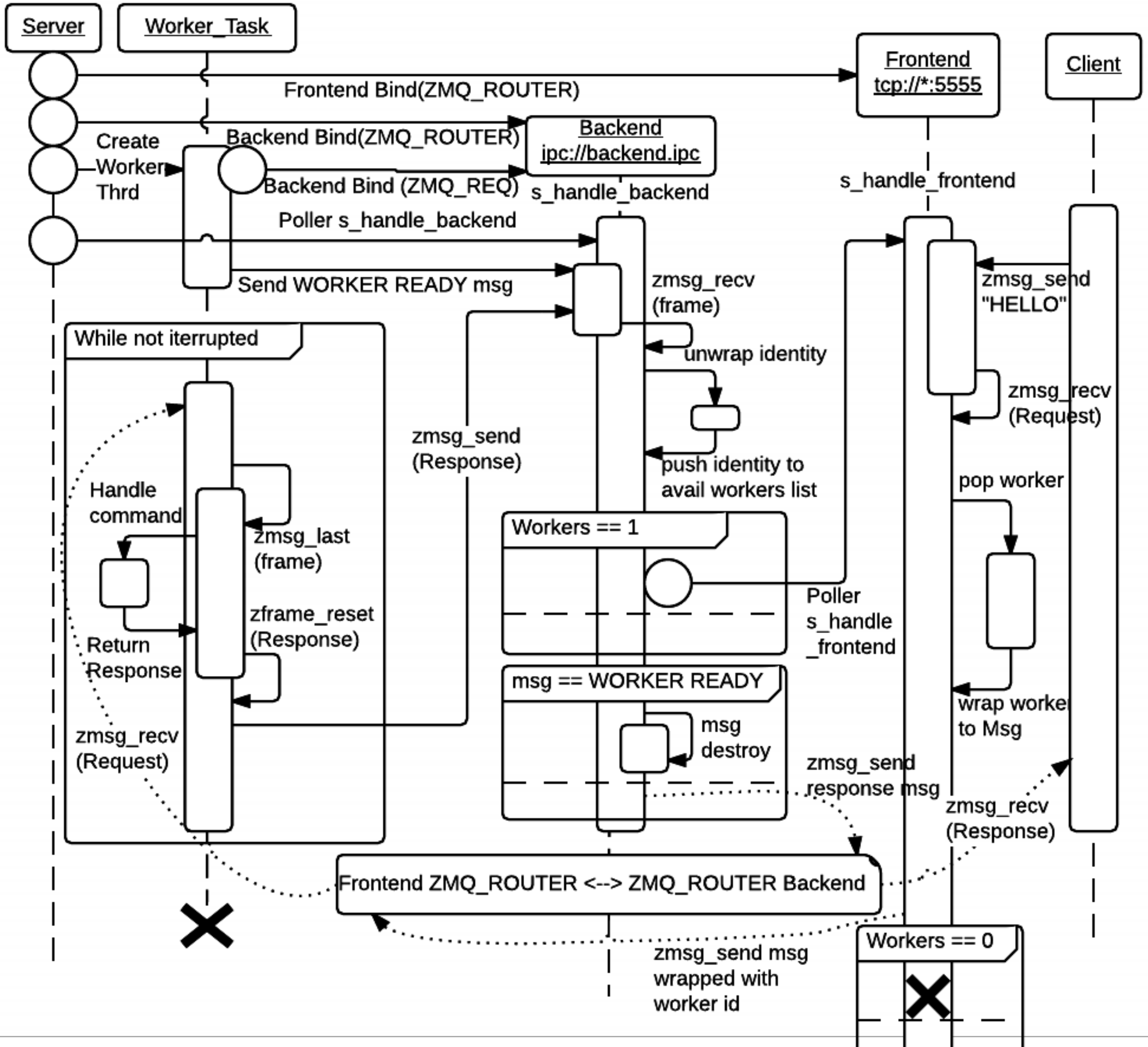
- Client Requests Server | Port: 5555
- Client -> Lazy Pirate Strategy
- Tries 3 Times, with an interval of 1 Second
  - Server is not available (started)
  - Server is to busy
  - Server crashed
- If no Response -> Break -> Terminate
- ZMQ_REQ -> "tcp://localhost:5555“ (Server:localhost, still hardcoded)

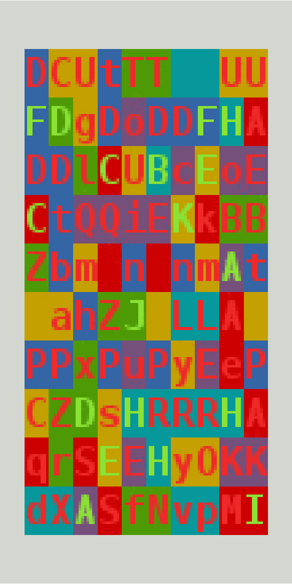
Load Balancer (Server)



