```cpp
#include "carte.h"
#include "iostream"

//constructeur
carte::carte():point_click(0,0),point_depart(0,0),point_release(0,0),point1_gps(
0,0),point1(0,0),point2_gps(0,0),point2(0,0),coul(255255255),md5(""),source(""),
source_chemin(""),carteDessiner(false),coord_gps(false),enregistrer(false),trace
r(false),nbpoint(0),flags(0),etendueZone(10)
{

    //largeur= QApplication::desktop()->width()-100;
    //hauteur = QApplication::desktop()->height()-100;
    this->setMinimumSize(largeur,hauteur);
    imageCarte = new QImage();
    imageAffichage= new QImage();
    tracerChemin= new QImage();
    copieTailleNormale= new QImage();

    QObject::connect(this, SIGNAL(ChangeZoomIn()),this, SLOT(augmenter_zoom()));
    QObject::connect(this, SIGNAL(ChangeZoom()),this, SLOT(diminuer_zoom()));
    QObject::connect(this, SIGNAL(signalDessinerChemin(QPoint)),this,
SLOT(dessinerChemin(QPoint)));
    QObject::connect(this, SIGNAL(changeRes2(QPoint)),this,
SLOT(attributCouleur(QPoint)));
    QObject::connect(this, SIGNAL(changeRes2(QPoint)),this,
SLOT(sauvegardeItineraire(QPoint)));
    QObject::connect(this,SIGNAL(SignalFlag(QPoint)),this,SLOT(placerFlag1(QPoin
t)));
    QObject::connect(this,SIGNAL(SignalFlag(QPoint)),this,SLOT(placerFlag2(QPoin
t)));
}

//destructeur
carte::~carte(){}

//accesseur
bool carte::getCarteDessiner(){return carteDessiner;}

QPoint carte::getPoint() {return point_click;}

int carte::getFlags() {return flags;}

QRgb carte::getCouleur() {return coul;}

QPoint carte::getPoint1() {return point1_gps;}

QPoint carte::getPoint2() {return point2_gps;}

coord_decimal carte::getCoordDec() {return dec;}

coord_decimal carte::getCoordDec1() {return dec1;}

bool carte::test_carte() {return carteDessiner;}

bool carte::test_enregistrer() { return enregistrer;}

//mutateur
void carte::setCartedessiner(bool choix) {carteDessiner= choix;}
```

```
void carte::setFlags(int f) {flags=f;}

void carte::setCouleur(QRgb c) {coul = c;}

void carte::setPoint(QPoint p) {point_click=p;}

void carte::setPoint1(QPoint p) {point1_gps=p;}

void carte::setPoint2(QPoint p) {point2_gps=p;}

void carte::setCoordDec(double la,double lo,double la1,double lo1)
{

        std::cout<<"lat : "<<dec.toSexaLa(la).toStdString()<<"long :
"<<dec.toSexaLo(lo).toStdString()<<std::endl;
            dec.setLatitude(la);
            dec.setLongitude(lo);

            dec1.setLatitude(la1);
            dec1.setLongitude(lo1);

}

void carte::setCoordSeg(int d1, int m1,double s1,int dd1, int mm1,double ss1,int
d2, int m2, double s2,int dd2, int mm2, double ss2)
{
  dec.toDecLa(d1,m1,s1);
  dec.toDecLo(dd1,mm1,ss1);
  dec1.toDecLa(d2,m2,s2);
  dec1.toDecLo(dd2,mm2,ss2);

}

void carte::setTest_enregistrer(bool b)
{
   enregistrer=b;
}

void carte::setTest_carte(bool b)
{
    carteDessiner=b;
}

//fonctions
void carte::calcul_md5(QString src)
{
    QFile image (src);
    if (image.open(QFile::ReadOnly)) {
        QByteArray contenuFichier = image.readAll();
        QByteArray hashData =
QCryptographicHash::hash(contenuFichier,QCryptographicHash::Md5);
        md5= hashData.toHex();
    }
}

void carte::afficherCarte(QString chemin){
    source_chemin= chemin;
    if (!chemin.isNull()) {
    echelle= 1.0;
```

```
    imageCarte= new QImage(chemin);
    std::cout<<"image: "<<chemin.toStdString()<<std::endl;
    int width= imageCarte->width();
    int height=imageCarte->height();

    if (width>height){
    QImage newImage= (imageCarte->scaledToWidth(1201,Qt::SmoothTransformation));
    imageCarte=new QImage(newImage);
    imageAffichage= new QImage(newImage);
    copieTailleNormale= new QImage(newImage);
    //std::cout<<"you piss off"<<std::endl;
    }
    else {
    QImage newImage= (imageCarte->scaledToHeight(651,Qt::SmoothTransformation));
    imageCarte=new QImage(newImage);
    imageAffichage= new QImage(newImage);
    copieTailleNormale= new QImage(newImage);
     //std::cout<<"you piss off"<<std::endl;
    }
     //std::cout<<"ça passe"<<std::endl;
    largeur=imageCarte->width();
    hauteur=imageCarte->height();
    carteDessiner=true;
    tracerChemin=new QImage(largeur,hauteur,QImage::Format_ARGB32);
