游戏设计
登录模块 消息模块 DB模块 匹配模块 sight move&path skill AI 属性 ssbattle
ThreadActor
AI结合

问题
INetSessionMgr::GetInstance()->SendMsgToSession(ST_CLIENT_C2Log, 0, sMsg, n32MsgID);
sid=0 logserver

INetSessionMgr::GetInstance()->TranMsgToSession(ST_SERVER_CS_OnlyGS,-1,
          sData.c_str(),sData.size(),n32MsgID,0,0);
sid=-1 广播

session = type + id + buf
buf = len + msgid + buf

保存指针，就不需要同步更新所有保存该对象的值。
priority_queue
concurrent_queue
tinyxml.h
boost::asio
c语言libcurl库的异步用法

CFunction::SplitInt(model.c_str(), teamcount, 10); 3v3
CCSCfgMgr::getInstance().GetHeroBuyCfg(un32HeroGoodsID);
#define RedisSCallBack(realCallBack, holder) std::bind(realCallBack, holder,
std::placeholders::_1, std::placeholders::_2, std::placeholders::_3)
std::function可以取代函数指针的作用，因为它可以延迟函数的执行，特别适合作为回调函数使用。
它比普通函数指针更加的灵活和便利。
timeBeginPeriod(1); 设置cpu定时器最小分辨率ms
split_vector_type ssIndexVec;
boost::algorithm::split(ssIndexVec, ssIndexStr, boost::algorithm::is_any_of(";"));
ref_ptr<CSSSkill>   m_pSkillRuntimeData;//技能的指针

          连接监听
connector => listener
gs => bs
gc => bs
bs => ls
gc => gs
gs => cs

cs => ls
cs => redis
cs => redis logic
gs => cs

ss => cs
rc => cs

ss => logs
gs => ss

<SSPort>10001</SSPort>
<GSPort>10002</GSPort>
<RSPort>10010</RSPort>

登录模块
lsserver
SDKAsyncHandler
// 收到第1消息： gc =>ls 请求登录，ls放入登录队列
gc2ls:clientSession:Msg_Handle_Init => CheckLogin 插入m_SDKCallbackQueue、m_UserLoginDataMap
newtimer_cb 定时取出m_SDKCallbackQueue => new_conn curl访问第三方 => write_cb 接收curl返回数据 =>
    AsynHandleLoiginCheckMsg 删除m_UserLoginDataMap => AddUserToLoginMap => SDKConnector:SendToInsertData

SDKConnector
SendToInsertData 插入m_DBCallbackQueue
Update 取出m_DBCallbackQueue 插入m_AllLoginUserInfo => PostMsgToGC_NotifyServerList

CBalanceSession
//第2条消息 bs => ls
bs2ls:Msg_Handle_OneClientLoginCheck 删除m_AllLoginUserInfo 返回结果给bs

bsserver
// 收到第2消息：gc => bs bs => ls 客户端连接bs，bs向ls请求用户是否合法连接
CClientSession::MSG_OneClientLogin

// 发送第3消息：ls => bs bs返回login结果给gc bs => gs 用户验证是否成功，如果验证成功，请求分配gs给用户
CB2LSession::Msg_User_Login

// 发送第4消息： gs => bs gs允许该用户连接，告知gs地址和连接密码
CGateSession::Msg_Handle_OneUserLoginTokenRet

消息模块
ISDListener 接收
CreateListener => SetSessionFactory 设置ISrvCliSession类，对应Msg_Handle_xxx就是接收处理函数

ISDConnector 接收
CreateConnector => 设置ICliSession类，对应Msg_Handle_xxx就是接收处理函数

ISDSession 发送
SetConnection 建立连接后自动设置ISDConnection
SendMsgToSession 根据sid发送 => INetSession.send => ISDConnection.send

gs相关
NET/xxxxSession => gsKernel:.HandleMsgFromxxxx

cs相关
CCSUser::PostMsgToGC => CCSKernel::PostMsgToGS

ss
SSBattleNetSessionMgr

接收消息
CCentralSession::Msg_Handle_Dispath =>CSSWorkThreadMgr::SendCSMsgToLogicThread =>
    BattleLogicThreadActor::EncodeCSMsgAndSendToLogicThread =>
    CSSBattleMgr::OnMsgFromCS/OnMsgFromGS_ReportGCMsg

DB模块

异步操作
设置callback: new DBActiveWrapper callback => Action:callback
DBActiveWrapper::EncodeAndSendToDBThread => Active::Send 发送到queue
Active::Run 定时执行 => Active::Consume 取出queue 执行callback
查询

DBAsynQueryUserCallBack => DBAsyn_QueryUser =>
CCSUserMgr::DBAsyn_QueryUser(SUserDBData & sUserData 参数, DBToCS::QueryUser&
sQueryUser 结果, IDBConnector* pConn){

CCSCfgMgr 获取map配置
CCSUserMgr
CCSUser
    CTaskMgr user friend item bag rune hero battle cdkey mail

CCSBattleMgr相关类
CCSBattleMgr
    m_cAllBattleMap[battleId]=CCSBattle
    m_BattleRoomList[roomId]=CCSBattleRoom

CCSMatchMgr
    mAllMatchList[matchType][mapId]=ICSMatchList
    mAllTeamMap[teamId]=CCSMatchTeam

CCSMatchTeam (CCSMatchMgr::UserCreateTeam)
    mTeamId
    mVPlayer[i]=IMatchPlayer

CCSMatchList_Nomal (CCSMatchMgr::TeamStartMatch)
    mRoomList[roomid]=CCSMatchRoom_Nomal Nomal根据map人数限制划分CCSMatchTeam到不同

CCSMatchRoom_Nomal中

CCSMatchRoom_Nomal (CCSMatchList_Nomal::AddOneTeam)
    mTeamMap[i]=CCSMatchTeam

CCSBattleRoom (CCSBattleMgr::AskCreateRoom)
    m_pPlayerList[i]=IRoomPlayer

CCSBattle (CCSBattleMgr::OnBattleRoomStart)

开始匹配
CCSBattleMgr => CCSMatchMgr => CCSMatchTeam => match => CCSMatchList_Nomal => CCSMatchRoom_Nomal

匹配成功
CCSBattleMgr::Initialize => CCSMatchMgr::Update => CCSMatchRoom_Nomal::Update => CCSBattleMgr::OnBattleMached =>
  CCSBattleRoom => CCSBattle
---
sight

== sight entity相关
ISSightLight
IsInSight 视野判断
GetSightX CSSGameUnit 中实现
GetSightY CSSGameUnit 中实现
GetLightDistance CSSGameUnit 中实现

