VT-4坦克车模拟训练平台

系统可行性分析报告

目录

[一．介绍 1](#_Toc68878670)

[二．项目分析 1](#_Toc68878671)

[三．技术分析 1](#_Toc68878672)

[1.物理引擎 2](#_Toc68878673)

[2.图形引擎 2](#_Toc68878674)

[3.同步方式 2](#_Toc68878675)

[四．设计分析 3](#_Toc68878676)

[1.设计类型 3](#_Toc68878677)

[2.设计模式 3](#_Toc68878678)

[3.设计目标 4](#_Toc68878679)

[五．可能存在的问题 4](#_Toc68878680)

[1.训练体验差，竞技体验差 4](#_Toc68878681)

[2.开发成本高，维护成本高，扩展性差 4](#_Toc68878682)

[3.部署复杂度高 5](#_Toc68878683)

[六．想法及解决思路 5](#_Toc68878684)

[1.模块划分 5](#_Toc68878685)

[2.硬件和软件配置应当具有通用性 5](#_Toc68878686)

[3.一个项目只选用一个引擎 5](#_Toc68878687)

[4.融合形式的改变 5](#_Toc68878688)

[5.开发自己的军事仿真模拟引擎 5](#_Toc68878689)

[七．提案 6](#_Toc68878690)

[1.用户管理平台 6](#_Toc68878691)

[2.数据分析平台 6](#_Toc68878692)

[3.VT4编辑器 7](#_Toc68878693)

[4.VT4主体程序 7](#_Toc68878694)

[八．总结 8](#_Toc68878695)

# 一．介绍

本文首先对项目进行了整体的分析。之后分别从技术的层面和设计的层面对该项目的可行性进行了分析阐述，其中主要分析了两款引擎之间的差异，两种设计类型的差异。之后结合分析所得，提出了三点可能会面临的问题，并在之后提供了五点想法及解决思路。在最后提出了一套与现有设计思路不太相同的可行性方案提供参考。

# 二．项目分析

根据对VT4启动会议纪要和VT4产品线组成方案的研究分析，了解到项目主要分为理论学习训练、单战位操作技能训练、单车内多战位协同训练、多车车组协同训练、战术推演训练五大模块。

其中理论学习训练、单战位操作技能训练为单独引擎开发的单机模块，而且之前已经有成功案例，所以在可行性方面是没问题的。且相对独立，融合性较强，通过些许调整就可以很好的融合于系统之中。

单车内多战位协同训练模块、多车车组协同训练模块两者可以合并为协同训练模块，两者是相同的业务逻辑，相同的开发逻辑，只是参与的人员数量不同，设置为不同的房间或不同的关卡设计即可。

战术推演训练模块与前面几大模块有较大的差异，尤其是于协同训练模块，差异主要表现在涉及了两种游戏模式的同时还使用了两种开发引擎。

所以接下来的内容主要讨论协同训练模块和战术推演训练模块结合的可行性。

|  |  |  |
| --- | --- | --- |
| 模块 | 协同训练 | 战术推演训练 |
| 类型 | FPS（第一人称射击）+ SLG（模拟驾驶） | RTS（即时战略） |
| 引擎 | CRYENGINE V | Unity |
| 物理引擎 | CryPhysics | PhysX |
| 图形引擎 | CRYENGINE V | Unity |
| 同步方式 | State Sync（状态同步） | Lockstep（帧同步） |

# 三．技术分析

该部分主要是从两大引擎的融合可行性方面做出分析。

## 1.物理引擎

Unity物理引擎基于PhysX开发，而CRYENGINE V的物理引擎为CryPhysics。CE官方文档中提到虽然CRYENGINE除了本机物理引擎（CryPhysics）之外，还允许CRYENGINE用户将Nvidia的第三方物理引擎（PhysX）用作CRYENGINE中的外部物理引擎。但同时也做出警告，指出此功能仍处于测试阶段，不建议在产生依赖关系的生产环境中使用它。

