Module `navisu-extensions`

0.1 La commande `TargetCmd`

Dans ce document on s'attache surtout à la description de l'utilisation de cette commande et non pas à son implémentation. Les `TargetCmd` permettent d'interroger NaVisu sur la position d'objets de type `NavigationData`, c'est à dire la plupart des objets d'une carte S57, par rapport à une position définie par l'utilisateur.
Construction d'une commande `TargetCmd` coté client :

```
Target target = new Target(new BeaconCardinal(), 48.3130, -4.6214));
Command navCmd = new Command("TargetCmd", target);
```

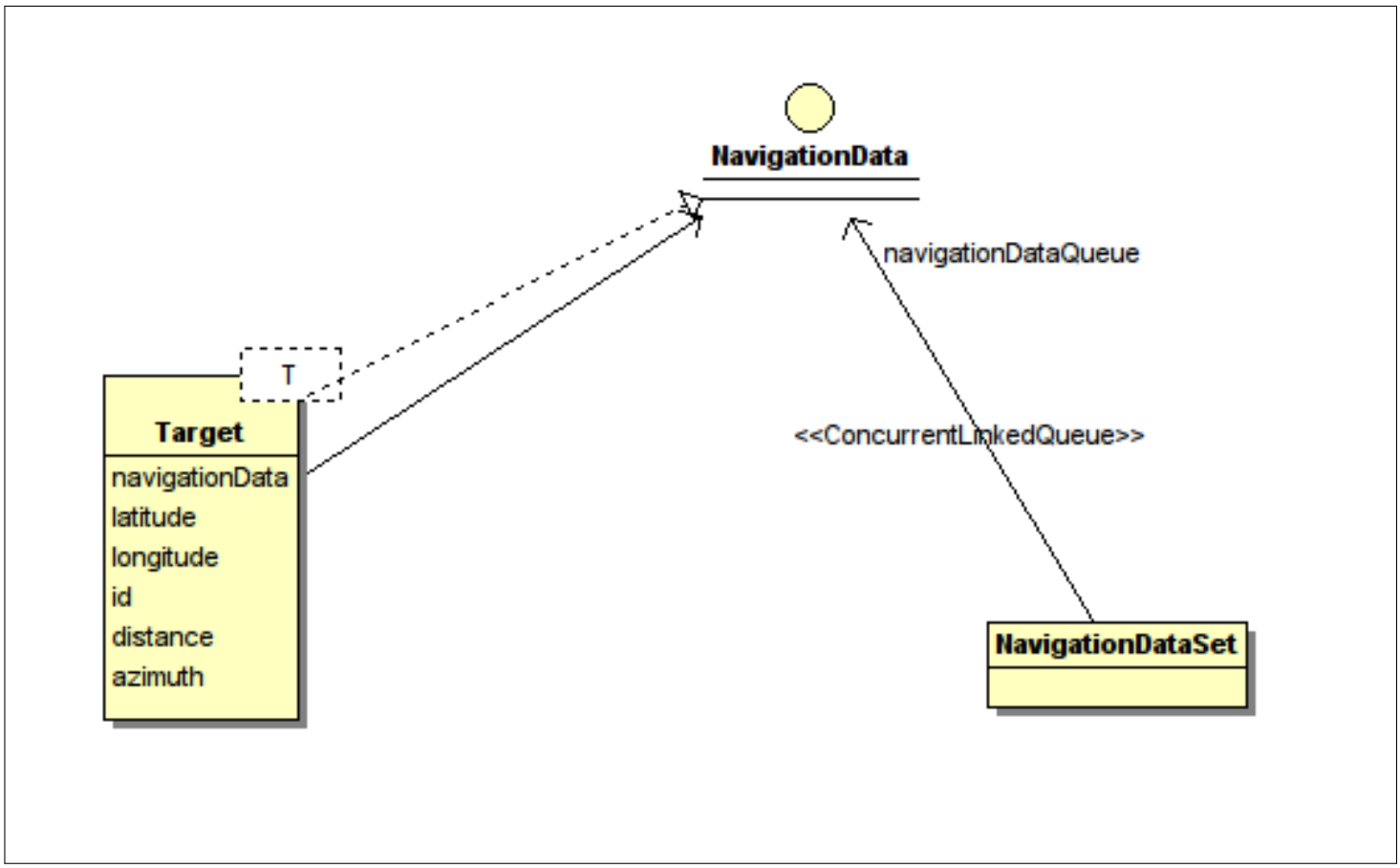

FIGURE 1 – *La classe* `Target`

Comme les autres commandes, le premier argument est le nom de la commande. Le second argument est du type `Target` qui implémente l'interface : `NavigationData`. Cet objet `Target` est aussi susceptible de posséder un objet de type `NavigationData`.
Après réception et traitement par NaVisu, un objet de type `NavigationDataSet` sera renvoyé.

0.1.1 Construction de l'objet `Target` pour la requête

— Lors de l'émission de la requête, cet argument sert à définir le type précis d'objets à cibler. On se contente alors d'utiliser un constructeur par défaut. On peut affiner la requête en précisant un type exact ou au contraire l'élargir en donnant une super classe. Par exemple `Buoyage` pour cibler tout type de balisage.