CSSightLight unuse &&

ISSightObject mOwner(CSSight)
SetOwner mOwner 设置视野归属
OnAppearInSight 把obj推送给watchers go 中实现
OnDisappearInSight 把obj推送给watchers go 中实现
GetSightPosX go 中实现
GetSightPosY CSSGameUnit 中实现
GetSightObjectType CSSGameUnit 中实现
IfInvisible CSSGameUnit 中实现

CSSGameUnit CSSGameObject、ISSightLight、ISSightObject的派生
m_sCurOASI(CSSGameUnit) m_sFpMgr(CSSGUParameterMgr)
== ISSightLight 相关
GetSightX 获取m_sCurOASI.cPos
GetSightY 获取m_sCurOASI.cPos
GetLightDistance 固定值 3000
==ISSightObject 相关
GetSightPosX 获取m_sCurOASI.cPos 和GetSightX功能相同 ??
GetSightPosY 获取m_sCurOASI.cPos
GetSightObjectType 设置obj为视野敏感对象,hero为true，技能为false

IfInvisible 隐身
    m_sFpMgr.GetValue(eEffectCate)

CSSGameObject   m_sCurOASI.cPos cDir   调用移动模块
SetGOActionState_Pos   设置go坐标
SetGOActionState_Dir   设置go朝向
== sight entity相关

CSSight 我方视野 静态管理&& m_sightLishtMap(ISSightLight) 视野判断
m_watcherVec(CSSUser) 视野消息被推送者   m_sightType 普通/OB
== m_sightLishtMap 相关
IsInSight 判断obj在m_sightLishtMap视野内 CSSightMgr中使用
    ISSightLight::IsInSight
AddSightLight   进入战场时
    第一步, 设置obj归属 ISSightObject::SetOwner
    第二步, 添加ISSightLight
RemoveSightLight
== m_watcherVec 相关
AddWatcher
RemoveWatcher

结构体
SSightObjectInfo
sightArray(CSSight) 所有可观察到obj的CSSight 动态管理&& childSet(ISSightObject) 召唤物 技能

CSSightMgr   m_SightVec(CSSight) m_SightObjectMap(SSightObjectInfo) 全图obj
== CSSight 相关
CreateSight   根据camp创建 CSSight
== ISSightObject 相关
AddSightObject   添加视野obj到全图
    第一步, 添加 m_SightObjectMap
    第二步, ISSightObject::OnAppearInSight
RemoveSightObject   从全图移除视野obj
    ISSightObject::OnDisappearInSight
CheckObjectInSight   检查所有的视野对象(非敏感对象+500ms周期)。当其出现或者消失在视野中的时候，发送对应的消息
    第一种, 出现
        添加 m_SightObjectMap
        ISSightObject::OnAppearInSight
    第二种, 消失
        移除 m_SightObjectMap
        ISSightObject::OnDisappearInSight
AddChildSightObject   添加子视野obj

```
    SSightObjectInfo::childSet.insert
RemoveChildSightObject
== watcher 相关
AddWatcherAndCheck    添加watcher到全图
    CSSight::AddWatcher
    ISSightObject::OnAppearInSight
OnHeartBeat
    CheckObjectInSight


CSSBattle::RemoveISSUserToSight
    CSSight::RemoveWatcher
----------------------------------------
== move & path
疑问
1 不是取寻路路径 而是取  m_stepMoveTarget  =>
CSSMoveMgr::SetNextMovePoint  从m_asPathCell中获取下一点 并设置 m_dir
ISSMoveObject::CalculateStepMoveTarget GetDir() * speed * timeDiff => m_stepMoveTarget
TryMove  调用 ISSMoveObject::Move  m_stepMoveTarget 设置为 m_sCurOASI.cPos
(AskStartMoveToTar  OnHeartBeat)


== move entity相关
ISSMoveObjectHolder
OnStopMove  CSSSkillEffect_Move 中实现
CSSSkillEffect_Move::OnStopMove 停止通知


ISSMoveObject
成员
          SSMoveObjectImpactType          m_savedImpactType; //缓存的碰撞类型
          SSMoveObjectStatus   m_moveStatus;//移动状态(停止，移动中)
          SSMoveObjectMoveType m_moveType;//移动类型（方向，寻路，强制）
          UINT8                                    m_grpID;//组ID
          TIME_MILSEC                              m_startMoveTime;//开始移动时间
          TIME_MILSEC                              m_lastFailStartTime;//上次停止移
动时间  ??
          FLOAT                                    m_oldSpeed;//缓存的速度
          bool                                     m_bIfSpeedChanged;//是否发生过速
度改变
          FLOAT                                    m_forceMoveSpeed;//强制位移速度
          ColVector                          m_askMoveDir;//请求移动的方向
          TIME_MILSEC                              m_redirectTime;//上次尝试自动转向
时间
          ColVector                          m_dir;//实际移动方向
          ColSphere                          m_stepMoveTarget;//自动寻路当前移动目标
          ColVector                          m_beforeMovePos;//上一次位移的坐标
          SIntPoint                          m_asPathCell[c_n32MaxOrderPathNode];//自动寻
```

路的路径点 [0]是targetPos
	UINT16 m_pathLength;//自动寻路路径长度
	UINT16 m_moveStep;//自动寻路当前坐标序列
号 m_asPathCell 的索引
	bool m_ifStepMoved;//no use
	bool m_ifForceMoveImpact;//强制位移是
否需要检测碰撞
方法
== setter 方法
SetBeforeMovePos m_beforeMovePos
SetMoveStatus m_moveStatus
SetAskMoveDir m_askMoveDir
SetDir m_dir
SetStartMoveTime m_startMoveTime
SetMoveStep 设置 m_moveStep
AddMoveStep m_moveStep++
GetNowMoveIntTar m_asPathCell[m_moveStep]
GetPathCell 获取 m_asPathCell进行赋值
SetStepMoveTarget 设置 m_stepMoveTarget
== tool 方法
CalculateStepMoveTarget 根据速度*当前时间差 => m_stepMoveTarget 不考虑碰撞
    SetStepMoveTarget
Move 上次位置 => 开始时间 => 移动
    第一步, 保存当前位置
        SetBeforeMovePos
        SetStartMoveTime
    第二步, 设置当前位置 m_stepMoveTarget => 设置为当前位置
         CSSGameUnit::OnMoved
    第三步, 移动动作
        CSSGameUnit::OnStartMove
Stop
    SetMoveStatus m_moveStatus
    SetLastFailStartTime m_lastFailStartTime
    SetStepMoveTarget CSSGameObject::m_sCurOASI.cPos => 设置 m_stepMoveTarget
    SetBeforeMovePos CSSGameObject::m_sCurOASI.cPos => 设置 m_beforeMovePos

