加绒　　于2025年5月20日

Screw Jam游戏请参见如下网址：

https://play.google.com/store/apps/details?id=com.quok.screwJam

本文通过建立数学模型，分析玩家完成Screw Jam游戏中新生成的盒子的难度，并将其量化。

对于中间的盒子的难度指标

1、指标函数

记生成的盒子颜色为 c，生成 c 色盒子时暂存区中已有的且颜色不为 c 的钉子数量为 M，游戏开始时同种颜色下的钉子数目最大为 N（单局游戏内不变），最上面两层彩条的 c 色钉子的数目为 n_c，当前盘面中的彩条层数为 l。

定义评估 c 色盒子的完成难度的 $dct(c)$ 指标函数如下。

① 当 $M = 5$ 时，$dct(c) = 100$. 即不可能完成

② 当 $M < 5$ 时，分两种情况：

a. 若 $l > 2$，则 $dct(c) = \frac{M}{5} \times 30 + \frac{N-n_c}{N} \times 70$.

b. 若 $l \leq 2$，则 $dct(c) = \frac{M}{5} \times 100$.

综上所述，完成 c 色盒子的难度评估为

$$dct(c) = \begin{cases} 100, & M = 5, \\ \frac{M}{5} \times 30 + \frac{N-n_c}{N} \times 70, & M < 5 \text{ 并且 } l > 2, \\ \frac{M}{5} \times 100 & M < 5 \text{ 并且 } l \leq 2. \end{cases}$$

2、说明

① 当 $M=5$ 时，游戏失败，直接赋值 $dct=100.$ 意为不可能完成

② 当 $M<5$ 时，希望难度指标 dct 能反应游戏本身的**客观难度**加上玩家**心理压力（容错）**难度。游戏中，玩家比较容易点击的钉子位于最顶上两层彩条，所以只需要统计前两层彩条中 c 色钉子的个数 n_c，n_c 越多难度越低，反之难度越大（引入 N 是做归一化处理，得到 $\frac{N-n_c}{N}$）。此外，暂存区中钉子的数量直接反映了玩家的容错，暂存区中的非 c 色的钉子越多，允许玩家失误的空间越少，难度越高，归一化处理后就是 $\frac{M}{5}$.

而 30 和 70 是默认使用的权重指标，因为假设玩家足够聪明，我认为 n_c 对玩家完成 c 色盒子的难度影响更大一些，赋值给高一些。

此外，当 $l\leq 2$ 作为特殊情况讨论，此时彩条层数不大于2，可以认为玩家已经能够点击到盘面上的所有钉子，所以只考虑了 M 对 c 色盒子的完成难度的影响。

3、代码

```
public int maxScrew = -1; //存储游戏开始时，单种颜色下最多的钉子数目
public override float EvalBoxDifficulty(BoardData data, TypeColor color)
{
    int M, ncolor;
    double dft;
    bool isLayerMoreThan2;

    Dictionary<TypeColor, int> allScrew = new Dictionary<TypeColor, int>();

    // 盘面上的螺钉
    ncolor = 0; //顶上两层同color的钉子数目
    isLayerMoreThan2 = false;
```

```
    foreach (var s in data.ScrewsOnBoard.Keys)
    {
        if (maxScrew < 0)
        {
            if (!allScrew.ContainsKey(s.TypeColor))
            {
                allScrew[s.TypeColor] = 0;
            }
            allScrew[s.TypeColor]++;
        }

        // 获取顶上两层的彩条
        List<BarItem> bars =
ScrewUtils.GetBarsBlockingScrew(s);
        if (bars.Count > 1)
        {
            isLayerMoreThan2 = true;
            continue;
        }
        if (s.TypeColor == color)
            ncolor++;
    }

    if (maxScrew < 0)
    {
        maxScrew = allScrew.OrderByDescending(kvp =>
kvp.Value).First().Value;
    }

    // 暂存区里面的钉子
    M = 0;
    foreach (var s in data.ScrewsOnTempArea)
    {
        if (s.TypeColor != color)
```

```
            {
                M++;
            }
        }

        // 评估函数
        if (isLayerMoreThan2)
            dft = M / 5.0 * 30.0 + (double)(maxScrew - ncolor)
/ (double)maxScrew * 70.0;
        else
            dft = M / 5.0 * 100.0;

        return (float)dft;
}
```

