FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ

UNIVERSITATEA TEHNICĂ A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT

LUCRARE DE LABORATOR #6

LUCRU IN ECHIPA.APLICATIE COMPLEXA

Autor:

st. gr. TI-142

Damaschin Ion

lector asistent:

Irina COJANU

lector superior:

Svetlana COJOCARU

LUCRARE DE LABORATOR #6

1. Obiectivele lucrării

- Crearea unei aplicatii complexe in echipa.
- Divizarea sarcinilor pe membrii echipei

2. Efectuarea lucrării de laborator

2.1. Task-uri implementate

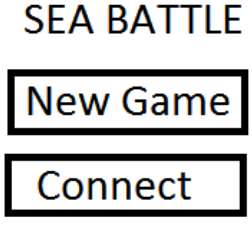
- Dezvoltarea unei aplicatii:
  - Game development (web, mobile, desktop)

2.2. Realizarea lucrării de laborator

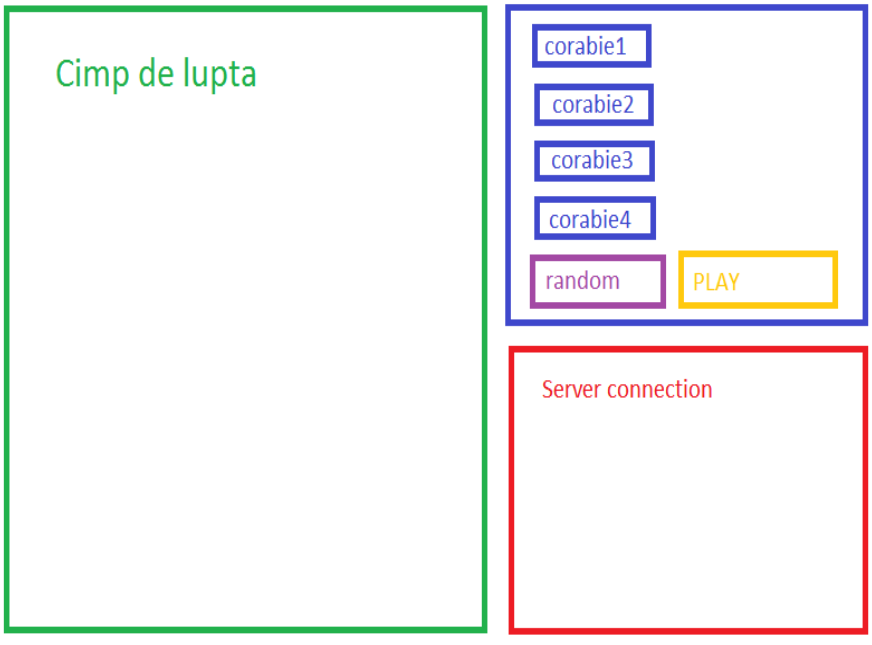
In aceasta lucrare de laborator am realizat un joc ,si anume cunoscutul SeaBattle. Proiectul a fost realizat in Qt.

Schematic jocul contine 3 ferestre de baza:

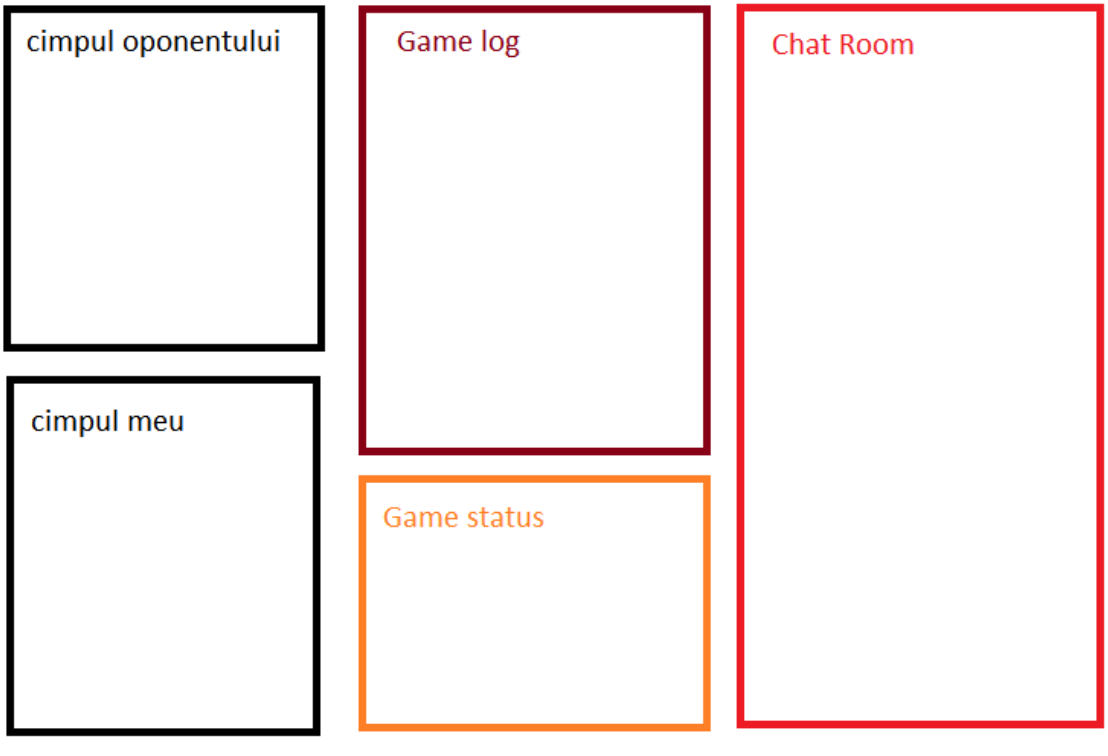
- Fereastra care apare la rularea jocului



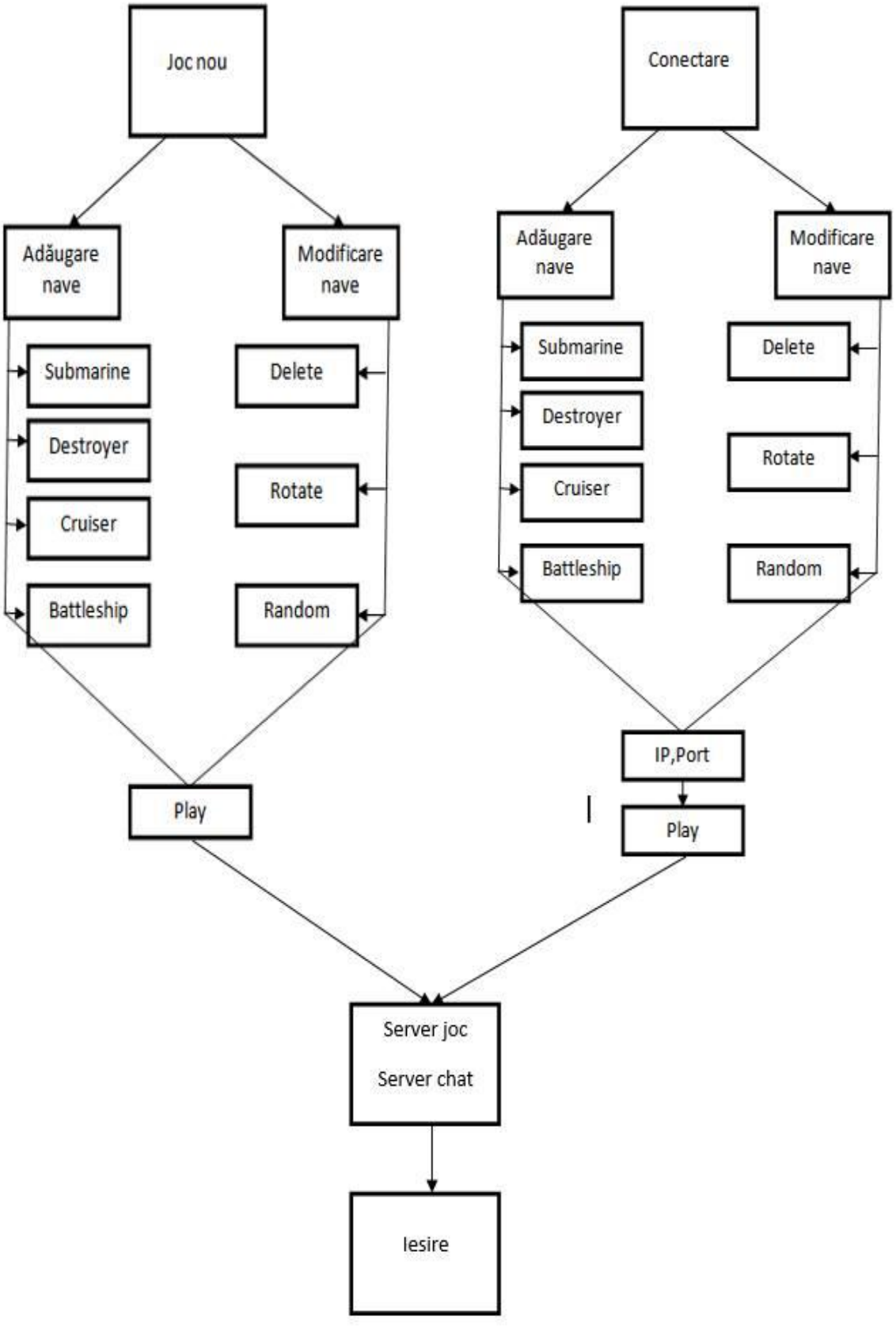
- Fereastra care apare la tastarea butonului new game(aproape identica cu cea la tastarea connect):