    tracerChemin2=new QImage(largeur,hauteur,QImage::Format_ARGB32);
    while(!pile.isEmpty()){
        pile.pop();
    }

    while(!pile_release.isEmpty()){
        pile_release.pop();
    }

    calcul_md5(chemin);

    p1=new QImage("gps2.png");
    p2=new QImage("gps2.png");

    //std::cout<<"hauteur : "<<hauteur<<std::endl;
    //std::cout<<"largeur : "<<largeur<<std::endl;
    update();

    }

}

void carte::exporter_gpx()
{
    if (tracer==true) {
        QString str =  QFileDialog::getSaveFileName(this, tr("Exporter le projet
en .gpx"),"/home/Export_gpx"+QDate::currentDate().toString()+".gpx",tr("Fichier
(*.gpx)"));
        QString entete = "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n<gpx
version=\"1.1\"creator=\"Projet Stage RAKOTONIARY SOMBI @ BEILLEAU
QUENTIN\">\n<trk>\n<name>Tracking GPS</name>\n<trkseg>\n";
        QString fin = "</trkseg>\n</trk>\n</gpx>";
        QString points = "";
        int i=0;
        QStack<QPoint> tmp = pile;
```

```
        while(!tmp.isEmpty())
        {
        QPoint var = tmp.pop();
        point_gps p = pt_gps(point1,point2,var);
        points =points+" <trkpt lat="+QString::number(p.X())+"
lon="+QString::number(p.Y())+"><cmt>Point "+QString::number(i)
+"</cmt></trkpt>\n";
        i++;
        }

        //str = str+"/Export_gpx"+QDate::currentDate().toString()+".gpx";//voir
comment mieux utiliser le chemin voir a créer un dossier

        QFile file(str);
        if (file.open(QFile::WriteOnly | QIODevice::Text | QIODevice::Truncate))
{
             QTextStream out(&file);
             out << entete<< points << fin;
        } else QMessageBox::critical(this, "Attention : ", trUtf8("Impossible
d'enregistrer le fichier à cet emplacement. Merci de choisir un emplacement
valide."));
    } else QMessageBox::critical(this, "Attention : ", trUtf8("Un chemin doit-
être d'abord tracé."));
}

void carte::sauvegarde_sous()
{
    if (enregistrer==false){
        source = QFileDialog::getSaveFileName(this, trUtf8("Sauvegarder le
projet "),"/home/projet_gpx-"+QDateTime::currentDateTime().toString()
+".xml",trUtf8("Fichier (*.xml)"));
        QStack<QPoint> tmp = pile;
        QStack<QPoint> tmp1 = pile_release;
        if (source=="") enregistrer = false;
        else {
           //
source=source+"/projet_gpx-"+QDateTime::currentDateTime().toString()
+".xml";//voir comment mieux utiliser le chemin voir a créer un dossier
            QFile file(source);

            if (file.open(QFile::WriteOnly| QIODevice::Text |
QIODevice::Truncate)) {
                QTextStream out(&file);
                out<<"<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n";
                out<<"<md5sum>\n"<<md5<<"\n</md5sum>\n";
                out<<"<image>\n"<<source_chemin<<"\n</image>\n";
                out<<"<point
icone>\n"<<point1_gps.x()<<"\n"<<point1_gps.y()<<"\n</point icone>\n";
                out<<"<point click
icone>\n"<<point1.x()<<"\n"<<point1.y()<<"\n</point click icone>\n";
                out<<"<point1
icone>\n"<<point2_gps.x()<<"\n"<<point2_gps.y()<<"\n</point1 icone>\n";
                out<<"<point1 click
icone>\n"<<point2.x()<<"\n"<<point2.y()<<"\n</point1 click icone>\n";
                out<<"<lat>\n"+QString::number(dec.getLatitude())
+"\n</lat>\n<lon>\n"+QString::number(dec.getLongitude())+"\n</lon>\n";
                out<<"<lat1>\n"+QString::number(dec1.getLatitude())
+"\n</lat1>\n<lon1>\n"+QString::number(dec1.getLongitude())+"\n</lon1>\n";
                while(!tmp.isEmpty())
                {
                      QPoint p = tmp.pop();
```

```
                        out<<"<point>\n"<<p.x()<<"\n"<<p.y()<<"\n</point>\n";
                        QPoint p1 = tmp1.pop();
                        out<<"<point
release>\n"<<p1.x()<<"\n"<<p1.y()<<"\n</point realease>\n";

                    }
                    file.close();
                }
                enregistrer = true;

            }
        } else if (enregistrer==true){

                    QStack<QPoint> tmp = pile;
                    QStack<QPoint> tmp1 = pile_release;
                    QFile file(source);
                    if (file.open(QFile::WriteOnly| QIODevice::Text |
