物质世界交互系统（CHWS）总体设计

Created by XiaoP

2002.6

CHWS 是一个在网络上实现的分布式交互系统，由**服务器系统 CHWServer** 和**客户端系统 CHWClient** 组成。

网络上应该有一个中心服务器系统，该系统包含一个动态的世界数据库，该数据库可以是分布的，但是必须同步。世界数据库将维持整个世界的所有数据，并且处理世界的进展与演化，动态修改世界数据。

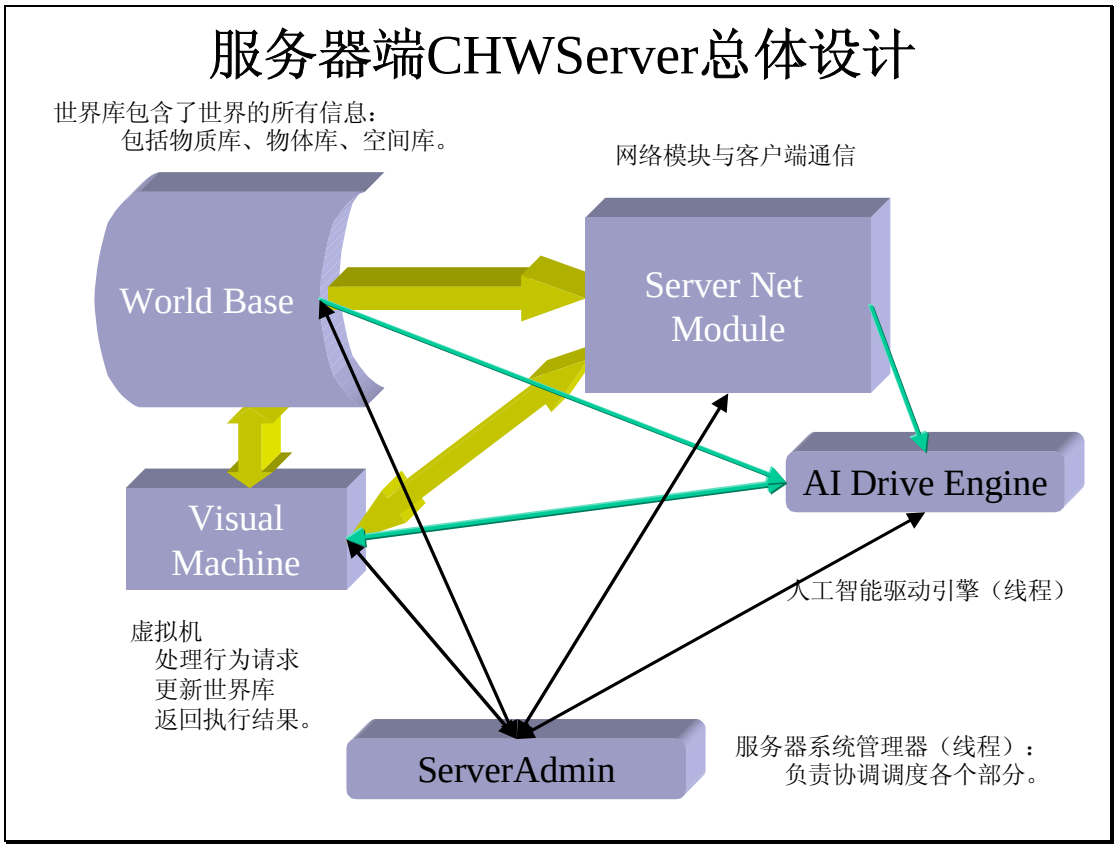
网络上分布着多个客户端系统，这些客户端系统将连接入中心服务系统从而窥探部分或整个世界。客户端系统也可以代表某个物体接入世界，按照 CHWS 物质的定义处理作用于自己的各种相互作用，并且做出某些行为响应。