- Schema cimpului de joc:



- Schema functionala a programului:



In proiectul respectiv am fost responsabil de crearea unui mini chat si de realizarea cimpului de lupta si functiile necesare.

Chatul este compus din partea de client si partea de server.

- Chatclient

```
private slots:
    void setConnected();
    void setDisconnected();
    void toggleConnection();
    void sendMessage();
    void receiveMessage();
```

Dupa cum vedem are doar 4 sloturi (functii) .

```
void ChatClient::sendMessage()
{
      // "<nick> message\n"
 if(ans = true)
     socket->write( "<font color=\"Red\">"+  nick->text().toLatin1() + ": "+"</font>" + message->text().toLatin1() + "\n");
   else
    if(ans = false)
       socket->write( "<font color=\"Aqua\">"+  nick->text().toLatin1() + ": "+"</font>" + message->text().toLatin1() + "\n");

    message->clear();
}

void ChatClient::receiveMessage()
{
    // missing some checks for returns values for the sake of simplicity
    qint64 bytes = buffer->write(socket->readAll());
    // go back as many bytes as we just wrote so that it can be read
    buffer->seek(buffer->pos() - bytes);
    // read only full lines, line by line
    while (buffer->canReadLine())
    {
        QString line = buffer->readLine();
        chat->append(line.simplified());
    }
}
```

Pentru transmiterea datelor am folosit protocolul TCP.Datele sunt transmise prin tipul qint64 care corespunde cu long int.Datele primite (mesajul oponentului) sunt inserate in fereastra chatului.

- Chatserver

```
private slots:
    void addConnection();
    void removeConnection();
    void receiveMessage();
```

Functiile sunt asemanatoarea cu cele din chatclient cu exceptia addConnection si removeConnection ,care sunt caracteristice pentru partea server.

```
void ChatServer::addConnection()
{
    QTcpSocket* connection = nextPendingConnection();
    connections.append(connection);
    QBuffer* buffer = new QBuffer(this);
    buffer->open(QIODevice::ReadWrite);
    buffers.insert(connection, buffer);
    connect(connection, SIGNAL(disconnected()), SLOT(removeConnection()));
    connect(connection, SIGNAL(readyRead()),    SLOT(receiveMessage()));
}

void ChatServer::removeConnection()
{
    QTcpSocket* socket = static_cast<QTcpSocket*>(sender());
    QBuffer* buffer = buffers.take(socket);
    buffer->close();
    buffer->deleteLater();
    connections.removeAll(socket);
    socket->deleteLater();
}
```