QIODevice::Truncate)) {
                        QTextStream out(&file);
                        out<<"<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n";
                        out<<"<md5sum>\n"<<md5<<"\n</md5sum>\n";
                        out<<"<image>\n"<<source_chemin<<"\n</image>\n";
                        out<<"<point
icone>\n"<<point1_gps.x()<<"\n"<<point1_gps.y()<<"\n</point icone>\n";
                        out<<"<point click
icone>\n"<<point1.x()<<"\n"<<point1.y()<<"\n</point click icone>\n";
                        out<<"<point1
icone>\n"<<point2_gps.x()<<"\n"<<point2_gps.y()<<"\n</point1 icone>\n";
                        out<<"<point1 click
icone>\n"<<point2.x()<<"\n"<<point2.y()<<"\n</point1 click icone>\n";
                        out<<"<lat>\n"+QString::number(dec.getLatitude())
+"\n</lat>\n<lon>\n"+QString::number(dec.getLongitude())+"\n</lon>\n";
                        out<<"<lat1>\n"+QString::number(dec1.getLatitude())
+"\n</lat1>\n<lon1>\n"+QString::number(dec1.getLongitude())+"\n</lon1>\n";
                        while(!tmp.isEmpty())
                        {
                              QPoint p = tmp.pop();
                              out<<"<point>\n"<<p.x()<<"\n"<<p.y()<<"\n</point>\n";
                              QPoint p1 = tmp1.pop();
                              out<<"<point
release>\n"<<p1.x()<<"\n"<<p1.y()<<"\n</point realease>\n";
                        }
                       file.close();
                    }

            }

}

void carte::zoom(float valeur){
    echelle = (valeur * echelle);
    valeurZoom->setText(QString::number(echelle) );

}

void carte::dessinerChemin(const QPoint &p){

    qDebug()<<"dessiner chemin";
    //parcours entier de la carte
    //parcoursImageAffichage();
```

```
    // capture point de release souris
    point_release=p;
    pile_release.push(p);
    tracer=true;

     // tracer départ du chemin
    tracerZone(point_click,coul);
    //int i=tracerZone(point_click,coul);
    //std::cout<<"zone : "<<i<<std::endl;

    // calcul orientation (release devient l'orientation)
    if (abs(point_release.x())>abs(point_release.y())){
            if (point_release.x()>0)
                point_release=QPoint (point_click.x()
+etendueZone,point_click.y());
                else
                point_release=QPoint (point_click.x()-
etendueZone,point_click.y());
    }
    else {
        if (point_release.y()>0)
            point_release=QPoint (point_click.x(),point_click.y()+etendueZone);
            else
            point_release=QPoint (point_click.x(),point_click.y()-etendueZone);
    }

   // std::cout<<"point de base: "<<point_click.x()<<"
"<<point_click.y()<<std::endl;
   // std::cout<<"point release "<<point_release.x()<<"
"<<point_release.y()<<std::endl;
   QPoint resultat=directionChemin();

   // trace suite chemin jusqu'à sortie de chemin
   while(resultat!=QPoint(0,0)&&
         (resultat.x()<=largeur)&&
         (resultat.x()>=0)&&
         (resultat.y()>=0)&&
         (resultat.y()<=hauteur))
   {
      //std::cout<<"resultat: "<<resultat.x()<<" "<<resultat.y()<<std::endl;

      point_release.setX(resultat.x()+(resultat.x()-point_click.x()));
      point_release.setY(resultat.y()+(resultat.y()-point_click.y()));

      point_click.setX(resultat.x());
      point_click.setY(resultat.y());
      resultat=directionChemin();
  //      std::cout<<"new point  de base: "<<point_click.x()<<"
"<<point_click.y()<<std::endl;
  //      std::cout<<"new release x: "<<point_release.x()<<" new release y:
"<<point_release.y()<<std::endl;
    }
update();
}

QPoint carte::directionChemin(){
    int margeErreur=2;
    int choix=0;
    //0: arret   1: gauche   2:haut    3:droite   4: bas
    QPoint ancienneOrientation (point_release.x()-
point_click.x(),point_release.y()-point_click.y());
```

```
    if (ancienneOrientation==QPoint(-etendueZone,0)) choix=1;
    else if (ancienneOrientation==QPoint(0,-etendueZone)) choix=2;
        else if (ancienneOrientation==QPoint(etendueZone,0)) choix=3;
            else if (ancienneOrientation==QPoint(0,etendueZone)) choix=4;
        //std::cout<<"ancienne DIrection: "<<ancienneOrientation.x()<<"
