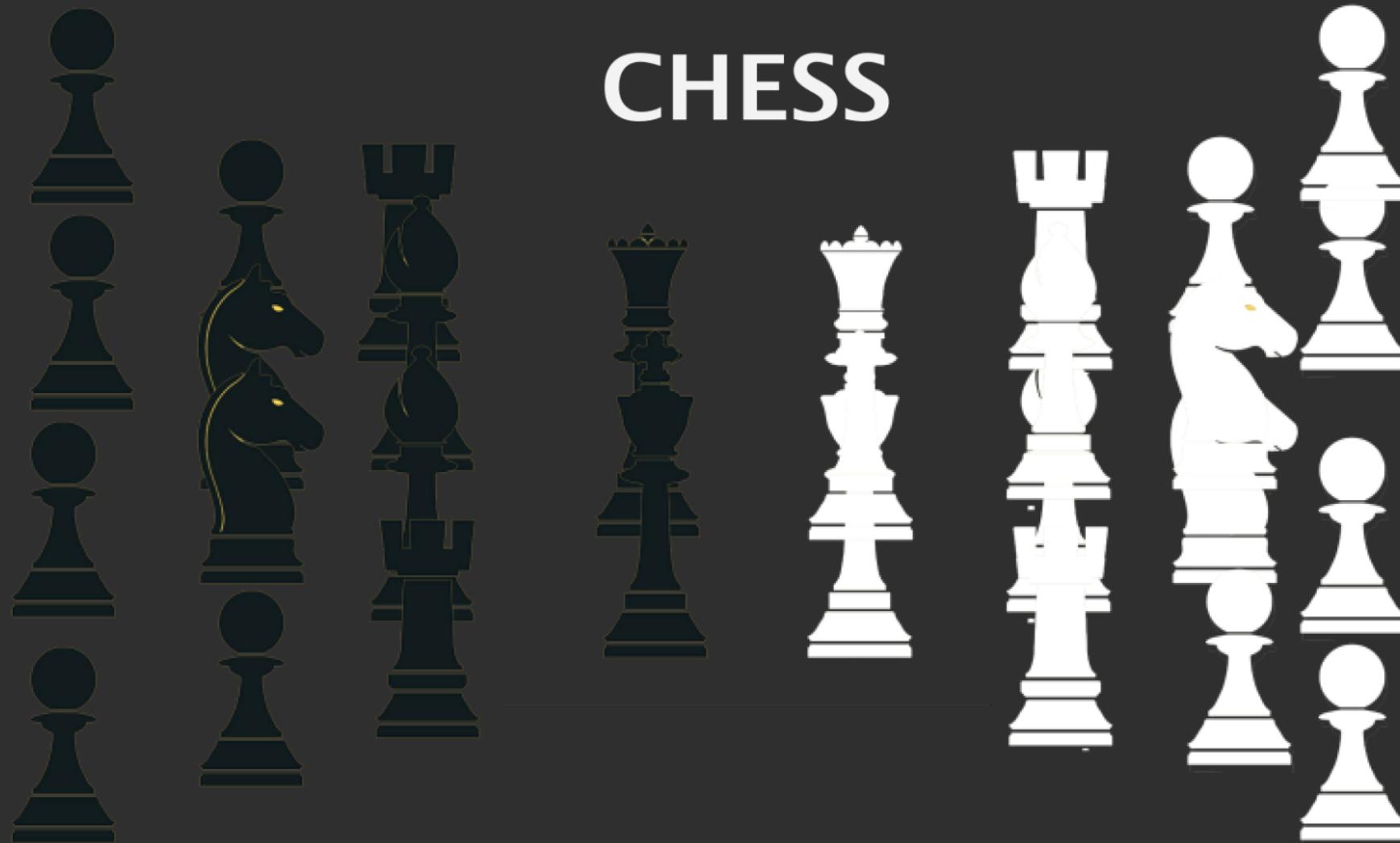




Game

CHESS

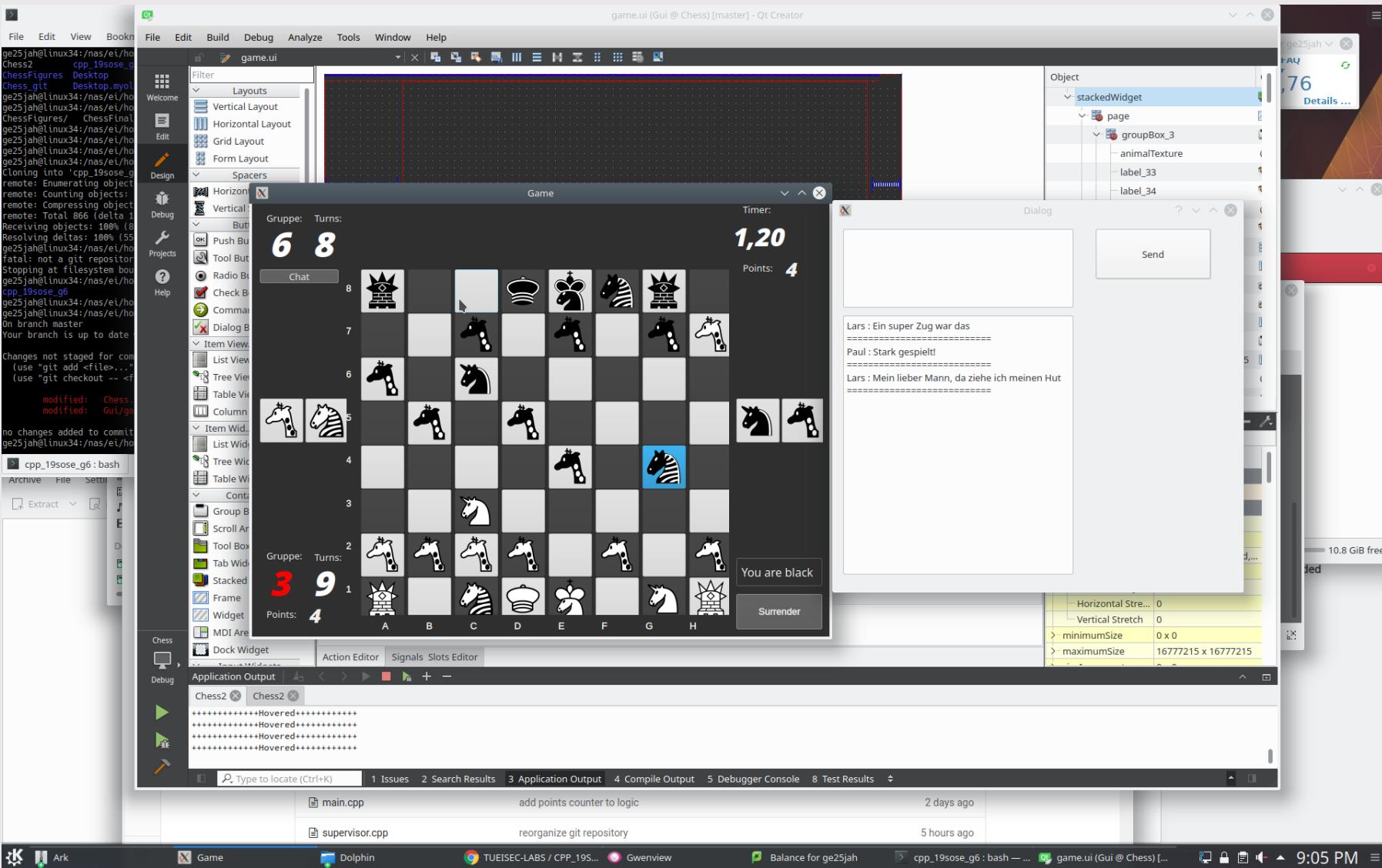


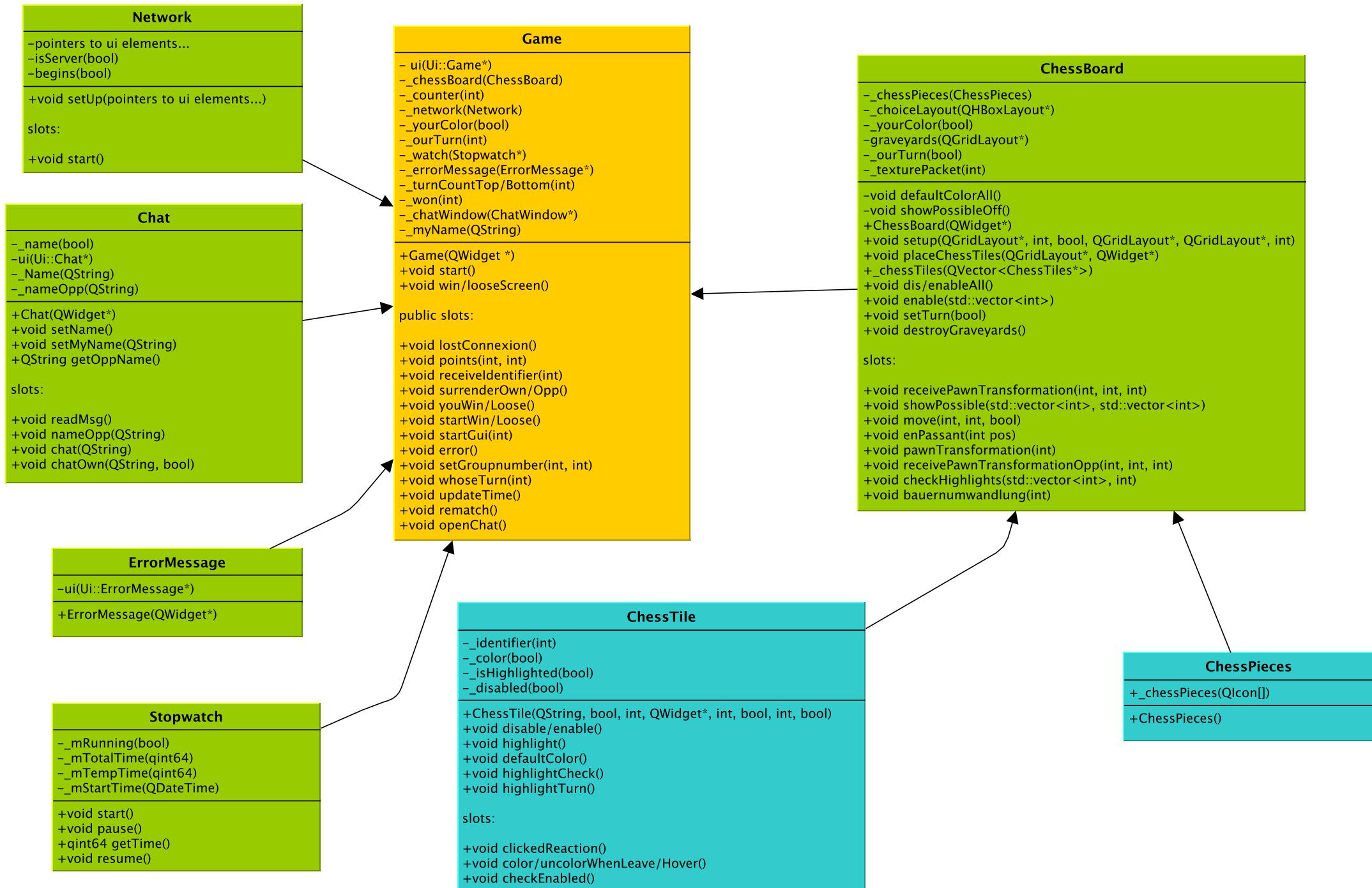