即使两者都使用PhysX物理引擎，两大引擎也可能基于PhysX加入了自身的一些处理逻辑，不敢保证可以实现相同的物理特性。也不能保证不会遇到CE的引擎bug，这都需要测试确认。如果物理引擎的差别导致程序无法达到预期效果，需要修改底层的逻辑这个工作量和工作难度将会很大。

## 2.图形引擎

图形引擎由光照，摄像机，材质纹理，粒子系统，渲染功能等组成，CE与Unity采用不同的图形引擎。将两者进行融合以后，虽然是相同的模型，相同的场景，相同的参数，但是在不同的程序中展示的效果是截然不同的，这将大大影响系统的整体性。虽然两者是在同一个场景做同样的事情，但是图形的展示效果却不尽相同，直观的感受会有很大差距，产生一种分离感。

## 3.同步方式

同步方式本身是没有硬性要求的，但是从使用性和扩展性出发，有一些建议性的选择方案。

协同训练融合了第一人称射击和模拟驾驶元素，要求操作反馈明锐，对操作手感的要求较高，状态同步配合客户端的一些预处理算法，可以很好的实现以上需求。但是问题在于状态同步数据量大，当实体过多之后，服务器软硬件压力都会很大，处理不好容易出现明显的卡顿等，影响体验。

而战术推演训练作为即时战略模式，往往会需要同时操作大量的军事单位，来实现战略战术目的，所以需要同步的实体数量上会很多，而且即时战略模式主要以鼠标点击地形进行操作，对操作手感没有过多的要求，而是对操作的即时性有较高要求，帧同步只是同步操作数据，数据量小，不受单位实体数量的影响，相同帧的数据，也保证了即时性和公平性，所以即时战略一般采用帧同步的方式。

所以如果采用系统方案中的硬融合，就必须在以上两种同步方式中进行取舍。如果选择状态同步，在战术推演训练的即时性和扩展性上可能会出问题，如果选择帧同步，可能会对协同训练精细的配合操作有所影响，而且不论用哪种方式，为了提高体验，技术方面都将面临巨大的挑战。

# 四．设计分析

该部分主要是从游戏类型的融合可行性方面做出分析。因为设计层面的分析是基于游戏的开发模式来展开的，所以我先简述一下我对游戏的看法。

电子游戏于1952年面世，虽然距今不足七十年，但其发展速度之快，已经深入当代人的生活。就像书籍，影视的发展一样，电子游戏已经不单单是指狭义的休闲娱乐工具，而已经逐渐发展为一种文化的载体，一种对现实世界或思维世界的模拟。如今电子游戏已经发展出其成熟的结构体系， 有很多现成的游戏设计模式可供我们参考，借鉴。这些都是前辈开发者们经过开发-发售-玩家反馈-反复调整后，一点点总结出的经验。我们决不能视而不见，更不应该去抵触这些宝贵的经验，甚至背道而驰。我们应当充分了解学习游戏的设计模式并灵活运用与我们的系统开发。取其精华，去其糟粕。在其基础上，实现我们更加专业化的需求。

## 1.设计类型

协同训练模块

协同训练区分为车长，炮长和驾驶员。对车长和炮长来说，类型为第一人称射击类。对于驾驶员来说为第一人称模拟驾驶类。这两种游戏类型的结合在游戏中是比较常见的，市面上也有大量的成熟案例可供参考。所以协同训练模块是可行的。

战术推演训练模块

战术推演训练模块属于策略类大类中的即时战略类，与协同训练模块中的第一人称射击类游戏和第一人称模拟驾驶类所属不同的体系架构，在融合方面比较困难。但也是可以通过一些设计来实现。

## 2.设计模式

《Game Programming Patterns》中写道，好的设计意味着当需要做出改动，整个程序就好像正等着这种改动。

协同训练模块

协同训练中一辆VT4坦克并不是一个整体，在设计时需要由三个实体组成，分别为车长，炮长，驾驶员，三者的基类可以设定为人员类，然后其具备其自身特有的方法和同步逻辑。

战术推演模块

战术推演中VT4车辆为一个整体，继承于车辆基类，因为操作方式不同，所以类内所包含的方法也是与协同训练中的VT4类完全不同的。

同样的一个实体，需要为其设计两个完全不同的类，当需求改变时，两个类都需要进行相应的修改。并不符合程序设计的结构化和复用性，而且很容易出现bug，代码健壮性和可维护性也会很差。

