帧同步的原理

1: 服务器接收 每个客户端的输入，然后每帧广播给其它的客户端;
2:所有的客户端拿到是服务器同样的输入, 然后自己独立运算;
3:输入一样，代码一样，就能得到一样的结果;
4:服务器比较简单，计算都放到了客户端上;
5: 传输的数据小, 游戏手感不错;
6: 缺点是容易作弊;
7: 玩家的操作的反应时间一般是50~100ms;

帧同步流程

1: 服务器同步的帧，我们叫做逻辑帧，每秒维持 15~20左右;
2:客户端收到逻辑帧后，先处理这帧所有用户的输入;
3: 把下一帧的当前用户操作发送给服务器;
4: 服务器会对每帧的用户的操作进行缓存，保存起来;
5: 当客户断线以后, 重新连接上后，服务器把断线没有处理的帧到最新的帧，发送给客户端, 客户端自己计算，同步到当前最新状态;
6: 当网络延时，用户的操作没有发送到服务器，会选择忽略;

(1)同步_lastFrameOpts的逻辑操作，调整位置到真实的逻辑位置:
调整完以后，客户端同步到syncFrameID

(2)从syncFrameID+1开始-->frame.frameid-1
同步丢失的帧，快速同步到当前帧
所有客户端的数据都被同步到frame.frameid-1
同步这些帧造成的由帧驱动的公共物体的位置(如小兵位置)

(3)获取最后一个操作 frame.frameid的操作，同时syncFrameID=frame,frameid
更新_lastFrameOpts为该帧
根据这个帧来处理，更新动画状态以及位移，产生的位移为“假位移”

(4)采集下一帧要发送给服务器的操作

1.之所以第一步要使用_lastFrameOpts的逻辑操作，而不是直接同步收到的最后一帧的操作，是因为
如果客户端从100帧掉线到199帧，那么第200帧应该同步的是断线之前的最后一次操作，而不是第199帧的操作。因为断线前的操作已经被服务器接收并转发给其他客户端，并已经同步，而此时本地客户端还未同步，所以要先同步这一次的操作。

2.之所以不在第x帧的第四步时直接将_lastFrameOpts更新，是因为需要在x+1帧时先使用这一次的操作更新动画以及位移(这时的位移相当于是一次预测，预测这次操作同步后的位移)，之后在x+2帧的第一步时，在同步到真正的逻辑位置，此时的位移抖动较小。

如果直接在第x帧的第四步时更新_lastFrameOpts，在x+1帧时，玩家会直接无动画的瞬移到逻辑同步的位置。之后在第x+1帧的第四部时，会从这次逻辑帧更新之后位置，在根据第x帧的操作播放动画与位移，会造成很大的抖动。

红字为逻辑位置，绿字为本地客户端的执行动画位移后的实际位置

第x帧的第四步时直接将_lastFrameOpts更新：

x帧(右移操作,0.0m)-->x+1帧(同步逻辑位置,0.0->1.0m)-->x+1帧(播放动画，并继续右移1.0m->1.5m)-->x+1帧(左移操作)-->x+2帧(同步逻辑位置,1.5m->0m) 这会造成很大抖动，影响游戏体验

第x帧的第三步时将_lastFrameOpts更新：

x帧(右移操作,0.0m)-->x+1帧(同步逻辑位置,0.0->0.0m)-->x+1帧(播放动画，并继续右移0.0m->0.5m)-->x+1帧(左移操作)-->x+2帧(同步逻辑位置,0.5m->1.0m)-->x+2帧(播放动画,并左移1.0m->0.5m)-->x+2帧(无操作)-->x+3帧(同步逻辑位置,0.5m->0.0m) 在没有使用其他防抖动措施的情况下，整体抖动较小