Chess Gruppe 6

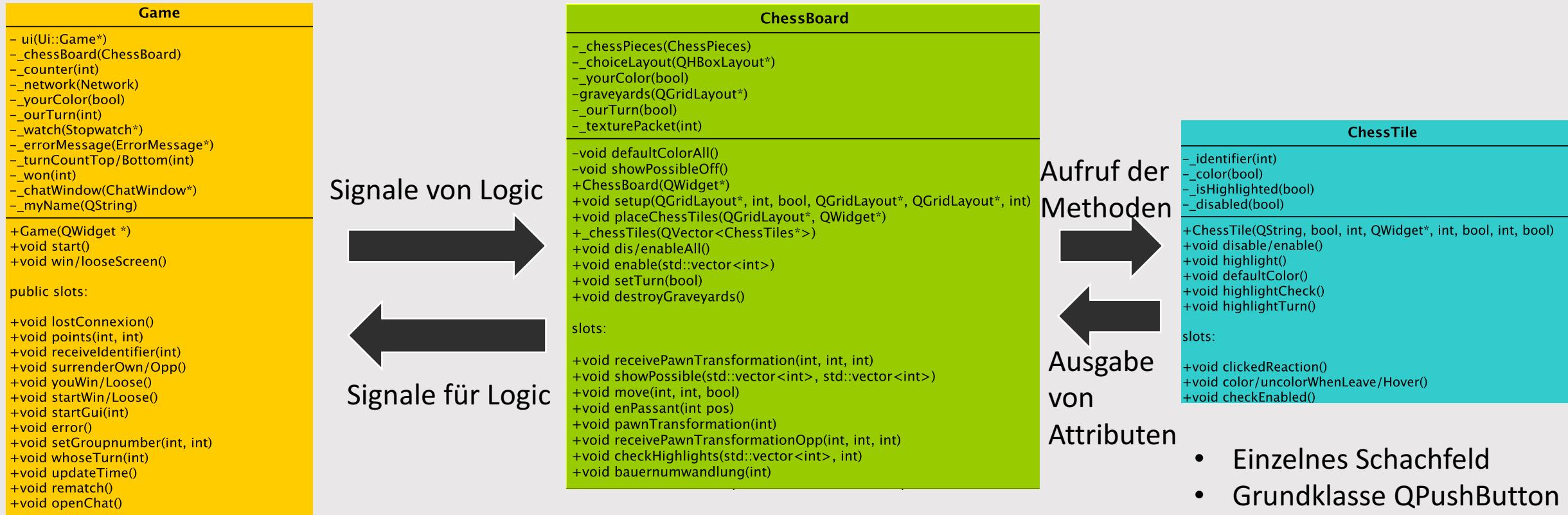
Aufgabenverteilung

- Logik: Rayene Messaoud