"<<ancienneOrientation.y()<<std::endl;

    switch (choix){
    //gauche
            case (1):
    {
                 int gauche=(tracerZone(QPoint(point_click.x()-
etendueZone,point_click.y()),coul));
                //int droite=(tracerZone(QPoint(point_click.x()
+etendueZone,point_click.y()),coul));
                int haut=(tracerZone(QPoint(point_click.x(),point_click.y()-
etendueZone),coul));
                int bas=(tracerZone(QPoint(point_click.x(),point_click.y()
+etendueZone),coul));
                int nouvelleOrientation=maximum(gauche,maximum(haut,bas));

                //std::cout<<"nouvelle DIrection:
"<<nouvelleOrientation<<std::endl;
                if (nouvelleOrientation>margeErreur){
                     //std::cout<<"nouvelle DIrection2: "<<nouvelleOrientation
<<std::endl;
                   // return QPoint(point_click.x()-
etendueZone,point_click.y());

                    if (nouvelleOrientation == gauche) return
QPoint(point_click.x()-etendueZone,point_click.y());
                    else if (nouvelleOrientation == haut) return
QPoint(point_click.x(),point_click.y()-etendueZone);
                               else if (nouvelleOrientation == bas) return
QPoint(point_click.x(),point_click.y()+etendueZone);

                }
                return QPoint(0,0);
    }
                break;

    //haut
            case (2):
    {
                int gauche=(tracerZone(QPoint(point_click.x()-
etendueZone,point_click.y()),coul));
                int droite=(tracerZone(QPoint(point_click.x()
+etendueZone,point_click.y()),coul));
                int haut=(tracerZone(QPoint(point_click.x(),point_click.y()-
etendueZone),coul));
                //int bas=(tracerZone(QPoint(point_click.x(),point_click.y()
+etendueZone),coul));
                int nouvelleOrientation=maximum(haut,maximum(gauche,droite));

                std::cout<<"nouvelle DIrection 3: "<<nouvelleOrientation
<<std::endl;
                if (nouvelleOrientation>margeErreur){
                    if (nouvelleOrientation == gauche) return
QPoint(point_click.x()-etendueZone,point_click.y());
                    else if (nouvelleOrientation == haut) return
QPoint(point_click.x(),point_click.y()-etendueZone);
```

```
                        else if (nouvelleOrientation == droite) return
QPoint(point_click.x()+etendueZone,point_click.y());
                   // std::cout<<"nouvelle DIrection 4: "<<nouvelleOrientation
<<std::endl;
                    //return QPoint(point_click.x(),point_click.y()-
etendueZone);

                }
                return QPoint(0,0);
      }
                break;

    //droite
            case (3):
    {
                //int gauche=(tracerZone(QPoint(point_click.x()-
etendueZone,point_click.y()),coul));
                int droite=(tracerZone(QPoint(point_click.x()
+etendueZone,point_click.y()),coul));
                int haut=(tracerZone(QPoint(point_click.x(),point_click.y()-
etendueZone),coul));
                int bas=(tracerZone(QPoint(point_click.x(),point_click.y()
+etendueZone),coul));
                int nouvelleOrientation=maximum(droite,maximum(haut,bas));
               // std::cout<<"nouvelle DIrection 5: "<<nouvelleOrientation
<<std::endl;
                if (nouvelleOrientation>margeErreur){
                     if (nouvelleOrientation == haut) return
QPoint(point_click.x(),point_click.y()-etendueZone);
                        else if (nouvelleOrientation == droite) return
QPoint(point_click.x()+etendueZone,point_click.y());
                             else if (nouvelleOrientation == bas) return
QPoint(point_click.x(),point_click.y()+etendueZone);
                    //std::cout<<"nouvelle DIrection 6: "<<nouvelleOrientation
<<std::endl;
                   // return QPoint(point_click.x()
+etendueZone,point_click.y());

                }
                return QPoint(0,0);
      }
                break;

    //bas
            case (4):
    {
                int gauche=(tracerZone(QPoint(point_click.x()-
etendueZone,point_click.y()),coul));
                int droite=(tracerZone(QPoint(point_click.x()
+etendueZone,point_click.y()),coul));
