文章编号：2095-2163(2022)07-0178-04 中图分类号：TP317 文献标志码：A

基于银河编辑器的即时战略游戏设计

佟子豪，乔秀明，马新宇
（北京信息科技大学 计算机学院，北京 100192）

摘 要：随着互联网行业的发展，网络游戏市场也随之兴起。网络游戏中，实时的战略对抗游戏一直受到玩家的喜爱。本文以银河编辑器为开发环境，开发并设计了一款实时的、可交互的、真实度高的即时战略游戏。游戏可模拟两方势力，在既定的规则下进行对战的场景。

关键词：游戏；规则；实体

Design of real-time strategy game based on galaxy editor

TONG Zihao, Qiao Xiuming, MA Xinyu
(School of Computing, Beijing Information Science and Technology University, Beijing 100192, China)

【Abstract】 In recent years, with the development of the Internet industry, the online game market has also sprung up. In online games, real-time strategic confrontation games have always been loved by players. Taking Galaxy editor as the development environment, this paper develops and designs a real-time, interactive and high authenticity real-time strategy game. The game can simulate the scene of two forces fighting under the established rules.

【Key words】 game; rules; entity

0 引 言

游戏作为大众生活中必不可少的娱乐方式之一，为满足使用者的各方面需求，其游戏类型和方式一直在不断地变化发展。目前，通常将其分为动作游戏、射击游戏、策略游戏、探索游戏等[1]。其中，即时战略游戏以博弈性强，节奏快等特色深受玩家喜爱。

即时战略游戏(Real-Time Strategy Game)，简称RTS，是策略游戏(Strategy Game)的一种[2]。游戏是即时进行的，而不是策略游戏多见的回合制。近年来，作为人工智能的研究对象的即时战略游戏有很多，星际争霸 2(StarCraftII)就是其中之一[3-4]。银河编辑器是一款功能强大的游戏制作工具，利用该工具可以进行几乎所有类型的游戏场景设计[5]。本文基于银河编辑器，在 Microsoft Windows10 64 位操作系统下进行即时战略游戏的设计和开发。游戏旨在模拟对战规则，制作一款兼具真实性和趣味性的战略游戏。具体设计过程有游戏地图设计、游戏规则设计、模型设计、实体属性设计等。

1 设计思路

本文开发的即时战略游戏可表述为：在一片海域中，两支海盗船队互相遭遇，并进行战斗。战斗的双方拥有不同的船只配置，以及各不相同的胜利条件。游戏中，玩家可操控一方势力，通过观察敌方船队态势来制定战斗计划，操控己方船只，力求达成胜利条件。同时，如果己方船队全灭或己方关键目标被摧毁，则游戏失败。本游戏为模拟船队的战斗场景，研究设计多种各具特色的船只及武器来增强策略性和对抗性，其模型、行为和特效动画进行设计，以求模拟战斗的真实感。

2 游戏实现

本游戏作为即时战略游戏，没有其他类型游戏的渐进流程，游戏的正常运行即为两方势力的对抗场景，在游戏规则下正常开始和结束。这需要实体对象，地图，游戏规则等的紧密配合。

2.1 游戏地图设计

地图设计包括地图的长度、宽度、像素值的设定，还包括地形、障碍物的设计。地形包括水体、礁

基金项目：北京信息科技大学计算机学院大学生创新创业训练项目(5102010805)；北京信息科技大学校基金(1825021)。

作者简介：佟子豪(2000-)，男，本科生，主要研究方向：银河编辑器平台开发；乔秀明(1989-)，女，博士，讲师，主要研究方向：语言处理、人工智能；马新宇(2001-)，男，本科生，主要研究方向：智能模型研究。

收稿日期：2021-10-10

石、陆地、丘陵等等。地图上的每一个点都有对应的高度以及纹理属性。

本游戏的地图设计基于水战，故地图中大部分地形皆为水面，配合船只无法通行的岛屿、礁石等。地图尺寸为 256 * 256，地形比例大致基于现实进行设计，以配合单位航速、武器射程等实体属性，确保游戏的真实性。

2.2　游戏规则设计

游戏规则的设计是高度自由的，由银河编辑器的“触发器”系统帮助实现。触发器由 3 部分组成：事件、条件与动作。运作流程如图 1 所示。

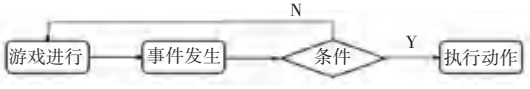

图 1　触发器流程

Fig. 1　Trigger process

其中“事件”指游戏中正在进行的或已发生的事件，作为触发器的开启判定条件。事件包括实体单位的一系列动作，如单位受到攻击、单位进入指定区域等。一些玩家的操作，例如点击某按钮、移动镜头等也可以作为事件。

“条件”是触发器内部的进一步的判定条件，当事件发生时，触发器开启，下一步则进行条件的判定。一个触发器可以不设置条件，也可以设置多个条件，但必须要有“事件”。在游戏进行过程中。当触发器设定的事件发生时，若当前所有条件为真，则执行设置好的动作。在触发器中，还可设置局部变量，帮助进行条件的编写。

当事件发生且条件为真，则执行设置好的动作。动作大致分为实体层面和游戏层面，实体层面的动作包括但不限于杀死或创建某单位，禁用单位技能等。游戏层面的动作则包含使某一方胜利、结束游戏等。图 2 展示了本游戏中当一方所有单位被消灭，则判定另一方游戏胜利的触发器。