CSSGameObject m_sCurOASI(cPos cDir) 当前位置 朝向 && m_pAI
SetGOActionState_Pos 设置当前位置
SetGOActionState_Dir 设置当前朝向

CSSGameUnit CSSGameObject ISSMoveObject的派生
BeginAction_Move 设置当前朝向 + 消息推送
    ISSMoveObject.m_dir => CSSGameObject::SetGOActionState_Dir
OnMoved
    CSSGameObject::SetGOActionState_Pos
OnChangeDir
    CSSGameObject::SetGOActionState_Dir

OnStartMove　移动动作
    判断没受控制
    BeginAction_Move
OnMoveBlock
    m_pAI::OnMoveBlock　调用AI模块
== move entity相关

CSSAStar
init　长宽、静态碰撞 => 初始化寻路数据
FindPath　起点、终点、动态碰撞数组 => 路径

CSSMoveMgr

CSSAStar
        m_Astar;//A星寻路类　init　FindPath
    hash_set<ISSMoveObject*>　　m_mObjectMap[6];//所有碰撞对象,按照组ID归类 AddMoveObject
    hash_set<ISSMoveObject*>　　m_allMObjectSet;//所有碰撞对象　AddMoveObject
    ISSMoveObject*
        m_heartBeatTempArray[1024];//心跳时用临时数组　OnHeartBeat

bool*
        m_staticBlockArray;//静态碰撞点信息
    map<UINT64,SIntTri>
        m_staticTriBlockMap;//三角形静态碰撞map　??
    hash_map<UINT64,SIntPoint*>　　m_MapCacheMap;//没有动态阻挡，把本次寻路缓存
    UINT16
        m_regionWidth;//地图宽度，单位格子
    UINT16
        m_regionHeight;//地图高度，单位格子
    UINT16
        m_regionSize;//格子尺寸，正方形
== tool 方法
GetPathByCache
SavePathToCache　设置 m_MapCacheMap
    GetPathID　startRegionId + targetRegionId => 唯一pathId
        GetRegionID_INT　pos => regionId
TestNextStepMove　碰撞检查
    IfStaticImpact
    IfDynamicImpact
SetNextMovePoint　将移动目标设置为下一个点　设置当前朝向　使用寻路缓存的主要方法 &&
    ISSMoveObject::GetNowMoveIntTar
    ISSMoveObject::SetDir
SetMapInfo　初始化 m_Astar

```
     m_Astar.Init
TryMove    实际移动方法  &&
    IfStaticImpact   仅请求朝方向移动需要静态碰撞检测
    IfDynamicImpact  => CSSMoveTool::IfImpact 动态碰撞检测
    ISSMoveObject::Move(now)   移动
== handler 方法
AskStartMoveForced 开始强制位移(可以在移动中调用)
AskStartMoveDir   请求朝方向移动   askdir =>
    第一步，设置移动朝向  ISSMoveObject::SetAskMoveDir
    第二步，计算下100ms位置  ISSMoveObject::CalculateStepMoveTarget
    第三步, CSSGameUnit::OnStartMove
AskStartMoveToTar  请求移动到指定坐标  targetPos =>
    第一步, 设置寻路缓存
        FindDynamicBlock  获取动态碰撞数据
        第一种, 如果有动态阻挡，或者找不到缓存
            m_Astar.FindPath(aStartInfo)  起点、终点、动态碰撞数据 => psPathBuff + 设置 ISSMoveObject.m_asPathCell
            SavePathToCache  如果没有动态阻挡插入缓存 m_MapCacheMap
        第二种, 相反情况下
            GetPathByCache  设置 ISSMoveObject.m_asPathCell
            ISSMoveObject::SetPathLength(i + 1)  设置寻路到边界点 !!
    第二步, 设置第一步为目标
        ISSMoveObject::SetMoveStep(1)
        SetNextMovePoint
    第三步, 如果下一个100毫秒后会阻挡，停止移动  ??
        TestNextStepMove  碰撞检查
            ISSMoveObject::Stop
    第四步, 设置朝向
        CSSGameUnit::OnStartMove
AskStopMoveObject
    StopLastStep
        TryMove 根据当前时间和最后一次移动的时间差，做最后一小步的移动
        ISSMoveObject::Stop(now,bIfCallBack)
OnHeartBeat
    第一步, 重置所有目标为未移动
        ISSMoveObject::CalculateStepMoveTarget(now)
        移动中的ISSMoveObject 放入 m_heartBeatTempArray
    第二步, 反复尝试移动所有对象，直到没有任何一个对象可以移动为止 至多被移动一次
        TryMove
        CheckTargetMoveStatus  自动寻路下：检查是否需要更换移动点，或已移动到目标  反复尝试条件 !!
            ISSMoveObject::AddMoveStep()
            SetNextMovePoint
    第三步, 剩下的目标都是不能移动的，将其停止
        ISSMoveObject::Stop(now)
    第四步, 重新检查所有目标，看是否可以在下一次心跳移动
        ISSMoveObject::CalculateStepMoveTarget(now + 100)
        TestNextStepMove
            ISSMoveObject::Stop
```

CSSBattle::LoadMapData 初始化地图
    第一步，CSSCfgMgr::LoadMapStaticData 加载/CSBattleMgr/Map/.map文件
    第二步，初始化静态碰撞数据
    CSSMoveMgr::AddStaticBlockInfo m_staticBlockArray 用于CSSAStar::init初始化寻路数据
    CSSMoveMgr::AddStaticTriBlockInfo m_staticTriBlockMap

CSSBattle::DoPlayHeartBeat 定时间隔100，执行所有实体timeDiff的一次移动
    CSSMoveMgr::OnHeartBeat
---
skill

== skill entity 相关
CSSGameObject   m_pNormalAttackSkill 普攻   m_sCurOASI(un32SkillID sSkillTarGUID eOAS)
SetGOActionState   m_sCurOASI.eOAS 玩家状态
SetGOActionState_SkillID   m_sCurOASI.un32SkillID
SetGOActionState_SkillTarGUID   m_sCurOASI.sSkillTarGUID