                //int haut=(tracerZone(QPoint(point_click.x(),point_click.y()-
etendueZone),coul));
                int bas=(tracerZone(QPoint(point_click.x(),point_click.y()
+etendueZone),coul));
                int nouvelleOrientation=maximum(bas,maximum(gauche,droite));

//std::cout<<"nouvelle DIrection 7: "<<nouvelleOrientation <<std::endl;
                if (nouvelleOrientation>margeErreur){
                    if (nouvelleOrientation == gauche) return
QPoint(point_click.x()-etendueZone,point_click.y());
                        else if (nouvelleOrientation == droite) return
```

```
QPoint(point_click.x()+etendueZone,point_click.y());
                                else if (nouvelleOrientation == bas) return
QPoint(point_click.x(),point_click.y()+etendueZone);

                    //std::cout<<"nouvelle DIrection 8: "<<nouvelleOrientation
<<std::endl;
                     //return QPoint(point_click.x(),point_click.y()
+etendueZone);
                }
                return QPoint(0,0);
     }
                break;
}
     return QPoint(0,0);
}

void carte::parcoursImageAffichage(){

    //parcourir toute la carte
    for (int i=0;i<largeur;i++){
         for (int j=0; j<hauteur;j++){
             if (comparerCouleurAvecMarge( imageCarte->pixel(i,j),coul)==true)
                {imageCarte->setPixel(QPoint(i,j),0xFF00FF00);}
            // else {imageAffichage->setPixel(QPoint(i,j),0xFFFFFF);}
           // std::cout<<"couleur : "<<coul<<std::endl;}
             //imageCarte->setPixel(QPoint(i,j),coul);
         }
     }

}

void carte::sauvegardeItineraire(const QPoint &p){
    pile.push(p);

}

void carte::charger()
{

    QString fichier = QFileDialog::getOpenFileName(this, "Charger un projet",
"/home", "Fichier (*.xml)");
    QFile file(fichier);
    QStack<QPoint> tmp;
    QStack<QPoint> tmp1;
    if (file.open(QFile::ReadOnly| QIODevice::Text | QIODevice::Truncate)) {
         flags=1;
         QTextStream in(&file);
         QStringList liste;
         QString ligne;
         QString src;
         while(!in.atEnd()){
           ligne = in.readLine();
           liste+=ligne;
           }

         bool valid = file.exists(liste.at(5));
         if (!valid) { src = QFileDialog::getOpenFileName(this, "Ouvrir un
fichier", QString(), "Images (*.png *.gif *.jpg *.jpeg)");

         } else src = liste.at(5);
```

```
          calcul_md5(src);
          QString md5s = liste.at(2);
          std::cout<<"md5 :"<<md5.toStdString()<<" md5s
:"<<md5s.toStdString()<<std::endl;
          if (md5==md5s) {
                for(int j=0;j<liste.size();j++) {
                    ligne = liste.at(j);
                    if(ligne=="<image>") afficherCarte(src);

                    if(ligne=="<lat>") dec.setLatitude(liste.at(j+1).toDouble());
                    if(ligne=="<lon>")
dec.setLongitude(liste.at(j+1).toDouble());
                    if(ligne=="<lat1>")
dec1.setLatitude(liste.at(j+1).toDouble());
                    if(ligne=="<lon1>")
dec1.setLongitude(liste.at(j+1).toDouble());
                    if (ligne=="<point icone>") {
                         QPoint p;
                         p.setX(liste.at(j+1).toInt());
                         p.setY(liste.at(j+2).toInt());

                         point1_gps=p;
                     }
                    if (ligne=="<point1 icone>") {
                         QPoint p;
                         p.setX(liste.at(j+1).toInt());
                         p.setY(liste.at(j+2).toInt());

                         point2_gps=p;
                     }
                    if (ligne=="<point click icone>") {

                         QPoint p;
                         p.setX(liste.at(j+1).toInt());
                         p.setY(liste.at(j+2).toInt());

                         point1=p;
                     }
                    if (ligne=="<point1 click icone>") {
                         std::cout<<"point :"<<liste.at(j+1).toStdString()<<std::
endl;
                         QPoint p;
                         p.setX(liste.at(j+1).toInt());
                         p.setY(liste.at(j+2).toInt());

                         point2=p;
                     }