人类操作者或者观察者使用具有人类能识别的多媒体界面的客户端系统接入，这样的系统具有如下典型的图形操作界面、音响系统。

非人类操作者（响应机器）直接与特定形式的客户端系统联系，接入世界系统。

客户端系统与服务系统的联系必须遵照特定的**协议 Protocol** 才能进行。

一、 CHWS 服务器系统（CHWServer）



服务器系统分为**世界数据库 World Base、虚拟机 Visual Machine、服务器网络模块 Server Net Module、AI 驱动引擎 AI Drive Engine、服务系统管理器 Server Admin** 五大部分。下面分别来讨论它们：

1、 世界数据库 WorldBase（WB）

为了实现整个世界的模拟，服务器系统上维持着一个巨大的数据库，我们将在近似程度上模拟一些世界状态（“状态”一词的理解可以参照非线性动力学等理论，我们的模拟是在硬件条件能支撑的基础上尽量做到**“多”**，当然我们也必须要使世界的处理过程进行得**“快”**，这样客户端接入的操作者才能够感觉同步流畅。一句话，尽量让它们觉得真实及时。）

世界由什么构成？很多概念吗？不多，世界就由**物体 Obj** 构成，只由各种各样的物体来构成！这里没别的实体，就只有物体，世界充满了物体。很抱歉我没用物质这个概念，因为我在设计一个系统，我必须将实体和定义（概念）区分开来。物质是一个抽象的概念，一种物质定义可以对应多个物体实体。世界中每一个物体都是属于一种物质的，物体之间的相互联系影响促成了整个世界的演化发展。也许你要问还有空间呢？空间算不算世界单独的一部分？哲学家早就讨论过了，但从来就没有讨论够。实际上物体和空间是不可分割的，任何物体都处于（属于）某一个空间中，而任何空间都应该属于某一个物体——包容器物体。在设计上只有一个总空间是预定义的，由于不能给它设置包容器，所以它就必须被预先定义。所有的物体在这个总空间内诞生了，它们又有各自的子空间，子空间里面又有各种各样的物体，这样的世界是无穷无尽的。

这个世界我将她安置在一个巨大的数据库系统里面。这样一个巨大的数据库必须是由几个部分来构成的，下面就让我们来看看 **WorldBase** 的组成。

一个**物质库 SubBase**。这个库比较复杂，幸好不会经常改变它。在里面我要定义各种物质。对于一种物质，它有哪些特征，也就是它具有什么属性和状态及其意义；创建该物质的一个新物体时它的所有状态的初始值是什么；它在某个状态组合下对外的图形表现是什么样的，这些图形信息是用来方便你我这样的人类接入者；它的行为有哪些，每个行为怎样实施；它执行一个行为会向行为对象给予什么样的影响，也就是发出什么样的相互作用；它自己接受（响应）什么样的相互作用，怎样去响应。总之，我们这个世界的物理定律将在 SubBase 中被定义。

有一点还需要特别说明，一种物质要定义一些经常性活动（**常规活动 Routine Activity**）。对于这种活动我将它解释为一系列行为响应序列，比如有生命物质对应的实体为了维持生命而进行的经常性活动。简单的生长现象，动物的获取水、空气、食物的过程等等，都可以看成这种经常性活动加以定义。这种序列是随时会进行的（起码会在空闲时候进行），它有内在的动机，看来这种动机我应该把它叫做“本能”。怎么样让世界中的物体们经常性的执行这些序列呢？它们会在一种 LifePulse 模式下被触发，由专门的部分(AIDE)来驱动，这将是以后的一个话题。

一个**物体库 ObjBase**。世界的所有数据就在于这些物体及其之间的关系，所有物体的状态勾勒出了整个世界的面貌，它们改变世界也随之而变。这个物体库总是在各种情况下被外界不断改变着，它应该被外界互斥访问的，这个在虚拟机处得以保证。世界服务器必须维护这个物体库，保证它的及时更新，保证它的稳定安全。这里我们要注意，服务器系统要随时备份的主要数据就是这个物体库，正是它保证了整个世界。