Fisierul cimplupta.cpp contine urmatoarele semnale si sloturi:

```
signals:
    void corabieActivata(bool val,int size); //semnal despre corabia de pe cimp aleasa de utilizator
    void corabieConflicta(bool val); //semnal despre aparitia/rezolvarea conflictului
    void celulaClicked(int x,int y); //semnal despre click in celula de pe cimp
    void miscareRaspuns(int x,int y,CELL_STATUS st,Corabie *s); //rs la miscarea oponentului
    void updateStatusLabel(); //semnal despre necesitatea unui update a statisticii

public slots:
    void rotatieCorabieActiva(); //rotatie corabie
    void adaugaCorabiePeCimp(int size); //adaugarea corabiei pe cimp
    void elimineCorabieCimp(); //stergerea corabiei
    void reset();
    void genereazaRandom();
    void deseneazaPunct(QPainter *desen,int x,int y); //desneaza punctul
    void deseneazaRanit(QPainter *desen,int x,int y); //deseneaza atunci cind -ranit
    void verificaMiscare(int x,int y); //verifica miscarea oponentului
    void verificaMiscareRaspuns(int x,int y,CELL_STATUS st, QPoint poz,int size,direction dir); //verifica raspunsul oponentului la miscare
    void puncteCorabieDistrusa(QPoint pos,int size,direction dir); //punctele imprejurul corabiei distruse
    void corabieNeactiva(); //deactivarea corabiei

protected:
    void paintEvent(QPaintEvent *event); //evenimentul desenarii cimpului
    void mousePressEvent(QMouseEvent *event);//evenimentul click pe cimp
    void mouseMoveEvent(QMouseEvent *event);//evenimentul deplasarii cursorului pe cimp
    void mouseReleaseEvent(QMouseEvent *event);//evenimentul eliberarii butonului mouse
```

Toate corabiile create in timpul jocului sunt salvate intr-o structura de tip lista. La distrugerea uneia din nave,de catre oponent sau a oponetului,ea se sterge din lista.
Pentru generarea random se obtine pointer la nava ,aleator se alege orientarea si pentru fiecare tip de nava se creaza un ciclu pentru amplasarea lor si verificarea de conflicte.
Functiile verificareMiscare si verificareMiscareRaspuns sunt respectiv pentru verificarea miscarilor.
La inceput celula e goala,apoi se creaza un ciclu in care se verifica daca oponentul a lovit in nava.Daca o deterioreaza sau distruge ,tot rindul ramine la el.Daca esueaza atunci rindul vine la noi,iar la el click-ul devine imposibil.
Datele despre miscarile efectuate se transmit prin TCP.si totodata se inscriu in aria pentru loguri.

Concluzie:

In urma efectuarii acestui proiect am studiat mai aprofundat Qt.
Pot sa spun ca Qt este o platforma foarte puternica ,in care se poate de creat tot feluri de proiecte.Este relativ usor de invatat,dar contine foarte multe functii.Am aflat mai multe despre semnale si sloturi,care sunt caracteristice doar pentru aceasta platforma.
Cel mai greu in partea mea de proiect a fost sa lucrez cu navele,deoarece tot designul a fost in cod.Nu am folosit redactorul de forme inclus in platforma.
Chatul a fost usor de creat deoarece Qt contine clase si functii predestinate pentru lucrul in retea.
Din pacate inca nu am reusit sa realizam ca proiectul sa lucreze in retele diferite,respectiv jocul e posibil doar la 2 PC-uri aflate in aceiasi retea,dar speram ca pe viitor sa inlaturam aceasta problema si sa imbunatatim considerabil jocul.