                    if (ligne=="<point>") {
                         std::cout<<"point :"<<liste.at(j+1).toStdString()<<std::
endl;
                         QPoint p;
                         p.setX(liste.at(j+1).toInt());
                         p.setY(liste.at(j+2).toInt());
                         std::cout<<"point click :"<<p.x()<<""<<p.y()<<std::endl;
                         tmp.push(p);
                     }
                    if (ligne=="<point release>") {
                         QPoint p;
                         p.setX(liste.at(j+1).toInt());
                         p.setY(liste.at(j+2).toInt());
                         std::cout<<"point
```

```
release:"<<p.x()<<""<<p.y()<<std::endl;
                        tmp1.push(p);
                    }
                   std::cout<<"affichage :"<<liste.at(j).toStdString()<<std::end
l;
               }
              // std::cout<<"lat : "<<dec.getLatitude()<<" lon :
"<<dec.getLongitude()<<std::endl;
               // std::cout<<"lat : "<<dec1.getLatitude()<<" lon :
"<<dec1.getLongitude()<<std::endl;
         } else QMessageBox::critical(this, "Attention", trUtf8("L'image chargée
n'est pas la même que celle du projet sauvegarder. Merci de charger celle qui
convient."));

    file.close();
    }
    nbpoint=2;
    qDebug()<<tmp.at(0)<<" tmp1 : "<<tmp1.at(0);

    while(!tmp.isEmpty()){
        if (!tmp1.isEmpty()) {
            point_click = tmp.pop();
            attributCouleur(point_click);
            sauvegardeItineraire(point_click);
            dessinerChemin(tmp1.pop());

         } else QMessageBox::critical(this, "Attention", trUtf8("Le fichier est
corrompu !"));

    }
    if (!tmp1.isEmpty()) dessinerChemin(tmp1.pop());

}

point_gps carte::pt_gps(QPoint a, QPoint b, QPoint c)
{

    point_gps res;
    double x1 = (b.x()-a.x());
    double y1 = (b.y()-a.y());
    //std::cout<<" x1 "<<x1<<" y1 "<<y1<<std::endl;
    double x2 = (c.x()-a.x());
    double y2 = (c.y()-a.y());
    //std::cout<<" x2 "<<x2<<" y2 "<<y2<<std::endl;
    double y_gps = (dec1.getLatitude()-dec.getLatitude());
    double x_gps = (dec1.getLongitude()-dec.getLongitude());
    //std::cout<<" x_gps "<<x_gps<<" y_gps "<<y_gps<<std::endl;
    double var = ((x2*x_gps)/x1);
    double var2 = ((y2*y_gps)/y1);
    //std::cout<<" var "<<var<<" var2 "<<var2<<std::endl;
    double res_x = dec.getLongitude()+var;
    double res_y = dec.getLatitude()+var2;

    res.setX(res_x);
    res.setY(res_y);

    return res;
}

//slots
```

```
void carte::augmenter_zoom(){
    std::cout<<"zoom in"<<std::endl;
    //zoom(1.25);
    echelle*=1.25;
    QImage newImage= (copieTailleNormale-
>scaled(largeur*echelle,hauteur*echelle));
    //QImage newImage2= (imageAffichage-
>scaled(largeur*echelle,hauteur*echelle));
    QImage newImage3= (tracerChemin2->scaled(largeur*echelle,hauteur*echelle));

    imageAffichage=new QImage(newImage);
    //imageAffichage=new QImage(newImage2);
    tracerChemin=new QImage(newImage3);
    largeur=imageCarte->width();
    hauteur=imageCarte->height();
    update();
}

void carte::diminuer_zoom(){
    std::cout<<"zoom out"<<std::endl;
    //zoom(0.8);
    echelle*=0.8;
    QImage newImage= (copieTailleNormale-
>scaled(largeur*echelle,hauteur*echelle));
    //QImage newImage2= (imageAffichage-
>scaled(largeur*echelle,hauteur*echelle));
    QImage newImage3= (tracerChemin2->scaled(largeur*echelle,hauteur*echelle));

    imageAffichage=new QImage(newImage);
    //imageAffichage=new QImage(newImage2);
    tracerChemin=new QImage(newImage3);
    largeur=imageCarte->width();
    hauteur=imageCarte->height();
    update();
}

void carte::attributCouleur(const QPoint &p){
    QRgb pt ;
    setPoint(p);
    pt = imageCarte->pixel(p);
    point_depart=p;
    coul=pt;
}

void carte::fermerProjet(){

  QImage newImage(0,0,QImage::Format_ARGB32);
  hauteur=0;
  largeur=0;
  imageCarte= new QImage(newImage);
  tracerChemin= new QImage(newImage);
  imageAffichage=new QImage(newImage);
  copieTailleNormale=new QImage(newImage);
  tracer=false;
  p1= new QImage();
  p2= new QImage();
  setFlags(0);
  dec.setLatitude(0);
  dec.setLongitude(0);
  dec1.setLatitude(0);