ObjBase 中保存每个物体的状态，对每一个物体也要定义一些其他部分使用的系统参数。

一个**空间库 SpaceBase**。虽然理论上物体和空间是不可分割的，但是我在实现中还是要使它们分开存在。除了总空间，每一个**子空间 SubSpace** 都与一个包容器物体联系起来，这样空间库和物体库将紧密联系，空间库中的每一个子空间离开物体库是没有意义的。而且，一个物体可以拥有数个子空间，就像你可以在身上同时带着几个包裹一样。

空间 Space 对物体的容纳采用分层结构。实际的空间本来应该是三维的，但是由于是近似的模拟，所以我采用多层二维平面来表示，每层二维平面上的一个点只能由一个物体占据。目前采用的是七层，将真实世界近似为“地表面”、“地面和天空之间”、“天空”这三个层次，其他层用于特殊物体。以后可能会增加更多的层次。

上面的三个数据库构成了我的世界 WorldBase，这个世界将自行维护这三个库的统一，它们构成了一个整体。这个整体（世界库）对外提供了一些接口，我称之为**世界管理界面 WroldAdminInterface（WAI）**，这个界面将用于外界来操作世界数据库。由于设计上将这三个库独立开来，所以 WAI 将进行所有的维护工作。我们来看看可以怎样操作。

你可以通过 WAI 读取世界中某个或多个物体的状态值。你当然可以把这些值拿去直接显示出来，只要你看得懂（电影《黑客帝国》里面哪些操作员就看得懂）。

你还可以通过界面上的接口读取物质库，根据物体的物质定义将物体的状态更图形化得显示出来。

如果你要改写某个物体的状态，你可能想要直接修改物体库，但是这太危险了。因为物体的状态联系着物质的定义，你不能随便就加上一个属性状态，必须要与物质定义相对应。更重要的是，若是物体的子空间状态值里指定了某个子空间的话，更不能随便修改这个状态值就了事，否则你很可能会丢失这个子空间。WAI 上定义了一些透明操作接口，比如移动物体、创建删除物体，修改物体的附加属性值（而不是属性）等等，你通过这些接口来操作

物体就行了。

2、虚拟机 VisualMachine（VM）

很自然的会问这些问题：既然定义了那么多，那么怎样来执行这些定义中的行为呢？物体对这些物质之间的相互作用的响应应该被怎样执行？物体的行为应该被怎样执行？那些物理定律应该被怎样执行？所以我们对世界的模拟，除了数据库和那些定义，我们还需要一个机制来使这些演化起来。我设计了一个**虚拟机 VisualMachine** 来完成这些运作和执行工作。

虚拟机处理一些行为指令和系统命令，对 WorldBase 进行操作更新，并且返回执行成功与否的结果。

哪些东西需要虚拟机来执行？

第一个显然是物体的行为。物质库定义行为的时候，将一种复杂行为分解成一些系统认可的逻辑操作序列，比如修改物体属性值、移动物体、创建删除物体、发出相互作用等等，我将这些逻辑操作序列称为**基本操作指令（BPCMD）**，它们使用**虚拟机语言（VMP）**写进物质的**简单行为（Simple Action）**定义。简单行为在**行为深度**（行为定义的一个数字字符串值，详细解释见 AI 驱动设计）上处于最底层，复杂行为可以由多个简单行为定义，层层递增，就可以定义出高级行为或者活动来。这些操作将被虚拟机按定义的流程来执行。这样如果物体要进行一项行为，可以将行为参数交给虚拟机，虚拟机执行它后更新整个世界（即按照定义的方式，行为对世界会有什么影响），发出相互作用给目标物体，并且反馈结果。