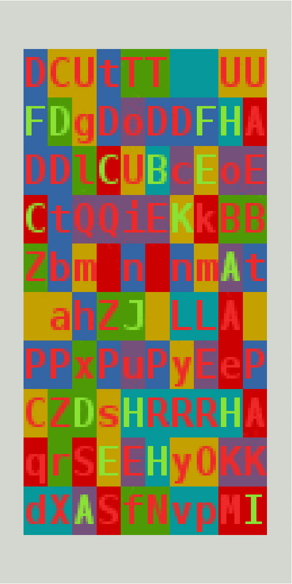
- Frontend: ZMQ_ROUTER “tcp://*:5555“
- Backend: ZMQ_ROUTER “ipc://backend.ipc”
- Workerthread: ZMQ_REQ “ipc://backend.ipc”
- Workerthread: Sending Worker Ready on “ipc://backend.ipc“
- Loop Workerthread: Receive Msg, get last Frame (Commando), Handle cmd, reset frame with answer, send (response) to “ipc://backend.ipc”
- Poll On Backend “ipc://backend.ipc” -> handle_Backend
- Handle_Backend: Receive Msg from “ipc://Backend.ipc”, unwrap identity from Msg, append identity to workers list. If there is just 1 Worker, poll “tcp://*:5555” -> handle_frontend if there is a Msg.
- Hanlde_Backend: If there is a Msg is ‘Worker Ready’ drop it otherwise, forward it to Frontend
- Hanlde_Frontend: Receive Msg from “tcp://*:5555”, get a available worker form list and wrap msg with the worker pointer. Send wrapped Msg to Backend “ipc://backend.ipc”. If then no more available workers left, cancel poller which handle_frontend on frontend

Load Balancer (Sequence)





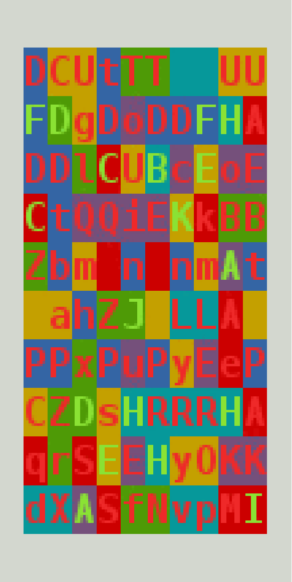
Shared Key Value Map



```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

- Server ZMQ_PUB -> “tcp://*:5556”
- Server Sendet asynchron Status updates.
- Jeder Client, FieldViewer kann diese Statusmeldungen Subscriben und local in einer KVHash speichern. Diese KVHash kann auf die gewünschten Werde im Programm abgefragt werden.

Shared Key Value Map

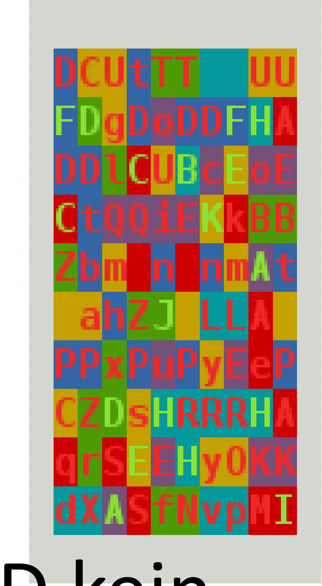


```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

- Verwendete KV States

– [fldlen]	int: Size N
– [%d][%d]	char *: Field Owner
– [state]	int: Game State FINISHED, RUNNING, WAITING4PLAYERS
– [winner]	char *: winner Name
– {%s}	int: PlayerId

PlayerId vs FieldOwner



```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

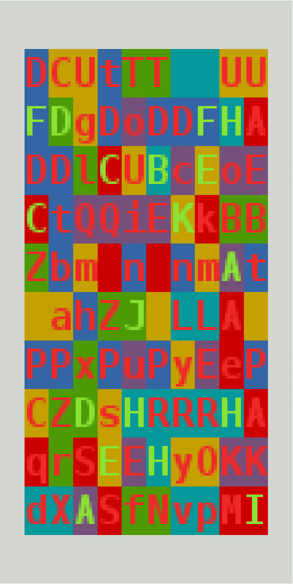
- Req: Name des Spielers steht im Feld.
- Owner der Fields als char *
- Problem: Mit char * lässt sich in der CMD kein schönes Feld zeichnen.
- Server published eindeutige PlayerId pro Spieler.

Synchronisation Task

```
DCUtTT  UU
FDgDoDDFHA
DDlCUBcEoE
CtQQiEKkBB
Zbm n nmAt
 ahZJ LLA 
PPxPuPyEeP
CZDsHRRRHA
qrSEEHyOKK
dXASfNvpMI
```

- Server bemerkt keine Disconnects.
- Client kann ohne Probleme reconnecten.
- Synchronisation tasks (~1s):
  - Game State
  - FieldSize
  - PlayerName:Id
  - Fieldowner

Environment



- Win 7 -> VMWare Fedora 17 x86
- Sublime
- Terminal
- ZeroMQ 4.2.0

Demo