## 3.设计目标

协同训练模块

协同训练的性质更适合用于小的战场环境，少量部队的小规模精细化协同战术配合训练。指挥官通过语音对车长进行指挥，听取车长的战场反馈，进行决断，达到在小型局部战争中真实的战场环境模拟效果。

战术推演模块

战术推演训练的性质更适合用于大的战场环境，多部队，多兵种，协同作战的模拟指挥训练。更多的是考验指挥官的战略部署，部队的调动，主要目标的占领等要素。小规模的局部协同作战并不应该是推演所关心的。

如果将两者结合，结合形式处理不好的话，就会显得训练目标不明确，甚至互相影响。

# 五．可能存在的问题

## 1.训练体验差，竞技体验差

战术推演端操作的单位远不如协同训练端的灵活，开始训练后，可能出现战术推演端操作的单位配合不好协同训练的单位，因而导致训练结果不理想。反之亦然。游戏类型的不同导致协同训练中的车组开火需要在3D环境中瞄准目标，预判目标的运动，然后开火打击，如果预判失误或者目标改变运动状态，则会出现无法击中的情况。到了战术推演之中，只需要选定攻击目标，执行攻击即可。这会导致训练毫无公平性可言，影响训练体验和竞技体验。

## 2.开发成本高，维护成本高，扩展性差

两个不同的引擎开发的两种不同类型的系统，很难通过一个统一的模块对其进行管理，其中包含的变数太多了，很难面面俱到。这就导致高昂的开发成本，当程序开发完成后，bug的修改，细节的调整，后期的维护都需要同时有CE，Unity，服务端三方的协同进行，协同调试测试。维护成本很高。扩展性方面，比如当要在当前基础上加入一个新的车辆，就需要修改Unity程序，修改CE程序，修改协同训练中VT4的类，还需要修改战术推演中VT4的类。如果是具备新功能的车辆，还需要调整服务端的传输协议，改动量是非常庞大的，并且还需要测试确保其在CE和Unity中的同步性，所以扩展性较差。

## 3.部署复杂度高

不符合产品化需求。需要部署Unity环境，CE环境，导调等，不同的终端需要不同的软硬件部署，灵活性差，需要专门的部署，难以达到产品级要求。作为对外功能展示，我们可以预先准备，安排好机器和环境。但是作为产品的话，一定要考虑到部署的便捷性和灵活性。

# 六．想法及解决思路

## 1.模块划分

理论学习训练、单战位操作技能训练两个模块，可以整合为一个独立的模块，也可以合并到协同训练之中，作为协同训练中的教学关卡。

## 2.硬件和软件配置应当具有通用性

作为产品的话，一定要保证其通用性。同一台硬件设备应当具备同时作为炮长，车长，驾驶员的能力，操作手柄都应当具备，根据不同的位置选择不同的操作方式。大大增加部署的便捷性和灵活性。

## 3.一个项目只选用一种引擎

每个引擎都有其特点，其独特的处理算法，处理逻辑，就好比apple的app就无法在Android手机上使用，即使是相同的功能也要开发两套，虽然说游戏引擎之间的差异没有操作系统之间那么大，但是两者间的差距也是技术层面很难克服的障碍。

## 4.融合形式的改变

策略类融合坦克仿真模拟。需要大的战略地图，高级指挥官在大战略地图中对进行整体战略的规划。当到达指定地点或遇到敌人后触发遭遇战，涉及到的部队进入一个独立房间的小的战场环境，在这个小的区域内进行像协同训练训练一样的战术级训练。