  dec1.setLongitude(0);
```

```
  update();
// penser a remettre la couleur a blanc

}

void carte::placerFlag1(const QPoint &p)
{
    if (nbpoint==0){
        point1=p;
        int x = (p.x()-4);
        int y = (p.y()-(p1->height()));
        QPoint pt(x,y);
        std::cout<<"image 1 : "<<x<<" "<<y<<std::endl;
        nbpoint++;
        point1_gps=pt;

        update();
    }

}

void carte::placerFlag2(const QPoint &p)
{
    if (nbpoint==1){
        point2=p;
        int x = (p.x()-4);
        int y = (p.y()-(p2->height()));
        std::cout<<"image 2 : "<<x<<" "<<y<<std::endl;
        QPoint pt(x,y);
        nbpoint++;
        point2_gps=pt;

        update();

    }

}

void carte::setNbpoint()
{
    nbpoint=0;
    update();
}

//gestion des évènements
void carte::paintEvent(QPaintEvent *event)
{
    if ((carteDessiner)&&(flags==1)){
        QPainter painter(this);
        QPoint point (0,0);

        //painter.drawImage(point,*imageCarte);

        painter.drawImage(point,*imageAffichage);
        painter.drawImage(point,*tracerChemin);
        point_gps p = pt_gps(point1_gps,point2_gps,point_depart);
        std::cout<<"point : "<<point_depart.x()<<" "<<point_depart.y()<<" coord
gps :"<<p.X()<<" "<<p.Y()<<std::endl;
    }
```

```
    else if ((carteDessiner)&&(flags==2)){
            QPainter painter(this);
            QPoint point (0,0);
            painter.drawImage(point,*imageAffichage);
            if (nbpoint==1) {
                 QPainter painter(this);
                 QPoint point (0,0);
                 painter.drawImage(point,*imageAffichage);
                 painter.drawImage(getPoint1(),*p1);
                  } else if (nbpoint==2) {
                            QPainter painter(this);
                            QPoint point (0,0);
                            painter.drawImage(point,*imageAffichage);
                            painter.drawImage(getPoint1(),*p1);
                            painter.drawImage(getPoint2(),*p2);
                           }
            }
}

void carte::mousePressEvent(QMouseEvent *event)
{

    //std::cout<<"click : "<<event->x()<<" "<<event->y()<<std::endl;
    if ((carteDessiner)&&(flags==1)){
           if (event->button() == Qt::LeftButton)
            {
               emit changeRes2(event->pos());
             }
    } else if ((carteDessiner)&&(flags==2)){
        if (nbpoint==0){

                emit placerFlag1(event->pos());

                }
        else if (nbpoint==1) {

                emit placerFlag2(event->pos());

                }

    }
}

void carte::mouseReleaseEvent(QMouseEvent *event)
{
    if ((carteDessiner)&&(flags==1)){

    if (event->button() == Qt::LeftButton)
        {

           emit ChangeRes();
           emit signalDessinerChemin(event->pos());
        }

    }
}

void carte::wheelEvent(QWheelEvent *event)
{
```

```
    if ((carteDessiner)&&(flags>0)){
        if (event->delta()>0){
            emit ChangeZoomIn();
            update();
        }
        else   {
            emit ChangeZoom();
            update();
        }
    }
}

/****************** fonction pour l'algo de couleur*********************/

int carte::maximum(int a, int b){
    if(a>b) return a;
        else return b;
}

int carte::minimum(int a, int b){
    if(a<b) return a;
        else return b;
}

bool carte::comparerCouleurAvecMarge(QRgb p1, QRgb p2){

    unsigned int differenceRouge = abs(qRed(p1) - qRed(p2));
    unsigned int differenceVert  = abs(qGreen(p1)- qGreen(p2));
    unsigned int differenceBleu  = abs(qBlue(p1)-qBlue(p2));
    /*int couleurMaxP1= maximum(qRed(p1),maximum(qGreen(p1), qBlue(p1)));
    int couleurMaxP2= maximum(qRed(p2),maximum(qGreen(p2), qBlue(p2)));
    int couleurMin= minimum(qRed(p1),minimum(qGreen(p1), qBlue(p1)));*/
    if ((differenceRouge + differenceVert + differenceBleu)>100) return false;
        else return true;

    /*int couleurDominante= maximum(differenceBleu,maximum(differenceRouge,
differenceVert));
    if(couleurDominante>40) return false;   //40
                else if ((differenceRouge + differenceVert +
differenceBleu)>100) return false;  //100
    else return true;*/