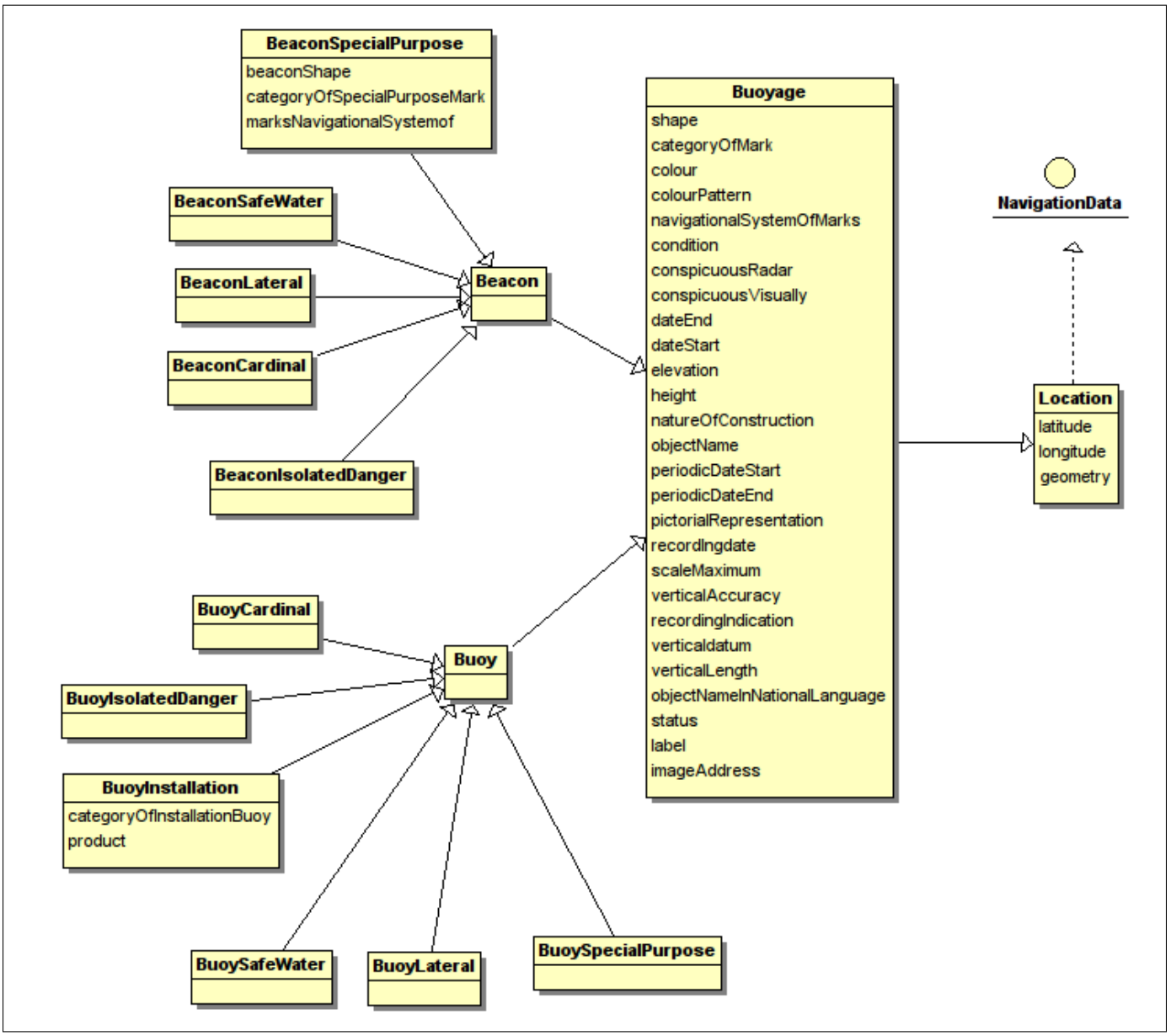

Figure 2 – *Les classes* `Buoyage`

— Les deux doubles suivants définissent la position de l'observateur. C'est par rapport à cette position que seront ciblés les objets dans NaVisu. Les arguments suivants sont optionnels.
— Un identifiant peut être ajouté, à l'usage du client.
— L'argument `distance` : s'il n'est pas présent : le premier objet répondant au type demandé sera renvoyé. S'il est présent le ou les objets, répondant au type demandé, se trouvant dans un cercle dont le rayon est défini par cette distance, seront renvoyés (l'unité est le mètre).
— Si l'argument `azimuth` est présent, les données dans un secteur `azimuth` $\pm 10°$ seront renvoyées, éventuellement filtrées par la distance.

0.1.2 Exemple de codage d'un client

Voir un exemple : https://github.com/terre-virtuelle/NaVisuClient
— Initialisation du client.
— Ouverture du canal "/navigation" sur le serveur
— Création des requêtes

```
public void init() {
        cmdVertx = VertxFactory.newVertx();
        cmdVertx.createHttpClient().setHost("localhost")
        .setPort(9090)
        .connectWebsocket("/navigation", (WebSocket websocket) -> {
        websocket.dataHandler((Buffer data) -> {
                if (data != null) {
                        response(data.toString());
                }
        });
        cmdWS = websocket;
        cmdRequest(new Target(new BeaconCardinal(), 48.3130, -4.6214));
        cmdRequest(new Target(new Buoyage(), 48.3130, -4.6214));
        cmdRequest(new Target(new BeaconSpecialPurpose(), 48.3130, -4.6214));
        });
}
```

— Instanciation de la commande