4、测试

实际游玩 Level 28关卡，测试发现 $dft > 55$ 的盒子较难完成，特别是游戏刚开始的时候，需要一些运气，而 $dft \leq 55$ 的盒子只需要玩家善于观察，没有误操作的情况出现，一般都能完成。

对于左边的盒子的难度指标

1、难度评估函数

记当前正在消除的盒子（中间的盒子）颜色为 a，新生成的盒子（左边的盒子）颜色为 b. 生成 b 色盒子时暂存区中已有的钉子数量为 M，其中 b 色钉子数量为 K，则玩家消除该生成的 b 色盒子时，暂存区中的钉子数目为 $M - \min\{K, 3\}$. 游戏开始时同种颜色下的钉子数目最大为 N. 最上面两层彩条的 c 色（c 可以取 a 或 b）钉子的数目为 n_c，当前盘面中的彩条层数为 l.

采用与上述相同的思路，定义评估 a 色盒子的完成难度的 $dct(a)$ 指标如下。

① 当 $M=5$ 时，$dct(a)=100$. 即不可能完成

② 当 $M<5$ 时，分两种情况：

a. 若 $l>2$，则 $dct(a)=\frac{M}{5}\times 30+\frac{N-n_a}{N}\times 70$.

b. 若 $l\leq 2$，则 $dct(a)=\frac{M}{5}\times 100$.

定义评估 b 色盒子的完成难度的 $dct(b)$ 指标如下。

① 当 $M=5$ 时，$dct(b)=100$. 游戏失败，不可能完成。

② 当 $M<5$ 时，分3种情况：（可以认为 b 色盒子能否完成也受 a 色盒子的影响，比方说玩家如果无法完成 a 色盒子那自然也完不成 b 色盒子，所以除了上述的因素外，还引入了 $dct(a)$，两者取最大值）

a. 当 $K\geq 3$ 时，$dct(b)=dct(a)$.

b. 若 $K<3$ 并且 $l>2$，则 $dct(b)=\max\{\frac{M-K}{5}\times 30+\frac{N-n_b}{N}\times 70, dct(a)\}$.

c. 若 $K<3$ 并且 $l\leq 2$，则 $dct(b)=\max\{\frac{M-K}{5}\times 100, dct(a)\}$.

2、生成难度位于某区间的盒子算法

遍历当前盘面和暂存区的钉子的颜色，找到一种颜色 c，使得 $low\leq dct(c)\leq high$，返回 c；如果没找到满足要求的颜色，则返回使得玩家最容易完成的盒子颜色。