第二个是物体对物质相互作用的响应。相互作用被统一定义，不单独属于某种物质，就像我们真实世界的物理定律一样，是本质存在的，统一于物质定义之外。你一定要记住我们设计上是近似模拟！比如人对人的“语音的作用”，我们就可以理解为一种相互作用，这种作用传递语言声音。它由作用源的“说话行为”发出，由作用对象接收，作用对象“人”如果能够处理它（当然能够）就进行响应的行为——“听并且记下行为”。这种行为的执行也将由虚拟机来完成，这时候我们将相互作用的参数信息输入给虚拟机。需要注意的是，一种物质定义的相互作用响应行为是与某种相互作用对应的，由于不同种物质对同一种相互作用很可能有不同的响应，所以每种物质都会单独定义它能接受的所有相互作用的响应行为。

虚拟机和世界库 WorldBase 是分开独立设计的，虚拟机通过 WAI 操作 WorldBase。虚拟机也有自己的界面，称为**虚拟机操作接口 VMPI**。包含行为的参数和信息的**行为指令 ActionCommand** 实体将通过 VMPI 送入虚拟机执行。虚拟机在执行解释程序时应该具有检测死循环的能力，这样可以保证世界的稳定性。

另外，对 WorldBase 的需要互斥访问的部分将由 VM 来保证。

3、 服务器网络模块 Server Net Module（SNM）

网络模块将负责监听特定的接入端口，接收来自客户端的连接。这里的客户端可能由人类操作，也可能是 AI 操作。客户端与服务系统的网络模块之间的联系通过一定的协议来进行，比如 **CHWITP 协议**就是用于客户系统和服务系统通信的主要协议。

当某个客户端连接到服务系统时，网络模块负责创建对该客户的连接服务体系，该体系对客户端进行身份的认证工作，如果客户端有效，将对其服务。这里的服务内容包括两方面：一方面接收客户端的各种请求，比如**命令请求 CommandRequest、操作请求 OperateRequest** 等等，另一方面在适当时候及时向客户端发回**通知响应 NotifyResponse**。这两方面将由两个不同的线程来运作。

客户端发出命令请求，一般用于获取世界的某些信息，比如角色物体当前的环境情况、指定空间范围内有哪些物体等等。当网络模块接收到客户端命令请求以后，将按照协议读取命令参数，然后立即处理这个命令，根据情况读取 WorldBase 的信息。命令的处理结果依据命令意义而定，可能是在本次接收过程（命令对话过程）中直接将结果响应给客户端，也可

能是通过通知响应传给客户端（比如请求一个指定物体的所有状态值）。在后者的情况下，本次命令对话过程发回客户端的结果就不是完全指明命令的执行情况了的。命令请求也可能用于客户端本地物质库的更新。

客户端的操作请求，是客户端对代表角色物体发出的行为控制指令，比如行走、说话等等。当网络模块接收到客户端操作请求以后，将按照协议读取行为参数，然后构造一个**行为指令 ActionCommand** 交给虚拟机去执行，直接响应一个执行结果给客户端。注意，这个返回的执行结果并不完全是行为执行的后果，仅仅代表行为执行成功与否。也许该行为会造成很多后果，这是由行为定义并且由虚拟机解释行为的过程产生的，这可能会更新部分或者整个世界。当虚拟机在执行过程中发现行为有向物体发出相互作用的时候，如果对象物体有在线控制客户端，虚拟机就构造一个通知响应发给客户端，否则直接将相互作用发给 **AI 驱动引擎（AI Drive Engine——AIDE）**,让 AI 引擎代表对象物体去处理。当世界状态变化的时候，虚拟机在执行行为后会返回给网络模块**物体更新通知信息**，该信息指出需要通知哪些物体它们周围的情况（实际上是周围的物体）改变了，以及改变成什么样了。SNM 接着分析这些需要通知的物体，如果它们有远程控制客户端在线，那么 SNM 通过一种叫做“**物体更新通知 ObjUpdateNotify**”的通知响应将更新信息发向这些客户端，如果它们没有远程控制者，SNM 将物体更新通知信息发向 AIDE，让 AI 引擎代表需要通知的物体去处理。

4、 AI 驱动引擎 AI Drive Engine（AIDE）

AI 驱动引擎将是我们世界里面大多数物体的角色代表，这些角色都没有远程在线的控制客户端，我们可以将这些角色称为**非远程控制物体 NRCObj**。看来是 AIDE 为这些物体行为提供内在动机了。AIDE 可以读取 WorldBase 获取信息，也可以让 VM 执行一些行为。