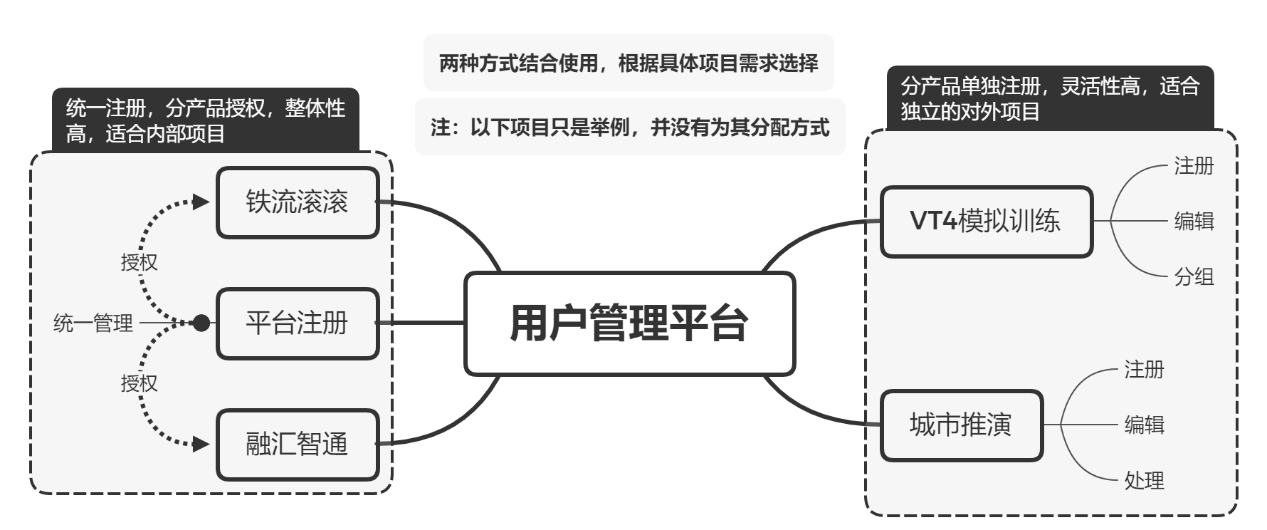
## 5.开发自己的军事仿真模拟引擎

引擎的组成是由编辑器和引擎主体程序组成的，比如像Unity我们通过其编辑器就可以实现很多的功能，包括加载地形，导入模型，修改参数等。一些基本操作我们甚至不需要编写代码就可以实现。我们要想做自己的军事仿真模拟引擎，也应该模仿其模式，利用编辑器加主体程序的思路来制作。也是利用沙盒游戏的思路。

