行军线逻辑

MarchInfo是行军线数据基类，解析行军线数据的方法是MarchInfo里的updateInfo。行军线协议大部分都是push类型，下面说下最常用的几个，

1. "push.world.march"：每当在你视角的一定范围内（范围由服务器定）、玩家自己和玩家同联盟的人有行军线数据发生了变化（除了因战斗胜利、失败或行军取消撤回而产生的变化外）时，都会由此协议推送下最新的行军数据。
2. "push.world.march.win"和"push.monster.siege.def.fail"：这两个协议的意思基本一样，都是代表行军到达目的地后战斗胜利变化。"push.world.march.win"代表玩家的行军线战斗胜利，"push.monster.siege.def.fail"只代表怪物攻城活动里怪物攻打玩家城市获得胜利。要注意的一点是，这两个协议不会下发行军线的所有数据，只会下发战斗胜利后行军线的更改数据。
3. "push.world.march.fail"和"push.monster.siege.def.win"：这两个协议的意思基本一样，都是代表行军到达目的地后战斗失败变化。"push.world.march.fail"代表玩家的行军战斗失败，"push.monster.siege.def.win"则代表怪物攻城活动里怪物攻打玩家城市失败。与上面胜利协议不同的是，这两个协议会把行军线的所有数据重新下发下来。
4. "push.world.march.spd"：行军线加速协议
5. "push.world.march.retreat"：行军线取消撤回协议，服务器在某些情况下需要撤回行军也会走这个协议，比如怪物攻城时，如果玩家已经连续两轮防守失败了，但仍旧还有新的怪物行军线在往玩家城市行军时，服务器就会发送该协议，取消行军。
6. "push.world.march.world.get"：每次从主城进入大地图的时候，服务器都会通过此协议推送玩家自己和玩家同联盟的人的所有行军线，以及一定范围内的其他玩家的行军线。
7. "world.user.march"：刷新玩家自己所有行军线数据的协议。该协议的详细意义可以看下面的TimeUpMarchProcesser类的解释

行军线的状态变化和展示刷新：

WorldController. updateMarchInfos() 方法是每帧刷新的，客户端通过该方法来刷新行军线的一切逻辑。

这里首先会调用WorldMapView2. updateMarchTarget() 方法，该方法里会根据行军状态并且通过processMarchAction() 方法来刷新战斗动画播放逻辑和刷新行军目标点的一些显示逻辑。通过clearMarchNode() 方法来清除无效的行军线，通过drawMarchNode() 方法来创建行军线和刷新行军队列坐标逻辑。

然后调用WorldController.checkOverTimeMarch() 方法来判断行军线是否已到达了目的地，如果已到达，则对行军线的一些数据进行修改或者删除行军线数据。

如果行军线数据被删除了，则通过WorldMapView2. clearMarchNode() 方法删除行军线展示。

TimeUpMarchProcesser：

这个类是为了刷新玩家自身行军线数据服务的，主要逻辑是：

1. 当玩家自身的行军线数据因为已经回到了自己的城市或者因为其他一些情况被删除时，被删除的行军线id会缓存到这个类里。于此同时会发送"common.res.syn"协议刷新出征界面的军团状态和其他的一些数据。如果在Global.MARCH\_TIMEUP\_DELAY\_REQUEST\_TIME秒内，没有收到"common.res.syn"协议的回复或者客户端也未收到同id的新的行军线数据，TimeUpMarchProcesser则会发送"world.user.march"协议同步玩家自己的所有行军线数据，确保玩家自己的行军线数据是正确的。
2. 后续新增加了个逻辑，每隔marchUpdateTime秒，都会重新发送此"world.user.march"协议，确保即使玩家自己的行军线数据因为某些情况下出错了也能在一段时间后复原。控制代码在refresh() 方法里。

行军线需要注意的两个坑点：

1. 服务器在推送"push.army.return"军团协议有个前提，必须是客户端至少发送"common.res.syn"和"world.user.march"这两个协议之一。正常情况下，玩家自己的行军线因回到自己的城市被删除后，会发送"common.res.syn"协议，如果没有发送或者没有收到"common.res.syn"的回复的话，再5秒后还会发送"world.user.march"协议。但如果连"world.user.march"这个协议这个协议都发送失败的话，服务器便不会主动推送"push.army.return"军团协议，然后就会导致出征界面的军团状态展示出错，并且该军团无法再次出征。
2. 行军线集结数据有异常。

目前情况下单人的行军数据没有发现异常，只需要按照行军数据展示就可以了，但如果是集结行军，行军数据里会有异常，不能完全按照行军数据展示。

MarchInfo. marchType代表行军类型，当类型为MethodRally时代表发起者的行军数据。当类型为MethodUnion代表参与者行军数据。当类型为这两种其中之一时，MarchInfo. teamUid就会被赋上发起者行军数据的uuid。

行军队列变为集结类型的方式：

1. 发起集结，玩家发起集结后就会创建一条类型为MethodRally的集结类型的行军线
2. 参与集结，玩家派军团参加集结，当玩家到达集结点后，玩家参与集结的行军线会变为MethodUnion类型，而在这之前，参与集结的行军线还不属于集结类型。
3. 这点也时容易引起误解的一点，当王座、投石机、城战城市和血岩矿里，同时驻扎了多支部队，那么服务器就会将这里所有的部队都转变成集结类型，即使里面的部队都是分批单人行军进去的。

还有要注意的是，如果多只队伍是合并在一起行军或者都驻扎在某个点，并且是集结类型，此时如果解散该行军，让这多只队伍都返回玩家自己的城市时，他们仍旧是集结类型，并不会因为解散了就会变为单人类型。即解散前是MethodRally类型解散后仍旧是MethodRally类型，解散前是MethodUnion类型解散后仍旧是MethodUnion类型。

因为上面的两点，导致客户端无法通过MarchInfo.marchType来判断行军线是不是集结类型。如果想要判断是不是集结类型，需要对MarchInfo里多个参数同时进行判断，比如行军目的地类型等等。

还有，参与行军类型MethodUnion还存在一个问题。当玩家a派出部队去参与玩家b发起的集结时，在部队到达目的地并且集结时间结束开始出发时，服务器会推送刷新玩家b发起集结行军数据，让其正常的从集结点往目的地行军。但服务器推送刷新玩家a参与者行军数据却是 一直驻扎在集结点不动，因此在玩家a的视角里，他参加集结的部队是一直驻扎在集结点不动的，因此客户端需要修改展现，不能展示玩家a参与的行军数据，而是应该展示玩家b发起的行军数据。如果行军的集结目标是可驻扎的目标点的话（比如王座，投石机等），在到达目的地后，玩家a还需要发送"world.user.march"协议，让服务器重新推送玩家a的参加集结的行军数据，将玩家a的部队驻扎在目标点。

上面所说的逻辑均在代码里写好，玩家a的视角里需要展示玩家b发起者的行军数据的代码在QueueController.addMarchQueueInfo() 方法里的else if (info.marchType == MarchMethodType.MethodUnion || info.marchType == MarchMethodType.MethodRally)

这行判断里，（这里没有只根据MethodUnion和MethodRally来判断是不是集结类型）。后面需要发送"world.user.march"协议的代码写在了WorldMarchWinPush. handleResponse() 里，通过调用TimeUpMarchProcesser.getInstance().sendRequest() 发送

后续集结行军有新增类似的需求可以参考这两个地方