              //if (somme<10) return true;
}

int carte::tracerZone(const QPoint &p,const QRgb &color){
    int nbPixel=0;
    for (int i=p.x()-etendueZone; i<p.x()+etendueZone;i++){
        for (int j=p.y()-etendueZone; j<p.y()+etendueZone;j++){
            if((i<=largeur)&&(i>=0)&&(j>=0)&&(j<=hauteur)){
            if ( comparerCouleurAvecMarge(color,imageAffichage-
>pixel(QPoint(i,j)))) {
                imageAffichage->setPixel(QPoint(i,j),255255255);
                tracerChemin->setPixel(QPoint(i,j),0xFF00FF00);
                tracerChemin2->setPixel(QPoint(i,j),0xFF00FF00);
                nbPixel++;
            }
            }
        }
    }
   return nbPixel;
}
```

```
/************************Brouillon
algo****************************************

   while(comparerCouleurAvecMarge( imageCarte->pixel(point_click),imageCarte-
>pixel(direction)==true)){
       imageCarte->setPixel(direction,255255255);
       std::cout<<"yes"<<std::endl;
      // QPoint direction1,direction2,direction3,direction4;
      QPoint direction1=QPoint(direction.x()+1,direction.y());
      QPoint direction2=QPoint(direction.x()-1,direction.y());
      QPoint direction3=QPoint(direction.x(),direction.y()+1);
      QPoint direction4=QPoint(direction.x(),direction.y()-1);
       if ((comparerCouleurAvecMarge( imageCarte->pixel(direction),imageCarte-
>pixel(direction1)==true))){
           point_click=direction;
           direction=direction1;
       }
       else
                   if ((comparerCouleurAvecMarge( imageCarte-
>pixel(direction),imageCarte->pixel(direction2)==true))){
                       point_click=direction;
                       direction=direction2;
       }
       else
                   if ((comparerCouleurAvecMarge( imageCarte-
>pixel(direction),imageCarte->pixel(direction3)==true))){
                       point_click=direction;
                       direction=direction3;
                   }
       else
                   if ((comparerCouleurAvecMarge( imageCarte-
>pixel(direction),imageCarte->pixel(direction4)==true))){
                       point_click=direction;
                       direction=direction4;
                   }

    }

      for (int i=0;i<largeur;i++){
        for (int j=0; j<hauteur;j++){
            if (comparerCouleurAvecMarge( imageCarte->pixel(i,j),coul)==true)

               {imageCarte->setPixel(QPoint(i,j),255255255);}
          // std::cout<<"couleur : "<<coul<<std::endl;}
            //imageCarte->setPixel(QPoint(i,j),coul);

        }
    }

    //tracer diagonale et parcourir rectangle
    if((directionX>=0)&&(directionY>=0))
    {
        for (int i=point_click.x();i<=direction.x();i++){
           for (int j=point_click.y();j<=direction.y();j++){
                if (comparerCouleurAvecMarge( imageCarte-
>pixel(i,j),coul)==true) tracerChemin->setPixel(QPoint(i,j),0xFF00FF00);
           }
        }
    }
    else
        if((directionX>=0)&&(directionY<0))
```

```
        {
            for (int i=point_click.x();i<=direction.x();i++){
               for (int j=point_click.y();j>=direction.y();j--){
                    if (comparerCouleurAvecMarge( imageCarte-
>pixel(i,j),coul)==true) tracerChemin->setPixel(QPoint(i,j),0xFF00FF00);
               }
            }
        }
        else
            if((directionX<0)&&(directionY>=0))
            {
                for (int i=point_click.x();i>=direction.x();i--){
                   for (int j=point_click.y();j<=direction.y();j++){
                        if (comparerCouleurAvecMarge( imageCarte-
>pixel(i,j),coul)==true) tracerChemin->setPixel(QPoint(i,j),0xFF00FF00);
                   }
                }
            }
            else
                {
                    for (int i=point_click.x();i>=direction.x();i--){
                       for (int j=point_click.y();j>=direction.y();j--){
                           if (comparerCouleurAvecMarge( imageCarte-
>pixel(i,j),coul)==true) tracerChemin->setPixel(QPoint(i,j),0xFF00FF00);
                       }
                    }
                }*/

   /*

*************************************************************************
*******/
```