CSSGameUnit   m_mPassiveSkillMap[eType] = u32PSkillID
== action 相关
BeginAction_PrepareSkill
    CSSGameObject::SetGOActionState
    CSSGameObject::SetGOActionState_SkillID
    CSSGameObject::SetGOActionState_SkillTarGUID
BeginAction_ReleaseSkill
BeginAction_UsingSkill
BeginAction_LastingSkill
== passiveSkill 相关
AddPassiveSkill   m_mPassiveSkillMap
RemovePassiveSkill
GetPassiveSkillVec   eType => m_mPassiveSkillMap
OnPassitiveSkillCalled   触发被动技能 &&
    CSSPassiveSkillMgr::Trigger
OnPassitiveSkillHeartBeat
    OnPassitiveSkillCalled
OnGameUnitHeartBeat
    OnPassitiveSkillHeartBeat

CSSHero   CSSGameUnit 的派生 CSSGameObject 的派生
LoadHeroCfg 设置 m_pNormalAttackSkill
OnHeartBeat
    CSSGameUnit::OnGameUnitHeartBeat
== skill entity 相关

== skill config 相关
CSSCfgMgr
map<EObjectType, SGoodsCfg>
m_cGoodsCfgMap; &&
map<UINT32, SSNewSkillCfg>
m_cSkillCfgMap; &&
map<UINT32, ESkillEffectType> m_cSkillModelTypeMap;
技能模块类型
map<UINT32, SSkillModelAccountCfg>
m_cSkillModelAccountCfgMap; 结算类技能模块配置(具体技能配置 以下均是)
map<UINT32, SSkillModelLeadingCfg>
m_cSkillModelLeadingCfgMap;
map<UINT32, SSkillModelEmitCfg>
m_cSkillModelEmitCfgMap;
map<UINT32, SSkillModelRangeCfg>
m_cSkillModelRangeCfgMap;
map<UINT32, SSkillModelBufCfg> m_cSkillModelBufCfgMap;
map<UINT32, SSkillModuleSummonCfg>
m_cSkillModelSummonCfgMap;
map<UINT32, SSkillModelMoveCfg>
m_cSkillModelMoveCfgMap;
map<UINT32, SSkillModelSwitchCfg>
m_cSkillModelSwitchCfgMap;
map<UINT32, SSkillModelPurificationCfg>
m_cSkillModelPurificationCfgMap;
map<UINT32, SSkillModelLinkCfg>
m_cSkillModelLinkCfgMap;

map<UINT32, SSPassiveSkilllCfg>
m_cPassitiveSkillCfgMap; &&
map<UINT32, EPassiveSkillType> m_cPassitiveTypeMap;
被动技能类型
map<UINT32, SSPassiveEffectCfg_BloodSeek>
m_cPassitiveEffectBloodSeekCfgMap;
map<UINT32, SSPassiveEffectCfg_Rebound> m_cPassitiveEffectReboundCfgMap;
map<UINT32, SSPassiveEffectCfg_Relive> m_cPassitiveEffectReliveCfgMap;
map<UINT32, SSNewSkillCfg*> m_cpSkillLevelMap;
&&
cs初始化
CCSCfgMgr::Initalize 加载CSBattleMgr目录和CSBattleMgr\NewSkill下的配置 =>
LoadNewSkillCfg 加载SkillCfg_manager.xml等配置
ss获取配置
CCSBattleMgr::OnMsgFromSS_AskAllCfgList =>
CCSBattleMgr::PostMsgToSS_NotifyNewSkillCfgList 等推送 =>
CSSCfgMgr::OnMsgFromCS_NotifyNewSkillCfgList 等加载 => CSSCfgMgr::AddSkillCfg
== skill config 相关

== 技能相关

```
enum ESkillState{
          eSkillState_Free,
          eSkillState_Preparing,  吟唱
          eSkillState_Releasing,  前摇
          eSkillState_Using,   使用中
          eSkillState_Lasting,  后摇
          eSkillState_End,
};
CSSSkill
    INT32  m_normalAttackReleaseTime;
    const SSNewSkillCfg        *cpsCfg;
    bool                       ifRunning;
    ref_ptr<CSSGameUnit>       pcMasterGU;  masterGu
    ref_ptr<CSSGameUnit>       pcTarGU;   targetGu
    CVector3D                                    cTargetPos;
    ESkillState                eSkillState;
    TIME_MILSEC                tStateMilsec; //上个状态切换时间
    TIME_MILSEC                tCooldownMilsec; //技能冷却时间
    TIME_MILSEC                tBeginMilsec;//技能开始时间
    UINT32                     un32UsingEffectsArr[c_n32UsingEffectsNum];  特效唯一ID
    bool                       bIfCanCooldown;
    bool                       bIfOpen;
    UINT32                     un32PassitiveSkillArr[c_n32MaxNextSkillEffectCfgNum];
    CVector3D                  cDir;
== 使用前检查
IfSkillUsable   消耗 => cd
IfHasPreTime    吟唱时间/前摇时间
IfImpactSkill   是否瞬发技能
IfSkillRunning  技能运行中
CheckConsume   消耗检查
CheckStatus   检查技能所有者状态   调用CSSGUParameterMgr::GetValue接口
CheckAndSetTarget   设置目标  指定技能 => aoe技能
    CSSHero::GetLockedTargetGUID  设置 pcTarGU
    CSSHero::CheckUseGasSkill   Gas技能检查
IfSkillUsableWithNowTarget  消耗检查 => 设置目标 => 营地检查 => 目标存活 => 目标在施法距离
    CheckConsume
    CheckAndSetTarget
== cd 相关
DoCoolDown   检查和设置cd  设置 tCooldownMilsec
    第一种, 普攻  周期= (cpsCfg->n32ReleaseMilsec + cd)* 攻速
        CSSGameUnit::GetFPData(eEffectCate_AttackSpeed)   获取实时攻速
    第二种, 技能  curtime + cd * cdReduce
ReFreshCD   重置tCooldownMilsec为当前时间
OnValueChanged   修改前摇阶段的普攻cd    ??
    修改 tBeginMilsec  tCooldownMilsec
== 特效相关
MakeSkillEffect    触发被动 =>  特效
    CSSGameUnit::OnPassitiveSkillCalled
```

```
    CSSEffectMgr::AddEffectsFromCfg
ClearUsingEffects
== 其它方法
CheckAndDoInstantSkill   瞬发技能free后处理
    MakeSkillEffect
    第一种, cpsCfg->n32SkillLastTime =0    End
    相反,  设置 eSkillState = eSkillState_Using  tStateMilsec = curtime
== 主要方法  &&
Start    开始使用技能
    第一步, 检查   IfSkillUsable
    第二步, 更新状态  设置 eSkillState = eSkillState_Free  tBeginMilsec = curtime
    第三步, 瞬发技能free后处理
        CheckAndDoInstantSkill
End     设置 eSkillState = eSkillState_End
TryCancle   取消技能
    第一步, 检查
        第一种, 前摇前阶段/后摇阶段, 允许
        第二种, 使用阶段,  CSSSkillEffect::IsCanStopUsing =>  判断所有特效是否允许
    第二步, 取消使用中技能特效
        使用阶段,  CSSSkillEffect::ForceStop  停止特效
    第三步, End
HeartBeat    检查使用者状态 =>  检查距离  =>  状态切换处理  =>  同步状态
    状态切换处理
        等待状态  eSkillState_Free =>  eSkillState_Preparing
            设置技能方向
        吟唱状态  eSkillState_Preparing => eSkillState_Releasing
            等待时间判断
        前摇状态  eSkillState_Releasing => eSkillState_Using
            第一步, 前摇时间判断  普攻需根据攻速实时计算
            第二步, 扣费判断  CheckConsume
            第三步, 特效和伤害计算 &&   MakeSkillEffect
        引导状态  清除技能特效
            CSSSkillEffect::IsUsingSkill =>  ClearUsingEffects
        后摇状态  eSkillState_Using => eSkillState_End
            后摇结束时间判断
        结束状态  End
    同步状态
        CSSGameUnit::BeginAction_LastingSkill 等action接口

CSSAI_Hero::AskUseSkill
CSSSkill::IfSkillUsableWithNowTarget
CSSSkill::Start

CSSAI::CancleAttack/CSSAI_Hero::ClearAction
CSSSkill::TryCancle

CSSGUParameter::Recount    更新配置  减cd等
CSSSkill::OnValueChanged
```