AIDE 需要运行的一个线程是**生命线程 LifeThread**，这个线程轮流对每一个 NRCObj 执行对应物质所定义的常规活动，在常规活动的行为序列都执行完毕以后，则执行该物体的剧本。必须注意的是，为了保证每个物体常规活动的执行看起来是连续的、同时进行的，则每次循环**（生命周期 LifeCircle）**中一个物体最多执行当前常规活动行为响应序列里面的下一个行为，或者在检测到下一个行为的**行为触发状态条件 Action ActivizeCondition（AAC）**不满足时就什么也不做。当常规活动的行为序列已经进行完或者中途允许执行剧本的时候，物体的本次生命周期中就会再检测剧本，从**总体剧本 Main Scenario** 中找到该物体演绎的部分，以在物体信息中获取的该物体目前进行剧本部分简单行为序列的位置（剧本最终被编译成简单行为序列）继续执行下一个行为或者检测 AAC 不满足时什么都不做。这种模式应该要不断地循环执行，我称之为**生命脉冲 LifePulse**。

AIDE 需要运行的另一个线程是**感知线程 ApperaceiveThread**，这个线程用来接收相互作用，这个时候它代表相互作用的目标物体进行响应，执行物质定义中对应该相互作用的响应行为序列。通过该线程，AIDE 还可以接收物体更新通知信息，这时候它又代表需要通知的对象物体进行**感知**响应。物体对周围物体更新通知信息的感知实际上也会产生一个行为响应序列，当然，这又是由 VM 来执行的。感知到相互作用或环境更新的响应处理过程中，将检测很多事务需求，比如是否开始进行某个常规活动。

注意，AIDE 在让 VM 执行一系列行为的时候，需要监控行为的执行过程。如果行为进入了死循环，VM 能通知 AIDE，然后 AIDE 要能够保证该物体进入稳定状态，简单的方式就是停止行为执行，对物体状态进行稳定同步即**状态稳定 StateStabilify**。同时，AIDE 必须保证各个物体的行为都可以得到执行，而且不会让系统死在执行的某个循环里头。