 - Schachlogik
 - Schnittstelle zwischen Netzwerk und Gui
- Netzwerk: Ali Oguz Can
 - Aufsetzen von Server und Client
 - Implementation Kommunikationsprotokoll
- Ui: Paul Delseith
 - Design der Ui

GUI



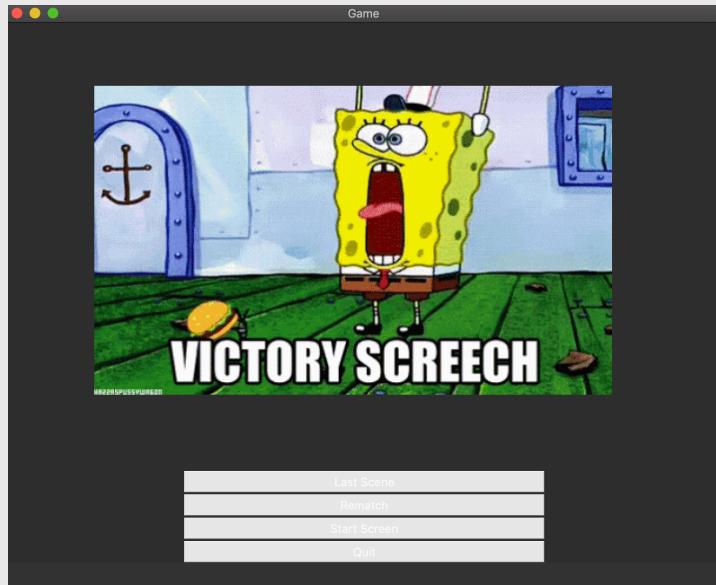
