Сложность должна формироваться за счет:

1. анлока контента

(2) изменения набора объектов в течении одной сессии

(3) именения дистанций между объектами в течении одной сессии

# **1 АНЛОК КОНТЕНТА**

Открываем объекты по мере того, как долго играет юзер.

***Это означает, что каждый объект в игре становится доступным для генерации только после того, как игрок суммарно (! за все время) “набегал” то кол-во секунд, которое мы присвоили для этого объекта.***

Например:

|  |  |
| --- | --- |
| **Object (id)** | **Time sum in game (sec)** |
| Tree 1 | 0s |
| Stone 1 | 60s |
| Stone 2 | 90s |
| Tree 2 | 120s |

// Из таблицы видно, что по дефолту до первых 60(сек) игрок сможет увидеть в игре только объект Tree 1 и никаких объектов больше, даже если веса для них будут выставлены.

# **2 ИЗМЕНЕНИЕ НАБОРА ОБЪЕКТОВ**

Это то, что мы, по идее, уже должны были сделать.

Все сводится к тому, что мы задаем временные интервалы, объекты, которые могут спауниться в этих интервалах, и веса этих объектов:

|  |  |  |
| --- | --- | --- |
| **Time interval** | **Object id** | **Objects’ weights** |
| 30 | Tree 1 | 20 |
| Stone 1 | 80 |
| 90 | Tree 1 | 30 |
| Tree 2 | 30 |
| Stone 1 | 25 |
| Deer | 15 |

// 30 в первой колонке означает интервал 0-30, 90 во второй - интервал от 30-90

// тайминги в рамках одного забега (!)

***Помимо одиночных объектов, было бы хорошо присваивать object id не только для одиночных объектов, но и для групп объектов. Например, заранее созданная группа из камня и дерева - хорошо бы иметь возможность считать такую группу одним объектом, например, назвать ее stone+tree, и задавать ей веса так же, как и просто камню или просто дереву отдельно***

# **3 ИЗМЕНЕНИЕ ДИСТАНЦИЙ МЕЖДУ ОБЪЕКТАМИ**

Очевидно нам надо варьировать следующими дистанциями (или у тебя это время):

* **Min distance to next object** *- минимальная дистанция до следующего объекта (слева и справа)*
* **Max distance before object** *- максимальная дистанция до следующего объекта (слева и справа)*

Вынеси эти настройки локально в префаб каждого из игровых объектов, чтобы можно было задать эти дистанции разными:

|  |  |  |
| --- | --- | --- |
| **Object id** | **Min dist to next obj** | **Max dist to next obj** |
| Tree1 | 5 | 10 |
| Stone 1 | 5 | 7 |

**Эти дистанции позволят нам задать дефолтную плотность объектов, но если она не будет меняться, будет скучно играть :)**

Поэтому, давай для каждого из наших временных промежутков, будем увеличивать плотность, домножая эти дефолтные расстояния на коэффициенты:

|  |  |
| --- | --- |
| Time, sec | Coeff |
| 30 | 1.2 |
| 60 | 0.9 |
| 90 | 0.8 |

// Это будет означать, что, например, на промежутке 60-90 секунд, все минимальные дистанции будут равны дефолтным, помноженным на 0.9 (то есть будут меньше, чем на предыдущем)

# **4 ПОВТОРЕНИЕ - МАТЬ УЧЕНИЯ**

Еще раз об алгоритме:

1. Смотрим суммарное время в игре и определяем, какие объекты мы вообще можем сейчас использовать в спаунере, а какие нет
2. Во время забега смотрим, есть ли у объекта какой-либо вес на текущем временном промежутке, если он не 0, то спауним объекты с определенными весами
3. Учитываем коэффициенты и спауним все объекты с дистанциями, актуальными для текущего временного интервала