— Sérialisation de l'objet commande en xml
— Envoi au serveur.

```
public void cmdRequest(NavigationData target) {
        Writer stringWriter = new StringWriter();
        Command navCmd = new Command("TargetCmd", target);
        try {
                ImportExportXML.exports(navCmd, stringWriter);
        } catch (JAXBException ex) {
        Logger.getLogger(AppTarget.class.getName()).log(Level.SEVERE, null, ex);
        }
        cmdWS.writeTextFrame(stringWriter.toString());
}
```

Le résultat est en xml

— Après lecture du résultat.
— Traduction du résultat en objets.
— Récupération de la liste des NavigationData et coercition de type si nécessaire.
— Utilisation des données.

```
private void response(String resp) {

        NavigationDataSet navigationDataSet = new NavigationDataSet();
        try {
                navigationDataSet = ImportExportXML.imports(navigationDataSet,
                 new StringReader(resp));
        } catch (JAXBException ex) {
        Logger.getLogger(App.class.getName()).log(Level.SEVERE, ex.toString(), ex);
        }

        List<NavigationData> result = navigationDataSet.getNavigationDataList();
        for (NavigationData n : result) {
                Target t = (Target) n;
                System.out.print("buoy : "
                + ((Buoyage) t.getNavigationData()).getObjectName());
                System.out.print(" type : "
                + t.getNavigationData().getClass().getSimpleName());
                System.out.printf(" distance : %.2f m", t.getDistance());
                System.out.printf(" azimuth : %.2f °\n", t.getAzimuth());
        }
}
```