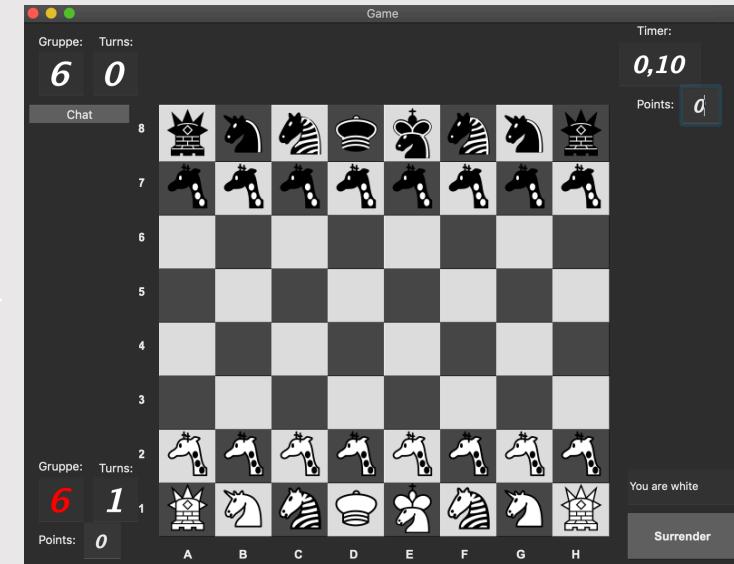
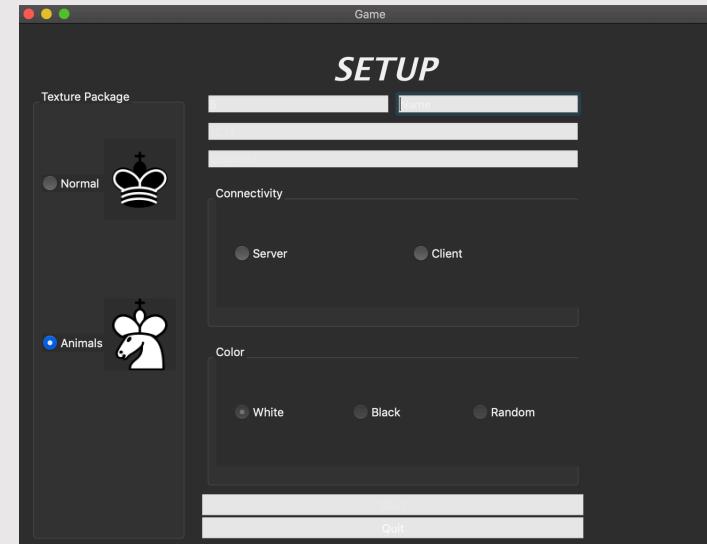
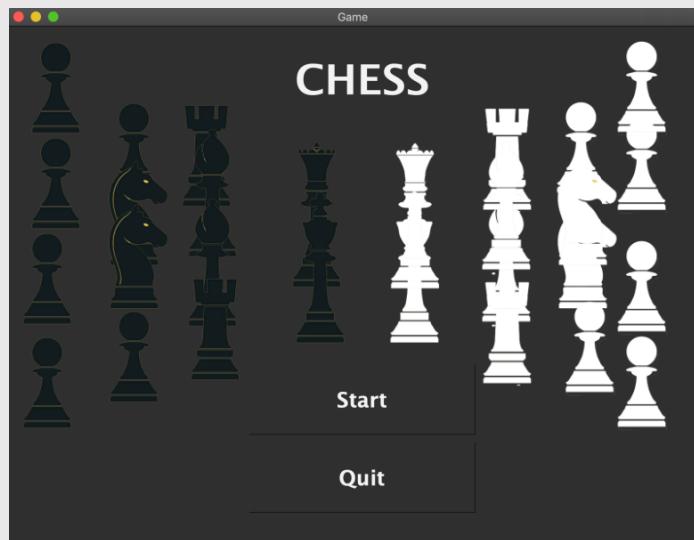