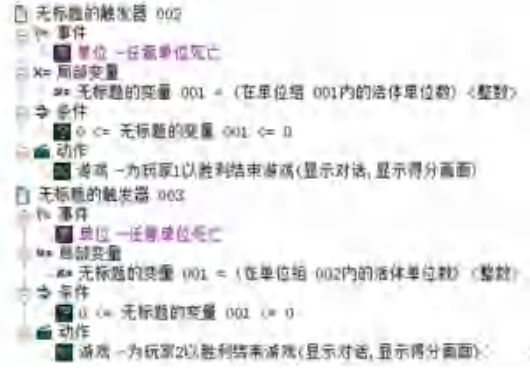

图 2　触发器设置

Fig. 2　Trigger settings

利用触发器，还可以设计出许多提高游戏真实性的规则。例如本游戏中飞弹命中概率的规则，即为设置某一单位被飞弹命中为事件，设置局部变量随机数作为条件，当随机数在某一范围时，执行免去单位受到的伤害的动作。

2.3　实体设计

本游戏设计了多种船只及武器，以实现角色间协同配合的战斗场景。船只的设计可分为属性、武器、模型等方面的设计。

此外，每种船只都设计有移动、发射武器、被摧毁的动画，与实体模型一同装载到单位上，组成单位的模型部分。

每种船只可编辑不同的武器，拥有不同的作战方式，具有自己的特性和属性值。比如载有火炮的海盗船，具有生命值、速度、加速度、视野范围等许多复杂的属性，船只模型的一切数据均以 XML 代码的形式存储，以下对海盗船模型的部分代码进行展示。

```
<CUnit id="q">//实体 ID
    <Acceleration value="2"/>//加速度
    <Speed value="1.1992"/> //速度
    <TurningRate value="39.902 3"/>//转身速率
    <LifeMax value="10"/>//最大生命值
    <Sight value="32"/>//视野半径
    <Height value="8"/>//高度
    <VisionHeight value="4"/>//视野高度
    < WeaponArray Link = " artillery " Turret = " AutoTurret"/>//使用武器:火炮
```

每种武器也具有独立的属性，比如火炮这一武器，具有飞行速度、射程范围等属性。火炮的部分代码如下。

```
    <CWeaponLegacy id="UnknownWeapon">
    <TargetFilters value = " Ground, Visible; Missile, Stasis, Dead, Hidden,Invulnerable"/>//目标筛选器
    <Range value="6"/>//射程
    <Speedvalue="2.086"/>//速率
    <TurningRate value="494.472 6"/>//转身速率
</CWeaponLegacy>
```

武器本身也是单位的一种属性，通过 XML 代码将武器链接到单位，即可实现武器对船只的装配。

此外，本游戏中海盗旗舰搭载的武器为“发射飞弹”，其代码实现如下。

```
<CWeaponLegacy id=" MissileLaunch">//武器 ID
        <PathingAmmoUnit value="f"/>//武器开
```

火时作为发射物发射出的单位

<Range value="30"/>//射程

旗舰在武器开火时创建飞弹单位,飞弹同样是实体单位,具有自己的属性值。

2.4 游戏场景设计

本游戏为模拟真实的海盗遭遇战,在游戏初始化并开始运行时,会先加载地图,并从实体列表读取已经设置好的实体单位部署在对应的坐标点。且本游戏假设战斗双方都能通过雷达手段获知敌方目标位置。图 3 展示了为本游戏预设的实体列表。

图 3 实体列表

Fig. 3 Entity list

游戏开始后,将“玩家 1”阵营的单位控制权分配给玩家,由玩家进行操控,动作指令使用鼠标和键盘进行输入。玩家可以控制己方的船只进行移动、攻击等动作,来完成制定的胜利目标。

3 游戏运行

游戏的地图、实体、规则等信息以 SC2Map 格式封装在一个地图文件中,并通过银河编辑器进行导入。游戏的按键、音量都可以通过内部的菜单来设置。当既定的胜利条件触发器被触发,则弹出游戏胜利画面,并结束游戏。

游戏的运行除玩家手动操作之外,也可由银河编辑器内置的代理库实现。将编写好的实体动作信息和地图文件输入代理库并运行,游戏中的实体单位会在每一个时间步(step)执行编写好的动作,来实现游戏的正常运行。下面是执行文件信息的核心代码:

```
#若动作类型为机动,返回机动动作
if BigStepCounter <= len(List2):
    aim =target[StepCounter][BigStepCounter]
    if action_type[StepCounter][BigStepCounter]
      == 0:
       print("动作类型的行索引:{},列索
       引:{},执行实体 tag:{},机动动作坐标为
       {}".format(StepCounter, BigStepCounter,
       Entity_bound_red[StepCounter].tag, aim))
       return actions.RAW_FUNCTIONS.Move_pt
("queued", Entity_bound_red[StepCounter].tag,
[aim.x, aim.y])## 若动作类型为打击,返回打击
动作
    elif
    action_type[StepCounter][BigStepCounter] == 1:
        print("动作类型的行索引:{},列索引:{},
        打击动作目标为{}".format(StepCounter,
        BigStepCounter, aim))
    return actions.RAW_FUNCTIONS.Attack_unit("
    queued", Entity_bound_red[StepCounter].tag,
    aim)
      else:
        return actions.RAW_FUNCTIONS.no_op()
    else:
        return
    actions.RAW_FUNCTIONS.no_op()
```

图 4 展示了存储实体动作信息的文件内容,对于每一个时间步,都存储了所有实体的编号、信息和执行的动作。

图 4 实体动作信息

Fig. 4 Entity action information

此外,本文进行了实时输出游戏信息的研究,能够在游戏运行中实时地输出每一时间步地图中的实体信息,如图 5 所示。

图 5 输出信息

Fig. 5 Output information

(下转第 184 页)