CSSAI_HeroRobot::HeartBeat => CSSAI_Hero::HeartBeat => CSSAI::AttackHeartBeat =>
CSSSkill::HeartBeat
== 技能相关

== 特效相关 包含伤害计算
CSSSkillEffect
Begin
End
Update
CheckCooldown
ForceStop
IsCanStopUsing

CSSPassiveEffect
Trigger

CSSEffectMgr
    m_WaitingEffectMap 等待中效果列表   AddEffect 添加 OnHeartBeat 删除
    m_UpdateEffectMap 执行中效果列表   AddEffect/OnHeartBeat 添加 OnHeartBeat 删除
RemoveEffect   移除特效 => 从列表移除
    CSSSkillEffect::ForceStop
RemoveAllEffectBySkillID
AddEffect 添加特效到列表
    第一步，特效有延迟时间，添加 m_WaitingEffectMap
    否则， CSSSkillEffect::Begin   并添加到 m_UpdateEffectMap
    第二步， CSSSkill::DoCoolDown
AddEffectsFromCfg   根据配置创建一个新的效果，一般在技能起点或者效果调用别的效果的时候使用
    CreateSkillEffect
    AddEffect
UseSkillDirect   跳过技能系统，直接添加一个效果 例如购买装备
    CSSSkillEffect_Caculate::CaculateSkillEffectOnce 对目标执行结算(加血等)
OnHeartBeat
    第一种，等待列表   检查 => 执行特效 => 放入执行列表
        CSSSkillEffect::Begin
    第二种，执行列表   每一帧需要运行Update方法。如果运行返回不是Normal(出错/结束)，则将其终止并移除
        CSSSkillEffect::Update
        结束特效   CSSSkillEffect::End 并从执行列表移除

伤害计算
CSSEffectMgr::UseSkillDirect / CSSEffectMgr::AddEffect =>
CSSSkillEffect_Caculate::CaculateSkillEffectOnce/CSSSkillEffect_Buf::AddBuffEffect/CSSSkillEffect_Buf::End
/CSSSkillEffect_Caculate::Begin =>
CSSHero::ApplyHurt

== 特效相关

== 被动技能相关
CSSPassiveSkill public ISSightObject
    m_tLastEffectMilsec 上次触发时间 + cd
    m_un32TriggerTimes 触发频率 ??
    m_lKeepedEffects   当前保持的特效 AddKeepedEffect 时添加 OnHeartBeat 时删除
AddKeepedEffect 添加当前状态被动特效
OnHeartBeat 删除失效特效
    DestroyAFreePassitiveEffect 删除 m_lKeepedEffects
Trigger 触发被动
    检查概率触发 CSSEffectMgr::CanEffectBeTriggered
    检查触发频率 CheckTriggerTimes ??
    刷新CD m_tLastEffectMilsec = GetUTCMiliSecond()
    调用主动技能 CSSEffectMgr::AddEffectsFromCfg
    调用被动效果 GetKeepedEffect => CreateEffect => CSSSkillEffect::Trigger =>
AddKeepedEffect

CSSPassiveSkillMgr
    m_PassiveSkillMap[id]=CSSPassiveSkill   AddPassiveSkill 时添加 OnHeartBeat 时删除
    m_DelayDeleteVec[i]=n32PassiveSkillUniqueID RevmovePassiveSkill 将准备删除的技能
加入延迟删除列表

AddPassiveSkill 初始化 装备等决定被动
    CreatePassiveSkill
    CSSGameUnit::AddPassiveSkill 目标添加被动技能
    CSSPassiveSkill::OnAdd
        GetEffectMgr::AddEffectsFromCfg
Trigger
    CSSPassiveSkill::Trigger 触发被动
OnHeartBeat
    第一步, 删除失效被动 遍历和删除 m_DelayDeleteVec 删除m_PassiveSkillMap => 移除目
标被动
        CSSGameUnit::RemovePassiveSkill
    第二步, 删除失效effect CSSPassiveSkill::OnHeartBeat
== 被动技能相关

AI
疑问
1 CSSAI_Hero是所有hero的AI CSSAI_HeroRobot是机器人AI

== 文件分类
HERO_ROBOT 包含behavior目录下所有文件
npc AI 包含CSSAI_Building CSSAI_Soldier CSSAI_Wild

== HERO_ROBOT 相关
xml配置

```
<info un32ID="3021">   n32ID   节点id
    <n32FunctionID>0</n32FunctionID>   n32ModelID
    <n32NodeID>1</n32NodeID>           n32NodeType  0-3
    <n32ExNodeID>3015</n32ExNodeID>    n32ParentID   父节点id
    <n32Value>0</n32Value>             an32Parameters
    <PostScript>读条打断</PostScript>
</info>
<info un32ID="184">
    <n32MapID>1001</n32MapID>
    <n32CampID>1</n32CampID>
    <n32NodeID>6</n32NodeID>
    <n32NodeX>8632</n32NodeX>
    <n32NodeY>6006</n32NodeY>
    <n32NodeZ>7176</n32NodeZ>
</info>
```