- Verwaltung der GUI, führt alle Teile zusammen
- Zugriff auf GUI immer über Game
- Implementiert Startup Routine um alle GUI Komponenten zu erzeugen/zerstören

- Verwaltet das Schachfeld (Erzeugt und zerstört es)
 - Ruft Unterfunktionen der ChessTiles auf
 - Implementiert alle Spezialzüge
 - Zugriff auf jedes ChessTile
 - GUI merkt sich so wenige Informationen wie möglich um Fehler im Zusammenspiel mit Logic zu vermeiden
 - Einheitliche Schnittstellen zur Verarbeitung von Informationen vom Netzwerk und der eigenen Logic
- Einzelnes Schachfeld
 - Grundklasse QPushButton
 - Kennt seine Farbe, Nummer und Zustand
 - Selbst implementiertes Disable -> nicht ausgraut wenn Disabled

Aufbau

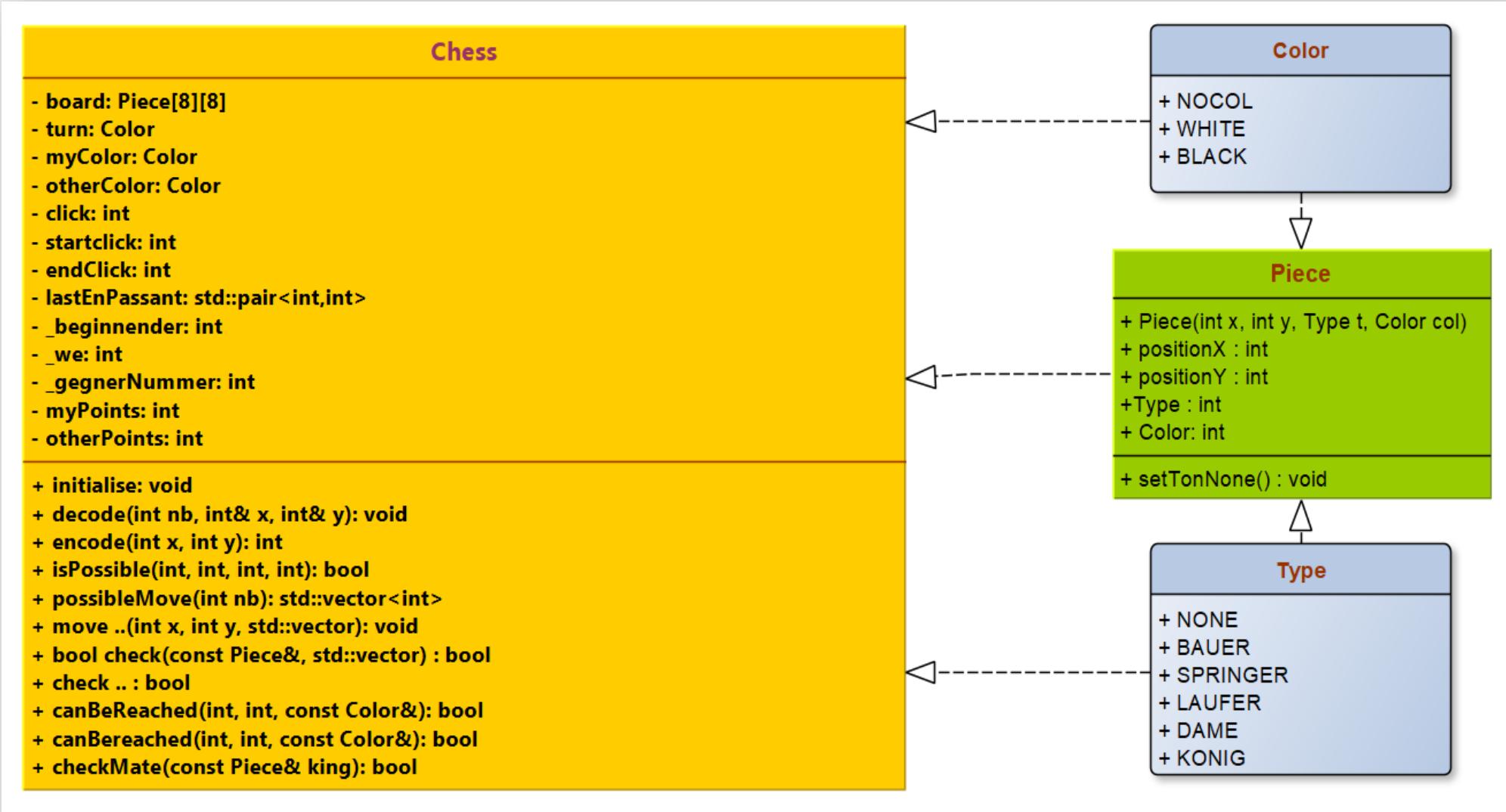


Main Menu

Rematch?

Quit

Logik Klassenstruktur



Verbindung zwischen Logik und Netzwerk

SIGNALS zwischen Logik und Netzwerk

ourMove(std::vector<quint8>)	Sendet move vom GUI zu Netzwerk
ourResponse(std::vector<quint8>)	Sendet Antwort von Logik zu Netzwerk
weWin()/weLose()	Sendet win und lose Signal
Surrender(std::vector<quint8>)	Sendet Signal für das Aufgeben



SLOTS zwischen Logik und Netzwerk

moveFromNetwork(int, int, int, int, int, int)	- Überprüfe Move von Gegner
ResponseFromNetwork(std::vector<quint8>)	- Überprüfe Netzwerksantwort
setColors(int weAre, int beginnender, int gegnerNummer)	- Nimmt parameter für das Herstellen vom Spiel Matrix
rematch()	- Spielfeld löschen und neu starten



Verbindung zwischen Logik und GUI

SIGNALS zwischen Logik und GUI

whereToMove(std::vector<quint8> possible, Sendet mögliche Wege für Pieces
std::vector<quint8> yourPiece)

modifyGui(int start, int end, bool schlagt) Sendet Signal um das GUI zu modifizieren

errorState() Sendet Signal dass es ein Error gibt

whoseTurn(bool Turn) Sendet Signal ob Spieler dran ist oder nicht



SLOTS zwischen Logik und GUI

moveFromGUI(int nb) Überprüfe Move von GUI und Turns
Verwaltung

myBauerChange(int piece) Modifiziert Game Matrix

Netzwerk

Die Zusammenarbeit zweier Programme

Ali Oguz Can

Zwei Große Klassen

```
#include "myserver.h"
```

```
MyServer::MyServer(quint16 port, quint8 beginnender, QObject *parent) :  
    QObject(parent), _port(port), _beginnender(beginnender)
```