0.1.3 Visualisation sur NaVisu

En plus de renvoyer la liste des objets ciblés, NaVisu visualise les requêtes et les éléments trouvés sur la carte. Résultat des trois requêtes précédentes :

```
cmdRequest(new Target(new BeaconCardinal(), 48.3130, -4.6214));
cmdRequest(new Target(new Buoyage(), 48.3130, -4.6214));
cmdRequest(new Target(new BeaconSpecialPurpose(), 48.3130, -4.6214));
```

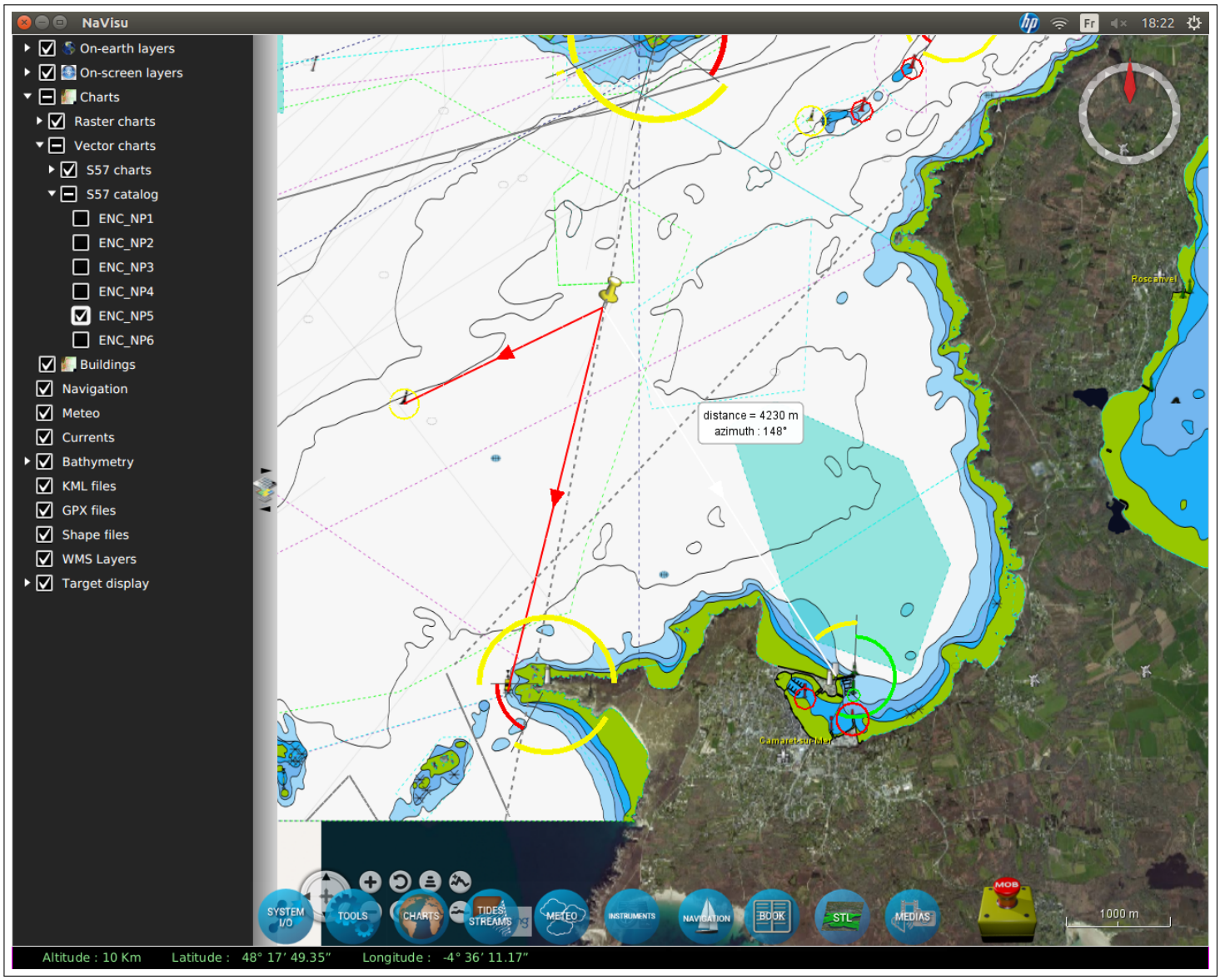

Figure 3 – *Ciblage des bouées dans NaVisu, plusieurs critères de choix d'objets, sans paramètres de distance*