继承

```
CSSBTreeNode => CSSBTreeCondition => CSSBTreeCon_xxx
             => CSSBTreeAction  =>  CSSBTreeAct_xxx
            => CSSBTreeSelector
            => CSSBTreeSequence
```

加载配置

```
CSSCfgMgr::CSSCfgMgr
    CSSCfgMgr::OnMsgFromCS_NotifyRobotAICfgList       ./CSBattleMgr/RobotAICfg.xml
    CSSCfgMgr::OnMsgFromCS_NotifyRobotAIPathCfgList   ./CSBattleMgr/RobotPath.xml
```


初始化

```
CSSAI_HeroRobot::CSSAI_HeroRobot
第一步, 创建路径  m_path[m_un8ChoosedPath][m_un16PathIndex] = CVector3D 当前目标路径
第二步, 创建action/con/seq/select各类节点+构建addchild关系 => m_pBTreeRoot
CSSBTreeSelector  满足一个跳出，执行action
CSSBTreeSequence  所有满足跳出，执行action


=====移动 AI
CSSAI  m_ifMoving
MoveToTar      和 CSSAI_Hero::AskMoveTar 区别: AskMoveTar仅被CSSAI_HeroRobot对象调
用 ??
    CSSMoveMgr::AskStartMoveToTar  调用移动模块
OnMoveBlock

CSSAI_Hero: public CSSAI
AskMoveTar
    第一步, 停止攻击、技能等
    第二步, CSSMoveMgr::AskStartMoveToTar  调用移动模块
AskMoveDir    unuse &&
    第一步, 停止攻击、技能等
    第二步, CSSMoveMgr::AskStartMoveDir  调用移动模块
```

CSSAI_HeroRobot: public CSSAI_Hero
MoveByPath　按照m_un8ChoosedPath继续移动 (CSSBTreeAct_MoveByPath::Action 设置 m_un8ChoosedPath)
    CSSAI_Hero::AskMoveTar

移动 action
CSSBTreeAct_MoveByPath::Action
    FindNearestPathNode　简单获取两点最短路径
    CSSAI_HeroRobot::MoveByPath
=====移动 AI

=====attack AI

CSSSkill
TryCancle
Start　施法
HeartBeat 技能在施法中，调用技能心跳

CSSAI　m_pAttackSkill　m_eAttackState
SetNormalAttackSkill　添加hero/下线 CSSHero::ResetAI时设置
AttackHeartBeat　判断m_eAttackState
    第一种，追击　施法距离 => 移动 => 停止移动
        CSSAI::MoveToTar
    第二种，施法　未施法 => 施法距离 => 施法 => 施法失败 => 技能心跳
        CSSSkill::Start　调用技能模块
        CSSSkill::HeartBeat　调用技能模块

CSSAI_Hero: public CSSAI　m_pWantUseSkill　m_pNowSkill m_nextSkill m_ifAutoAttack
AskUseSkill　未施法 m_nextSkill => 施法距离 m_pWantUseSkill => 追击 => 瞬发技能 => 施法 m_pNowSkill
    CSSSkill::Start　调用技能模块
UseSkillHeartBeat　m_nextSkill 存在时：
    CSSSkill::HeartBeat　调用技能模块
TryUseSkillWithAnyType
    第一种，物品技能　m_pcHeroGU->AskUseGoods　调用角色模块
    第二种，施法　AskUseSkill
WantUseSkillHeartBeat　m_pWantUseSkill 存在时：　施法距离 => 追击 => 施法
    TryUseSkillWithAnyType
HeartBeat
    WantUseSkillHeartBeat
    UseSkillHeartBeat
    CSSAI::AttackHeartBeat　自动攻击时
    DoStandNormalAttack　查找可以主动攻击目标
AskStartAutoAttack　自动攻击
    CSSAI::AttackHeartBeat

攻击 action

CSSBTreeAct_NormalAttack::Action
    CSSAI_Hero::AskStartAutoAttack

技能 action
CSSBTreeAct_UseSkill::Action   锁定目标 => 施法
    第一种, CSSHero::AskUseGoods 物品技能 调用角色模块
    第二种, CSSAI_HeroRobot::AskUseSkill

CSSAI_HeroRobot 移动和攻击核心 &&
HeartBeat   执行动作 => 移动状态 => 移动 => CSSAI_Hero心跳
     m_pBTreeRoot->Travel 执行action 执行3次break
     MoveByPath 自动寻路
     CSSAI_Hero::HeartBeat
=====attack AI

== NPC 相关

class CSSNPC : public CSSGameUnit   m_aspSkillArr CSSGameObject::m_pNormalAttackSkill
m_pAI
LoadNPCCfg   加载 CSBattleMgr\NPCCfg.xml的 un32SkillType1 技能ID 设置
m_pNormalAttackSkill
    SetAI 设置 m_pAI
==视野相关
GetSight
OnAppearInSight   重写 ISSightObject::OnAppearInSight
OnDisappearInSight
IfNotNeedSynSight 建筑和召唤物不需要异步视野
==其它
LookForEnemy   敌人数组 + 视野内 + 仇恨等级 => 目标
OnHeartBeat   回血 + AI心跳 + 被动技能心跳 + CSSGUParameterMgr::OnHeartBeat 属性改变
推送
     m_pAI->HeartBeat
== NPC 相关

== NPC AI相关

初始化
CSSNPC::LoadNPCCfg
    CSBattleMgr\MapConfig\SS_SoldierWaypoints.xml 小兵路线图

CSSAI_Building: public CSSAI   m_pcBuildingGU
HeartBeat 寻敌 => 普攻
    CSSNPC::LookForEnemy   调用npc模块
    CSSSkill::Start 调用技能模块

CSSAI_Wild : public CSSAI   m_pcWildGU m_eGuardState CSSAI::m_eAttackState
StopAI   仅吸附技能时 CSSAI_Hero::AskAbsorbMonster

ResumeAI
LookForEnemy 简单重写CSSNPC方法
HeartBeat 判断设置 m_eGuardState m_eAttackState
    第一种, 等待状态 寻敌 => 设置追击
    第二种, 攻击状态 未施法 => 是否返回 => 继续攻击 => 攻击心跳 => 失败继续寻敌/返回
        CSSAI::MoveToTar 返回
        CSSAI::AttackHeartBeat 攻击心跳
    第三种, 返回状态 在出生点 => 返回