# 七．提案

可以将开发分为四个组成部分，其中包括两个通用平台，一个编辑工具，一个主体程序。我以下只是将大的思路体系做阐述，具体的实施内容还需要进一步策划。

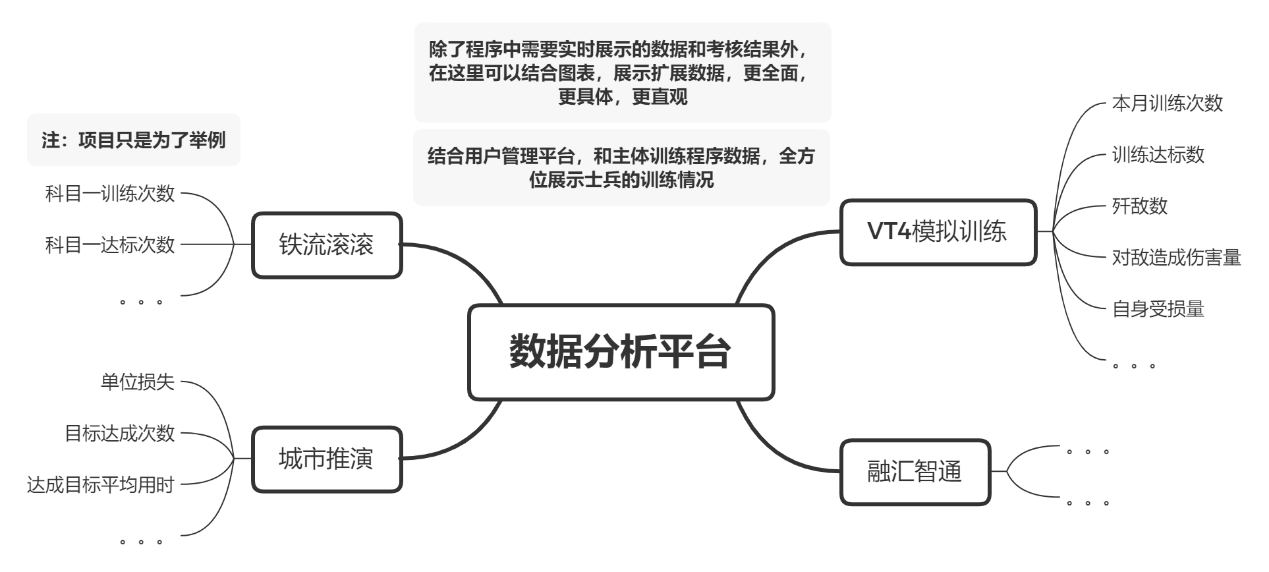
## 1.用户管理平台



可以利用web形式开发，用于用户注册，用户编辑，小组分配等功能。该平台可以负责人员管理相关的所有功能，所有需要人员管理的项目的人员管理都可以在该平台进行处理。比如可以包括VT4，城市推演等，都可以在平台中注册，管理。可以采用以上两种形式相结合，由我们主导的可以实行通行证机制，统一管理。对外需要灵活性的可以采用单产品管理。