3、代码

```
public int maxScrew = -1; //存储游戏开始时，单种颜色下最多的钉子数目
```

```
private TypeColor TryGenerateBoxQ4(BoardData data, float
low, float high)
{
    float dft, mindft;
    TypeColor mincolor = TypeColor.NONE;

    Dictionary<TypeColor, int> allScrew = new
Dictionary<TypeColor, int>();

    // 盘面上的螺钉
    foreach (var s in data.ScrewsOnBoard.Keys)
    {
        if (!allScrew.ContainsKey(s.TypeColor))
        {
            allScrew[s.TypeColor] = 0;
        }
        allScrew[s.TypeColor]++;
    }

    // 暂存区里面的钉子
    foreach (var s in data.ScrewsOnTempArea)
    {
        if (!allScrew.ContainsKey(s.TypeColor))
        {
            allScrew[s.TypeColor] = 0;
        }
        allScrew[s.TypeColor]++;
    }

    mindft = 100;
    foreach (var s in allScrew.Keys)
    {
        dft = EvalBoxDifficulty(data, s);
        if (low <= dft && dft <= high)
```

```
        {
            return s;
        }
        if(dft < mindft)
        {
            mindft = dft;
            mincolor = s;
        }
    }

    Debug.Log($"无满足条件的盒子颜色");
    //return ScrewUtils.GenerateRandomBox(data);
    if (mincolor != TypeColor.NONE)
        return mincolor;
    else
        return allScrew.OrderByDescending(kvp =>
kvp.Value).First().Key;

}

public override float EvalBoxDifficulty(BoardData data,
TypeColor colorb)
{
    int M, K, na, nb;
    double dfta, dftb;
    bool isLayerMoreThan2;
    TypeColor colora = TypeColor.NONE;

    foreach (var a in data.Boxes)
    {
        colora = a.TypeColor;
    }
```

```
    Dictionary<TypeColor, int> allScrew = new
Dictionary<TypeColor, int>();

    // 盘面上的螺钉
    na = 0;
    nb = 0;
    isLayerMoreThan2 = false;
    foreach (var s in data.ScrewsOnBoard.Keys)
    {
        if (maxScrew < 0)
        {
            if (!allScrew.ContainsKey(s.TypeColor))
            {
                allScrew[s.TypeColor] = 0;
            }
            allScrew[s.TypeColor]++;
        }

        // 获取顶上两层的彩条
        List<BarItem> bars =
ScrewUtils.GetBarsBlockingScrew(s);
        if (bars.Count > 1)
        {
            isLayerMoreThan2 = true;
            continue;
        }
        if (s.TypeColor == colorb)
            nb++;
        if (s.TypeColor == colora)
            na++;
    }

    if (maxScrew < 0)
    {
```

```
            maxScrew = allScrew.OrderByDescending(kvp =>
kvp.Value).First().Value;
        }

        // 暂存区里面的钉子
        M = 0;
        K = 0;
        foreach (var s in data.ScrewsOnTempArea)
        {
            M++;
            if (s.TypeColor == colorb)
            {
                K++;
            }
        }

        // 评估函数 a
        if (colora == TypeColor.NONE)
        {
            dfta = 0;
        }
        else
        {
            if (isLayerMoreThan2)
                dfta = M / 5.0 * 30.0 + (double)(maxScrew - na)
/ (double)maxScrew * 70.0;
            else
                dfta = M / 5.0 * 100.0;
        }

        // 评估函数 b
        if (K >= 3)
        {
            dftb = dfta;
```

```
    }
    else
    {
        if (isLayerMoreThan2)
        {
            dftb = (M - K) / 5.0 * 30.0 + (double)(maxScrew
- nb) / (double)maxScrew * 70.0;
            dftb = MAX(dftb, dfta);
        }
        else
        {
            dftb = (M - K) / 5.0 * 100.0;
            dftb = MAX(dftb, dfta);
        }
    }

    //Debug.Log($"isLayerMoreThan2: {isLayerMoreThan2}");
    //Debug.Log($"M: {M}");
    //Debug.Log($"maxScrew: {maxScrew}");
    //Debug.Log($"ncolor: {ncolor}");
    //return (float)dftb;


    return (float)dftb;
}

private double MAX(double a,double b)
{
    return (a > b) ? a : b;
}
```

4、测试

测试中重复游玩 Level 28关卡，发现 $dft(b) > 55$ 的盒子较难完成，需要一些运气，特别是当出现了一个 $dft(b) > 55$ 的盒子后接着后面的几个盒子会越来越难完成，体现了数学模型中 a 色盒子的完成情况对 b 色盒子的完成情况的影响，而 $dft \leq 55$ 的盒子只需要玩家善于观察，没有误操作的情况出现，一般都能完成。