CSSAI_Soldier : public CSSAI   m_eOccupyState m_pcSoldierGU m_acOrderPath
m_n32CurNodeIdx
SetOccupyPath   初始化出兵路线 m_acOrderPath
    CSSMoveMgr::AskStartMoveToTar 调用移动模块
SetAttackTarget 简单设置当前敌人
HeartBeat 修改 m_n32CurNodeIdx 判断设置 m_eOccupyState
    第一种, 走路状态 寻敌 => 设置攻击参数 => 返回
        CSSAI::MoveToTar
    第二种, 攻击状态 检查 => 攻击心跳 => 失败继续寻敌/返回
        CSSAI::AttackHeartBeat
    第三种, 返回状态
        CSSAI::MoveToTar

== NPC AI相关

== 属性相关

enum EEffectCate 属性种类

ISSParameterChangeCallBack
    OnValueChanged   CSSSkill 中实现

CSSGUParameter
== 成员
    bool   m_bIfChanged;//值是否被改变过
    CSSGUParameter* m_pMaxValueCallBack;//最大值改变的回调接口
          ISSParameterChangeCallBack* m_pValueCallback;//当前值改变的回调接口
          map<INT32, INT8> m_uniqueMap   uniqueId所有业务只用了默认值0   unuse
== 方法
AddBase
GetValue
OnSend   设置 m_bIfChanged = false
Recount 值改变时触发
    OnMaxValueChanged
    CSSSkill::OnValueChanged   只影响到改变普攻攻速 ??
SetMaxValueChangeCallback 设置 m_pMaxValueCallBack 仅血蓝用到

CSSGUParameterMgr
构造时调用 SetMaxValueChangeCallback
== 成员

CSSGUParameter*   paras[eEffectCate_End]; 各种属性
== 方法
GetValue
AddBaseValue
    CSSGUParameter::AddBase
AddBaseGroup
ChangeHP   基础血量 => 添加
SetValueChangeCallBack   设置 CSSGUParameter。m_pValueCallback
OnHeartBeat 属性改变推送 + CSSGUParameter::OnSend
== 属性相关

CSSGameUnit   m_sFpMgr(CSSGUParameterMgr)
== 常规消耗
ChangeCurHP
ChangeCurMP
CheckRecover 回复
== 常规消耗
GetFPData   获取实时属性 &&
    CSSGUParameterMgr::GetValue
ChangeFPData   修改属性
    CSSGUParameterMgr::AddBaseValue/AddPercentValue
    CSSGUParameterMgr::RemoveBaseValue/RemovePercentValue
ApplyHurt
    ChangeCurHP 根据属性计算伤害
    CSSPassiveSkillMgr::Trigger 调用技能模块
OnPassitiveSkillHeartBeat
    OnPassitiveSkillCalled   触发被动技能
        CSSPassiveSkillMgr::Trigger
OnGameUnitHeartBeat   检查 => 回复 => 心跳
    CSSAI::HeartBeat
    OnPassitiveSkillHeartBeat
    CSSGUParameterMgr::OnHeartBeat

CSSHero   CSSGameUnit 的派生
LoadHeroCfg   添加属性组 => 设置回调
    CSSGUParameterMgr::SetValueChangeCallBack
AskUseSkill
    CSSAI::AskUseSkill   调用AI模块
AskBuyGoods
    CSSEffectMgr::UseSkillDirect
AskUseGoods   物品技能   查找配置 => 使用技能 => 扣减物品
    AskUseSkillInGoods
        AskUseSkill
    CSSGUParameterMgr::OnHeartBeat
----
ssbattle

```
enum ESSBattleState
{
        eSSBS_SelectHero = 0,
        eSSBS_SelectRune,
        eSSBS_Loading,
        eSSBS_Playing,
        eSSBS_Finished,
};

CSSBattle
== 成员
        CSSBattleMgr*                   m_pCSSBattleMgr;
        CSSEffectMgr*                   m_pEffectMgr;
        CSSPassiveSkillMgr* m_pPassiveSkillMgr;
        CSSightMgr*                                     m_pSightMgr;
        CSSMoveMgr*                                     m_pMoveMgr;
    CSSVMBornMgr*               m_SSVMMgr;
    map<INT32,CSSight*>         m_pSightMap;
    GameObjectMap               m_cGameObjectMap; 用户上下线使用    EnterBattle 时添加
RemoveGameUnit 时删除
    SBattleUserInfo     m_asBattleUserInfo[c_un32MaxUserInBattle]; 战斗使用
InsertUser 时添加
    set<DealySendMsg*>          m_DealySendMsgSet;  延迟消息
== 主要方法
AskAttack  攻击相关  &&
    CSSHero::AskStartAutoAttack
AskMoveDir  移动相关  &&
    CSSHero::AskMoveDir
AddSightLight  视野相关  &&
    CSSight::AddSightLight
== 方法
KickoutAllUser
    RemoveISSUserToSight
    CSSBattleMgr::RemoveUser
InsertUser
AddHero     玩家初始化 => 进入战场
    CSSHero::LoadHeroCfg
    CSSHero::LoadPassitiveSkill
    EnterBattle
    CSSHero::ResetAI
DoPlayHeartBeat
    第一步, 检查掉线时间。如果玩家掉线超过30秒，则将其AI替换为机器人AI
        ResetAIAtOnUserOffline
    第二步, 管理器心跳
        CSSMoveMgr::OnHeartBeat 位移需要在游戏对象心跳之前运行
        CSSGameUnit::OnHeartBeat    所有玩家心跳
        CSSEffectMgr::OnHeartBeat
        CSSPassiveSkillMgr::OnHeartBeat
        CSSightMgr::OnHeartBeat
```

第三步，剔除离线玩家
RemoveGameUnit
第四步，进入刷兵时间了，才开始调用脚本。
OnAllNPCFunc
第五步，延迟消息推送
CheckLoadingTimeOut 加载数据
第一步，如果有非空位而玩家又尚未连接，则继续等待
第二步，地图静态NPC先加载
LoadMapConfigNpc
第三步，英雄相关
将英雄的位置适当分散，修改出生位置
然后创建英雄
AddHero
加入符文页属性
加入初始金钱
将英雄信息同步到客户端
第四步，添加到视野系统
AddISSUserToSight
OnHeartBeat 检查玩家状态 => 选角 => 选符文 => 加载数据 => 心跳
CheckPlayTimeout
CheckSelectHeroTimeout
CheckSelectRuneTimeout
CheckLoadingTimeOut
DoPlayHeartBeat