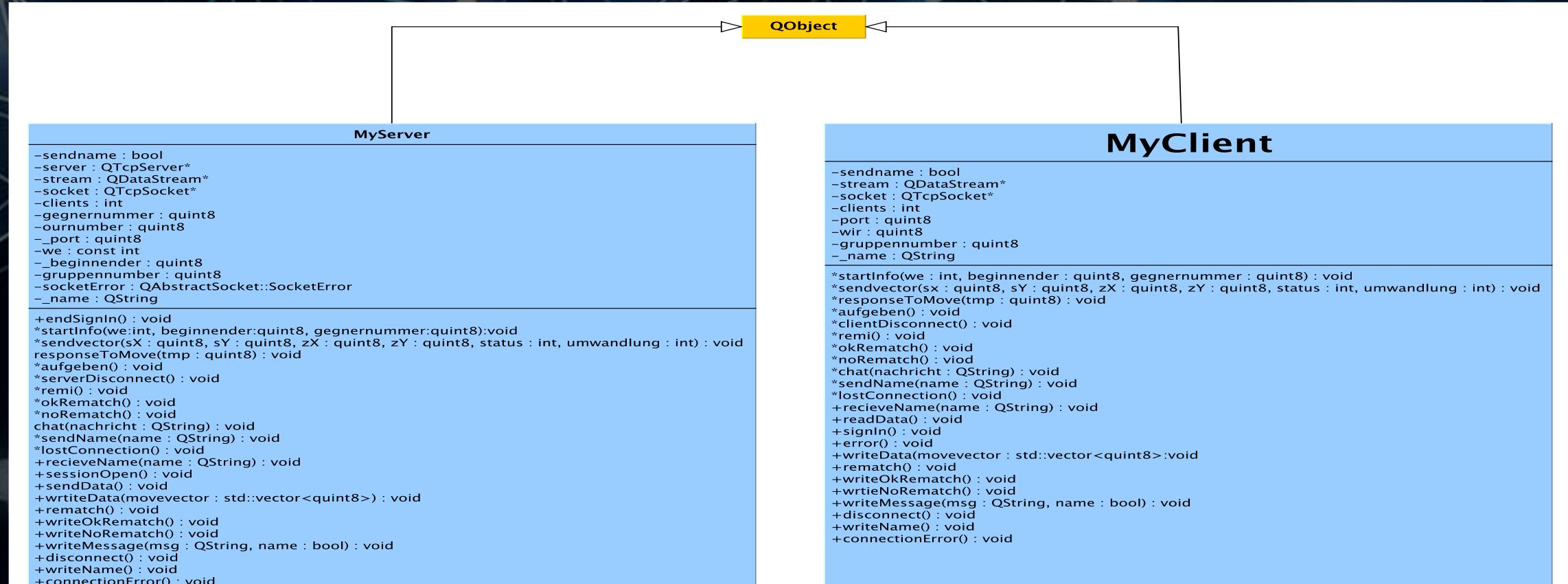
Verbindung

```
#include "myclient.h"
```

```
MyClient::MyClient(QString _host, quint16 _port, QObject* parent):QObject(parent),  
    host(_host),port(_port)
```

if(MyServer && MyClient) return Verbindung;

- Sorgt für die Verbindung und stellt sicher, dass der Port geöffnet wird.
- Der Client verbindet sich mit dem Port



Probleme

- Ein ganz neues Konzept -> etwas wenig Beispiele
- QT ist eine riesige Welt mit vielen Funktionilitäten -> was wann?
- Mit steigender Funktionalität steigende Unübersichtlichkeit,
kompliziertere Fehlersuche
- Erst mit der Zeit merkt man, dass ausgewählte QT Klassen nicht alles
können was man benötigt
- Unpassende Slots für Signale wegen der Anzahl der Parameter beim
Verbinden.

Testen

- Network braucht leider die anderen Bauteile, um getestet werden zu können.
- Deshalb wurde erst auf Compiler Errors geachtet und erst dann wenn alles zusammengefügt wurde, wurde an den Logic-/Laufzeiterrors weitergearbeitet.
- Beim Testen ist die folgende Warning häufig aufgetreten.