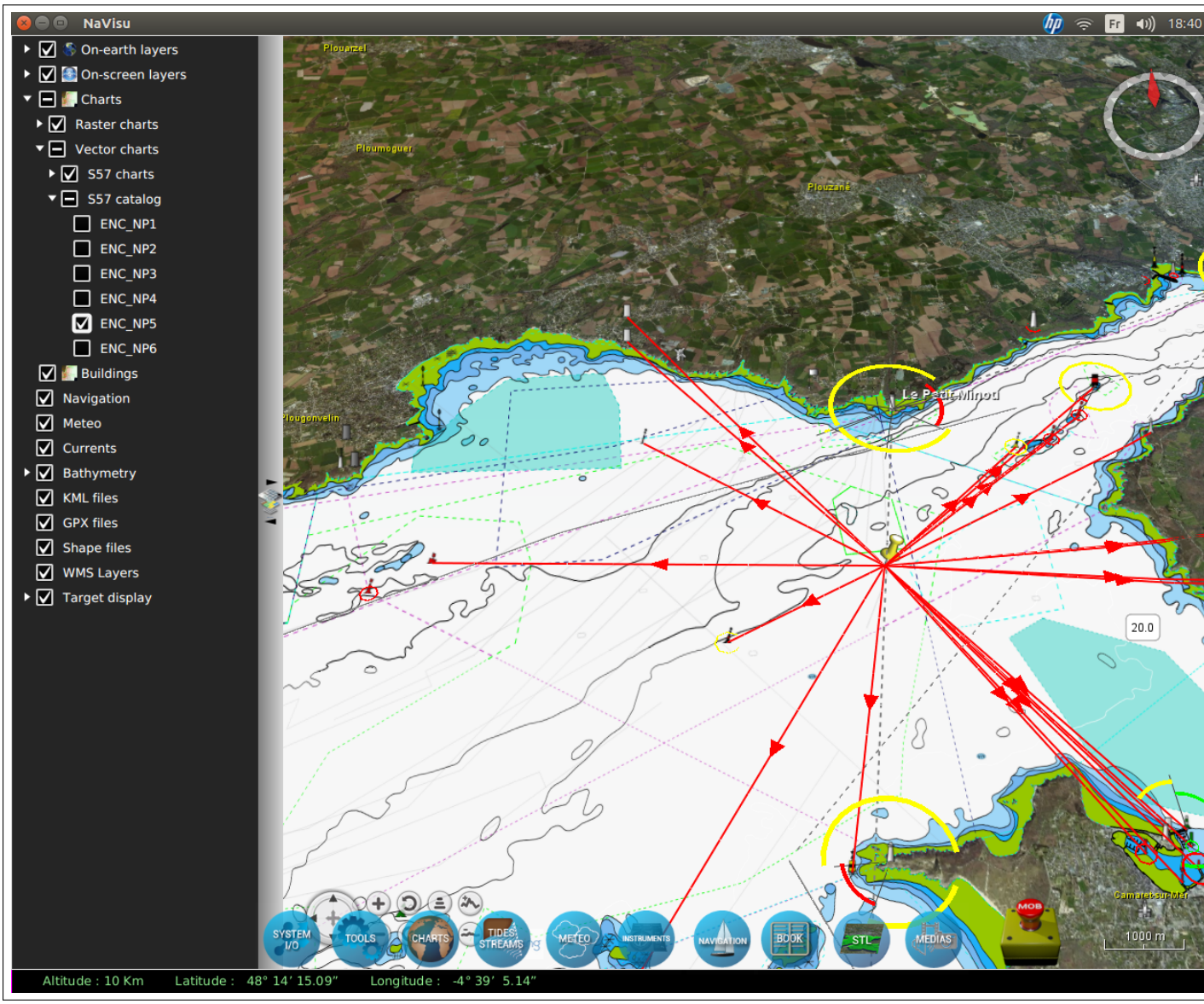

Figure 4 – *Ciblage des bouées dans NaVisu avec une distance de 6000 m*