5、 服务系统管理器 ServerAdmin（SA）

SA 实际上是在 CHWServer 上长期运行一个**服务系统管理器线程 ServerAdminThread（SAT）**。该线程将负责以下几个方面：

接受来自系统的启动命令，启动各个部分

在运行过程中协调各个部分的相互配合

接受来自系统停止命令，向各个部分发出停止指示，在各部分停止运行以后停止自己的运行，通知系统。

将各个系统发出的通知信息反馈给系统知晓。

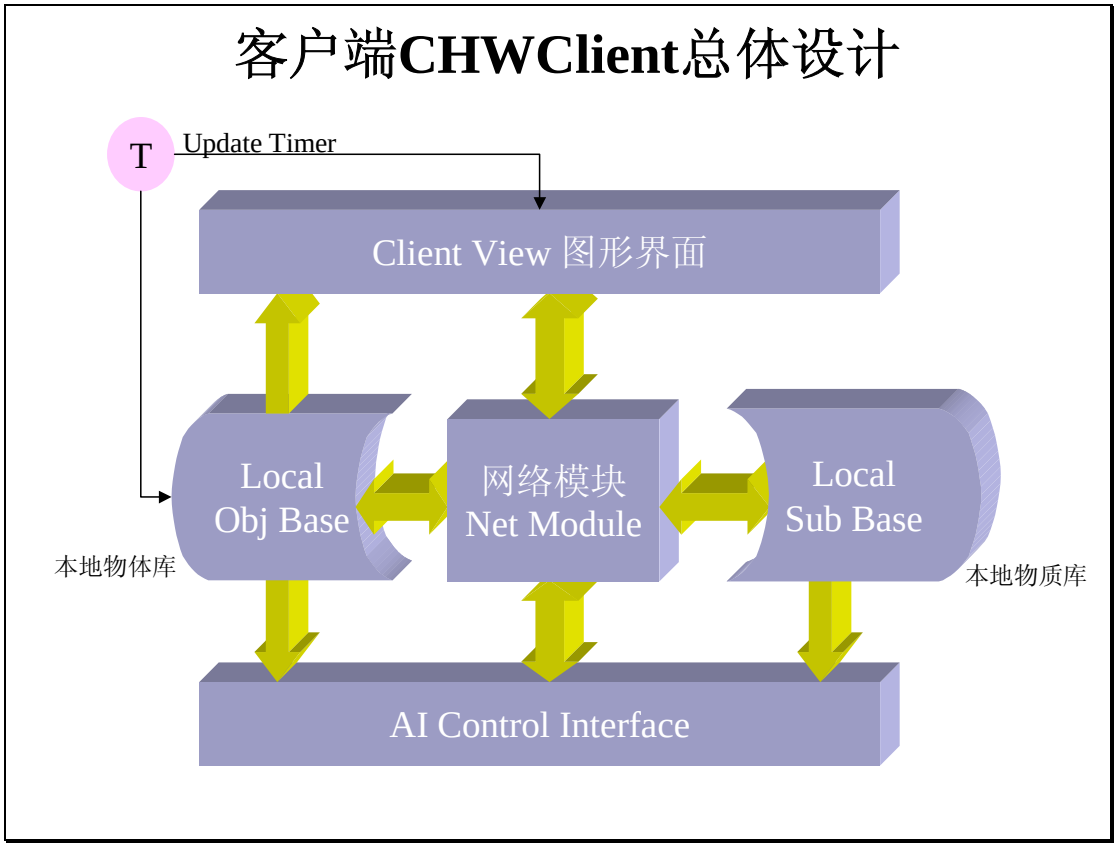
最后再说说整个服务器系统。

我将设计一个单独的包容器将这五个部分包含起来，对外将看成是一个整体，这个整体就是**服务器系统核心 ServerMain**。

ServerMain 对外界提供一个**服务器系统接口 ServerSysInterface(SSI)**。外界（也许就是个窗口）通过 SSI 来启动和停止服务器，还可以通过 SSI 来获取 SA 的信息反馈通道，这让外界能够得知各部分的运行情况。

SSI 还有一个时钟输入点，由外界向这点输入时钟脉冲。这个脉冲的作用是用于**状态稳定 StateStabilify**。由于网络故障或者虚拟机执行过程非法中断等等都可能造成物体的行为不稳定，即物体一直处在暂时的状态上面。系统以固定时间间隔检测所有物体的暂态，如果暂态超时，那么就让物体进入该暂态对应的稳态。举个例子，比如一个角色“人”正在向东走动，这时候他的远程控制端突然断线，由于物质定义已经将人的行走和站立解释成两种不同的行为，所以服务系统并不会收到客户端应该发来的站立请求，那么这个人就会一直走下去。人在这种情况下至少应该进入某个不影响大局的稳定状态，“行走”对“人”这种物质来说是暂态，它对应的稳态自然应该是“站立”。

二、CHWS 客户端系统（CHWClient）



客户端系统分为**图形界面 ClientView、本地物体库 Local Obj Base、客户端网络模块 Client Net Module、本地物质库 Local Sub Base、AI 控制接入界面 AI Control Interface** 五

大部分。下面分别来讨论它们：

1、 图形界面 ClientView（CV）

这个界面是专门针对人类操作者的，客户端获取的部分世界信息将被图形化的表示出来。这个部分世界的信息一般是以操作者所控制的角色物体为中心，周围一定范围（**可感知范围 AppercriveArea**）的所有物体的信息。