⚠ implicit conversion loses integer precision: 'const int' to 'quint8' (aka 'unsigned char')

myclient.cpp

46

Maßnahmen zur Sicherheit und Stabilität

- An manchen Stellen wurden mit Absicht zusätzliche Scopes erzeugt.
- Überflüssige bzw. unwichtige Bytes wie die Länge bei der read() Funktion wurden von wichtigen abgetrennt und entsorgt.
- Error Signale/Slots wurden implementiert, falls eine ungültige Eingabe eingeliefert wird bzw. das Programm verfügt über die entsprechende Implementierung nicht.

```
case(0x03) : {
    quint8 lenght; //um das erste Byte zu entfernen
    quint8 letztesbyte;
    stream>> lenght;
    for(int i=0; i<4; i++){
        quint8 tmp;
        stream >> tmp;
        vector.push_back(tmp);
        qDebug() << "recieved from server" << tmp;
    }
}
```

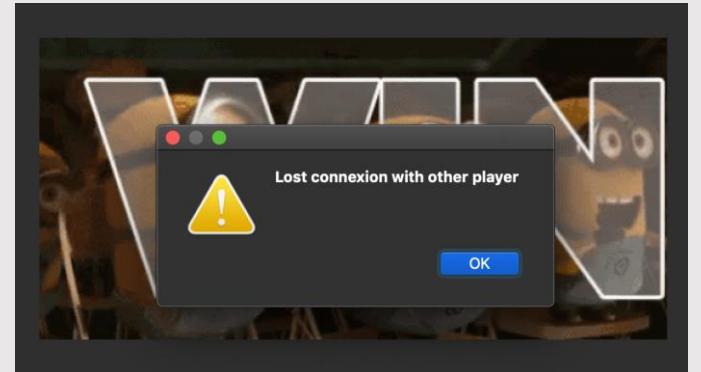
Testen

- Command Line ausgaben und QT Debugger
- Logik Tests durch Ausgabe des Spielfelds in der Command Line
- Einfache Chat Anwendung zum Test des Netzwerks
- Nach Zusammenführen Lokales Spiel auf einem Gerät, später gegen andere Teams

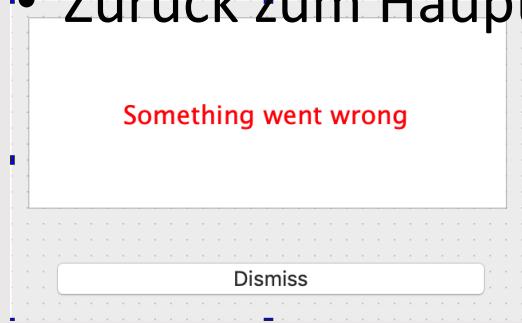
T	S	.	D	K	L	.	T
.	B	.	.	B	.	B	B
.	.	B	.	L	.	.	S
B	B	.	.
.	.	.	.	B	.	.	B
.	D	S
B	B	.	B	.	B	B	.
T	.	L	.	K	L	S	T

```
other pieces 16
other pieces 19
other pieces 22
other pieces 23
other pieces 24
other pieces 24
other pieces 25
other pieces 28
other pieces 29
other pieces 32
other pieces 33
other pieces 38
other pieces 39
my Points 0 your points 2 =====
my max Points 39 your max points 39 ==
write data to server
finisheds
was server hat 0 0
```

Stabilität



- Nach Beenden einer Spielsitzung werden alle Netzwerkverbindungen getrennt
- Wenig Potential für Fehleingaben da Buttons deaktiviert werden
- Verbindungsabbruch (unerwartet)
 - Empfangen von Signalen die laut eigener Logik fehlerhaft sind
 - Generell Verbindungsabbruch
- Error() Signal geworfen bei Fehlern in der internen Kommunikation
 - Zurück zum Hauptbildschirm, Verbindungsabbruch und Meldung



Zusatzfunktionalität

- Surrender Funktionalität mit Abbruch der Verbindung
 - 0xFF
- Rematch anfrage mit Antwort des Gegners
 - 0XF5(Anfrage)&0XF6(Antwort)
- Eigenständiges Chatfenster
 - 0XF7(Nachrichten)&0XF8(Name)
- Punktecounter
- Rundenanzeige
- Hervorheben des Spielers der am Zug ist
- Off Field für geschlagene Figuren
- Zwei Figurenpakete zur Auswahl

Vielen Dank für Ihre
Aufmerksamkeit

