AI 实现分享

dtcxzyw

May 12, 2019

目录

安全措施

我的个性签名是——安全第一。
由于我不会附加 gdb 到 AI 上调试，只能通过 SDK 的 debug 接口输出调试信息。
下面从三个角度防范 bug：

- 执行时：拦截并报告无效操作，尽可能记录调用位置。
- 执行后：在下一次决策时检查上一次决策是否有效，统计性能信息（掉帧过多，技能利用率过低都可能是 bug 引起的）
- 无药可救时：吞下在计算过程中出现的异常，让 AI 尽可能在下一帧继续工作。

这些无关 AI 决策的代码帮助我快速发现并修复 bug。

安全措施——执行时

将所有的执行指令集中到 SafeOp 名称空间中进行，它将检查每个参数的有效性。
至于调用位置，C++20 中有 std::source_location，调用时传进去。
注意它并不能代替决策时对参数有效性的检查，要把它独立于 AI 逻辑之外。

安全措施——执行后

存储上一帧的决策与调用方，检查该帧的状态是否为期望状态。
此外还统计掉帧，检查行走是否被卡死，统计移动、火球、陨石阵的利用率。

安全措施——无药可救时

C++ Exception

以帧为处理粒度，在 playerAI 函数套一个就完事了。

signal

段错误会触发 SIGSEGV 信号，可以替换信号处理函数。

errno

对于数学函数的异常，只能在单帧处理结束后检查 errno。

局面分析

分析全局局面对于决策是很有必要的。
Analyzer 提供了如下信息：

- 上一次 bonus 消失的时间
- 上一帧敌人移动的方向
- 检查单位向目标的行进速度是否过慢（用于突击）
- 列出最近攻击该单位的敌人编号

采样系统

起源

当没有数学模型来指导决策怎么办？随机!!!
方法简单粗暴：随机满足要求的决策，选取估值最高的。

进阶

但是一堆 if 和 for 的嵌套好难写!!! 还有更复杂的需求：

- 以一定概率忽视某一考虑项
- 各类估值按照字典序比较
- 尽可能选取满足某一条件的元素

GROES

将这些需求归类，我搞出了一个通用的采样系统!!!
采样系统由 3 类估值函数组成：

- request：强制要求，无视优先级
- optional：可选要求，会被转换为 estimate
- estimate：估值，将 Point 映射为 double，按照字典序比较

GROES 实现

感谢伟大的 lambda、std::function、Templates 和 Proxy Classes!!! ~~但这并不能改变我将来转 Rust 的决定~~。
整个系统由 PointSampler 类维护，使用 Generator 来构造 PointSampler，然后添加一堆的 ROE 描述需求，最后调用 sample 方法得到结果。
像 Parallel Patterns Library 的一连串.then()，可以使用链式调用一句搞定 sample。
对于那些可有可无的需求，增加一个 apply 方法让 lambda 修改自己以避免打断链式调用。
对于 bonus 这种需要精确降落的移动，可以通过 hotPoint 来引导选择。

GROES 扩展

选择攻击/跟踪对象时，还有从一些 id 中选出一个或几个 id 的需求。
再写一个 DiscreteSampler 来满足这个需求。

estimate 的优先级不应过高，除非使用一些舍入/分段函数嵌套在外面。
Sampler 自带一些 Helper Function 和重载运算符避免写 lambda。
Map 和 Attack 模块提供了基本的 Sampler 以确保决策有效。
它们的 Generator 使用的不是 std::mt19937_64 而是 Halton 序列。序列生成代码是从我的 CUDA 软件渲染器上扒下来的。

GROES 表达

是不是很像 Scratch???

```
makeAttackBaseSampler()
    .request([](const Point& cp) {
        for(Point bp : getMap().bonus_places)
            if(dist(bp, cp) < 2.0 * CONST::bonus_radius)
                return false;
        return true;
    })
    .optional(unitHP() < CONST::human_hp * 0.2)
    .estimate(minv(distance(p)))
    .sample();
```

任务分配

经典的 1-3-1 分配，在输送单位安全时，空闲的单位会返回去输送新的水晶。
bonus 处的单位是永久职业的，不允许残血的单位到那里去送死。

移动

- A* 仅提供前进方向指导。
- 优先躲避 meteor，其次是 fireball。
- 有时会与己方单位/敌方单位保持距离。
- bonus 处使用保守策略。

攻击

- 以消灭敌人有生力量为目标。
- 尽可能同时影响到多个敌人。
- 选取近的敌人作为目标。
- meteor 仅用于阻挡敌人前进。
- 在无法打到目标时，识别转角并封锁。
- bonus 处使用保守策略。

未使用的 idea

- 远程扫射(看不出效果)
- 预测攻击估值函数(准确率 30% 以下)
- 惰性分块 Dijkstra(已被 A* 代替)
- 寻找掩体(不会写)
- 平行同步齐射(射速太慢)
- 惯性预瞄(虽然惯性预测的准确率在 60% 到 80% 左右,但是剩下的 miss 应该都是敌人躲火球的时候发生的)

End

感谢大家的聆听！
感谢秦岳同学、吕紫剑同学在 APIO、THUWC、THUSC 上的宣传。
该 slides 的模板：

`https://github.com/edasubert/beamerMaterialDesign`