图形界面仅仅是及时显示本地构造的一个部分世界的数据，这些数据来源于本地物体库。对于一些特有的世界信息，比如说话这样的相互作用，界面上会有专门人性化的显示，可以做得好像漫画书里人物头顶的语言框框。又比如对物品的察看，将会弹出一个详细资料介绍窗口，这个窗口图形化表示该物体的详细信息，假如该物体又是个包裹类的物质，则窗口还将列出该物体子空间（包裹内部）中的物体列表。

主窗口中显示这些物体状态需要的图形信息则来自于本地物质库。图形界面必须在本地物体库或者物质库更新的时候及时更新，使界面展现出这部分世界的变化发展。

图形界面上还具有专用的几种窗口，一些用于图形化操作输入，一些用于系统信息的显示。

2、 本地物体库 Local Obj Base（LOB）

本地物体库存储用于构造需要察看的部分世界的信息拷贝。网络模块不断的接收来自服务器的更新通知，不断更新本地物体库。本地物体一旦被更新，将立即通知图形界面刷新显示。

本地物体库还要存储一些额外的信息，这些信息将用于显示刷新。

3、 客户端网络模块 Client Net Module（CNM）

CNM 首先会启动一个**客户端管理线程 ClientAdminThread（CAT）**，这个线程负责启动连接网络的其他线程，以及对上级（窗口）反馈各个线程的工作情况。

一个**客户端控制线程 ClientControlThread（CCT）**将会在操作者准备接入世界系统的时候启动。它首先连接服务器，然后按照协议与服务器沟通。这个线程用于登陆、发送命令请求与操作请求等等。

登陆成功以后，CAT 启动一个**客户端感知线程 ClientAppearceiveThread（CAPT）**，这个感知线程将一直接收客户端传来的响应。客户端传来通知响应的时候，CAPT 按协议处理这些通知响应。

如果是**物体更新通知 ObjUpdateNotify（OUN）**，CAPT 就构造**本地物体信息 LocalObjInfo** 更新 LOB，然后立即通知界面刷新显示或者通知 AICI 角色自己或者环境更新了。

如果是**相互作用通知 EffectNotify（EN）**（客户端应该只处理对人类的相互作用），CAPT 就构造相互作用信息，如果信息本身需要就以人性化的多媒体方式表示出来，比如别人在说话的图形表示、爆炸的声音等等。当然，这是人类操作者的情况，如果是 AI 操作，CAPT 就不必将相互作用传给 AI 了，因为 AI 操作者我将设计成为**状态响应机器 StateResponseMachine（SRM）**，它只对自己或周围的状态改变做出响应。

请注意，当相互作用到来后，操作者一般会做出必要的响应来。

4、 本地物质库 Local Sub Base（LSB）

本地物质库主要用于为物体状态显示提供显示信息。

原则上本地物质库应该与服务器物质库相同，但是由于服务器上的物质库不断有新的物质加入，如果保持严格同步将非常浪费网络带宽。所以采用这样一种同步策略，当客户端获取一个物体的状态后，发现该物体所对应物质在本地物体库上没有被定义，则显示界面上将暂时不显示该物体（代替标志）。那么则可以肯定是服务器上新加入了该物质的定义。这时候 CNM 可以通过 CCT 发出一个**获取物质信息 GetSubInfo** 命令请求到服务器端，并接受服

务器的响应来更新 LSB。

5、 AI 控制接入界面 AI Control Interface（AICI）

该界面用于 AI 操作者接入控制。

AI 操作者设计成 SRM，对自己和周围的状态改变做出响应，这个接入过程使用 AICI 的统一接口。AI 操作者的设计将是下一个阶段的目标了。

在 AI 接入的时候，CNM 的通知将直接通过 AICI 发向 AI。

最后再说说整个客户端系统。

我将设计一个单独的包容器将除 CV 的其余四个部分包含起来，对外将看成是一个整体，这个整体就是**客户端系统核心 ClientMain**。

ClientMain 对外界提供一个**客户端系统接 ClientSysInterface(CSI)**。外界（也许就是个窗口）通过 CSI 来启动和停止客户端，还可以通过 CSI 来获取 CAT 的信息反馈通道，这让外界能够得知各部分的运行情况。