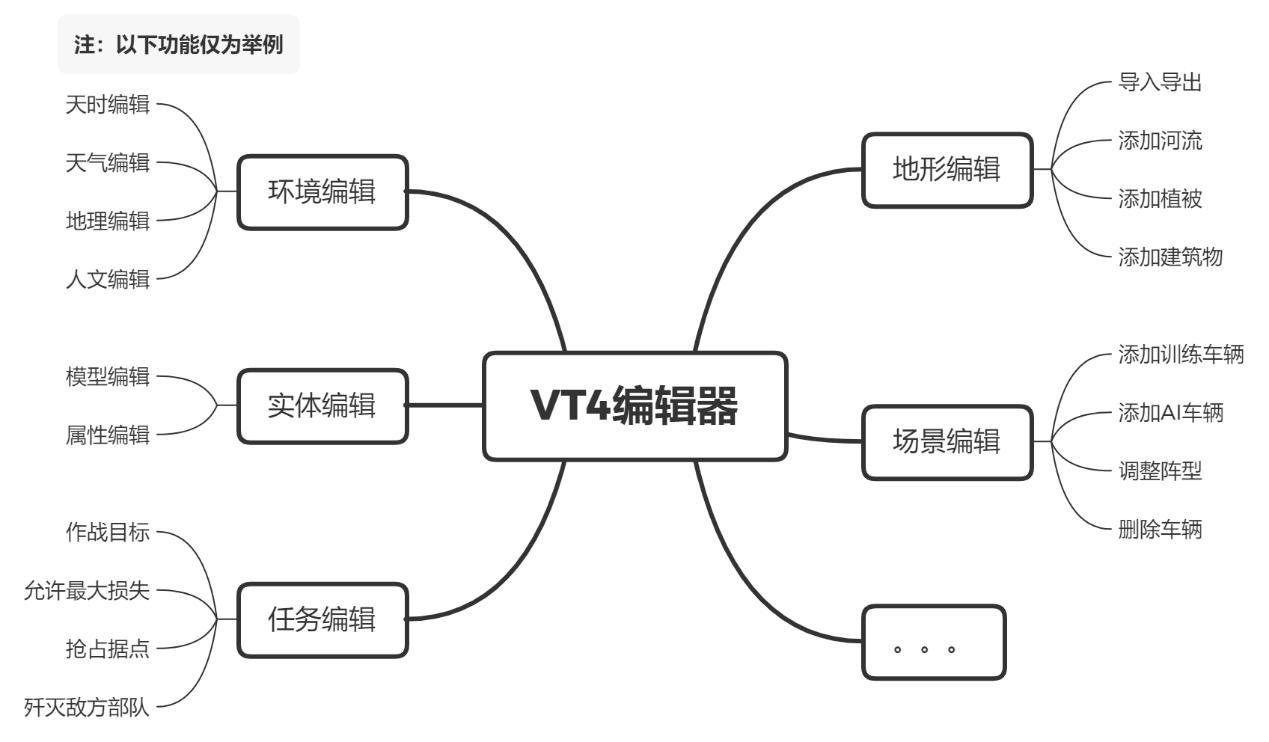
后续可以进一步扩展为项目管理平台。

## 2.数据分析平台



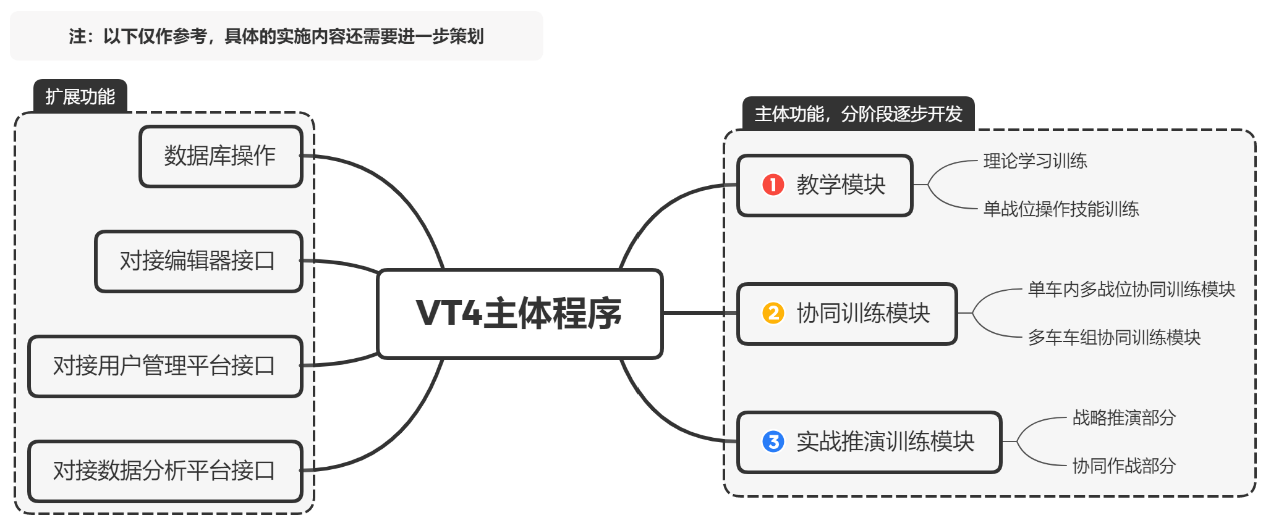
可以利用web形式开发， 在主体程序中进行的训练的数据，都将存入数据库之中，在主体程序中将进行一部分数据分析展示，但更多更全面的数据分析展示则可以交给该平台来做处理，该平台通过读取数据库数据，通过各种算法进行分析，图表进行展示。同样该平台也是可以融入公司的大部分产品，包括VT4，城市推演等，与用户管理平台相结合，全面分析展示不同人员在不同的训练科目中的表现。

## 3.VT4编辑器



作为程序主体的外部编辑工具，通过与程序主体的接口对接，实现在不修改代码的基础上，对人物，场景，单位进行添加，属性参数进行编辑等功能。

## 4.VT4主体程序



利用游戏引擎开发，承载程序主体逻辑，实现功能需求，并对编辑器开放接口。

后续配合编辑器，可以进一步扩展成军事仿真模拟引擎。初期，相同类型的项目都可以融入其中，等到主体逻辑设计全面后，比如包括了飞机，坦克，步兵等军事设备的逻辑后，就可以作为军事仿真模拟引擎，以后的项目就可以直接利用该引擎进行开发。

# 八．总结

总体而言，以项目目前的设计思路来看，可行性方面面临诸多问题。从技术层面来讲，该系统的设计思路是可以实现的。但是，实现难度巨大，而且可能遇到诸多不可预见的问题，在实现程度和实现成本上很难把控。从设计的角度来看，项目完成后可能也无法做到预想的训练目标，并且可能无法达到预期的高复用性，高扩展性，高融合性，产品化等需求。建议通过会议，进一步分析讨论，结合多方意见，完善系统方案。