CSSBattleMgr
== 成员
UINT64
m_un64MaxBattleID;
map<UINT64, CSSBattle*> m_cAllBattleMap;
OnMsgFromCS_CreateBattle 时添加
CSMsgHandlerMap m_asCSMsgHandler;
GSMsgHandlerMap m_asGSMsgHandler;
GCMsgHandlerMap m_asGCMsgHandler;
map<UINT64, CSSUser*> m_cUserGUIDMap; 用户上下线使用
AddUser 时添加
map<SUserNetInfo, CSSUser*> m_cUserNetInfoMap; 用户上下线使用 AddUser 时添加
TIME_MILSEC
m_tHeartBeartUTCMilsec;
== 方法
CreateNewBattle unuse
AddUser 创建 CSSUser 并加入 m_cUserGUIDMap
Initialize 初始化消息handler
OnMsgFromCS_CreateBattle
第一步，创建战场 CSSBattle/CSSGuideBattle
第二步，读取地图信息 CSSBattle::LoadMapData
第三步，

```
        CSSBattle::InsertUser
        AddUser
    第四步, 加入到战场管理器中  m_cAllBattleMap
    第五步, 发送创建结果给CS服务器
OnBattleHeartBeat   遍历 m_cAllBattleMap 所有战斗心跳
    CSSBattle::OnHeartBeat
    CSSGameLogMgr::RefreshLog
---------------------------------------
ThreadActor

BattleLogicThreadActor  执行战斗管理器心跳, 服务于CSSWorkThreadMgr
== 成员
          Active*                          m_pActive;     使用active发消息给cs/gs
          CSSBattleMgr*        m_pBattleMgr;
          int                                        m_BattleNum;
== 构造方法
    Active::CreateActive  使用Active::GetTimer定时执行 CSSBattleMgr::OnBattleHeartBeat


结构体
struct BattleActor{
                    BattleLogicThreadActor* pActor;
                    vector<SGUID_KEY> guidVec;
          };
typedef map<INT64,   BattleActor>          BattleID2ActorIDMap;
struct UserActor{
                    BattleLogicThreadActor* pActor;
                    SGUID_KEY sSGUID_KEY;
                    SUserNetInfo sNetInfo;
          };
typedef map<SGUID_KEY, UserActor*>        User2ActorMapByGuid;
typedef map<SUserNetInfo, UserActor*>     User2ActorMapByNetInfo;


CSSWorkThreadMgr
== 成员
    INT32                                  m_ThreadNum;   unuse
    CBattleTimer               m_BattleTimer;
    ActorVec         m_pBattleLogicThreadActorVec;   Start 时添加
    Concurrency::concurrent_queue<INT64>            m_ToDelBattleCallbackQueue;
NotifyDelBattle 时添加
    //3个map 共享相同
    BattleID2ActorIDMap        m_BattleID2ActorIDMap;
    User2ActorMapByGuid        m_User2ActorMapByGuid;//真实表：用户GUID信息(截取
CS->SS创建战场，SS自己删除战场生成的)
                                          SendCSMsgToLogicThread 用户上线/创建战场
时添加 下线时删除
    User2ActorMapByNetInfo     m_User2ActorMapByNetInfo;//网络快表：用户网络信息(截取
CS->SS创建战场、用户上下线消息生成的)
```

SendCSMsgToLogicThread 用户上线/创建战场时
添加 下线时删除
== 方法
构造
    CBattleTimer::AddTimer 执行 CSSWorkThreadMgr::CheckCSConnect
    CBattleTimer::AddTimer 执行 CSSWorkThreadMgr::ProfileReport
SendCSMsgToLogicThread   创建战场/ping/用户上下线等
    从 m_pBattleLogicThreadActorVec 获取一个 BattleLogicThreadActor 存到3个map中
NotifyDelBattle 打扫战场, 加入 m_ToDelBattleCallbackQueue, map等信息清除
Init 初始化消息
    INetSessionMgr::CreateListener
    INetSessionMgr::CreateConnector
Start 根据系统核数创建多个 BattleLogicThreadActor 并放入 m_pBattleLogicThreadActorVec
&&
Update
    INetSessionMgr::Update
    CheckDelBattleCallbackQueue 检查 m_ToDelBattleCallbackQueue并删除map等信息
    CBattleTimer::run

main
          DbgLib::CDebugFx::SetExceptionHandler(true);
          DbgLib::CDebugFx::SetExceptionCallback(ExceptionCallback, NULL);
          timeBeginPeriod(1);
          CSSCfgMgr::GetSSCfgInstance().Initialize();
          CSSWorkThreadMgr::GetInstance().Init();
          CSSWorkThreadMgr::GetInstance().Start();
          CSSWorkThreadMgr::GetInstance().Update();
          google::protobuf::ShutdownProtobufLibrary();
          timeEndPeriod(1);

AI结合 &&

== 实体
CSSGameObject m_pAI(CSSAI)

CSSHero: public CSSGameObject
ResetAI 设置 m_pAI 为CSSAI_Hero/CSSAI_HeroRobot
== 实体

==== 技能模块
CSSAI_Hero::AskUseSkill
CSSSkill::Start

CSSHero::AskUseSkill   hero调用
    CSSAI_Hero::AskUseSkill
CSSBTreeAct_UseSkill::Action heroRobot调用
    m_pAI.AskUseSkill
==== 技能模块

==== 移动模块
CSSAI_Hero::AskMoveTar
    CSSMoveMgr::AskStartMoveToTar

CSSHero::AskMoveTar   hero调用
    CSSAI_Hero::AskMoveTar
CSSBTreeAct_MoveByPath::Action heroRobot调用
    CSSAI_HeroRobot::MoveByPath
        CSSAI_Hero::AskMoveTar

CSSBattle::DoPlayHeartBeat 定时间隔100，执行所有实体timeDiff的一次移动
    CSSMoveMgr::OnHeartBeat
       CSSMoveMgr::TryMove
==== 移动模块

CSSAI_HeroRobot::HeartBeat
CSSAI_Hero::HeartBeat
CSSSkill::HeartBeat

CSSBattle::DoPlayHeartBeat
    CSSMoveMgr::OnHeartBeat 定时间隔100，执行所有实体timeDiff的一次移动   移动心跳
        CSSMoveMgr::TryMove
    CSSHero::OnHeartBeat
        CSSGameUnit::OnGameUnitHeartBeat hero/heroRobot调用
            m_pAI::HeartBeat   技能心跳
